

WG3:HBA-008

H2-2003-310

August, 2003

ISO
International Organization for Standardization
ANSI
American National Standards Institute

ANSI TC NCITS H2
ISO/IEC JTC 1/SC 32/WG 3
Database

Title: (ISO-ANSI Working Draft) Information and Definition Schemas (SQL/Schemata)

Author: Jim Melton (Editor)

References:

- 1) WG3:HBA-002 = H2-2003-304 = 5WD-01-Framework-2003-09, *WD 9075-1 (SQL/Framework)*, September, 2003
- 2) WG3:HBA-003 = H2-2003-305 = 5WD-02-Foundation-2003-09, *WD 9075-2 (SQL/Foundation)*, September, 2003
- 3) WG3:HBA-004 = H2-2003-306 = 5WD-03-CLI-2003-09, *WD 9075-3 (SQL/CLI)*, September, 2003
- 4) WG3:HBA-005 = H2-2003-307 = 5WD-04-PSM-2003-09, *WD 9075-4 (SQL/PSM)*, September, 2003
- 5) WG3:HBA-006 = H2-2003-308 = 5WD-09-MED-2003-09, *WD 9075-9 (SQL/MED)*, September, 2003
- 6) WG3:HBA-007 = H2-2003-309 = 5WD-10-OLB-2003-09, *WD 9075-10 (SQL/OLB)*, September, 2003
- 7) WG3:HBA-008 = H2-2003-310 = 5WD-11-Schemata-2003-09, *WD 9075-11 (SQL/Schemata)*, September, 2003
- 8) WG3:HBA-009 = H2-2003-311 = 5WD-13-JRT-2003-09, *WD 9075-13 (SQL/JRT)*, September, 2003

- 9) WG3:HBA-010 = H2-2003-312 = 5WD-14-XML-2003-09, *WD 9075-14 (SQL/XML)*, September, 2003

ISO/IEC JTC 1/SC 32

Date: 2003-07-25

ISO/IEC 9075-11:2003 (E)

ISO/IEC JTC 1/SC 32/WG 3

United States of America (ANSI)

Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)

Technologies de l'information — Langages de base de données — SQL — Partie 11: Information et Definition Schémas (SQL/Schemata)

Document type: International standard

Document subtype:

Document stage: (4) Approval

Document language: English

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, photocopying, recording, or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to ISO at the address below or ISO's member body in the country of the requester.

*Copyright Manager
ISO Central Secretariat
1 rue de Varembé
1211 Geneva 20 Switzerland
tel. +41 22 749 0111
fax +41 22 734 1079
internet: iso@iso.ch*

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

Page

Foreword.....	ix
Introduction.....	x
1 Scope.....	1
2 Normative references.....	3
2.1 JTC1 standards.....	3
3 Definitions, notations, and conventions.....	5
3.1 Conventions.....	5
4 Concepts.....	7
4.1 Introduction to the Definition Schema.....	7
4.2 Introduction to the Information Schema.....	8
5 Information Schema.....	9
5.1 INFORMATION_SCHEMA Schema.....	9
5.2 INFORMATION_SCHEMA_CATALOG_NAME base table.....	10
5.3 CARDINAL_NUMBER domain.....	11
5.4 CHARACTER_DATA domain.....	12
5.5 SQL_IDENTIFIER domain.....	13
5.6 TIME_STAMP domain.....	14
5.7 ADMINISTRABLE_ROLE_AUTHORIZATIONS view.....	15
5.8 APPLICABLE_ROLES view.....	16
5.9 ASSERTIONS view.....	17
5.10 ATTRIBUTES view.....	18
5.11 CHARACTER_SETS view.....	20
5.12 CHECK_CONSTRAINT_ROUTINE_USAGE view.....	21
5.13 CHECK_CONSTRAINTS view.....	22
5.14 COLLATIONS view.....	23
5.15 COLLATION_CHARACTER_SET_APPLICABILITY view.....	24
5.16 COLUMN_COLUMN_USAGE view.....	25
5.17 COLUMN_DOMAIN_USAGE view.....	26
5.18 COLUMN_PRIVILEGES view.....	27
5.19 COLUMN_UDT_USAGE view.....	28
5.20 COLUMNS view.....	29
5.21 CONSTRAINT_COLUMN_USAGE view.....	31
5.22 CONSTRAINT_TABLE_USAGE view.....	33
5.23 DATA_TYPE_PRIVILEGES view.....	35
5.24 DIRECT_SUPERTABLES view.....	37
5.25 DIRECT_SUPERTYPES view.....	38

5.26	DOMAIN_CONSTRAINTS view.....	39
5.27	DOMAINS view.....	40
5.28	ELEMENT_TYPES view.....	42
5.29	ENABLED_ROLES view.....	43
5.30	FIELDS view.....	44
5.31	KEY_COLUMN_USAGE view.....	45
5.32	METHOD_SPECIFICATION_PARAMETERS view.....	47
5.33	METHOD_SPECIFICATIONS view.....	49
5.34	PARAMETERS view.....	51
5.35	REFERENCED_TYPES view.....	53
5.36	REFERENTIAL_CONSTRAINTS view.....	54
5.37	ROLE_COLUMN_GRANTS view.....	55
5.38	ROLE_ROUTINE_GRANTS view.....	56
5.39	ROLE_TABLE_GRANTS view.....	57
5.40	ROLE_TABLE_METHOD_GRANTS view.....	58
5.41	ROLE_USAGE_GRANTS view.....	59
5.42	ROLE_UDT_GRANTS view.....	60
5.43	ROUTINE_COLUMN_USAGE view.....	61
5.44	ROUTINE_PRIVILEGES view.....	62
5.45	ROUTINE_ROUTINE_USAGE view.....	63
5.46	ROUTINE_SEQUENCE_USAGE view.....	64
5.47	ROUTINE_TABLE_USAGE view.....	65
5.48	ROUTINES view.....	66
5.49	SCHEMATA view.....	69
5.50	SEQUENCES view.....	70
5.51	SQL_FEATURES view.....	71
5.52	SQL_IMPLEMENTATION_INFO view.....	72
5.53	SQL_LANGUAGES view.....	73
5.54	SQL_PACKAGES view.....	74
5.55	SQL_PARTS view.....	75
5.56	SQL_SIZING view.....	76
5.57	SQL_SIZING_PROFILES view.....	77
5.58	TABLE_CONSTRAINTS view.....	78
5.59	TABLE_METHOD_PRIVILEGES view.....	79
5.60	TABLE_PRIVILEGES view.....	80
5.61	TABLES view.....	81
5.62	TRANSFORMS view.....	82
5.63	TRANSLATIONS view.....	83
5.64	TRIGGERED_UPDATE_COLUMNS view.....	84
5.65	TRIGGER_COLUMN_USAGE view.....	85
5.66	TRIGGER_ROUTINE_USAGE view.....	86
5.67	TRIGGER_SEQUENCE_USAGE view.....	87

5.68	TRIGGER_TABLE_USAGE view.....	88
5.69	TRIGGERS view.....	89
5.70	UDT_PRIVILEGES view.....	91
5.71	USAGE_PRIVILEGES view.....	92
5.72	USER_DEFINED_TYPES view.....	93
5.73	VIEW_COLUMN_USAGE view.....	95
5.74	VIEW_ROUTINE_USAGE view.....	96
5.75	VIEW_TABLE_USAGE view.....	97
5.76	VIEWS view.....	98
5.77	Short name views.....	99
6	Definition Schema.....	117
6.1	DEFINITION_SCHEMA Schema.....	117
6.2	EQUAL_KEY_DEGREES assertion.....	118
6.3	KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 assertion.....	119
6.4	UNIQUE_CONSTRAINT_NAME assertion.....	120
6.5	ASSERTIONS base table.....	121
6.6	ATTRIBUTES base table.....	123
6.7	AUTHORIZATIONS base table.....	125
6.8	CATALOG_NAMES base table.....	126
6.9	CHARACTER_ENCODING_FORMS base table.....	127
6.10	CHARACTER_REPERTOIRES base table.....	128
6.11	CHARACTER_SETS base table.....	129
6.12	CHECK_COLUMN_USAGE base table.....	131
6.13	CHECK_CONSTRAINT_ROUTINE_USAGE base table.....	132
6.14	CHECK_CONSTRAINTS base table.....	133
6.15	CHECK_TABLE_USAGE base table.....	134
6.16	COLLATIONS base table.....	135
6.17	COLLATION_CHARACTER_SET_APPLICABILITY base table.....	136
6.18	COLUMN_COLUMN_USAGE base table.....	137
6.19	COLUMN_PRIVILEGES base table.....	138
6.20	COLUMNS base table.....	140
6.21	DATA_TYPE_DESCRIPTOR base table.....	145
6.22	DIRECT_SUPERTABLES base table.....	153
6.23	DIRECT_SUPERTYPES base table.....	155
6.24	DOMAIN_CONSTRAINTS base table.....	157
6.25	DOMAINS base table.....	159
6.26	ELEMENT_TYPES base table.....	160
6.27	FIELDS base table.....	162
6.28	KEY_COLUMN_USAGE base table.....	164
6.29	METHOD_SPECIFICATION_PARAMETERS base table.....	167
6.30	METHOD_SPECIFICATIONS base table.....	169
6.31	PARAMETERS base table.....	174

6.32	REFERENCED_TYPES base table.	177
6.33	REFERENTIAL_CONSTRAINTS base table.	179
6.34	ROLE_AUTHORIZATION_DESCRIPTORs base table.	182
6.35	ROUTINE_COLUMN_USAGE base table.	184
6.36	ROUTINE_PRIVILEGES base table.	186
6.37	ROUTINE_ROUTINE_USAGE base table.	188
6.38	ROUTINE_SEQUENCE_USAGE base table.	189
6.39	ROUTINE_TABLE_USAGE base table.	190
6.40	ROUTINES base table.	192
6.41	SCHEMATA base table.	200
6.42	SEQUENCES base table.	202
6.43	SQL_CONFORMANCE base table.	204
6.44	SQL_IMPLEMENTATION_INFO base table.	206
6.45	SQL_LANGUAGES base table.	207
6.46	SQL_SIZING base table.	210
6.47	SQL_SIZING_PROFILES base table.	212
6.48	TABLE_CONSTRAINTS base table.	214
6.49	TABLE_METHOD_PRIVILEGES base table.	217
6.50	TABLE_PRIVILEGES base table.	219
6.51	TABLES base table.	222
6.52	TRANSFORMS base table.	226
6.53	TRANSLATIONS base table.	228
6.54	TRIGGERED_UPDATE_COLUMNS base table.	230
6.55	TRIGGER_COLUMN_USAGE base table.	232
6.56	TRIGGER_ROUTINE_USAGE base table.	234
6.57	TRIGGER_SEQUENCE_USAGE base table.	235
6.58	TRIGGER_TABLE_USAGE base table.	236
6.59	TRIGGERS base table.	238
6.60	USAGE_PRIVILEGES base table.	241
6.61	USER_DEFINED_TYPE_PRIVILEGES base table.	243
6.62	USER_DEFINED_TYPES base table.	245
6.63	VIEW_COLUMN_USAGE base table.	249
6.64	VIEW_ROUTINE_USAGE base table.	250
6.65	VIEW_TABLE_USAGE base table.	251
6.66	VIEWS base table.	252
7	Conformance.	255
7.1	Claims of conformance to SQL/Schemata.	255
7.2	Additional conformance requirements for SQL/Schemata.	255
7.3	Implied feature relationships of SQL/Schemata.	255
Annex A	SQL Conformance Summary.	257
Annex B	Implementation-defined elements.	273

Annex C Deprecated features.....275

Annex D Incompatibilities with ISO/IEC 9075:1999.....277

Annex E SQL feature taxonomy.....279

Index.....283

Tables

Table	Page
1 Implied feature relationships of SQL/Schemata.	255
2 Feature taxonomy and definition for mandatory features.	279
3 Feature taxonomy for optional features.	281

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9075-11 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ISO/IEC 9075 consists of the following parts, under the general title *Information technology — Database languages — SQL*:

- Part 1: Framework (SQL/Framework)
- Part 2: Foundation (SQL/Foundation)
- Part 3: Call-Level Interface (SQL/CLI)
- Part 4: Persistent Stored Modules (SQL/PSM)
- Part 9: Management of External Data (SQL/MED)
- Part 10: Object Language Bindings (SQL/OLB)
- Part 11: Information and Definition Schema (SQL/Schemata)
- Part 13: Routines and Types Using the Java™ Programming Language (SQL/JRT)
- Part 14: XML-Related Specifications (SQL/XML)

Annexes A, B, C, D, and E of this part of ISO/IEC 9075 are for information only.

Introduction

The organization of this part of ISO/IEC 9075 is as follows:

- 1) [Clause 1, “Scope”](#), specifies the scope of this part of ISO/IEC 9075.
- 2) [Clause 2, “Normative references”](#), identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.
- 3) [Clause 3, “Definitions, notations, and conventions”](#), defines the notations and conventions used in this part of ISO/IEC 9075.
- 4) [Clause 4, “Concepts”](#), presents concepts used in the definition of Persistent SQL modules.
- 5) [Clause 5, “Information Schema”](#), defines viewed tables that contain schema information.
- 6) [Clause 6, “Definition Schema”](#), defines base tables on which the viewed tables containing schema information depend.
- 7) [Clause 7, “Conformance”](#), defines the criteria for conformance to this part of ISO/IEC 9075.
- 8) [Annex A, “SQL Conformance Summary”](#), is an informative Annex. It summarizes the conformance requirements of the SQL language.
- 9) [Annex B, “Implementation-defined elements”](#), is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.
- 10) [Annex C, “Deprecated features”](#), is an informative Annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 9075.
- 11) [Annex D, “Incompatibilities with ISO/IEC 9075:1999”](#), is an informative Annex. It lists the incompatibilities between this edition of this part of ISO/IEC 9075 and ISO/IEC 9075:1999.
- 12) [Annex E, “SQL feature taxonomy”](#), is an informative Annex. It identifies features of the SQL language specified in this part of ISO/IEC 9075 by a numeric identifier and a short descriptive name. This taxonomy is used to specify conformance and may be used to develop other profiles involving the SQL language.

In the text of this part of ISO/IEC 9075, Clauses begin a new odd-numbered page, and in [Clause 5, “Information Schema”](#), through [Clause 7, “Conformance”](#), Subclauses begin a new page. Any resulting blank space is not significant.

Information technology— Database languages —SQL —

Part 11: Information and Definition Schemas (SQL/Schemata)

1 Scope

This part of ISO/IEC 9075 specifies an Information Schema and a Definition Schema that describes:

- The SQL object identifier of ISO/IEC 9075.
- The structure and integrity constraints of SQL-data.
- The security and authorization specifications relating to SQL-data.
- The features, subfeatures, and packages of ISO/IEC 9075, and the support that each of these has in an SQL-implementation.
- The SQL-implementation information and sizing items of ISO/IEC 9075 and the values supported by an SQL-implementation.

This page intentionally left blank.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 JTC1 standards

[Framework] ISO/IEC FCD 9075-1:2003, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[Foundation] ISO/IEC FCD 9075-2:2003, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

This page intentionally left blank.

3 Definitions, notations, and conventions

This Clause modifies Clause 3, “Definitions, notations, and conventions”, in ISO/IEC 9075-2.

3.1 Conventions

This Subclause modifies Subclause 3.3, “Conventions”, in ISO/IEC 9075-2.

Insert this paragraph The Descriptions in Clause 6, “Definition Schema”, sometimes specify values that are to appear in rows of base tables. When such a value is given as a sequence of capital letters enclosed in <double quote>s, it denotes the same value as would be denoted by the <character string literal> obtained by replacing the enclosing <double quote>s by <quote>s. The need for such notation arises when the column in question sometimes, in other rows, contains character strings denoting SQL expressions, possibly even <character string literal>s.

This page intentionally left blank.

4 Concepts

This Clause modifies [Clause 4](#), “Concepts”, in ISO/IEC 9075-2.

4.1 Introduction to the Definition Schema

The Definition Schema base tables are defined as being in a schema named DEFINITION_SCHEMA. The table definitions are as complete as the definitional power of SQL allows. The table definitions are supplemented with assertions where appropriate.

The only purpose of the Definition Schema is to provide a data model to support the Information Schema and to assist understanding. An SQL-implementation need do no more than simulate the existence of the Definition Schema, as viewed through the Information Schema views. The specification does not imply that an SQL-implementation shall provide the functionality in the manner described in the Definition Schema.

The Definition Schema allows information to be stored about multiple catalogs. The defined constraints guarantee consistency, not only within a single catalog, but also over all catalogs described by the Definition Schema.

The way in which certain constraints are expressed caters for the possibility that an object is being referenced that exists in a catalog that is outside the purview of the Definition Schema containing the reference in question. For example, the definition of the VIEW_TABLE_USAGE base table in [Subclause 6.65](#), “VIEW_TABLE_USAGE base table”, includes the following constraint:

```
CONSTRAINT VIEW_TABLE_USAGE_CHECK_REFERENCES_TABLES
CHECK ( TABLE_CATALOG NOT IN
      ( SELECT CATALOG_NAME
        FROM SCHEMATA )
      OR
      ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
      ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM TABLES ) )
```

Either the table being used by the view exists in a catalog within this Definition Schema's purview, in which case its existence is guaranteed, or it is assumed but not guaranteed to exist in some catalog that is outside this Definition Schema's purview.

Because <unqualified schema name>s are prohibited by [SR 10](#)) of [Subclause 5.4](#), “Names and identifiers”, in ISO/IEC 9075-2, from specifying DEFINITION_SCHEMA, the Definition Schema cannot normally be accessed in an SQL-statement. However, view definitions in the Information Schema assume the existence of the Definition Schema and reference base tables whose <schema name> is DEFINITION_SCHEMA. They use the Definition Schema to define the content of the Information Schema. Regardless of [SR 14](#)) of [Subclause 5.4](#), “Names and identifiers”, in ISO/IEC 9075-2, the <schema name> DEFINITION_SCHEMA is never qualified by a <catalog name>. It is implementation-defined whether the DEFINITION_SCHEMA referenced by an INFORMATION_SCHEMA describes schemas in catalogs other than the catalog in which the INFORMATION_SCHEMA is located.

4.2 Introduction to the Information Schema

The views of the Information Schema are viewed tables defined in terms of the base tables of the Definition Schema.

The Information Schema views are defined as being in a schema named INFORMATION_SCHEMA, enabling these views to be accessed in the same way as any other tables in any other schema. SELECT on most of these views is granted to PUBLIC WITH GRANT OPTION, so that they can be queried by any user and so that SELECT privilege can be further granted on views that reference these Information Schema views. No other privilege is granted on them, so they cannot be updated.

In order to provide access to the same information that is available via the INFORMATION_SCHEMA to an SQL-Agent in an SQL-environment where the SQL-implementation does not support Feature F391, “Long identifiers”, alternative views are provided that use only short identifiers. The Information Schema also contains a small number of domains on which the columns of the Definition Schema are based. USAGE on all these domains is granted to PUBLIC WITH GRANT OPTION, so that they can be used by any user.

An SQL-implementation may define objects that are associated with INFORMATION_SCHEMA that are not defined in this Clause. An SQL-implementation or any future version of ISO/IEC 9075 may also add columns to tables that are defined in this Clause.

NOTE 1 — The Information Schema tables may be supposed to be represented in the Definition Schema in the same way as any other tables, and are hence self-describing.

NOTE 2 — The Information Schema is a definition of the SQL data model, specified as an SQL-schema, in terms of <SQL schema statement>s as defined in ISO/IEC 9075. Constraints defined in this Clause are not actual SQL constraints.

The representation of an <identifier> in the base tables and views of the Information Schema is by a character string corresponding to its <identifier body> (in the case of a <regular identifier>) or its <delimited identifier body> (in the case of a <delimited identifier>). Within this character string, any lower-case letter appearing in a <regular identifier> is replaced by the equivalent upper-case letter, and any <doublequote symbol> appearing in a <delimited identifier body> is replaced by a <double quote>. Where an <actual identifier> has multiple forms that are equal according to the rules of Subclause 8.2, “<comparison predicate>”, in ISO/IEC 9075-2, the form stored is that encountered at definition time.

5 Information Schema

5.1 INFORMATION_SCHEMA Schema

Function

Identify the schema that is to contain the Information Schema tables.

Definition

```
CREATE SCHEMA INFORMATION_SCHEMA  
  AUTHORIZATION INFORMATION_SCHEMA;
```

Conformance Rules

None.

5.2 INFORMATION_SCHEMA_CATALOG_NAME base table

Function

Identify the catalog that contains the Information Schema.

Definition

```
CREATE TABLE INFORMATION_SCHEMA_CATALOG_NAME
( CATALOG_NAME          SQL_IDENTIFIER,

  CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_PRIMARY_KEY
    PRIMARY KEY ( CATALOG_NAME ),

  CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_CHECK
    CHECK ( 1 = ( SELECT COUNT(*)
                  FROM INFORMATION_SCHEMA_CATALOG_NAME ) ),

  CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_FK_CATALOG_NAMES
    FOREIGN KEY ( CATALOG_NAME ) REFERENCES DEFINITION_SCHEMA.CATALOG_NAMES
);

GRANT SELECT ON TABLE INFORMATION_SCHEMA_CATALOG_NAME
TO PUBLIC WITH GRANT OPTION;
```

Description

- 1) The value of CATALOG_NAME is the name of the catalog in which this Information Schema resides.

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.
- 2) Without Feature F651, “Catalog name qualifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.

5.3 CARDINAL_NUMBER domain

Function

Define a domain that contains a non-negative number.

Definition

```
CREATE DOMAIN CARDINAL_NUMBER AS INTEGER
CONSTRAINT CARDINAL_NUMBER_DOMAIN_CHECK
CHECK ( VALUE >= 0 );
```

```
GRANT USAGE ON DOMAIN CARDINAL_NUMBER
TO PUBLIC WITH GRANT OPTION;
```

Description

- 1) The domain CARDINAL_NUMBER contains any non-negative number that is less than or equal to the implementation-defined maximum for INTEGER.

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.CARDINAL_NUMBER.

5.4 CHARACTER_DATA domain

Function

Define a domain that contains any character data.

Definition

```
CREATE DOMAIN CHARACTER_DATA AS  
    CHARACTER VARYING (ML)  
    CHARACTER SET SQL_TEXT;  
  
GRANT USAGE ON DOMAIN CHARACTER_DATA  
    TO PUBLIC WITH GRANT OPTION;
```

Description

- 1) This domain specifies any character data.
- 2) *ML* is the implementation-defined maximum length of a variable-length character string.

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHARACTER_DATA.

5.5 SQL_IDENTIFIER domain

Function

Define a domain that contains all valid <identifier body>s and <delimited identifier body>s.

Definition

```
CREATE DOMAIN SQL_IDENTIFIER AS  
    CHARACTER VARYING (L)  
    CHARACTER SET SQL_IDENTIFIER;  
  
GRANT USAGE ON DOMAIN SQL_IDENTIFIER  
    TO PUBLIC WITH GRANT OPTION;
```

Description

- 1) This domain specifies all variable-length character values that conform to the rules for formation and representation of an SQL <identifier body> or an SQL <delimited identifier body>.

NOTE 3 — There is no way in SQL to specify a <domain constraint> that would be true for the body of any valid SQL <regular identifier> or <delimited identifier> and false for all other character string values.

- 2) *L* is the implementation-defined maximum length of <identifier body> and <delimited identifier body>.

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IDENTIFIER.

5.6 TIME_STAMP domain

Function

Define a domain that contains a timestamp.

Definition

```
CREATE DOMAIN TIME_STAMP AS TIMESTAMP(2) WITH TIME ZONE
    DEFAULT CURRENT_TIMESTAMP(2);

GRANT USAGE ON DOMAIN TIME_STAMP
    TO PUBLIC WITH GRANT OPTION;
```

Description

- 1) The domain TIME_STAMP contains an SQL timestamp value.

Conformance Rules

- 1) Without Feature F251, “Domain support”, and Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TIME_STAMP.

5.7 ADMINISTRABLE_ROLE_AUTHORIZATIONS view

Function

Identify role authorizations for which the current user or role has WITH ADMIN OPTION.

Definition

```
CREATE VIEW ADMINISTRABLE_ROLE_AUTHORIZATIONS AS
  SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORs
  WHERE ROLE_NAME IN
    ( SELECT ROLE_NAME
      FROM INFORMATION_SCHEMA.APPLICABLE_ROLES
      WHERE IS_GRANTABLE = 'YES' );

GRANT SELECT ON TABLE ADMINISTRABLE_ROLE_AUTHORIZATIONS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.

5.8 APPLICABLE_ROLES view

Function

Identifies the applicable roles for the current user.

Definition

```
CREATE RECURSIVE VIEW APPLICABLE_ROLES ( GRANTEE, ROLE_NAME, IS_GRANTABLE ) AS
  ( ( SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
      FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORs
      WHERE ( GRANTEE IN
              ( CURRENT_USER, 'PUBLIC' )
            OR
              GRANTEE IN
              ( SELECT ROLE_NAME
                FROM ENABLED_ROLES ) ) ) )
  UNION
  ( SELECT RAD.GRANTEE, RAD.ROLE_NAME, RAD.IS_GRANTABLE
    FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORs RAD
    JOIN
      APPLICABLE_ROLES R
    ON
      RAD.GRANTEE = R.ROLE_NAME ) );

GRANT SELECT ON TABLE APPLICABLE_ROLES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.APPLICABLE_ROLES.

5.9 ASSERTIONS view

Function

Identify the assertions defined in this catalog that are owned by a given user or role.

Definition

```
CREATE VIEW ASSERTIONS AS
  SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
         IS_DEFERRABLE, INITIALLY_DEFERRED
  FROM DEFINITION_SCHEMA.ASSERTIONS
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    CONSTRAINT_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ASSERTIONS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F521, “Assertions”, conforming SQL language shall not reference INFORMATION_SCHEMA.ASSERTIONS.

5.10 ATTRIBUTES view

Function

Identify the attributes of user-defined types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW ATTRIBUTES AS
  SELECT DISTINCT
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    A.ATTRIBUTE_NAME, ORDINAL_POSITION, ATTRIBUTE_DEFAULT,
    IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    D1.CHARACTER_SET_CATALOG, D1.CHARACTER_SET_SCHEMA, D1.CHARACTER_SET_NAME,
    D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
    D1.USER_DEFINED_TYPE_CATALOG AS ATTRIBUTE_UDT_CATALOG,
    D1.USER_DEFINED_TYPE_SCHEMA AS ATTRIBUTE_UDT_SCHEMA,
    D1.USER_DEFINED_TYPE_NAME AS ATTRIBUTE_UDT_NAME,
    D1.SCOPE_CATALOG, D1.SCOPE_SCHEMA, D1.SCOPE_NAME,
    MAXIMUM_CARDINALITY, A.DTD_IDENTIFIER, IS_DERIVED_REFERENCE_ATTRIBUTE
  FROM ( DEFINITION_SCHEMA.ATTRIBUTES AS A
    LEFT JOIN
      DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
    ON ( ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME,
          'USER-DEFINED TYPE', A.DTD_IDENTIFIER )
      = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
          D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
  WHERE ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME ) IN
    ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG,
      UDTP.USER_DEFINED_TYPE_SCHEMA,
      UDTP.USER_DEFINED_TYPE_NAME
    FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
    WHERE ( UDTP.GRANTEE IN
      ( 'PUBLIC', CURRENT_USER )
    OR
      UDTP.GRANTEE IN
      ( SELECT ROLE_NAME
        FROM ENABLED_ROLES ) ) )
  AND
    A.UDT_CATALOG
  = ( SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ATTRIBUTES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.

5.11 CHARACTER_SETS view

Function

Identify the character sets defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW CHARACTER_SETS AS
  SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
         CHARACTER_REPERTOIRE, FORM_OF_USE, NUMBER_OF_CHARACTERS,
         DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, DEFAULT_COLLATE_NAME
  FROM DEFINITION_SCHEMA.CHARACTER_SETS
 WHERE ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
        'CHARACTER SET') IN
        ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME,
              UP.OBJECT_TYPE
          FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
          WHERE ( UP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  UP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
        AND
          CHARACTER_SET_CATALOG
          = ( SELECT CATALOG_NAME
              FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE CHARACTER_SETS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHARACTER_SETS.

5.12 CHECK_CONSTRAINT_ROUTINE_USAGE view

Function

Identify each SQL-invoked routine owned by a given user or role on which a domain constraint, table check constraint or assertion defined in this catalog is dependent.

Definition

```
CREATE VIEW CHECK_CONSTRAINT_ROUTINE_USAGE AS
  SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
         SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
  FROM DEFINITION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE CHECK_CONSTRAINT_ROUTINE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.

5.13 CHECK_CONSTRAINTS view

Function

Identify the check constraints defined in this catalog that are owned by a given user or role.

Definition

```
CREATE VIEW CHECK_CONSTRAINTS AS
  SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
         CHECK_CLAUSE
  FROM DEFINITION_SCHEMA.CHECK_CONSTRAINTS
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND CONSTRAINT_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE CHECK_CONSTRAINTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.14 COLLATIONS view

Function

Identify the character collations defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW COLLATIONS AS
  SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
         PAD_ATTRIBUTE, COLLATION_TYPE, COLLATION_DEFINITION,
         COLLATION_DICTIONARY
  FROM DEFINITION_SCHEMA.COLLATIONS
 WHERE ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
        'COLLATION' ) IN
        ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME,
              UP.OBJECT_TYPE
          FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
          WHERE ( UP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  UP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
        AND COLLATION_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE COLLATIONS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS.

5.15 COLLATION_CHARACTER_SET_APPLICABILITY view

Function

Identify the character sets to which each collation is applicable.

Definition

```
CREATE VIEW COLLATION_CHARACTER_SET_APPLICABILITY AS
  SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
         CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME
  FROM DEFINITION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( COLLATION_CATALOG, COLLATION_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND COLLATION_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE COLLATION_CHARACTER_SET_APPLICABILITY
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.
- 2) Without Feature F690, “Collation support ”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.

5.16 COLUMN_COLUMN_USAGE view

Function

Identify each case where a generated column depends on a base column in a base table owned by a given user or role.

Definition

```
CREATE VIEW COLUMN_COLUMN_USAGE AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         COLUMN_NAME, DEPENDENT_COLUMN
  FROM DEFINITION_SCHEMA.COLUMN_COLUMN_USAGE C
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( C.TABLE_CATALOG, C.TABLE_SCHEMA )
    = ( S.CATALOG_NAME, S.SCHEMA_NAME )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    C.TABLE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE COLUMN_COLUMN_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.
- 2) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.
- 3) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.

5.17 COLUMN_DOMAIN_USAGE view

Function

Identify the columns defined that are dependent on a domain defined in this catalog and owned by a user or role.

Definition

```
CREATE VIEW COLUMN_DOMAIN_USAGE AS
  SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
  FROM ( DEFINITION_SCHEMA.COLUMNS C
        JOIN
          ( DEFINITION_SCHEMA.DOMAINS D
        JOIN
          DEFINITION_SCHEMA.SCHEMATA S
          ON ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA )
              = ( S.CATALOG_NAME, S.SCHEMA_NAME ) ) )
        USING ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    DOMAIN_NAME IS NOT NULL
  AND
    DOMAIN_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE COLUMN_DOMAIN_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE.

5.18 COLUMN_PRIVILEGES view

Function

Identify the privileges on columns of tables defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW COLUMN_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
 WHERE ( GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR
        = CURRENT_USER
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND
    TABLE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE COLUMN_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_PRIVILEGES.

5.19 COLUMN_UDT_USAGE view

Function

Identify the columns defined that are dependent on a user-defined type defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW COLUMN_UDT_USAGE AS
  SELECT D.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         D.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         D.USER_DEFINED_TYPE_NAME AS UDT_NAME,
         C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME
  FROM ( DEFINITION_SCHEMA.COLUMNS AS C
        JOIN
          ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
        JOIN
          DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( D.USER_DEFINED_TYPE_CATALOG, D.USER_DEFINED_TYPE_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
        ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
              'TABLE', C.DTD_IDENTIFIER )
          = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
              D.OBJECT_TYPE, D.DTD_IDENTIFIER ) ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    D.DATA_TYPE = 'USER-DEFINED' ;

GRANT SELECT ON TABLE COLUMN_UDT_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_UDT_USAGE.

5.20 COLUMNS view

Function

Identify the columns of tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW COLUMNS AS
SELECT DISTINCT
    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    C.COLUMN_NAME, ORDINAL_POSITION, COLUMN_DEFAULT, IS_NULLABLE,
    COALESCE (D1.DATA_TYPE, D2.DATA_TYPE) AS DATA_TYPE,
    COALESCE (D1.CHARACTER_MAXIMUM_LENGTH, D2.CHARACTER_MAXIMUM_LENGTH)
    AS CHARACTER_MAXIMUM_LENGTH,
    COALESCE (D1.CHARACTER_OCTET_LENGTH, D2.CHARACTER_OCTET_LENGTH)
    AS CHARACTER_OCTET_LENGTH,
    COALESCE (D1.NUMERIC_PRECISION, D2.NUMERIC_PRECISION)
    AS NUMERIC_PRECISION,
    COALESCE (D1.NUMERIC_PRECISION_RADIX, D2.NUMERIC_PRECISION_RADIX)
    AS NUMERIC_PRECISION_RADIX,
    COALESCE (D1.NUMERIC_SCALE, D2.NUMERIC_SCALE)
    AS NUMERIC_SCALE,
    COALESCE (D1.DATETIME_PRECISION, D2.DATETIME_PRECISION)
    AS DATETIME_PRECISION,
    COALESCE (D1.INTERVAL_TYPE, D2.INTERVAL_TYPE)
    AS INTERVAL_TYPE,
    COALESCE (D1.INTERVAL_PRECISION, D2.INTERVAL_PRECISION)
    AS INTERVAL_PRECISION,
    COALESCE (D1.CHARACTER_SET_CATALOG, D2.CHARACTER_SET_CATALOG)
    AS CHARACTER_SET_CATALOG,
    COALESCE (D1.CHARACTER_SET_SCHEMA, D2.CHARACTER_SET_SCHEMA)
    AS CHARACTER_SET_SCHEMA,
    COALESCE (D1.CHARACTER_SET_NAME, D2.CHARACTER_SET_NAME)
    AS CHARACTER_SET_NAME,
    COALESCE (D1.COLLATION_CATALOG, D2.COLLATION_CATALOG)
    AS COLLATION_CATALOG,
    COALESCE (D1.COLLATION_SCHEMA, D2.COLLATION_SCHEMA)
    AS COLLATION_SCHEMA,
    COALESCE (D1.COLLATION_NAME, D2.COLLATION_NAME)
    AS COLLATION_NAME,
    DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
    COALESCE (D1.USER_DEFINED_TYPE_CATALOG, D2.USER_DEFINED_TYPE_CATALOG)
    AS UDT_CATALOG,
    COALESCE (D1.USER_DEFINED_TYPE_SCHEMA, D2.USER_DEFINED_TYPE_SCHEMA)
    AS UDT_SCHEMA,
    COALESCE (D1.USER_DEFINED_TYPE_NAME, D2.USER_DEFINED_TYPE_NAME)
    AS UDT_NAME,
    COALESCE (D1.SCOPE_CATALOG, D2.SCOPE_CATALOG) AS SCOPE_CATALOG,
    COALESCE (D1.SCOPE_SCHEMA, D2.SCOPE_SCHEMA) AS SCOPE_SCHEMA,
    COALESCE (D1.SCOPE_NAME, D2.SCOPE_NAME) AS SCOPE_NAME,
    COALESCE (D1.MAXIMUM_CARDINALITY, D2.MAXIMUM_CARDINALITY)
```

ISO/IEC 9075-11:2003 (E)
5.20 COLUMNS view

```
        AS MAXIMUM_CARDINALITY,
    COALESCE (D1.DTD_IDENTIFIER, D2.DTD_IDENTIFIER)
        AS DTD_IDENTIFIER,
    IS_SELF_REFERENCING, IS_IDENTITY, IDENTITY_GENERATION,
    IDENTITY_START, IDENTITY_INCREMENT,
    IDENTITY_MAXIMUM, IDENTITY_MINIMUM, IDENTITY_CYCLE,
    IS_GENERATED, GENERATION_EXPRESSION, IS_UPDATABLE
FROM ( ( DEFINITION_SCHEMA.COLUMNS AS C
    LEFT JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
    ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
        'TABLE', C.DTD_IDENTIFIER )
        = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
        D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) ) )
    LEFT JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
    ON ( ( C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME,
        'DOMAIN', C.DTD_IDENTIFIER )
        = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
        D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME ) IN
    ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME, CP.COLUMN_NAME
    FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
    WHERE ( CP.GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        CP.GRANTEE IN
        ( SELECT ROLE_NAME
        FROM ENABLED_ROLES ) ) )
AND
    C.TABLE_CATALOG
    = ( SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE COLUMNS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.
- 2) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.IS_GENERATED.
- 3) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.GENERATION_EXPRESSION.
- 4) Without Feature T111, “Updatable joins, unions, and columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.IS_UPDATABLE.

5.21 CONSTRAINT_COLUMN_USAGE view

Function

Identify the columns used by referential constraints, unique constraints, check constraints, and assertions defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW CONSTRAINT_COLUMN_USAGE AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
         CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
  FROM ( ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.CHECK_COLUMN_USAGE )
        UNION
        ( SELECT PK.TABLE_CATALOG, PK.TABLE_SCHEMA, PK.TABLE_NAME, PK.COLUMN_NAME,
                 FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA, FK.CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS FK
           JOIN
             DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS PK
           ON
             ( FK.UNIQUE_CONSTRAINT_CATALOG, FK.UNIQUE_CONSTRAINT_SCHEMA,
               FK.UNIQUE_CONSTRAINT_NAME )
             = ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                 PK.CONSTRAINT_NAME ) ) )
        UNION
        ( SELECT KC.TABLE_CATALOG, KC.TABLE_SCHEMA, KC.TABLE_NAME, KC.COLUMN_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KC
           JOIN
             DEFINITION_SCHEMA.TABLE_CONSTRAINTS
           USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
           WHERE CONSTRAINT_TYPE IN
             ( 'UNIQUE', 'PRIMARY KEY' ) ) )
  JOIN
    DEFINITION_SCHEMA.SCHEMATA
  ON
    ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( CATALOG_NAME, SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND CONSTRAINT_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE CONSTRAINT_COLUMN_USAGE
```

TO PUBLIC WITH GRANT OPTION;

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE.

5.22 CONSTRAINT_TABLE_USAGE view

Function

Identify the tables that are used by referential constraints, unique constraints, check constraints, and assertions defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW CONSTRAINT_TABLE_USAGE AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
  FROM ( ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.CHECK_TABLE_USAGE )
        UNION
        ( SELECT PK.TABLE_CATALOG, PK.TABLE_SCHEMA, PK.TABLE_NAME,
                 FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA, FK.CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS FK
           JOIN
             DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS PK
           ON ( FK.UNIQUE_CONSTRAINT_CATALOG, FK.UNIQUE_CONSTRAINT_SCHEMA,
                FK.UNIQUE_CONSTRAINT_NAME )
              = ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                 PK.CONSTRAINT_NAME ) ) )
        UNION
        ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.TABLE_CONSTRAINTS
           WHERE CONSTRAINT_TYPE IN
             ( 'UNIQUE', 'PRIMARY KEY' ) ) )
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND CONSTRAINT_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE CONSTRAINT_TABLE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.

5.23 DATA_TYPE_PRIVILEGES view

Function

Identify those schema objects whose included data type descriptors are accessible to a given user or role.

Definition

```
CREATE VIEW DATA_TYPE_PRIVILEGES
( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
  OBJECT_TYPE, DTD_IDENTIFIER ) AS
  SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         'USER-DEFINED TYPE', DTD_IDENTIFIER
    FROM ATTRIBUTES
  UNION
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         'TABLE', DTD_IDENTIFIER
    FROM COLUMNS
  UNION
  SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
         'DOMAIN', DTD_IDENTIFIER
    FROM DOMAINS
  UNION
  SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         'USER-DEFINED TYPE', DTD_IDENTIFIER
    FROM METHOD_SPECIFICATIONS
  UNION
  SELECT PARAMETER_UDT_CATALOG, PARAMETER_UDT_SCHEMA, PARAMETER_UDT_NAME,
         'USER-DEFINED TYPE', DTD_IDENTIFIER
    FROM METHOD_SPECIFICATION_PARAMETERS
  UNION
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         'ROUTINE', DTD_IDENTIFIER
    FROM PARAMETERS
  UNION
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         'ROUTINE', DTD_IDENTIFIER
    FROM ROUTINES
   WHERE DTD_IDENTIFIER IS NOT NULL
  UNION
  SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', SOURCE_DTD_IDENTIFIER
    FROM USER_DEFINED_TYPES
   WHERE SOURCE_DTD_IDENTIFIER IS NOT NULL
  UNION
  SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', REF_DTD_IDENTIFIER
    FROM USER_DEFINED_TYPES
   WHERE REF_DTD_IDENTIFIER IS NOT NULL;

GRANT SELECT ON TABLE DATA_TYPE_PRIVILEGES
```

TO PUBLIC WITH GRANT OPTION;

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES.

5.24 DIRECT_SUPERTABLES view

Function

Identify the direct supertables related to a table that are defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW DIRECT_SUPERTABLES AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, SUPERTABLE_NAME
  FROM DEFINITION_SCHEMA.DIRECT_SUPERTABLES
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    TABLE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE DIRECT_SUPERTABLES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S081, “Subtables”, conforming SQL language shall not reference INFORMATION_SCHEMA.DIRECT_SUPERTABLES.

5.25 DIRECT_SUPERTYPES view

Function

Identify the direct supertypes related to a user-defined type that are defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW DIRECT_SUPERTYPES AS
  SELECT USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         USER_DEFINED_TYPE_NAME AS UDT_NAME,
         SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
  FROM DEFINITION_SCHEMA.DIRECT_SUPERTYPES
 WHERE ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME ) IN
        ( ( SELECT UDTF.USER_DEFINED_TYPE_CATALOG, UDTF.USER_DEFINED_TYPE_SCHEMA,
                  UDTF.USER_DEFINED_TYPE_NAME
            FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTF
          JOIN
            DEFINITION_SCHEMA.SCHEMATA AS S
          ON ( ( UDTF.USER_DEFINED_TYPE_CATALOG,
                  UDTF.USER_DEFINED_TYPE_SCHEMA )
              = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
          WHERE ( S.SCHEMA_OWNER = CURRENT_USER
                OR
                  S.SCHEMA_OWNER IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) ) )
  AND
    USER_DEFINED_TYPE_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE DIRECT_SUPERTYPES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.DIRECT_SUPERTYPES.

5.26 DOMAIN_CONSTRAINTS view

Function

Identify the domain constraints of domains in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW DOMAIN_CONSTRAINTS AS
  SELECT DISTINCT
    CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
    DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
    IS_DEFERRABLE, INITIALLY_DEFERRED
  FROM DEFINITION_SCHEMA.DOMAIN_CONSTRAINTS
  WHERE ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN' ) IN
    ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE
      FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
      WHERE ( UP.GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR UP.GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) ) )
    AND CONSTRAINT_CATALOG
      = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE DOMAIN_CONSTRAINTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAIN_CONSTRAINTS.

5.27 DOMAINS view

Function

Identify the domains defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW DOMAINS AS
  SELECT DISTINCT
    DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
    DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION, DOMAIN_DEFAULT,
    MAXIMUM_CARDINALITY, D1.DTD_IDENTIFIER
  FROM DEFINITION_SCHEMA.DOMAINS AS D1
  JOIN
    DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
  ON ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
        'DOMAIN', D1.DTD_IDENTIFIER )
      = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
  WHERE ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN' ) IN
    ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE
      FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
      WHERE ( UP.GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        UP.GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) ) )
    OR
    ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IN
    ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
      FROM COLUMNS ) )
  AND
    DOMAIN_CATALOG
  = ( SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE DOMAINS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.

- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.

5.28 ELEMENT_TYPES view

Function

Identify the collection element types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW ELEMENT_TYPES AS
  SELECT DISTINCT
    OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER, DATA_TYPE,
    CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
    USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
    USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
    USER_DEFINED_TYPE_NAME AS UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAXIMUM_CARDINALITY, DTD_IDENTIFIER
  FROM DEFINITION_SCHEMA.ELEMENT_TYPES AS E
  JOIN
    DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
  USING ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER )
  WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER ) IN
    ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER
      FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );

GRANT SELECT ON TABLE ELEMENT_TYPES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S091, “Basic array support”, or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.

5.29 ENABLED_ROLES view

Function

Identify the enabled roles for the current SQL-session.

Definition

```
CREATE RECURSIVE VIEW ENABLED_ROLES ( ROLE_NAME ) AS
  VALUES ( CURRENT_ROLE )
UNION
  SELECT RAD.ROLE_NAME
  FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTOR RAD
  JOIN
    ENABLED_ROLES R
  ON RAD.GRANTEE = R.ROLE_NAME;

GRANT SELECT ON TABLE ENABLED_ROLES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ENABLED_ROLES.

5.30 FIELDS view

Function

Identify the field types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW FIELDS AS
  SELECT DISTINCT
    OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
    ORDINAL_POSITION, DATA_TYPE,
    CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
    USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
    USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
    USER_DEFINED_TYPE_NAME AS UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAXIMUM_CARDINALITY, DTD_IDENTIFIER
  FROM DEFINITION_SCHEMA.FIELDS AS F
  JOIN
    DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
  USING ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER )
  WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, ROOT_DTD_IDENTIFIER ) IN
    ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER
      FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );

GRANT SELECT ON TABLE FIELDS
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature T051, “Row types”, conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.

5.31 KEY_COLUMN_USAGE view

Function

Identify the columns defined in this catalog that are constrained as keys and that are accessible by a given user or role.

Definition

```
CREATE VIEW KEY_COLUMN_USAGE AS
  SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
         KCU1.TABLE_CATALOG, KCU1.TABLE_SCHEMA, KCU1.TABLE_NAME,
         KCU1.COLUMN_NAME, KCU1.ORDINAL_POSITION, KCU1.POSITION_IN_UNIQUE_CONSTRAINT
  FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU1
  JOIN
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC
  USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
 WHERE ( ( SELECT MAX ( KCU3.ORDINAL_POSITION )
          FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU3
          WHERE KCU3.CONSTRAINT_CATALOG = CONSTRAINT_CATALOG
            AND
            KCU3.CONSTRAINT_SCHEMA = CONSTRAINT_SCHEMA
            AND
            KCU3.CONSTRAINT_NAME = CONSTRAINT_NAME
        )
    = ( SELECT COUNT ( * )
        FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU2
        WHERE ( KCU2.TABLE_CATALOG, KCU2.TABLE_SCHEMA,
              KCU2.TABLE_NAME, KCU2.COLUMN_NAME )
            IN ( SELECT CP2.TABLE_CATALOG, CP2.TABLE_SCHEMA,
                  CP2.TABLE_NAME, CP2.COLUMN_NAME
                FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP2
                WHERE ( CP2.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                    OR
                    CP2.GRANTEE IN
                      ( SELECT ROLE_NAME
                        FROM ENABLED_ROLES )
                    )
                )
        )
    )
  AND
    KCU2.CONSTRAINT_CATALOG = CONSTRAINT_CATALOG
  AND
    KCU2.CONSTRAINT_SCHEMA = CONSTRAINT_SCHEMA
  AND
    KCU2.CONSTRAINT_NAME = CONSTRAINT_NAME
  )
  AND
    CONSTRAINT_CATALOG
  = ( SELECT CATALOG_NAME
```

```
FROM INFORMATION_SCHEMA_CATALOG_NAME );  
  
GRANT SELECT ON TABLE KEY_COLUMN_USAGE  
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE.

5.32 METHOD_SPECIFICATION_PARAMETERS view

Function

Identify the SQL parameters of method specifications described in the METHOD_SPECIFICATIONS view that are accessible to a given user or role.

Definition

```
CREATE VIEW METHOD_SPECIFICATION_PARAMETERS AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         P.ORDINAL_POSITION, P.PARAMETER_MODE, P.IS_RESULT,
         P.AS_LOCATOR, P.PARAMETER_NAME,
         P.FROM_SQL_SPECIFIC_CATALOG, P.FROM_SQL_SPECIFIC_SCHEMA,
         P.FROM_SQL_SPECIFIC_NAME, D.DATA_TYPE,
         D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
         D.CHARACTER_SET_CATALOG, D.CHARACTER_SET_SCHEMA, D.CHARACTER_SET_NAME,
         D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
         D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
         D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
         D.USER_DEFINED_TYPE_CATALOG AS PARAMETER_UDT_CATALOG,
         D.USER_DEFINED_TYPE_SCHEMA AS PARAMETER_UDT_SCHEMA,
         D.USER_DEFINED_TYPE_NAME AS PARAMETER_UDT_NAME,
         D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
         D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER
  FROM ( DEFINITION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS AS P
        JOIN
          DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
        ON
          ( P.SPECIFIC_CATALOG, P.SPECIFIC_SCHEMA, P.SPECIFIC_NAME,
            'USER-DEFINED TYPE', P.DTD_IDENTIFIER )
          = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
            D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
        JOIN
          DEFINITION_SCHEMA.METHOD_SPECIFICATIONS AS M
        USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
  WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
         M.USER_DEFINED_TYPE_NAME ) IN
        ( SELECT UDTF.USER_DEFINED_TYPE_CATALOG, UDTF.USER_DEFINED_TYPE_SCHEMA,
              UDTF.USER_DEFINED_TYPE_NAME
          FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTF
          WHERE ( UDTF.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  UDTF.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
        AND
          M.USER_DEFINED_TYPE_CATALOG
          = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME );
```

```
GRANT SELECT ON TABLE METHOD_SPECIFICATION_PARAMETERS  
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.

5.33 METHOD_SPECIFICATIONS view

Function

Identify the SQL-invoked methods in the catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW METHOD_SPECIFICATIONS AS
  SELECT M.SPECIFIC_CATALOG, M.SPECIFIC_SCHEMA, M.SPECIFIC_NAME,
         M.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         M.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         M.USER_DEFINED_TYPE_NAME AS UDT_NAME,
         M.METHOD_NAME, IS_STATIC, IS_OVERRIDING, IS_CONSTRUCTOR,
         D.DATA_TYPE, D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
         D.CHARACTER_SET_CATALOG, D.CHARACTER_SET_SCHEMA, D.CHARACTER_SET_NAME,
         D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
         D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
         D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
         D.USER_DEFINED_TYPE_CATALOG AS RETURN_UDT_CATALOG,
         D.USER_DEFINED_TYPE_SCHEMA AS RETURN_UDT_SCHEMA,
         D.USER_DEFINED_TYPE_NAME AS RETURN_UDT_NAME,
         D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
         D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER, M.METHOD_LANGUAGE,
         M.PARAMETER_STYLE, M.IS_DETERMINISTIC, M.SQL_DATA_ACCESS,
         M.IS_NULL_CALL,
         M.TO_SQL_SPECIFIC_CATALOG, M.TO_SQL_SPECIFIC_SCHEMA,
         M.TO_SQL_SPECIFIC_NAME,
         M.CREATED, M.LAST_ALTERED,
         DT.DATA_TYPE AS RESULT_CAST_FROM_DATA_TYPE,
         RESULT_CAST_AS_LOCATOR,
         DT.CHARACTER_MAXIMUM_LENGTH AS RESULT_CAST_CHAR_MAX_LENGTH,
         DT.CHARACTER_OCTET_LENGTH AS RESULT_CAST_CHAR_OCTET_LENGTH,
         DT.CHARACTER_SET_CATALOG AS RESULT_CAST_CHAR_SET_CATALOG,
         DT.CHARACTER_SET_SCHEMA AS RESULT_CAST_CHAR_SET_SCHEMA,
         DT.CHARACTER_SET_NAME AS RESULT_CAST_CHAR_SET_NAME,
         DT.COLLATION_CATALOG AS RESULT_CAST_COLLATION_CATALOG,
         DT.COLLATION_SCHEMA AS RESULT_CAST_COLLATION_SCHEMA,
         DT.COLLATION_NAME AS RESULT_CAST_COLLATION_NAME,
         DT.NUMERIC_PRECISION AS RESULT_CAST_NUMERIC_PRECISION,
         DT.NUMERIC_PRECISION_RADIX AS RESULT_CAST_NUMERIC_RADIX,
         DT.NUMERIC_SCALE AS RESULT_CAST_NUMERIC_SCALE,
         DT.DATETIME_PRECISION AS RESULT_CAST_DATETIME_PRECISION,
         DT.INTERVAL_TYPE AS RESULT_CAST_INTERVAL_TYPE,
         DT.INTERVAL_PRECISION AS RESULT_CAST_INTERVAL_PRECISION,
         DT.USER_DEFINED_TYPE_CATALOG AS RESULT_CAST_TYPE_UDT_CATALOG,
         DT.USER_DEFINED_TYPE_SCHEMA AS RESULT_CAST_TYPE_UDT_SCHEMA,
         DT.USER_DEFINED_TYPE_NAME AS RESULT_CAST_TYPE_UDT_NAME,
         DT.SCOPE_CATALOG AS RESULT_CAST_SCOPE_CATALOG,
         DT.SCOPE_SCHEMA AS RESULT_CAST_SCOPE_SCHEMA,
         DT.SCOPE_NAME AS RESULT_CAST_SCOPE_NAME,
         DT.MAXIMUM_CARDINALITY AS RESULT_CAST_MAX_CARDINALITY,
```

```

        DT.DTD_IDENTIFIER AS RESULT_CAST_DTD_IDENTIFIER
FROM ( DEFINITION_SCHEMA.METHOD_SPECIFICATIONS AS M
      JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
      ON ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
          M.USER_DEFINED_TYPE_NAME,
          'USER-DEFINED TYPE', M.DTD_IDENTIFIER )
      = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA,
          D.OBJECT_NAME,
          D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
LEFT JOIN
  DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DT
ON ( M.SPECIFIC_CATALOG, M.SPECIFIC_SCHEMA, M.SPECIFIC_NAME,
    'USER-DEFINED TYPE', RESULT_CAST_FROM_DTD_IDENTIFIER )
= ( DT.OBJECT_CATALOG, DT.OBJECT_SCHEMA, DT.OBJECT_NAME,
    DT.OBJECT_TYPE, DT.DTD_IDENTIFIER )
WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
        M.USER_DEFINED_TYPE_NAME ) IN
      ( SELECT UDTF.USER_DEFINED_TYPE_CATALOG, UDTF.USER_DEFINED_TYPE_SCHEMA,
          UDTF.USER_DEFINED_TYPE_NAME
        FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTF
        WHERE ( UDTF.GRANTEE IN
                ( 'PUBLIC', CURRENT_USER )
              OR
                UDTF.GRANTEE IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) )
      AND
        M.USER_DEFINED_TYPE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

```

```

GRANT SELECT ON TABLE METHOD_SPECIFICATIONS
  TO PUBLIC WITH GRANT OPTION;

```

NOTE 4 — The METHOD_SPECIFICATIONS view contains two sets of columns that each describe a data type. While the set of columns that are prefixed with “RESULT_CAST_” describes the data type specified in the <result cast>, if any, contained in the <method specification>, the other set of columns describes the data type specified in the <returns data type> contained in the <method specification>.

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.
- 2) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.
- 3) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.CREATED.
- 4) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.LAST_ALTERED.

5.34 PARAMETERS view

Function

Identify the SQL parameters of SQL-invoked routines defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW PARAMETERS AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         P1.ORDINAL_POSITION, PARAMETER_MODE,
         P1.IS_RESULT, P1.AS_LOCATOR, PARAMETER_NAME,
         FROM_SQL_SPECIFIC_CATALOG, FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME,
         TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME,
         DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         D1.CHARACTER_SET_CATALOG, D1.CHARACTER_SET_SCHEMA, D1.CHARACTER_SET_NAME,
         D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
         NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
         DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
         D1.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         D1.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         D1.USER_DEFINED_TYPE_NAME AS UDT_NAME,
         D1.SCOPE_CATALOG, D1.SCOPE_SCHEMA, D1.SCOPE_NAME,
         D1.MAXIMUM_CARDINALITY, D1.DTD_IDENTIFIER
  FROM ( DEFINITION_SCHEMA.PARAMETERS P1
        LEFT JOIN
          DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR D1
            ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', P1.DTD_IDENTIFIER )
              = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, D1.DTD_IDENTIFIER ) )
        JOIN
          DEFINITION_SCHEMA.ROUTINES R1
        USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
  WHERE ( ( ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IS NULL
            AND
              ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
                ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
                  FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
                  WHERE ( GRANTEE IN
                        ( 'PUBLIC', CURRENT_USER )
                      OR
                        GRANTEE IN
                          ( SELECT ROLE_NAME
                            FROM ENABLED_ROLES ) ) ) ) )
        AND SPECIFIC_CATALOG
          = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE PARAMETERS
```

TO PUBLIC WITH GRANT OPTION;

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.PARAMETERS.

5.35 REFERENCED_TYPES view

Function

Identify the referenced types of reference types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW REFERENCED_TYPES AS
  SELECT DISTINCT
    OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
    CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
    USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
    USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
    USER_DEFINED_TYPE_NAME AS UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAXIMUM_CARDINALITY, DTD_IDENTIFIER
  FROM ( DEFINITION_SCHEMA.REFERENCED_TYPES AS R
        JOIN
          DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
        USING ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, DTD_IDENTIFIER ) )
  WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, ROOT_DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, DTD_IDENTIFIER
          FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );

GRANT SELECT ON TABLE REFERENCED_TYPES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S041, “Basic reference types”, conforming SQL language shall not reference INFORMATION_SCHEMA.REFERENCED_TYPES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.REFERENCED_TYPES.

5.36 REFERENTIAL_CONSTRAINTS view

Function

Identify the referential constraints defined on tables in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW REFERENTIAL_CONSTRAINTS AS
  SELECT DISTINCT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
    TC2.CONSTRAINT_CATALOG AS UNIQUE_CONSTRAINT_CATALOG,
    TC2.CONSTRAINT_SCHEMA AS UNIQUE_CONSTRAINT_SCHEMA,
    TC2.CONSTRAINT_NAME AS UNIQUE_CONSTRAINT_NAME,
    RC.MATCH_OPTION, RC.UPDATE_RULE, RC.DELETE_RULE
  FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
  JOIN
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC1
  USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
  LEFT JOIN
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC2
  ON ( ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
    RC.UNIQUE_CONSTRAINT_NAME )
    = ( TC2.CONSTRAINT_CATALOG, TC2.CONSTRAINT_SCHEMA, TC2.CONSTRAINT_NAME ) )
  WHERE CONSTRAINT_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE REFERENTIAL_CONSTRAINTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS.

5.37 ROLE_COLUMN_GRANTS view

Function

Identifies the privileges on columns defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_COLUMN_GRANTS AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         COLUMN_NAME, PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
 WHERE ( GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND TABLE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_COLUMN_GRANTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, and Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_COLUMN_GRANTS.

5.38 ROLE_ROUTINE_GRANTS view

Function

Identify the privileges on SQL-invoked routines defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_ROUTINE_GRANTS AS
  SELECT GRANTOR, GRANTEE,
         SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
  JOIN
    DEFINITION_SCHEMA.ROUTINES
  USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
 WHERE ( GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
  AND
    SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_ROUTINE_GRANTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS.
- 3) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS.

5.39 ROLE_TABLE_GRANTS view

Function

Identifies the privileges on tables defined in this catalog that are available to or granted by the currently applicable roles.

Definition

```
CREATE VIEW ROLE_TABLE_GRANTS AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE, WITH_HIERARCHY
  FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES
 WHERE ( GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
       OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND TABLE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_TABLE_GRANTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_GRANTS.
- 2) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_GRANTS.

5.40 ROLE_TABLE_METHOD_GRANTS view

Function

Identify the privileges on methods of tables of structured types defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_TABLE_METHOD_GRANTS AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
         TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
         SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.TABLE_METHOD_PRIVILEGES
 WHERE ( GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
       OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND
    TABLE_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_TABLE_METHOD_GRANTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.
- 2) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.
- 3) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.

5.41 ROLE_USAGE_GRANTS view

Function

Identify the USAGE privileges on objects defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_USAGE_GRANTS AS
  SELECT GRANTOR, GRANTEE,
         OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE,
         'USAGE' AS PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
 WHERE ( GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND
    OBJECT_CATALOG
  = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_USAGE_GRANTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_USAGE_GRANTS.
- 2) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_USAGE_GRANTS.

5.42 ROLE_UDT_GRANTS view

Function

Identify the privileges on user-defined types defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_UDT_GRANTS AS
  SELECT GRANTOR, GRANTEE,
         USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         USER_DEFINED_TYPE_NAME AS UDT_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
 WHERE ( GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
       OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
  AND
    USER_DEFINED_TYPE_CATALOG
  = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_UDT_GRANTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_UDT_GRANTS.
- 2) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_UDT_GRANTS.

5.43 ROUTINE_COLUMN_USAGE view

Function

Identify the columns owned by a given user or role on which SQL routines defined in this catalog are dependent.

Definition

```
CREATE VIEW ROUTINE_COLUMN_USAGE AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, ROUTINE_CATALOG,
         ROUTINE_SCHEMA, ROUTINE_NAME, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         COLUMN_NAME
  FROM ( DEFINITION_SCHEMA.ROUTINE_COLUMN_USAGE
        JOIN
          DEFINITION_SCHEMA.ROUTINES
        USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) )
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
 WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    ROUTINE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROUTINE_COLUMN_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COLUMN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COLUMN_USAGE.

5.44 ROUTINE_PRIVILEGES view

Function

Identify the privileges on SQL-invoked routines defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW ROUTINE_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE, SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
  JOIN
    DEFINITION_SCHEMA.ROUTINES
  USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
 WHERE ( GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR
        = CURRENT_USER
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        AND
        ROUTINE_CATALOG
        = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME ) );

GRANT SELECT ON TABLE ROUTINE_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_PRIVILEGES.

5.45 ROUTINE_ROUTINE_USAGE view

Function

Identify each SQL-invoked routine owned by a given user or role on which an SQL routine defined in this catalog is dependent.

Definition

```
CREATE VIEW ROUTINE_ROUTINE_USAGE AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         RRU.ROUTINE_CATALOG, RRU.ROUTINE_SCHEMA, RRU.ROUTINE_NAME
  FROM DEFINITION_SCHEMA.ROUTINE_ROUTINE_USAGE RRU
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( RRU.ROUTINE_CATALOG, RRU.ROUTINE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROUTINE_ROUTINE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_ROUTINE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_ROUTINE_USAGE.

5.46 ROUTINE_SEQUENCE_USAGE view

Function

Identify each external sequence generator owned by a given user or role on which some SQL routine defined in this catalog is dependent.

Definition

```
CREATE VIEW ROUTINE_SEQUENCE_USAGE AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
  FROM DEFINITION_SCHEMA.ROUTINE_SEQUENCE_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    SEQUENCE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROUTINE_SEQUENCE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.
- 3) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.

5.47 ROUTINE_TABLE_USAGE view

Function

Identify the tables owned by a given user or role on which SQL routines defined in this catalog are dependent.

Definition

```
CREATE VIEW ROUTINE_TABLE_USAGE AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
  FROM ( DEFINITION_SCHEMA.ROUTINE_TABLE_USAGE
        JOIN
          DEFINITION_SCHEMA.ROUTINES
        USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) )
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
 WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROUTINE_TABLE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_TABLE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_TABLE_USAGE.

5.48 ROUTINES view

Function

Identify the SQL-invoked routines in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW ROUTINES AS
  SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_TYPE,
         MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
         R.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         R.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         R.USER_DEFINED_TYPE_NAME AS UDT_NAME,
         D.DATA_TYPE, D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
         D.CHARACTER_SET_CATALOG, D.CHARACTER_SET_SCHEMA, D.CHARACTER_SET_NAME,
         D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
         D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
         D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
         D.USER_DEFINED_TYPE_CATALOG AS TYPE_UDT_CATALOG,
         D.USER_DEFINED_TYPE_SCHEMA AS TYPE_UDT_SCHEMA,
         D.USER_DEFINED_TYPE_NAME AS TYPE_UDT_NAME,
         D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
         D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER, ROUTINE_BODY,
         CASE
           WHEN EXISTS
             ( SELECT *
               FROM DEFINITION_SCHEMA.SCHEMATA AS S
               WHERE ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                 AND
                     ( SCHEMA_OWNER = CURRENT_USER
                     OR
                       SCHEMA_OWNER IN
                         ( SELECT ROLE_NAME
                           FROM ENABLED_ROLES ) ) )
             THEN ROUTINE_DEFINITION
           ELSE NULL
         END AS ROUTINE_DEFINITION,
         EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE,
         IS_DETERMINISTIC, SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
         SCHEMA_LEVEL_ROUTINE, MAX_DYNAMIC_RESULT_SETS,
         IS_USER_DEFINED_CAST, IS_IMPLICITLY_INVOCABLE, SECURITY_TYPE,
         TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME,
         AS_LOCATOR, CREATED, LAST_ALTERED, NEW_SAVEPOINT_LEVEL, IS_UDT_DEPENDENT,
         DT.DATA_TYPE AS RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,
         DT.CHARACTER_MAXIMUM_LENGTH AS RESULT_CAST_CHAR_MAX_LENGTH,
         DT.CHARACTER_OCTET_LENGTH AS RESULT_CAST_CHAR_OCTET_LENGTH,
         DT.CHARACTER_SET_CATALOG AS RESULT_CAST_CHAR_SET_CATALOG,
         DT.CHARACTER_SET_SCHEMA AS RESULT_CAST_CHAR_SET_SCHEMA,
         DT.CHARACTER_SET_NAME AS RESULT_CAST_CHARACTER_SET_NAME,
```

```

DT.COLLATION_CATALOG AS RESULT_CAST_COLLATION_CATALOG,
DT.COLLATION_SCHEMA AS RESULT_CAST_COLLATION_SCHEMA,
DT.COLLATION_NAME AS RESULT_CAST_COLLATION_NAME,
DT.NUMERIC_PRECISION AS RESULT_CAST_NUMERIC_PRECISION,
DT.NUMERIC_PRECISION_RADIX AS RESULT_CAST_NUMERIC_RADIX,
DT.NUMERIC_SCALE AS RESULT_CAST_NUMERIC_SCALE,
DT.DATETIME_PRECISION AS RESULT_CAST_DATETIME_PRECISION,
DT.INTERVAL_TYPE AS RESULT_CAST_INTERVAL_TYPE,
DT.INTERVAL_PRECISION AS RESULT_CAST_INTERVAL_PRECISION,
DT.USER_DEFINED_TYPE_CATALOG AS RESULT_CAST_TYPE_UDT_CATALOG,
DT.USER_DEFINED_TYPE_SCHEMA AS RESULT_CAST_TYPE_UDT_SCHEMA,
DT.USER_DEFINED_TYPE_NAME AS RESULT_CAST_TYPE_UDT_NAME,
DT.SCOPE_CATALOG AS RESULT_CAST_SCOPE_CATALOG,
DT.SCOPE_SCHEMA AS RESULT_CAST_SCOPE_SCHEMA,
DT.SCOPE_NAME AS RESULT_CAST_SCOPE_NAME,
DT.MAXIMUM_CARDINALITY AS RESULT_CAST_MAX_CARDINALITY,
DT.DTD_IDENTIFIER AS RESULT_CAST_DTD_IDENTIFIER
FROM ( ( DEFINITION_SCHEMA.ROUTINES AS R
LEFT JOIN
DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
'ROUTINE', R.DTD_IDENTIFIER )
= ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
LEFT JOIN
DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DT
ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
'ROUTINE', RESULT_CAST_FROM_DTD_IDENTIFIER )
= ( DT.OBJECT_CATALOG, DT.OBJECT_SCHEMA, DT.OBJECT_NAME,
DT.OBJECT_TYPE, DT.DTD_IDENTIFIER ) )
WHERE ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IS NULL
AND
( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
WHERE ( GRANTEE IN
( 'PUBLIC', CURRENT_USER )
OR
GRANTEE IN
( SELECT ROLE_NAME
FROM ENABLED_ROLES ) ) )
AND SPECIFIC_CATALOG
= ( SELECT CATALOG_NAME
FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE ROUTINES
TO PUBLIC WITH GRANT OPTION;

```

NOTE 5 — The ROUTINES view contains two sets of columns that each describe a data type. While the set of columns that are prefixed with “RESULT_CAST_” describes the data type specified in the <result cast>, if any, contained in the <SQL-invoked routine>, the other set of columns describes the data type specified in the <returns data type> contained in the <SQL-invoked routine>.

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.
- 2) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.CREATED.
- 3) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.LAST_ALTERED.
- 4) Without Feature T272, “Enhanced savepoint management”, conforming SQL-language shall not reference INFORMATION_SCHEMA.ROUTINES.NEW_SAVEPOINT_LEVEL.

5.49 SCHEMATA view

Function

Identify the schemata in a catalog that are owned by a given user or role.

Definition

```
CREATE VIEW SCHEMATA AS
  SELECT CATALOG_NAME, SCHEMA_NAME, SCHEMA_OWNER,
         DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
         DEFAULT_CHARACTER_SET_NAME, SQL_PATH
  FROM DEFINITION_SCHEMA.SCHEMATA
 WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
        SCHEMA_OWNER IN
          ( SELECT ROLE_NAME
            FROM ENABLED_ROLES ) )
 AND
   CATALOG_NAME
   = ( SELECT CATALOG_NAME
       FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE SCHEMATA
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SCHEMATA.

5.50 SEQUENCES view

Function

Identify the external sequence generators defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW SEQUENCES AS
  SELECT S.SEQUENCE_CATALOG, S.SEQUENCE_SCHEMA, S.SEQUENCE_NAME,
         D.DATA_TYPE, D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX,
         D.NUMERIC_SCALE, S.MAXIMUM_VALUE, S.MINIMUM_VALUE,
         S.INCREMENT, S.CYCLE_OPTION
  FROM DEFINITION_SCHEMA.SEQUENCES AS S
  JOIN
    DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
  ON ( ( S.SEQUENCE_CATALOG, S.SEQUENCE_SCHEMA,
        S.SEQUENCE_NAME, 'SEQUENCE',
        S.DTD_IDENTIFIER )
      = ( OBJECT_CATALOG, OBJECT_SCHEMA,
        OBJECT_NAME, OBJECT_TYPE,
        D.DTD_IDENTIFIER ) )
  WHERE ( S.SEQUENCE_CATALOG, S.SEQUENCE_SCHEMA, S.SEQUENCE_NAME, 'SEQUENCE' ) IN
    ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME, UP.OBJECT_TYPE
      FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
      WHERE ( UP.GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        UP.GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) ) )
  AND S.SEQUENCE_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE SEQUENCES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES.

5.51 SQL_FEATURES view

Function

List the features and subfeatures of this standard, and indicate which of these the SQL-implementation supports.

Definition

```
CREATE VIEW SQL_FEATURES AS
  SELECT ID AS FEATURE_ID, NAME AS FEATURE_NAME, SUB_ID AS SUB_FEATURE_ID,
         SUB_NAME AS SUB_FEATURE_NAME, IS_SUPPORTED, IS_VERIFIED_BY, COMMENTS
  FROM DEFINITION_SCHEMA.SQL_CONFORMANCE
 WHERE TYPE IN
        ( 'FEATURE', 'SUBFEATURE' );

GRANT SELECT ON TABLE SQL_FEATURES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.52 SQL_IMPLEMENTATION_INFO view

Function

List the SQL-implementation information items defined in this standard and, for each of these, indicate the value supported by the SQL-implementation.

Definition

```
CREATE VIEW SQL_IMPLEMENTATION_INFO AS
  SELECT IMPLEMENTATION_INFO_ID, IMPLEMENTATION_INFO_NAME,
         INTEGER_VALUE, CHARACTER_VALUE, COMMENTS
  FROM DEFINITION_SCHEMA.SQL_IMPLEMENTATION_INFO;
```

```
GRANT SELECT ON TABLE SQL_IMPLEMENTATION_INFO
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO.

5.53 SQL_LANGUAGES view

Function

Identify the conformance levels, options, and dialects supported by the SQL-implementation processing data defined in this catalog.

NOTE 6 — The SQL_LANGUAGES view provides, among other information, the same information provided by the SQL object identifier specified in [Subclause 6.4](#), “Object identifier for Database Language SQL”, in ISO/IEC 9075-1.

Definition

```
CREATE VIEW SQL_LANGUAGES AS
  SELECT SQL_LANGUAGE_SOURCE, SQL_LANGUAGE_YEAR, SQL_LANGUAGE_CONFORMANCE,
         SQL_LANGUAGE_INTEGRITY, SQL_LANGUAGE_IMPLEMENTATION,
         SQL_LANGUAGE_BINDING_STYLE, SQL_LANGUAGE_PROGRAMMING_LANGUAGE
  FROM DEFINITION_SCHEMA.SQL_LANGUAGES;

GRANT SELECT ON TABLE SQL_LANGUAGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_LANGUAGES.

5.54 SQL_PACKAGES view

Function

List the packages of this standard, and indicate which of these the SQL-implementation supports.

Definition

```
CREATE VIEW SQL_PACKAGES AS
  SELECT ID AS FEATURE_ID, NAME AS FEATURE_NAME,
         IS_SUPPORTED, IS_VERIFIED_BY, COMMENTS
  FROM DEFINITION_SCHEMA.SQL_CONFORMANCE
 WHERE TYPE = 'PACKAGE';
```

```
GRANT SELECT ON TABLE SQL_PACKAGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_PACKAGES.

5.55 SQL_PARTS view

Function

List the parts of this standard, and indicate which of these the SQL-implementation supports.

Definition

```
CREATE VIEW SQL_PARTS AS
  SELECT ID AS FEATURE_ID, NAME AS FEATURE_NAME,
         IS_SUPPORTED, IS_VERIFIED_BY, COMMENTS
  FROM DEFINITION_SCHEMA.SQL_CONFORMANCE
  WHERE TYPE = 'PART';
```

```
GRANT SELECT ON TABLE SQL_PARTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_PARTS.

5.56 SQL_SIZING view

Function

List the sizing items defined in this standard and, for each of these, indicate the size supported by the SQL-implementation.

Definition

```
CREATE VIEW SQL_SIZING AS
  SELECT SIZING_ID, SIZING_NAME, SUPPORTED_VALUE, COMMENTS
  FROM DEFINITION_SCHEMA.SQL_SIZING;
```

```
GRANT SELECT ON TABLE SQL_SIZING
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.57 SQL_SIZING_PROFILES view

Function

List the sizing items defined in this standard and, for each of these, indicate the size required by one or more profiles of the standard.

Definition

```
CREATE VIEW SQL_SIZING_PROFILES AS
  SELECT SIZING_ID, SS.SIZING_NAME, SSP.PROFILE_ID,
         SSP.PROFILE_NAME, SSP.REQUIRED_VALUE, SSP.COMMENTS
  FROM DEFINITION_SCHEMA.SQL_SIZING_PROFILES AS SSP
  JOIN
    DEFINITION_SCHEMA.SQL_SIZING AS SS
  USING ( SIZING_ID );

GRANT SELECT ON TABLE SQL_SIZING_PROFILES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_SIZING_PROFILES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_SIZING_PROFILES.

5.58 TABLE_CONSTRAINTS view

Function

Identify the table constraints defined on tables in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TABLE_CONSTRAINTS AS
  SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         CONSTRAINT_TYPE, IS_DEFERRABLE, INITIALLY_DEFERRED
  FROM DEFINITION_SCHEMA.TABLE_CONSTRAINTS
 WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
        ( ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
            FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
            WHERE TP.PRIVILEGE_TYPE <> 'SELECT'
            AND
              ( TP.GRANTEE IN
                ( 'PUBLIC', CURRENT_USER )
              OR
                TP.GRANTEE IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) )
        UNION
        ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
          FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
          WHERE CP.PRIVILEGE_TYPE <> 'SELECT'
          AND ( CP.GRANTEE IN
                ( 'PUBLIC', CURRENT_USER )
              OR
                CP.GRANTEE IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) ) )
  AND
  CONSTRAINT_CATALOG
  = ( SELECT CATALOG_NAME FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TABLE_CONSTRAINTS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.59 TABLE_METHOD_PRIVILEGES view

Function

Identify the privileges on methods of tables of structured type defined in those catalogs that are available to or granted by a given user or role.

Definition

```
CREATE VIEW TABLE_METHOD_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.TABLE_METHOD_PRIVILEGES
 WHERE ( GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR
        = CURRENT_USER
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND
    TABLE_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TABLE_METHOD_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES.

5.60 TABLE_PRIVILEGES view

Function

Identify the privileges on tables defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW TABLE_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE, WITH_HIERARCHY
  FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES
 WHERE ( GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR
        = CURRENT_USER
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND
    TABLE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TABLE_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_PRIVILEGES.

5.61 TABLES view

Function

Identify the tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TABLES AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         TABLE_TYPE, SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,
         USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME, IS_INSERTABLE_INTO, IS_TYPED,
         COMMIT_ACTION
  FROM DEFINITION_SCHEMA.TABLES
 WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
        ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
          FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
          WHERE ( TP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) )
        UNION
          SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
          FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
          WHERE ( CP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  CP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
 AND
   TABLE_CATALOG
 = ( SELECT CATALOG_NAME
     FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TABLES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLES.

5.62 TRANSFORMS view

Function

Identify the transforms on user-defined types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRANSFORMS AS
  SELECT USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         USER_DEFINED_TYPE_NAME AS UDT_NAME,
         SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
         GROUP_NAME, TRANSFORM_TYPE
  FROM DEFINITION_SCHEMA.TRANSFORMS
 WHERE ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME ) IN
       ( ( SELECT UDTF.USER_DEFINED_TYPE_CATALOG, UDTF.USER_DEFINED_TYPE_SCHEMA,
                 UDTF.USER_DEFINED_TYPE_NAME
           FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTF
           WHERE ( UDTF.GRANTEE IN
                   ( 'PUBLIC', CURRENT_USER )
                 OR
                 UDTF.GRANTEE IN
                   ( SELECT ROLE_NAME
                     FROM ENABLED_ROLES ) ) ) )
       AND
         USER_DEFINED_TYPE_CATALOG
       = ( SELECT CATALOG_NAME
           FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRANSFORMS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature S241, “Transform functions”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSFORMS.

5.63 TRANSLATIONS view

Function

Identify the character transliterations defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRANSLATIONS AS
  SELECT TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME,
         SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
         SOURCE_CHARACTER_SET_NAME,
         TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
         TARGET_CHARACTER_SET_NAME,
         TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
         TRANSLATION_SOURCE_NAME
  FROM DEFINITION_SCHEMA.TRANSLATIONS
 WHERE ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME, 'TRANSLATION' ) IN
        ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME, UP.OBJECT_TYPE
          FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
          WHERE ( UP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  UP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) ) )
 AND
    TRANSLATION_CATALOG
  = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRANSLATIONS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSLATIONS.
- 2) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSLATIONS.
- 3) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA, or TRANSLATION_SOURCE_NAME.

5.64 TRIGGERED_UPDATE_COLUMNS view

Function

Identify the columns in this catalog that are identified by the explicit UPDATE trigger event columns of a trigger defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRIGGERED_UPDATE_COLUMNS AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
         EVENT_OBJECT_COLUMN
  FROM DEFINITION_SCHEMA.TRIGGERED_UPDATE_COLUMNS
 WHERE ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
         EVENT_OBJECT_TABLE, EVENT_OBJECT_COLUMN ) IN
        ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA,
                CP.TABLE_NAME, CP.COLUMN_NAME
          FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
          WHERE CP.PRIVILEGE_TYPE <> 'SELECT'
            AND
              ( CP.GRANTEE IN
                ( 'PUBLIC', CURRENT_USER )
              OR
                CP.GRANTEE IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) )
  AND
    TRIGGER_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGERED_UPDATE_COLUMNS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS.
- 2) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS.

5.65 TRIGGER_COLUMN_USAGE view

Function

Identify the columns on which triggers defined in this catalog and owned by a given user are dependent because of their reference by the search condition or in their appearance in a triggered SQL statement of a trigger owned by a given user or role.

Definition

```
CREATE VIEW TRIGGER_COLUMN_USAGE AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
  FROM DEFINITION_SCHEMA.TRIGGER_COLUMN_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND TRIGGER_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGER_COLUMN_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_COLUMN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERED_COLUMN_USAGE.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_COLUMN_USAGE.

5.66 TRIGGER_ROUTINE_USAGE view

Function

Identify each SQL-invoked routine owned by a given user or role on which some trigger defined in this catalog is dependent.

Definition

```
CREATE VIEW TRIGGER_ROUTINE_USAGE AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
  FROM DEFINITION_SCHEMA.TRIGGER_ROUTINE_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGER_ROUTINE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE.

5.67 TRIGGER_SEQUENCE_USAGE view

Function

Identify each external sequence generator owned by a given user or role on which some trigger defined in this catalog is dependent.

Definition

```
CREATE VIEW TRIGGER_SEQUENCE_USAGE AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
  FROM DEFINITION_SCHEMA.TRIGGER_SEQUENCE_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    SEQUENCE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGER_SEQUENCE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.
- 3) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.
- 4) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

5.68 TRIGGER_TABLE_USAGE view

Function

Identify the tables on which triggers defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW TRIGGER_TABLE_USAGE AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
  FROM DEFINITION_SCHEMA.TRIGGER_TABLE_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    TRIGGER_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGER_TABLE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE.

5.69 TRIGGERS view

Function

Identify the triggers defined on tables in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRIGGERS AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         EVENT_MANIPULATION,
         EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
         ACTION_ORDER,
         CASE
           WHEN EXISTS
             ( SELECT *
               FROM DEFINITION_SCHEMA.SCHEMATA AS S
               WHERE ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                 AND
                   ( S.SCHEMA_OWNER = CURRENT_USER
                     OR
                     S.SCHEMA_OWNER IN
                       ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) ) )
             THEN ACTION_CONDITION
           ELSE NULL
         END AS ACTION_CONDITION,
         CASE
           WHEN EXISTS
             ( SELECT *
               FROM DEFINITION_SCHEMA.SCHEMATA AS S
               WHERE ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                 AND
                   ( S.SCHEMA_OWNER = CURRENT_USER
                     OR
                     S.SCHEMA_OWNER IN
                       ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) ) )
             THEN ACTION_STATEMENT
           ELSE NULL
         END AS ACTION_STATEMENT,
         ACTION_ORIENTATION, ACTION_TIMING,
         ACTION_REFERENCE_OLD_TABLE, ACTION_REFERENCE_NEW_TABLE,
         ACTION_REFERENCE_OLD_ROW, ACTION_REFERENCE_NEW_ROW,
         CREATED
  FROM DEFINITION_SCHEMA.TRIGGERS
 WHERE ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE ) IN
        ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
          FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
          WHERE TP.PRIVILEGE_TYPE <> 'SELECT'
```

ISO/IEC 9075-11:2003 (E)
5.69 TRIGGERS view

```

        AND
        ( TP.GRANTEE IN
          ( 'PUBLIC', CURRENT_USER )
        OR
          TP.GRANTEE IN
          ( SELECT ROLE_NAME
            FROM ENABLED_ROLES ) )
    UNION
    SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
    FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
    WHERE CP.PRIVILEGE_TYPE <> 'SELECT'
    AND
        ( CP.GRANTEE IN
          ( 'PUBLIC', CURRENT_USER )
        OR
          CP.GRANTEE IN
          ( SELECT ROLE_NAME
            FROM ENABLED_ROLES ) ) )
    AND
    TRIGGER_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGERS
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.
- 2) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.TRIGGER_CREATED.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.

5.70 UDT_PRIVILEGES view

Function

Identify the privileges on user-defined types defined in this catalog that are accessible to or granted by a given user or role.

Definition

```
CREATE VIEW UDT_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE,
         USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         USER_DEFINED_TYPE_NAME AS UDT_NAME,
         PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
 WHERE ( GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
      OR
        GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
      OR
        GRANTOR
        = CURRENT_USER
      OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
  AND
    USER_DEFINED_TYPE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE UDT_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.UDT_PRIVILEGES.

5.71 USAGE_PRIVILEGES view

Function

Identify the USAGE privileges on objects defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW USAGE_PRIVILEGES AS
  SELECT GRANTOR, GRANTEE,
         OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, 'USAGE' AS PRIVILEGE_TYPE, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
 WHERE ( GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        GRANTEE IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES )
        OR
        GRANTOR
        = CURRENT_USER
        OR
        GRANTOR IN
        ( SELECT ROLE_NAME
          FROM ENABLED_ROLES ) )
 AND
    OBJECT_CATALOG
  = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE USAGE_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.USAGE_PRIVILEGES.

5.72 USER_DEFINED_TYPES view

Function

Identify the user-defined types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW USER_DEFINED_TYPES AS
  SELECT U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA, U.USER_DEFINED_TYPE_NAME,
        USER_DEFINED_TYPE_CATEGORY, IS_INSTANTIABLE, IS_FINAL,
        ORDERING_FORM, ORDERING_CATEGORY,
        ORDERING_ROUTINE_CATALOG, ORDERING_ROUTINE_SCHEMA, ORDERING_ROUTINE_NAME,
        REFERENCE_TYPE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
        D1.CHARACTER_SET_CATALOG, D1.CHARACTER_SET_SCHEMA, D1.CHARACTER_SET_NAME,
        D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
        NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
        SOURCE_DTD_IDENTIFIER, REF_DTD_IDENTIFIER
  FROM ( DEFINITION_SCHEMA.USER_DEFINED_TYPES AS U
        LEFT JOIN
          DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
        ON ( ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
              U.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
              U.SOURCE_DTD_IDENTIFIER )
          = ( OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE,
              D1.DTD_IDENTIFIER )
        OR
          ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
              U.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
              U.REF_DTD_IDENTIFIER )
          = ( OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE,
              D1.DTD_IDENTIFIER ) ) )
  WHERE ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
        U.USER_DEFINED_TYPE_NAME ) IN
        ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG, UDTP.USER_DEFINED_TYPE_SCHEMA,
              UDTP.USER_DEFINED_TYPE_NAME
          FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
          WHERE ( UDTP.GRANTEE IN
                ( 'PUBLIC', CURRENT_USER )
              OR
                UDTP.GRANTEE IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) )
  AND
        U.USER_DEFINED_TYPE_CATALOG
        = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

```
GRANT SELECT ON TABLE USER_DEFINED_TYPES  
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.USER_DEFINED_TYPES.

5.73 VIEW_COLUMN_USAGE view

Function

Identify the columns on which viewed tables defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW VIEW_COLUMN_USAGE AS
  SELECT VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
  FROM DEFINITION_SCHEMA.VIEW_COLUMN_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    VIEW_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE VIEW_COLUMN_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_COLUMN_USAGE.

5.74 VIEW_ROUTINE_USAGE view

Function

Identify each routine owned by a given user or role on which a view defined in this catalog is dependent.

Definition

```
CREATE VIEW VIEW_ROUTINE_USAGE AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
  FROM DEFINITION_SCHEMA.VIEW_ROUTINE_USAGE T
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( T.TABLE_CATALOG, T.TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
  WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND
    T.SPECIFIC_CATALOG
  = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE VIEW_ROUTINE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_ROUTINE_USAGE.

5.75 VIEW_TABLE_USAGE view

Function

Identify the tables on which viewed tables defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW VIEW_TABLE_USAGE AS
  SELECT VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
  FROM DEFINITION_SCHEMA.VIEW_TABLE_USAGE
  JOIN
    DEFINITION_SCHEMA.SCHEMATA S
  ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
WHERE ( SCHEMA_OWNER = CURRENT_USER
      OR
        SCHEMA_OWNER IN
          ( SELECT ROLE_NAME
            FROM ENABLED_ROLES ) )
      AND
        VIEW_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE VIEW_TABLE_USAGE
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_TABLE_USAGE.

5.76 VIEWS view

Function

Identify the viewed tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW VIEWS AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    CASE
      WHEN EXISTS
        ( SELECT *
          FROM DEFINITION_SCHEMA.SCHEMATA AS S
          WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
            = ( S.CATALOG_NAME, S.SCHEMA_NAME )
            AND
              ( SCHEMA_OWNER = CURRENT_USER
                OR
                  SCHEMA_OWNER IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
        THEN VIEW_DEFINITION
      ELSE NULL
    END AS VIEW_DEFINITION,
    CHECK_OPTION, IS_UPDATABLE,
    ( SELECT IS_INSERTABLE_INTO
      FROM DEFINITION_SCHEMA.TABLES AS T
      WHERE ( V.TABLE_CATALOG, V.TABLE_SCHEMA, V.TABLE_NAME )
        = ( T.TABLE_CATALOG, T.TABLE_SCHEMA, T.TABLE_NAME )
    ) AS INSERTABLE_INTO
  FROM DEFINITION_SCHEMA.VIEWS AS V
  WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
    ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
      FROM TABLES )
  AND
    TABLE_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE VIEWS
  TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.77 Short name views

Function

Provide alternative views that use only identifiers that do not require Feature F391, “Long identifiers”.

Definition

```
CREATE VIEW CATALOG_NAME
    ( CATALOG_NAME ) AS
    SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME;

GRANT SELECT ON TABLE CATALOG_NAME
    TO PUBLIC WITH GRANT OPTION;

CREATE VIEW ADMIN_ROLE_AUTHS
    ( GRANTEE, ROLE_NAME, IS_GRANTABLE ) AS
    SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
    FROM INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS;

GRANT SELECT ON TABLE ADMIN_ROLE_AUTHS
    TO PUBLIC WITH GRANT OPTION;

CREATE VIEW ATTRIBUTES_S
    ( UDT_CATALOG,          UDT_SCHEMA,          UDT_NAME,
      ATTRIBUTE_NAME,      ORDINAL_POSITION,    ATTRIBUTE_DEFAULT,
      IS_NULLABLE,        DATA_TYPE,          CHAR_MAX_LENGTH,
      CHAR_OCTET_LENGTH,  CHAR_SET_CATALOG,  CHAR_SET_SCHEMA,
      CHARACTER_SET_NAME, COLLATION_CATALOG,    COLLATION_SCHEMA,
      COLLATION_NAME,     NUMERIC_PRECISION,    NUMERIC_PREC_RADIX,
      NUMERIC_SCALE,      DATETIME_PRECISION, INTERVAL_TYPE,
      INTERVAL_PRECISION, ATT_UDT_CAT,          ATT_UDT_SCHEMA,
      ATT_UDT_NAME,        SCOPE_CATALOG,        SCOPE_SCHEMA,
      SCOPE_NAME,          MAX_CARDINALITY,    DTD_IDENTIFIER,
      IS_DERIVED_REF_ATT ) AS
    SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
      ATTRIBUTE_NAME, ORDINAL_POSITION, ATTRIBUTE_DEFAULT,
      IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
      CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
      COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
      NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
      INTERVAL_PRECISION, ATTRIBUTE_UDT_CATALOG, ATTRIBUTE_UDT_SCHEMA,
      ATTRIBUTE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
      SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
      IS_DERIVED_REFERENCE_ATTRIBUTE
    FROM INFORMATION_SCHEMA.ATTRIBUTES;

GRANT SELECT ON TABLE ATTRIBUTES_S
```

ISO/IEC 9075-11:2003 (E)
5.77 Short name views

TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW CHARACTER_SETS_S
( CHAR_SET_CATALOG, CHAR_SET_SCHEMA, CHARACTER_SET_NAME,
  CHAR_REPERTOIRE, FORM_OF_USE, NUMBER_OF_CHARS,
  DEF_COLLATE_CAT, DEF_COLLATE_SCHEMA, DEF_COLLATE_NAME ) AS
SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
       CHARACTER_REPERTOIRE, FORM_OF_USE, NUMBER_OF_CHARACTERS,
       DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, DEFAULT_COLLATE_NAME
FROM INFORMATION_SCHEMA.CHARACTER_SETS;
```

GRANT SELECT ON TABLE CHARACTER_SETS_S
TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW COLLATION_APPLIC_S
( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
  CHAR_SET_CATALOG, CHAR_SET_SCHEMA, CHARACTER_SET_NAME ) AS
SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
       CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME
FROM INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY;
```

GRANT SELECT ON TABLE COLLATION_APPLIC_S
TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW COLLATIONS_S
( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
  PAD_ATTRIBUTE, COLLATION_TYPE, COLLATION_DEFN,
  COLLATION_DICT ) AS
SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
       PAD_ATTRIBUTE, COLLATION_TYPE, COLLATION_DEFINITION,
       COLLATION_DICTIONARY
FROM INFORMATION_SCHEMA.COLLATIONS;
```

GRANT SELECT ON TABLE COLLATIONS_S
TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW COL_COL_USAGE
( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
  COLUMN_NAME, DEPENDENT_COLUMN ) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       COLUMN_NAME, DEPENDENT_COLUMN
FROM INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE;
```

GRANT SELECT ON TABLE COL_COL_USAGE
TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW COL_DOMAIN_USAGE
( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
  COLUMN_NAME ) AS
SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
```



```
TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
FROM INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE;
```

```
GRANT SELECT ON TABLE COL_DOMAIN_USAGE
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW COLUMNS_S
( TABLE_CATALOG,      TABLE_SCHEMA,      TABLE_NAME,
  COLUMN_NAME,         ORDINAL_POSITION,   COLUMN_DEFAULT,
  IS_NULLABLE,        DATA_TYPE,         CHAR_MAX_LENGTH,
  CHAR_OCTET_LENGTH,  NUMERIC_PRECISION,  NUMERIC_PREC_RADIX,
  NUMERIC_SCALE,      DATETIME_PRECISION, INTERVAL_TYPE,
  INTERVAL_PRECISION, CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,
  CHARACTER_SET_NAME, COLLATION_CATALOG,  COLLATION_SCHEMA,
  COLLATION_NAME,     DOMAIN_CATALOG,    DOMAIN_SCHEMA,
  DOMAIN_NAME,        UDT_CATALOG,      UDT_SCHEMA,
  UDT_NAME,           SCOPE_CATALOG,   SCOPE_SCHEMA,
  SCOPE_NAME,         MAX_CARDINALITY,  DTD_IDENTIFIER,
  IS_SELF_REF,        IS_IDENTITY,      ID_GENERATION,
  ID_START,           ID_INCREMENT,    ID_MAXIMUM,
  ID_MINIMUM,         ID_CYCLE,        IS_GENERATED,
  GENERATION_EXPR,    IS_UPDATABLE ) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
  COLUMN_NAME, ORDINAL_POSITION, COLUMN_DEFAULT,
  IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
  CHARACTER_OCTET_LENGTH, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
  INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
  CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
  COLLATION_NAME, DOMAIN_CATALOG, DOMAIN_SCHEMA,
  DOMAIN_NAME, UDT_CATALOG, UDT_SCHEMA,
  UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
  SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
  IS_SELF_REFERENCING, IS_IDENTITY, IDENTITY_GENERATION,
  IDENTITY_START, IDENTITY_INCREMENT, IDENTITY_MAXIMUM,
  IDENTITY_MINIMUM, IDENTITY_CYCLE, IS_GENERATED,
  GENERATION_EXPRESSION, IS_UPDATABLE
FROM INFORMATION_SCHEMA.COLUMNS;
```

```
GRANT SELECT ON TABLE COLUMNS_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW CONSTR_ROUT_USE_S
( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
  SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,   SPECIFIC_NAME ) AS
SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
  SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
FROM INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE;
```

```
GRANT SELECT ON TABLE CONSTR_ROUT_USE_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW CONSTR_COL_USAGE
```

ISO/IEC 9075-11:2003 (E)
5.77 Short name views

```
        ( TABLE_CATALOG,      TABLE_SCHEMA,      TABLE_NAME,
          COLUMN_NAME,         CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
          CONSTRAINT_NAME ) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       COLUMN_NAME, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
       CONSTRAINT_NAME
FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE;
```

```
GRANT SELECT ON TABLE CONSTR_COL_USAGE
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW CONSTR_TABLE_USAGE
        ( TABLE_CATALOG,      TABLE_SCHEMA,      TABLE_NAME,
          CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
FROM INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE;
```

```
GRANT SELECT ON TABLE CONSTR_TABLE_USAGE
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW DOMAINS_S
        ( DOMAIN_CATALOG,      DOMAIN_SCHEMA,      DOMAIN_NAME,
          DATA_TYPE,          CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,
          CHAR_SET_CATALOG,    CHAR_SET_SCHEMA,    CHARACTER_SET_NAME,
          COLLATION_CATALOG,    COLLATION_SCHEMA, COLLATION_NAME,
          NUMERIC_PRECISION,    NUMERIC_PREC_RADIX, NUMERIC_SCALE,
          DATETIME_PRECISION,   INTERVAL_TYPE,     INTERVAL_PRECISION,
          DOMAIN_DEFAULT,       MAX_CARDINALITY,    DTD_IDENTIFIER ) AS
SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
       DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
       CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
       COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
       NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
       DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
       DOMAIN_DEFAULT, MAXIMUM_CARDINALITY, DTD_IDENTIFIER
FROM INFORMATION_SCHEMA.DOMAINS;
```

```
GRANT SELECT ON TABLE DOMAINS_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW ELEMENT_TYPES_S
        ( OBJECT_CATALOG,      OBJECT_SCHEMA,      OBJECT_NAME,
          OBJECT_TYPE,          COLLECTION_TYPE_ID, DATA_TYPE,
          CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
          CHAR_SET_SCHEMA,     CHARACTER_SET_NAME, COLLATION_CATALOG,
          COLLATION_SCHEMA,     COLLATION_NAME,     NUMERIC_PRECISION,
          NUMERIC_PREC_RADIX,    NUMERIC_SCALE,       DATETIME_PRECISION,
          INTERVAL_TYPE,        INTERVAL_PRECISION,   UDT_CATALOG,
          UDT_SCHEMA,           UDT_NAME,            SCOPE_CATALOG,
          SCOPE_SCHEMA,         SCOPE_NAME,          MAX_CARDINALITY,
          DTD_IDENTIFIER ) AS
SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
```

```
OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER, DATA_TYPE,  
CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,  
CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,  
COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,  
NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,  
INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,  
UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,  
SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,  
DTD_IDENTIFIER  
FROM INFORMATION_SCHEMA.ELEMENT_TYPES;
```

```
GRANT SELECT ON TABLE ELEMENT_TYPES_S  
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW FIELDS_S  
( OBJECT_CATALOG,      OBJECT_SCHEMA,      OBJECT_NAME,  
  OBJECT_TYPE,         ROW_IDENTIFIER,     FIELD_NAME,  
  ORDINAL_POSITION,   DATA_TYPE,         CHAR_MAX_LENGTH,  
  CHAR_OCTET_LENGTH,  CHAR_SET_CATALOG,  CHAR_SET_SCHEMA,  
  CHARACTER_SET_NAME, COLLATION_CATALOG,  COLLATION_SCHEMA,  
  COLLATION_NAME,     NUMERIC_PRECISION,  NUMERIC_PREC_RADIX,  
  NUMERIC_SCALE,      DATETIME_PRECISION, INTERVAL_TYPE,  
  INTERVAL_PRECISION, UDT_CATALOG,        UDT_SCHEMA,  
  UDT_NAME,           SCOPE_CATALOG,      SCOPE_SCHEMA,  
  SCOPE_NAME,         MAX_CARDINALITY,    DTD_IDENTIFIER ) AS  
SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,  
  OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,  
  ORDINAL_POSITION, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,  
  CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,  
  CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,  
  COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,  
  NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,  
  INTERVAL_PRECISION, UDT_CATALOG, UDT_SCHEMA,  
  UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,  
  SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER  
FROM INFORMATION_SCHEMA.FIELDS;
```

```
GRANT SELECT ON TABLE FIELDS_S  
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW KEY_COLUMN_USAGE_S  
( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,  CONSTRAINT_NAME,  
  TABLE_CATALOG,     TABLE_SCHEMA,    TABLE_NAME,  
  COLUMN_NAME,        ORDINAL_POSITION,  POSITION_IN_UC ) AS  
SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,  
  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,  
  COLUMN_NAME, ORDINAL_POSITION, POSITION_IN_UNIQUE_CONSTRAINT  
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE;
```

```
GRANT SELECT ON TABLE KEY_COLUMN_USAGE_S  
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW METHOD_SPECS
```

ISO/IEC 9075-11:2003 (E)
5.77 Short name views

```

( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
  UDT_CATALOG,        UDT_SCHEMA,        UDT_NAME,
  METHOD_NAME,        IS_STATIC,        IS_OVERRIDING,
  IS_CONSTRUCTOR,    DATA_TYPE,        CHAR_MAX_LENGTH,
  CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,    CHAR_SET_SCHEMA,
  CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
  COLLATION_NAME,    NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
  NUMERIC_SCALE,    DATETIME_PRECISION, INTERVAL_TYPE,
  INTERVAL_PRECISION, RETURN_UDT_CATALOG, RETURN_UDT_SCHEMA,
  RETURN_UDT_NAME,  SCOPE_CATALOG,    SCOPE_SCHEMA,
  SCOPE_NAME,        MAX_CARDINALITY,    DTD_IDENTIFIER,
  METHOD_LANGUAGE,    PARAMETER_STYLE,    IS_DETERMINISTIC,
  SQL_DATA_ACCESS,    IS_NULL_CALL,    TO_SQL_SPEC_CAT,
  TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME,    AS_LOCATOR,
  CREATED,            LAST_ALTERED,        RC_FROM_DATA_TYPE,
  RC_AS_LOCATOR,      RC_CHAR_MAX_LENGTH, RC_CHAR_OCT_LENGTH,
  RC_CHAR_SET_CAT,    RC_CHAR_SET_SCHEMA, RC_CHAR_SET_NAME,
  RC_COLLATION_CAT,   RC_COLLATION_SCH,   RC_COLLATION_NAME,
  RC_NUMERIC_PREC,    RC_NUMERIC_RADIX,   RC_NUMERIC_SCALE,
  RC_DATETIME_PREC,   RC_INTERVAL_TYPE,   RC_INTERVAL_PREC,
  RC_TYPE_UDT_CAT,    RC_TYPE_UDT_SCHEMA, RC_TYPE_UDT_NAME,
  RC_SCOPE_CATALOG,   RC_SCOPE_SCHEMA,    RC_SCOPE_NAME,
  RC_MAX_CARDINALITY, RC_DTD_IDENTIFIER ) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
  USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME,
  METHOD_NAME, IS_STATIC, IS_OVERRIDING,
  IS_CONSTRUCTOR, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
  CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
  CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
  COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
  INTERVAL_PRECISION, RETURN_UDT_CATALOG, RETURN_UDT_SCHEMA,
  RETURN_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
  SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
  METHOD_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
  SQL_DATA_ACCESS, IS_NULL_CALL, TO_SQL_SPECIFIC_CATALOG,
  TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, AS_LOCATOR,
  CREATED, LAST_ALTERED, RESULT_CAST_FROM_DATA_TYPE,
  RESULT_CAST_AS_LOCATOR, RESULT_CAST_CHAR_MAX_LENGTH,
RESULT_CAST_CHAR_OCTET_LENGTH,
  RESULT_CAST_CHAR_SET_CATALOG, RESULT_CAST_CHAR_SET_SCHEMA,
RESULT_CAST_CHAR_SET_NAME,
  RESULT_CAST_COLLATION_CATALOG, RESULT_CAST_COLLATION_SCHEMA,
RESULT_CAST_COLLATION_NAME,
  RESULT_CAST_NUMERIC_PRECISION, RESULT_CAST_NUMERIC_RADIX,
RESULT_CAST_NUMERIC_SCALE,
  RESULT_CAST_DATETIME_PRECISION, RESULT_CAST_INTERVAL_TYPE,
RESULT_CAST_INTERVAL_PRECISION,
  RESULT_CAST_TYPE_UDT_CATALOG, RESULT_CAST_TYPE_UDT_SCHEMA,
RESULT_CAST_TYPE_UDT_NAME,
  RESULT_CAST_SCOPE_CATALOG, RESULT_CAST_SCOPE_SCHEMA, RESULT_CAST_SCOPE_NAME,
  RESULT_CAST_MAX_CARDINALITY, RESULT_CAST_DTD_IDENTIFIER
FROM INFORMATION_SCHEMA.METHOD_SPECIFICATIONS;

GRANT SELECT ON TABLE METHOS_SPECS
  TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW METHOD_SPEC_PARAMS
( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
  ORDINAL_POSITION,   PARAMETER_MODE,    IS_RESULT,
  AS_LOCATOR,         PARAMETER_NAME,    FROM_SQL_SPEC_CAT,
  FROM_SQL_SPEC_SCH,  FROM_SQL_SPEC_NAME, DATA_TYPE,
  CHAR_MAX_LENGTH,    CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
  CHAR_SET_SCHEMA,    CHARACTER_SET_NAME, COLLATION_CATALOG,
  COLLATION_SCHEMA,   COLLATION_NAME,    NUMERIC_PRECISION,
  NUMERIC_PREC_RADIX, NUMERIC_SCALE,    DATETIME_PRECISION,
  INTERVAL_TYPE,      INTERVAL_PRECISION, PARM_UDT_CATALOG,
  PARM_UDT_SCHEMA,    PARM_UDT_NAME,    SCOPE_CATALOG,
  SCOPE_SCHEMA,       SCOPE_NAME,      MAX_CARDINALITY,
  DTD_IDENTIFIER ) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
  ORDINAL_POSITION, PARAMETER_MODE, IS_RESULT,
  AS_LOCATOR, PARAMETER_NAME, FROM_SQL_SPECIFIC_CATALOG,
  FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME, DATA_TYPE,
  CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
  CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
  COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
  NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
  INTERVAL_TYPE, INTERVAL_PRECISION, PARAMETER_UDT_CATALOG,
  PARAMETER_UDT_SCHEMA, PARAMETER_UDT_NAME, SCOPE_CATALOG,
  SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
  DTD_IDENTIFIER
FROM INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS;

GRANT SELECT ON TABLE METHOD_SPEC_PARAMS
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW PARAMETERS_S
( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
  ORDINAL_POSITION,   PARAMETER_MODE,    IS_RESULT,
  AS_LOCATOR,         PARAMETER_NAME,    FROM_SQL_SPEC_CAT,
  FROM_SQL_SPEC_SCH,  FROM_SQL_SPEC_NAME, TO_SQL_SPEC_CAT,
  TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME,  DATA_TYPE,
  CHAR_MAX_LENGTH,    CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
  CHAR_SET_SCHEMA,    CHARACTER_SET_NAME, COLLATION_CATALOG,
  COLLATION_SCHEMA,   COLLATION_NAME,    NUMERIC_PRECISION,
  NUMERIC_PREC_RADIX, NUMERIC_SCALE,    DATETIME_PRECISION,
  INTERVAL_TYPE,      INTERVAL_PRECISION, UDT_CATALOG,
  UDT_SCHEMA,         UDT_NAME,          SCOPE_CATALOG,
  SCOPE_SCHEMA,       SCOPE_NAME,      MAX_CARDINALITY,
  DTD_IDENTIFIER ) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
  ORDINAL_POSITION, PARAMETER_MODE, IS_RESULT,
  AS_LOCATOR, PARAMETER_NAME, FROM_SQL_SPECIFIC_CATALOG,
  FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME, TO_SQL_SPECIFIC_CATALOG,
  TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, DATA_TYPE,
  CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
  CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
  COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
  NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,

```

ISO/IEC 9075-11:2003 (E)

5.77 Short name views

```
INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
DTD_IDENTIFIER
FROM INFORMATION_SCHEMA.PARAMETERS;
```

```
GRANT SELECT ON TABLE PARAMETERS_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW REFERENCED_TYPES_S
( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
  OBJECT_TYPE, REFERENCE_TYPE_ID, DATA_TYPE,
  CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
  CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
  COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
  NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
  INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
  UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
  SCOPE_SCHEMA, SCOPE_NAME, MAX_CARDINALITY,
  DTD_IDENTIFIER ) AS
SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
DTD_IDENTIFIER
FROM INFORMATION_SCHEMA.REFERENCED_TYPES;
```

```
GRANT SELECT ON TABLE REFERENCED_TYPES_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW REF_CONSTRAINTS
( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
  UNIQUE_CONSTR_CAT, UNIQUE_CONSTR_SCH, UNIQUE_CONSTR_NAME,
  MATCH_OPTION, UPDATE_RULE, DELETE_RULE ) AS
SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA, UNIQUE_CONSTRAINT_NAME,
MATCH_OPTION, UPDATE_RULE, DELETE_RULE
FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS;
```

```
GRANT SELECT ON TABLE REF_CONSTRAINTS
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW ROLE_ROUT_GRANTS
( GRANTOR, GRANTEE, SPECIFIC_CATALOG,
  SPECIFIC_SCHEMA, SPECIFIC_NAME, ROUTINE_CATALOG,
  ROUTINE_SCHEMA, ROUTINE_NAME, PRIVILEGE_TYPE,
  IS_GRANTABLE ) AS
SELECT GRANTOR, GRANTEE, SPECIFIC_CATALOG,
```

```

        SPECIFIC_SCHEMA, SPECIFIC_NAME, ROUTINE_CATALOG,
        ROUTINE_SCHEMA, ROUTINE_NAME, PRIVILEGE_TYPE,
        IS_GRANTABLE
    FROM INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS;

```

```

GRANT SELECT ON TABLE ROLE_ROUT_GRANTS
    TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW ROL_TAB_METH_GRNTS
    ( GRANTOR, GRANTEE, TABLE_CATALOG,
      TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
      SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE ) AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
           TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
           SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
    FROM INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS;

```

```

GRANT SELECT ON TABLE ROL_TAB_METH_GRNTS
    TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW ROUT_ROUT_USAGE_S
    ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
      ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           ROUTINE_CATALOG, RRU.ROUTINE_SCHEMA, RRU.ROUTINE_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_ROUTINE_USAGE RRU;

```

```

GRANT SELECT ON TABLE ROUT_ROUT_USAGE_S
    TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW ROUT_SEQ_USAGE_S
    ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
      SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE;

```

```

GRANT SELECT ON TABLE ROUT_SEQ_USAGE_S
    TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW ROUTINE_COL_USAGE
    ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
      ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      COLUMN_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           COLUMN_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_COLUMN_USAGE;

```

```

GRANT SELECT ON TABLE ROUTINE_COL_USAGE

```

ISO/IEC 9075-11:2003 (E)
5.77 Short name views

TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW ROUT_TABLE_USAGE
( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
  ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
       ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
       TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.ROUTINE_TABLE_USAGE;
```

GRANT SELECT ON TABLE ROUT_TABLE_USAGE
 TO PUBLIC WITH GRANT OPTION;

```
CREATE VIEW ROUTINES_S
( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
  ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
  ROUTINE_TYPE, MODULE_CATALOG, MODULE_SCHEMA,
  MODULE_NAME, UDT_CATALOG, UDT_SCHEMA,
  UDT_NAME, DATA_TYPE, CHAR_MAX_LENGTH,
  CHAR_OCTET_LENGTH, CHAR_SET_CATALOG, CHAR_SET_SCHEMA,
  CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
  COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
  INTERVAL_PRECISION, TYPE_UDT_CATALOG, TYPE_UDT_SCHEMA,
  TYPE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
  SCOPE_NAME, MAX_CARDINALITY, DTD_IDENTIFIER,
  ROUTINE_BODY, ROUTINE_DEFINITION, EXTERNAL_NAME,
  EXTERNAL_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
  SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
  SCH_LEVEL_ROUTINE, MAX_DYN_RESLT_SETS, IS_USER_DEFND_CAST,
  IS_IMP_INVOCABLE, SECURITY_TYPE, TO_SQL_SPEC_CAT,
  TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME, AS_LOCATOR,
  CREATED, LAST_ALTERED, NEW_SAVEPOINT_LVL,
  IS_UDT_DEPENDENT, RC_FROM_DATA_TYPE, RC_AS_LOCATOR,
  RC_CHAR_MAX_LENGTH, RC_CHAR_OCT_LENGTH, RC_CHAR_SET_CAT,
  RC_CHAR_SET_SCHEMA, RC_CHAR_SET_NAME, RC_COLLATION_CAT,
  RC_COLLATION_SCH, RC_COLLATION_NAME, RC_NUM_PREC,
  RC_NUMERIC_RADIX, RC_NUMERIC_SCALE, RC_DATETIME_PREC,
  RC_INTERVAL_TYPE, RC_INTERVAL_PREC, RC_TYPE_UDT_CAT,
  RC_TYPE_UDT_SCHEMA, RC_TYPE_UDT_NAME, RC_SCOPE_CATALOG,
  RC_SCOPE_SCHEMA, RC_SCOPE_NAME, RC_MAX_CARDINALITY,
  RC_DTD_IDENTIFIER ) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
       ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
       ROUTINE_TYPE, MODULE_CATALOG, MODULE_SCHEMA,
       MODULE_NAME, UDT_CATALOG, UDT_SCHEMA,
       UDT_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
       CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
       CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
       COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
       NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
       INTERVAL_PRECISION, TYPE_UDT_CATALOG, TYPE_UDT_SCHEMA,
       TYPE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
```



```

SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
ROUTINE_BODY, ROUTINE_DEFINITION, EXTERNAL_NAME,
EXTERNAL_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
SCHEMA_LEVEL_ROUTINE, MAX_DYNAMIC_RESULT_SETS, IS_USER_DEFINED_CAST,
IS_IMPLICITLY_INVOCABLE, SECURITY_TYPE, TO_SQL_SPECIFIC_CATALOG,
TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, AS_LOCATOR,
CREATED, LAST_ALTERED, NEW_SAVEPOINT_LEVEL,
IS_UDT_DEPENDENT, RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,
RESULT_CAST_CHAR_MAX_LENGTH, RESULT_CAST_CHAR_OCTET_LENGTH,
    RESULT_CAST_CHAR_SET_CATALOG,
RESULT_CAST_CHAR_SET_SCHEMA, RESULT_CAST_CHAR_SET_NAME,
    RESULT_CAST_COLLATION_CATALOG,
RESULT_CAST_COLLATION_SCHEMA, RESULT_CAST_COLLATION_NAME,
    RESULT_CAST_NUMERIC_PRECISION,
RESULT_CAST_NUMERIC_RADIX, RESULT_CAST_NUMERIC_SCALE,
    RESULT_CAST_DATETIME_PRECISION,
RESULT_CAST_INTERVAL_TYPE, RESULT_CAST_INTERVAL_PRECISION,
    RESULT_CAST_TYPE_UDT_CATALOG,
RESULT_CAST_TYPE_UDT_SCHEMA, RESULT_CAST_TYPE_UDT_NAME,
    RESULT_CAST_SCOPE_CATALOG,
RESULT_CAST_SCOPE_SCHEMA, RESULT_CAST_SCOPE_NAME,
    RESULT_CAST_MAX_CARDINALITY,
RESULT_CAST_DTD_IDENTIFIER
FROM INFORMATION_SCHEMA.ROUTINES;

```

```

GRANT SELECT ON TABLE ROUTINES_S
    TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW SCHEMATA_S
    ( CATALOG_NAME,          SCHEMA_NAME,          SCHEMA_OWNER,
      DEF_CHAR_SET_CAT,     DEF_CHAR_SET_SCH,     DEF_CHAR_SET_NAME,
      SQL_PATH ) AS
SELECT CATALOG_NAME, SCHEMA_NAME, SCHEMA_OWNER,
    DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
    DEFAULT_CHARACTER_SET_NAME, SQL_PATH
FROM INFORMATION_SCHEMA.SCHEMATA;

```

```

GRANT SELECT ON TABLE SCHEMATA_S
    TO PUBLIC WITH GRANT OPTION;

```

```

CREATE VIEW SEQUENCES_S
    ( SEQUENCE_CATALOG,    SEQUENCE_SCHEMA,    SEQUENCE_NAME,
      DATA_TYPE,          NUMERIC_PRECISION,    NUMERIC_PREC_RADIX,
      NUMERIC_SCALE,       MAXIMUM_VALUE,        MINIMUM_VALUE,
      INCREMENT,           CYCLE_OPTION ) AS
SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME,
    DATA_TYPE, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
    NUMERIC_SCALE, MAXIMUM_VALUE, MINIMUM_VALUE,
    INCREMENT, CYCLE_OPTION
FROM INFORMATION_SCHEMA.SEQUENCES;

```

```

GRANT SELECT ON TABLE SEQUENCES_S
    TO PUBLIC WITH GRANT OPTION;

```

ISO/IEC 9075-11:2003 (E)

5.77 Short name views

```
CREATE VIEW SQL_IMPL_INFO
( IMPL_INFO_ID,          IMPL_INFO_NAME,          INTEGER_VALUE,
  CHARACTER_VALUE,      COMMENTS ) AS
SELECT IMPLEMENTATION_INFO_ID, IMPLEMENTATION_INFO_NAME, INTEGER_VALUE,
       CHARACTER_VALUE, COMMENTS
FROM INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO;

GRANT SELECT ON TABLE SQL_IMPL_INFO
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW SQL_SIZING_PROFS
( SIZING_ID,          SIZING_NAME,          PROFILE_ID,
  PROFILE_NAME,      REQUIRED_VALUE,      COMMENTS ) AS
SELECT SIZING_ID, SIZING_NAME, PROFILE_ID,
       PROFILE_NAME, REQUIRED_VALUE, COMMENTS
FROM INFORMATION_SCHEMA.SQL_SIZING_PROFILES;

GRANT SELECT ON TABLE SQL_SIZING_PROFS
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW SQL_LANGUAGES_S
( SOURCE,          SQL_LANGUAGE_YEAR,  CONFORMANCE,
  INTEGRITY,      IMPLEMENTATION,      BINDING_STYLE,
  PROGRAMMING_LANG ) AS
SELECT SQL_LANGUAGE_SOURCE, SQL_LANGUAGE_YEAR, SQL_LANGUAGE_CONFORMANCE,
       SQL_LANGUAGE_INTEGRITY, SQL_LANGUAGE_IMPLEMENTATION, SQL_LANGUAGE_BINDING_STYLE,

       SQL_LANGUAGE_PROGRAMMING_LANGUAGE
FROM INFORMATION_SCHEMA.SQL_LANGUAGES;

GRANT SELECT ON TABLE SQL_LANGUAGES_S
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW TABLE_METHOD_PRIVS
( GRANTOR,          GRANTEE,          TABLE_CATALOG,
  TABLE_SCHEMA,    TABLE_NAME,      SPECIFIC_CATALOG,
  SPECIFIC_SCHEMA,  SPECIFIC_NAME,    IS_GRANTABLE ) AS
SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
       TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
       SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
FROM INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES;

GRANT SELECT ON TABLE TABLE_METHOD_PRIVS
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW TABLES_S
( TABLE_CATALOG,    TABLE_SCHEMA,    TABLE_NAME,
  TABLE_TYPE,      SELF_REF_COL_NAME, REF_GENERATION,
  UDT_CATALOG,       UDT_SCHEMA,        UDT_NAME,
  IS_INSERTABLE_INT, IS_TYPED,          COMMIT_ACTION ) AS
```

```
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       TABLE_TYPE, SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,
       USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME,
       IS_INSERTABLE_INTO, IS_TYPED, COMMIT_ACTION
FROM INFORMATION_SCHEMA.TABLES;
```

```
GRANT SELECT ON TABLE TABLES_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRANSLATIONS_S
( TRANS_CATALOG,      TRANSLATION_SCHEMA, TRANSLATION_NAME,
  SRC_CHAR_SET_CAT,   SRC_CHAR_SET_SCH,   SRC_CHAR_SET_NAME,
  TGT_CHAR_SET_CAT,   TGT_CHAR_SET_SCH,   TGT_CHAR_SET_NAME,
  TRANS_SRC_CATALOG,  TRANS_SRC_SCHEMA,    TRANS_SRC_NAME ) AS
SELECT TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME,
       SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
       SOURCE_CHARACTER_SET_NAME,
       TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
       TARGET_CHARACTER_SET_NAME,
       TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
       TRANSLATION_SOURCE_NAME
FROM INFORMATION_SCHEMA.TRANSLATIONS;
```

```
GRANT SELECT ON TABLE TRANSLATIONS_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRIG_ROUT_USAGE_S
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
  SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,     SPECIFIC_NAME ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
FROM INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE;
```

```
GRANT SELECT ON TABLE TRIG_ROUT_USAGE_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRIG_SEQ_USAGE_S
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
  SEQUENCE_CATALOG,   SEQUENCE_SCHEMA,     SEQUENCE_NAME ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
FROM INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE;
```

```
GRANT SELECT ON TABLE TRIG_SEQ_USAGE_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRIG_UPDATE_COLS
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
  EVENT_OBJECT_CAT,   EVENT_OBJECT_SCH,    EVENT_OBJECT_TABLE,
  EVENT_OBJECT_COL ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
```

ISO/IEC 9075-11:2003 (E)

5.77 Short name views

```
        EVENT_OBJECT_COLUMN
FROM INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS;
```

```
GRANT SELECT ON TABLE TRIG_UPDATE_COLS
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRIG_COLUMN_USAGE
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,    TRIGGER_NAME,
  TABLE_CATALOG,    TABLE_SCHEMA,    TABLE_NAME,
  COLUMN_NAME ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
FROM INFORMATION_SCHEMA.TRIGGER_COLUMN_USAGE;
```

```
GRANT SELECT ON TABLE TRIG_COLUMN_USAGE
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRIG_TABLE_USAGE
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,    TRIGGER_NAME,
  TABLE_CATALOG,    TABLE_SCHEMA,    TABLE_NAME ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE;
```

```
GRANT SELECT ON TABLE TRIGGER_TABLE_USAGE
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW TRIGGERS_S
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,    TRIGGER_NAME,
  EVENT_MANIPULATION, EVENT_OBJECT_CAT,  EVENT_OBJECT_SCH,
  EVENT_OBJECT_TABLE, ACTION_ORDER,      ACTION_CONDITION,
  ACTION_STATEMENT,   ACTION_ORIENTATION, ACTION_TIMING,
  ACT_REF_OLD_TABLE,  ACT_REF_NEW_TABLE,  ACT_REF_OLD_ROW,
  ACT_REF_NEW_ROW,    CREATED ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       EVENT_MANIPULATION, EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
       EVENT_OBJECT_TABLE, ACTION_ORDER, ACTION_CONDITION,
       ACTION_STATEMENT, ACTION_ORIENTATION, ACTION_TIMING,
       ACTION_REFERENCE_OLD_TABLE, ACTION_REFERENCE_NEW_TABLE,
       ACTION_REFERENCE_OLD_ROW, ACTION_REFERENCE_NEW_ROW, CREATED
FROM INFORMATION_SCHEMA.TRIGGERS;
```

```
GRANT SELECT ON TABLE TRIGGERS_S
TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW UDT_S
( UDT_CATALOG,        UDT_SCHEMA,        UDT_NAME,
  UDT_CATEGORY,       IS_INSTANTIABLE,   IS_FINAL,
  ORDERING_FORM,      ORDERING_CATEGORY,  ORDERING_ROUT_CAT,
  ORDERING_ROUT_SCH,  ORDERING_ROUT_NAME,  REFERENCE_TYPE,
  DATA_TYPE,         CHAR_MAX_LENGTH,    CHAR_OCTET_LENGTH,
  CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,     CHARACTER_SET_NAME,
```

```
COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,  
NUMERIC_PRECISION, NUMERIC_PREC_RADIX, NUMERIC_SCALE,  
DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,  
SOURCE_DTD_ID, REF_DTD_IDENTIFIER ) AS  
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME,  
USER_DEFINED_TYPE_CATEGORY, IS_INSTANTIABLE, IS_FINAL,  
ORDERING_FORM, ORDERING_CATEGORY, ORDERING_ROUTINE_CATALOG,  
ORDERING_ROUTINE_SCHEMA, ORDERING_ROUTINE_NAME, REFERENCE_TYPE,  
DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,  
CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,  
COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,  
NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,  
DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,  
SOURCE_DTD_IDENTIFIER, REF_DTD_IDENTIFIER  
FROM INFORMATION_SCHEMA.USER_DEFINED_TYPES;
```

```
GRANT SELECT ON TABLE UDT_S  
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUT_GRANTS.
- 2) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS_S.
- 3) Without Feature F341, “Usage tables”, conforming SQL language shall not reference the INFORMATION_SCHEMA.TRIG_TABLE_USAGE view.
- 4) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_UPDATE_COLS
- 5) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COL_DOMAIN_USAGE.
- 6) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONST_COL_USAGE.
- 7) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONST_TABLE_USAGE.
- 8) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE_S.
- 9) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COL_USAGE.
- 10) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_TABLE_USAGE.
- 11) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_ROUT_USAGE_S.

- 12) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTR_ROUT_USE_S.
- 13) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_ROUT_USAGE_S.
- 14) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_SEQ_USAGE_S.
- 15) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_COLUMN_USAGE.
- 16) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE.
- 17) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPL_INFO.
- 18) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_SIZING_PROFS.
- 19) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS_S.
- 20) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSLATIONS_S.
- 21) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANS_SRC_CATALOG, TRANS_SRC_SCHEMA, or TRANS_SRC_NAME.
- 22) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES_S.
- 23) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECS.
- 24) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC_PARAMS.
- 25) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVS.
- 26) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC.CREATED.
- 27) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC.LAST_ALTERED.
- 28) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.CREATED.
- 29) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.LAST_ALTERED.
- 30) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.CREATED.

- 31) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COL_COL_USAGE.
- 32) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS_S.GENERATION_EXPR.
- 33) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_SEQ_USAGE_S.
- 34) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES_S.
- 35) Without Feature T111, “Updatable joins, unions, and columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS_S.IS_UPDATABLE.
- 36) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_UPDATE_COLS
- 37) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference the INFORMATION_SCHEMA.TRIG_TABLE_USAGE view.
- 38) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_ROUT_USAGE_S.
- 39) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_SEQ_USAGE_S.
- 40) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.CREATED.
- 41) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS_S.
- 42) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_COLUMN_USAGE.
- 43) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMIN_ROLE_AUTHS.
- 44) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUT_GRANTS.

This page intentionally left blank.

6 Definition Schema

6.1 DEFINITION_SCHEMA Schema

Function

Create the schema that is to contain the base tables that underlie the Information Schema

Definition

```
CREATE SCHEMA DEFINITION_SCHEMA  
    AUTHORIZATION DEFINITION_SCHEMA
```

Description

None.

6.2 EQUAL_KEY_DEGREES assertion

Function

The assertion EQUAL_KEY_DEGREES ensures that every foreign key is of the same degree as the corresponding unique constraint.

Definition

```
CREATE ASSERTION EQUAL_KEY_DEGREES
CHECK
( NOT EXISTS
  ( SELECT *
    FROM ( SELECT COUNT ( DISTINCT FK.COLUMN_NAME ),
                  COUNT ( DISTINCT PK.COLUMN_NAME )
          FROM KEY_COLUMN_USAGE AS FK,
               REFERENTIAL_CONSTRAINTS AS RF,
               KEY_COLUMN_USAGE AS PK
          WHERE ( FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA,
                  FK.CONSTRAINT_NAME ) =
                ( RF.CONSTRAINT_CATALOG, RF.CONSTRAINT_SCHEMA,
                  RF.CONSTRAINT_NAME )
          AND
                ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                  PK.CONSTRAINT_NAME ) =
                ( RF.UNIQUE_CONSTRAINT_CATALOG, RF.UNIQUE_CONSTRAINT_SCHEMA,
                  RF.UNIQUE_CONSTRAINT_NAME )
          GROUP BY
                RF.CONSTRAINT_CATALOG, RF.CONSTRAINT_SCHEMA, RF.CONSTRAINT_NAME )
        AS R ( FK_DEGREE, PK_DEGREE )
    WHERE FK_DEGREE <> PK_DEGREE ) )
```

6.3 KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 assertion**6.3 KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 assertion****Function**

The assertion `KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1` ensures that every unique or primary key constraint has at least one unique column and that every referential constraint has at least one referencing column.

Definition

```
CREATE ASSERTION KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1
CHECK
( NOT EXISTS
  ( SELECT *
    FROM TABLE_CONSTRAINTS
      FULL OUTER JOIN
        KEY_COLUMN_USAGE
      USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
    WHERE COLUMN_NAME IS NULL
      AND
        CONSTRAINT_TYPE IN
          ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) ) )
```

6.4 UNIQUE_CONSTRAINT_NAME assertion

Function

The UNIQUE_CONSTRAINT_NAME assertion ensures that the same combination of <schema name> and <constraint name> is not used by more than one constraint.

NOTE 7 — The UNIQUE_CONSTRAINT_NAME assertion avoids the need for separate checks on DOMAINS, TABLE_CONSTRAINTS, and ASSERTIONS.

Definition

```
CREATE ASSERTION UNIQUE_CONSTRAINT_NAME
CHECK ( 1 =
  ( SELECT MAX ( OCCURRENCES )
    FROM ( SELECT COUNT (*) AS OCCURRENCES
      FROM ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
        FROM DOMAIN_CONSTRAINTS
        UNION ALL
        SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
        FROM TABLE_CONSTRAINTS
        UNION ALL
        SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
        FROM ASSERTIONS )
      GROUP BY
        CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) ) )
```

6.5 ASSERTIONS base table

Function

The ASSERTIONS table has one row for each assertion. It effectively contains a representation of the assertion descriptors.

Definition

```
CREATE TABLE ASSERTIONS
(
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  IS_DEFERRABLE           INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ASSERTIONS_IS_DEFERRABLE_NOT_NULL
    NOT NULL,
  INITIALLY_DEFERRED      INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ASSERTIONS_INITIALLY_DEFERRED_NOT_NULL
    NOT NULL,
  CONSTRAINT ASSERTIONS_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),
  CONSTRAINT ASSERTIONS_FOREIGN_KEY_CHECK_CONSTRAINTS
    FOREIGN KEY (CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
    REFERENCES CHECK_CONSTRAINTS,
  CONSTRAINT ASSERTIONS_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
    REFERENCES SCHEMATA,
  CONSTRAINT ASSERTIONS_DEFERRED_CHECK
    CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED ) IN
      ( VALUES ( 'NO', 'NO' ),
        ( 'YES', 'NO' ),
        ( 'YES', 'YES' ) ) )
)
```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the assertion being described.
- 2) The values of IS_DEFERRABLE have the following meanings:

YES	The assertion is deferrable.
NO	The assertion is not deferrable.

ISO/IEC 9075-11:2003 (E)
6.5 ASSERTIONS base table

3) The values of INITIALLY_DEFERRED have the following meanings:

YES	The assertion is initially deferred.
NO	The assertion is initially immediate.

6.6 ATTRIBUTES base table

Function

The ATTRIBUTES base table contains one row for each attribute. It effectively contains a representation of the attribute descriptors.

Definition

```
CREATE TABLE ATTRIBUTES (
    UDT_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    UDT_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    UDT_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ATTRIBUTE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION           INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT ORDINAL_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT ATTRIBUTES_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT ATTRIBUTES_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                           FROM ATTRIBUTES
                           GROUP BY UDT_CATALOG, UDT_SCHEMA, UDT_NAME ) ),
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ATTRIBUTE_DEFAULT          INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_NULLABLE                INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ATTRIBUTES_IS_NULLABLE_NOT_NULL
        NOT NULL
    CONSTRAINT ATTRIBUTES_IS_NULLABLE_CHECK
        CHECK ( IS_NULLABLE IN ( 'YES', 'NO' ) ),
    IS_DERIVED_REFERENCE_ATTRIBUTE INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_NOT_NULL
        NOT NULL
    CONSTRAINT ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_CHECK
        CHECK ( IS_DERIVED_REFERENCE_ATTRIBUTE IN ( 'YES', 'NO' ) ),
    CONSTRAINT ATTRIBUTES_PRIMARY_KEY
        PRIMARY KEY ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME, ATTRIBUTE_NAME ),
    CONSTRAINT ATTRIBUTES_UNIQUE
        UNIQUE ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME, ORDINAL_POSITION ),
    CONSTRAINT ATTRIBUTES_CHECK_DATA_TYPE
        CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
                  'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
                ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                      OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR ) ),
    CONSTRAINT ATTRIBUTES_UDT_IS_STRUCTURED_CHECK
        CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME ) IN
                ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME
                  FROM USER_DEFINED_TYPES
                  WHERE USER_DEFINED_TYPE_CATEGORY = 'STRUCTURED' ) )
```

)

Description

- 1) The values of UDT_CATALOG, UDT_SCHEMA, and UDT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type containing the attribute being described.
- 2) The value of ATTRIBUTE_NAME is the name of the attribute being described.
- 3) The values of UDT_CATALOG, UDT_SCHEMA, UDT_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the attribute.
- 4) The value of ORDINAL_POSITION is the ordinal position of the attribute in the user-defined type.
- 5) The value of ATTRIBUTE_DEFAULT is null if the attribute being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of ATTRIBUTE_DEFAULT is “TRUNCATED”. Otherwise, the value of ATTRIBUTE_DEFAULT is a character representation of the default value for the column that obeys the rules specified for <default option> in Subclause 11.5, “<default clause>”.

NOTE 8 — “TRUNCATED” is different from other values like CURRENT_USER or CURRENT_TIMESTAMP in that it is not an SQL <key word> and does not correspond to a defined value in SQL.

- 6) The values of IS_NULLABLE have the following meanings:

YES	The attribute is possibly nullable.
NO	The attribute is known not nullable.

6.7 AUTHORIZATIONS base table

Function

The AUTHORIZATIONS table has one row for each <role name> and one row for each <authorization identifier> referenced in the Information Schema. These are the <role name>s and <authorization identifier>s that may grant privileges as well as those that may create a schema, or currently own a schema created through a <schema definition>.

Definition

```
CREATE TABLE AUTHORIZATIONS (  
  AUTHORIZATION_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  AUTHORIZATION_TYPE      INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT AUTHORIZATIONS_AUTHORIZATION_TYPE_NOT_NULL  
      NOT NULL  
    CONSTRAINT AUTHORIZATIONS_AUTHORIZATION_TYPE_CHECK  
      CHECK ( AUTHORIZATION_TYPE IN ( 'USER', 'ROLE' ) ),  
  CONSTRAINT AUTHORIZATIONS_PRIMARY_KEY  
    PRIMARY KEY (AUTHORIZATION_NAME)  
)
```

Description

- 1) The values of AUTHORIZATION_TYPE have the following meanings:

USER	The value of AUTHORIZATION_NAME is a known <user identifier>.
ROLE	The value of AUTHORIZATION_NAME is a <role name> defined by a <role definition>.

6.8 CATALOG_NAMES base table

Function

The CATALOG_NAMES table identifies the catalog or catalogs that are described by this Definition Schema.

Definition

```
CREATE TABLE CATALOG_NAMES (  
    CATALOG_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CONSTRAINT CATALOG_NAMES_PRIMARY_KEY  
        PRIMARY KEY ( CATALOG_NAME )  
)
```

Description

- 1) The values of CATALOG_NAME are the names of the catalogs that are described by this Definition Schema.

6.9 CHARACTER_ENCODING_FORMS base table

Function

The CHARACTER_ENCODING_FORMS table has one row for each character encoding form descriptor.

Definition

```
CREATE TABLE CHARACTER_ENCODING_FORMS (
  CHARACTER_REPERTOIRE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT CHARACTER_ENCODING_FORMS_CHARACTER_REPERTOIRE_NAME_NOT_NULL
    NOT NULL,
  CHARACTER_ENCODING_FORM_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT CHARACTER_ENCODING_FORMS_CHARACTER_ENCODING_FORM_NAME_NOT_NULL
    NOT NULL,
  CONSTRAINT CHARACTER_ENCODING_FORMS_PRIMARY_KEY
    PRIMARY KEY ( CHARACTER_ENCODING_FORM_NAME, CHARACTER_REPERTOIRE_NAME ),
  CONSTRAINT CHARACTER_ENCODING_FORMS_FOREIGN_KEY_CHARACTER_REPERTOIRES
    FOREIGN KEY ( CHARACTER_REPERTOIRE_NAME )
      REFERENCES CHARACTER_REPERTOIRES
)
```

Description

- 1) The value of CHARACTER_ENCODING_FORM_NAME is the name of the character encoding form being described.
- 2) The value of CHARACTER_REPERTOIRE_NAME is the name of the character repertoire to which this character encoding form applies.
- 3) There is one row in this table for every character encoding form supported by the SQL-implementation.

6.10 CHARACTER_REPERTOIRES base table

Function

The CHARACTER_REPERTOIRES table has one row for each character repertoire descriptor.

Definition

```
CREATE TABLE CHARACTER_REPERTOIRES (
  CHARACTER_REPERTOIRE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT CHARACTER_REPERTOIRE_NAME_NOT_NULL
      NOT NULL,
  DEFAULT_COLLATION_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_CATALOG_NOT_NULL
      NOT NULL,
  DEFAULT_COLLATION_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_SCHEMA_NOT_NULL
      NOT NULL,
  DEFAULT_COLLATION_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_NAME_NOT_NULL
      NOT NULL,
  CONSTRAINT CHARACTER_REPERTOIRES_PRIMARY_KEY
    PRIMARY KEY ( CHARACTER_REPERTOIRE_NAME ),
  CONSTRAINT CHARACTER_REPERTOIRES_FOREIGN_KEY_COLLATIONS
    FOREIGN KEY ( DEFAULT_COLLATION_CATALOG, DEFAULT_COLLATION_SCHEMA,
                  DEFAULT_COLLATION_NAME )
      REFERENCES COLLATIONS
)
```

Description

- 1) The value of DEFAULT_COLLATION_CATALOG, DEFAULT_COLLATION_SCHEMA, and DEFAULT_COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the default collation of the character repertoire being described.
- 2) The value of CHARACTER_REPERTOIRE_NAME is the name of the character repertoire being described.
- 3) There is one row in this table for every character repertoire supported by the SQL-implementation.

6.11 CHARACTER_SETS base table

Function

The CHARACTER_SETS table has one row for each character set descriptor.

Definition

```
CREATE TABLE CHARACTER_SETS (
  CHARACTER_SET_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_REPERTOIRE       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FORM_OF_USE                INFORMATION_SCHEMA.SQL_IDENTIFIER,
  NUMBER_OF_CHARACTERS       INFORMATION_SCHEMA.CARDINAL_NUMBER,
  DEFAULT_COLLATE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_CATALOG_NOT_NULL
    NOT NULL,
  DEFAULT_COLLATE_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_SCHEMA_NOT_NULL
    NOT NULL,
  DEFAULT_COLLATE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_NAME_NOT_NULL
    NOT NULL,
  CONSTRAINT CHARACTER_SETS_PRIMARY_KEY
    PRIMARY KEY ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ),
  CONSTRAINT CHARACTER_SETS_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA )
      REFERENCES SCHEMATA,
  CONSTRAINT CHARACTER_SETS_FOREIGN_KEY_CHARACTER_ENCODING_FORMS
    FOREIGN KEY (FORM_OF_USE, CHARACTER_REPERTOIRE )
      REFERENCES CHARACTER_ENCODING_FORMS,
  CONSTRAINT CHARACTER_SETS_CHECK_REFERENCES_COLLATIONS
    CHECK ( DEFAULT_COLLATE_CATALOG NOT IN
      ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      ( DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA,
        DEFAULT_COLLATE_NAME ) IN
      ( SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
        FROM COLLATIONS ) )
)
```

Description

- 1) The values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set being described.

6.11 CHARACTER_SETS base table

- 2) The value of CHARACTER_REPERTOIRE is the name of the character repertoire of the character set being described.
- 3) The value of FORM_OF_USE is the name of the character encoding form used by the character set being described.
- 4) The value of NUMBER_OF_CHARACTERS is the null value.
- 5) The values of DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the explicit or implicit default collation for the character set.
- 6) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_TEXT. In that row:
 - a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.
 - b) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.
- 7) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_IDENTIFIER. In that row:
 - a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.
 - b) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.
- 8) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_CHARACTER. In that row:
 - a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.
 - b) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

6.12 CHECK_COLUMN_USAGE base table

Function

The CHECK_COLUMN_USAGE table has one row for each column identified by a <column reference> contained in the <search condition> of a check constraint, domain constraint, or assertion.

Definition

```
CREATE TABLE CHECK_COLUMN_USAGE (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT CHECK_COLUMN_USAGE_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),
  CONSTRAINT CHECK_COLUMN_USAGE_FOREIGN_KEY_CHECK_CONSTRAINTS
    FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
      REFERENCES CHECK_CONSTRAINTS,
  CONSTRAINT CHECK_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
    CHECK ( TABLE_CATALOG NOT IN
      ( SELECT CATALOG_NAME FROM SCHEMATA )
      OR
      ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
      ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
        FROM COLUMNS ) )
)
```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.

6.13 CHECK_CONSTRAINT_ROUTINE_USAGE base table

Function

The CHECK_CONSTRAINT_ROUTINE_USAGE base table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in an <assertion definition>, a <domain constraint>, or a <table constraint definition>.

Definition

```
CREATE TABLE CHECK_CONSTRAINT_ROUTINE_USAGE (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT CHECK_CONSTRAINT_ROUTINE_USAGE_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                  SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),
  CONSTRAINT CHECK_CONSTRAINT_ROUTINE_USAGE_CHECK_REFERENCES_ROUTINES
    CHECK ( SPECIFIC_CATALOG NOT IN
      ( SELECT CATALOG_NAME
        FROM SCHEMATA )
    OR
      ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
      ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
        FROM ROUTINES ) ),
  CONSTRAINT CHECK_CONSTRAINT_ROUTINE_USAGE_FOREIGN_KEY_CHECK_CONSTRAINTS
    FOREIGN KEY (CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
      REFERENCES CHECK_CONSTRAINTS
)
```

Description

- 1) The CHECK_CONSTRAINT_ROUTINE_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in an <assertion definition> or in the <check constraint definition> contained in either a <domain constraint> or a <table constraint definition>.
- 2) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the assertion or check constraint being described.
- 3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

6.14 CHECK_CONSTRAINTS base table

Function

The CHECK_CONSTRAINTS table has one row for each domain constraint, table check constraint, and assertion.

Definition

```
CREATE TABLE CHECK_CONSTRAINTS (  
    CONSTRAINT_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CONSTRAINT_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CONSTRAINT_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CHECK_CLAUSE        INFORMATION_SCHEMA.CHARACTER_DATA,  
    CONSTRAINT CHECK_CONSTRAINTS_PRIMARY_KEY  
        PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),  
    CONSTRAINT CHECK_CONSTRAINTS_SOURCE_CHECK  
        CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN  
            ( SELECT *  
              FROM ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME  
                    FROM ASSERTIONS  
                    UNION  
                      SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME  
                    FROM TABLE_CONSTRAINTS  
                    UNION  
                      SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME  
                    FROM DOMAIN_CONSTRAINTS ) ) )  
)
```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) Case:
 - a) If the character representation of the <search condition> contained in the <check constraint definition>, <domain constraint definition>, or <assertion definition> that defined the check constraint being described can be represented without truncation, then the value of CHECK_CLAUSE is that character representation.
 - b) Otherwise, the value of CHECK_CLAUSE is the null value.

NOTE 9 — Any implicit column references that were contained in the <search condition> associated with a <check constraint definition> or an <assertion definition> are replaced by explicit column references in CHECK_CONSTRAINTS.

6.15 CHECK_TABLE_USAGE base table

Function

The CHECK_TABLE_USAGE table has one row for each table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of a check constraint, domain constraint, or assertion.

Definition

```
CREATE TABLE CHECK_TABLE_USAGE
(
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT CHECK_TABLE_USAGE_PRIMARY_KEY
  PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
               TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),
  CONSTRAINT CHECK_TABLE_USAGE_FOREIGN_KEY_CHECK_CONSTRAINTS
  FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
  REFERENCES CHECK_CONSTRAINTS,
  CONSTRAINT CHECK_TABLE_USAGE_CHECK_REFERENCES_TABLES
  CHECK ( TABLE_CATALOG NOT IN
        ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
        ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
          FROM TABLES ) )
)
```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of the constraint being described.

6.16 COLLATIONS base table

Function

The COLLATIONS table has one row for each character collation descriptor.

Definition

```
CREATE TABLE COLLATIONS (
  COLLATION_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PAD_ATTRIBUTE               INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT COLLATIONS_PAD_ATTRIBUTE_CHECK
  CHECK ( PAD_ATTRIBUTE IN
    ( 'NO PAD', 'PAD SPACE' ) ),
  COLLATION_TYPE              INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_DEFINITION        INFORMATION_SCHEMA.CHARACTER_DATA,
  COLLATION_DICTIONARY        INFORMATION_SCHEMA.CHARACTER_DATA,
  CHARACTER_REPERTOIRE_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT CHARACTER_REPERTOIRE_NAME_NOT_NULL
  NOT NULL,
  CONSTRAINT COLLATIONS_PRIMARY_KEY
  PRIMARY KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ),
  CONSTRAINT COLLATIONS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA )
  REFERENCES SCHEMATA
)
```

Description

- 1) The values of COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation being described.
- 2) The values of COLLATION_TYPE, COLLATION_DICTIONARY, and COLLATION_DEFINITION are the null value.
- 3) The values of PAD_ATTRIBUTE have the following meanings:

NO PAD	The collation being described has the NO PAD characteristic.
PAD SPACE	The collation being described has the PAD SPACE characteristic.

- 4) The value of CHARACTER_REPERTOIRE_NAME is the name of the character repertoire to which the collation being described is applicable.

6.17 COLLATION_CHARACTER_SET_APPLICABILITY base table

Function

The COLLATION_CHARACTER_SET_APPLICABILITY table has one row for each applicability of a collation to a character set.

Definition

```
CREATE TABLE COLLATION_CHARACTER_SET_APPLICABILITY (
  COLLATION_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT COLLATION_CHARACTER_SET_APPLICABILITY_PRIMARY_KEY
    PRIMARY KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
                  CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ),
  CONSTRAINT COLLATION_CHARACTER_SET_APPLICABILITY_FOREIGN_KEY_COLLATIONS
    FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
      REFERENCES COLLATIONS,
  CONSTRAINT COLLATION_CHARACTER_SET_APPLICABILITY_CHECK_REFERENCES_CHARACTER_SETS
    CHECK ( CHARACTER_SET_CATALOG NOT IN
      ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ) IN
      ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME
        FROM CHARACTER_SETS ) )
)
```

Description

- 1) The values of COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation whose applicability is being described.
- 2) The values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set to which the collation is applicable.

6.18 COLUMN_COLUMN_USAGE base table

Function

The COLUMN_COLUMN_USAGE table has one row for each case where a generated column depends on a base column.

Definition

```
CREATE TABLE COLUMN_COLUMN_USAGE (
    TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DEPENDENT_COLUMN     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT COLUMN_COLUMN_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      COLUMN_NAME, DEPENDENT_COLUMN ),
    CONSTRAINT COLUMN_COLUMN_USAGE_FOREIGN_KEY_COLUMNS_1
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      COLUMN_NAME ) REFERENCES COLUMNS,
    CONSTRAINT COLUMN_COLUMN_USAGE_FOREIGN_KEY_COLUMNS_2
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      DEPENDENT_COLUMN ) REFERENCES COLUMNS
)
```

Description

- 1) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME and DEPENDENT_COLUMN are the catalog name, unqualified schema name and qualified identifier, respectively, of a generated column *GC*.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME and COLUMN_NAME are the catalog name, unqualified schema name and qualified identifier, respectively, of a column on which *GC* depends.

6.19 COLUMN_PRIVILEGES base table

Function

The COLUMN_PRIVILEGES table has one row for each column privilege descriptor. It effectively contains a representation of the column privilege descriptors.

Definition

```
CREATE TABLE COLUMN_PRIVILEGES (
  GRANTOR          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG   INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PRIVILEGE_TYPE    INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT COLUMN_PRIVILEGE_TYPE_CHECK
    CHECK ( PRIVILEGE_TYPE IN
      ( 'SELECT', 'INSERT', 'UPDATE', 'REFERENCES' ) ),
  IS_GRANTABLE     INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT COLUMN_PRIVILEGE_IS_GRANTABLE_NOT_NULL
    NOT NULL
  CONSTRAINT COLUMN_PRIVILEGE_IS_GRANTABLE_CHECK
    CHECK ( IS_GRANTABLE IN ( 'YES', 'NO' ) ),
  CONSTRAINT COLUMN_PRIVILEGE_PRIMARY_KEY
    PRIMARY KEY ( GRANTOR, GRANTEE,
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      PRIVILEGE_TYPE, COLUMN_NAME ),
  CONSTRAINT COLUMN_PRIVILEGE_FOREIGN_KEY_COLUMNS
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
      REFERENCES COLUMNS,
  CONSTRAINT COLUMN_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
    FOREIGN KEY ( GRANTOR )
      REFERENCES AUTHORIZATIONS,
  CONSTRAINT COLUMN_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTEE
    FOREIGN KEY ( GRANTEE )
      REFERENCES AUTHORIZATIONS
)
```

Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted column privileges, on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME, to the user or role identified by the value of GRANTEE for the column privilege being described.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or “PUBLIC” to indicate all users, to whom the column privilege being described is granted.

- 3) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the column on which the privilege being described was granted.
- 4) The values of PRIVILEGE_TYPE have the following meanings:

SELECT	The user has SELECT privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
INSERT	The user has INSERT privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
UPDATE	The user has UPDATE privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
REFER- ENCE	The user has REFERENCES privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.

- 5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.20 COLUMNS base table

Function

The COLUMNS table has one row for each column. It effectively contains a representation of the column descriptors.

Definition

```
CREATE TABLE COLUMNS (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION        INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT COLUMNS_ORDINAL_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT COLUMNS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT COLUMNS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM COLUMNS
                          GROUP BY
                              TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) ),
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_DEFAULT         INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_NULLABLE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IS_NULLABLE_NOT_NULL
        NOT NULL
    CONSTRAINT COLUMNS_IS_NULLABLE_CHECK
        CHECK ( IS_NULLABLE IN ( 'YES', 'NO' ) ),
    IS_SELF_REFERENCING    INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IS_SELF_REFERENCING_NOT_NULL
        NOT NULL
    CONSTRAINT COLUMNS_IS_SELF_REFERENCING_CHECK
        CHECK ( IS_SELF_REFERENCING IN ( 'YES', 'NO' ) ),
    IS_IDENTITY            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IS_IDENTITY_NOT_NULL NOT NULL
    CONSTRAINT COLUMNS_IS_IDENTITY_CHECK
        CHECK ( IS_IDENTITY IN ( 'YES', 'NO' ) ),
    IDENTITY_GENERATION    INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_START         INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_INCREMENT     INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_MAXIMUM       INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_MINIMUM       INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_CYCLE         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IDENTITY_CYCLE_CHECK
        CHECK ( IDENTITY_CYCLE IN ( 'YES', 'NO' ) ),
```



```
IS_GENERATED          INFORMATION_SCHEMA.CHARACTER_DATA,
GENERATION_EXPRESSION INFORMATION_SCHEMA.CHARACTER_DATA,
IS_UPDATABLE          INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IS_UPDATABLE_NOT_NULL NOT NULL
    CONSTRAINT COLUMNS_IS_UPDATABLE_CHECK
        CHECK ( IS_UPDATABLE IN ( 'YES', 'NO' ) ),

CONSTRAINT COLUMNS_PRIMARY_KEY
    PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

CONSTRAINT COLUMNS_UNIQUE
    UNIQUE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, ORDINAL_POSITION ),

CONSTRAINT COLUMNS_FOREIGN_KEY_TABLES
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES TABLES,

CONSTRAINT COLUMNS_CHECK_REFERENCES_DOMAIN
    CHECK ( DOMAIN_CATALOG NOT IN
        ( SELECT CATALOG_NAME
          FROM SCHEMATA )
      OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IN
        ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
          FROM DOMAINS ) ),

CONSTRAINT COLUMNS_CHECK_IDENTITY_COMBINATIONS
    CHECK ( ( IS_IDENTITY = 'NO' ) =
        ( ( IDENTITY_GENERATION, IDENTITY_START, IDENTITY_INCREMENT,
            IDENTITY_MAXIMUM, IDENTITY_MINIMUM, IDENTITY_CYCLE ) IS NULL ) ),

CONSTRAINT COLUMNS_CHECK_DATA_TYPE
    CHECK ( DOMAIN_CATALOG NOT IN
        ( SELECT CATALOG_NAME FROM SCHEMATA )
      OR
        ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
          IS NOT NULL
        AND
          ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            'TABLE', DTD_IDENTIFIER ) NOT IN
          ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, DTD_IDENTIFIER
            FROM DATA_TYPE_DESCRIPTOR ) )
      OR
        ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
          IS NULL
        AND
          ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            'TABLE', DTD_IDENTIFIER ) IN
          ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, DTD_IDENTIFIER
            FROM DATA_TYPE_DESCRIPTOR ) ) )
    )
```

Description

- 1) Case:
 - a) If a column is described by a column descriptor included in a table descriptor, then the table descriptor and the column descriptor are associated with that column.
 - b) If a column is described by a column descriptor included in a view descriptor, then the view descriptor and the corresponding column descriptor of the table of the <query expression> are associated with that column.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table containing the column being described.
- 3) The value of COLUMN_NAME is the name of the column being described.
- 4) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the column.
- 5) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are null if the column being described is not defined using a <domain name>. Otherwise, the values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the domain used by the column being described.
- 6) The value of ORDINAL_POSITION is the ordinal position of the column in the table.
- 7) Let *DC* be the descriptor of the column being described. If *DC* includes a <default option>, then let *DO* be that <default option>. The value of COLUMN_DEFAULT is

Case:

- a) If *DC* does not include a <default option>, then the null value.
 - b) If CHARACTER_LENGTH(*DO*) > *ML*, then "TRUNCATED".
 - c) Otherwise, *DO*.
- 8) The values of IS_NULLABLE have the following meanings:

YES	The column is possibly nullable.
NO	The column is known not nullable.

- 9) The values of IS_SELF_REFERENCING have the following meanings:

YES	The column is a self-referencing column.
NO	The column is not a self-referencing column.

- 10) The values of IS_IDENTITY have the following meanings:

YES	The column is an identity column.
NO	The column is not an identity column.

11) The values of IDENTITY_GENERATION have the following meanings:

ALWAYS	The column is an identity column whose values are always generated.
BY DEFAULT	The column is an identity column whose values are generated by default.
<i>null</i>	The column is not an identity column.

- 12) The value of IDENTITY_START is null if the column is not an identity column; otherwise, it is a character representation of the start value of the column being described.
- 13) The value of IDENTITY_INCREMENT is null if the column is not an identity column; otherwise, it is a character representation of the increment of the column being described.
- 14) The value of IDENTITY_MAXIMUM is null if the column is not an identity column; otherwise, it is a character representation of the maximum value of the column being described.
- 15) The value of IDENTITY_MINIMUM is null if the column is not an identity column; otherwise, it is a character representation of the minimum value of the column being described.
- 16) The value of IDENTITY_CYCLE is null if the column is not an identity column; otherwise, it is either YES or NO. The values of IDENTITY_CYCLE have the following meanings:

YES	The cycle option of the column is CYCLE.
NO	The cycle option of the column is NO CYCLE.
<i>null</i>	The column is not an identity column.

17) The values of IS_GENERATED have the following meanings:

NEVER	The column is not a generated column.
ALWAYS	The column is generated and stored.

- 18) The value of GENERATION_EXPRESSION is the text of the <generation expression> specified in the <column definition> when the column identified by COLUMN_NAME is defined.
- 19) The values of IS_UPDATABLE have the following meanings:

YES	The column is updatable.
-----	--------------------------

NO	The column is not updatable.
----	------------------------------

6.21 DATA_TYPE_DESCRIPTOR base table

Function

The DATA_TYPE_DESCRIPTOR table has one row for each domain, one row for each column (in each table) and for each attribute (in each structured type) that is defined as having a data type rather than a domain, one row for each distinct type, one row for the result type of each SQL-invoked function, one row for the data type contained in the <result cast> of each SQL-invoked function that specifies a <result cast>, one row for each SQL parameter of each SQL-invoked routine, one row for the result type of each method specification, one row for each parameter of each method specification, one row for each structured type whose associated reference type has a user-defined representation, one row for each field in a row type, one row for each element type of a collection type, one row for each referenced type of a reference type, and one row for each maximal supertype of each field, element type, or referenced type. It effectively contains a representation of the data type descriptors.

Definition

```
CREATE TABLE DATA_TYPE_DESCRIPTOR (
    OBJECT_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME                    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE                    INFORMATION_SCHEMA.CHARACTER_DATA,
    CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_OBJECT_TYPE
        CHECK ( OBJECT_TYPE IN
            ( 'TABLE', 'DOMAIN', 'USER-DEFINED TYPE',
              'ROUTINE', 'SEQUENCE' ) ),
    DTD_IDENTIFIER                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DATA_TYPE                     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT DATA_TYPE_DESCRIPTOR_OBJECT_DATA_TYPE_NOT_NULL
        NOT NULL,
    CHARACTER_SET_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DATA_TYPE_DESCRIPTOR_CHARACTER_SET_CATALOG_NOT_NULL
        NOT NULL,
    CHARACTER_SET_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DATA_TYPE_DESCRIPTOR_CHARACTER_SET_SCHEMA_NOT_NULL
        NOT NULL,
    CHARACTER_SET_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DATA_TYPE_DESCRIPTOR_CHARACTER_SET_NAME_NOT_NULL
        NOT NULL,
    CHARACTER_MAXIMUM_LENGTH       INFORMATION_SCHEMA.CARDINAL_NUMBER,
    CHARACTER_OCTET_LENGTH        INFORMATION_SCHEMA.CARDINAL_NUMBER,
    COLLATION_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    NUMERIC_PRECISION              INFORMATION_SCHEMA.CARDINAL_NUMBER,
    NUMERIC_PRECISION_RADIX       INFORMATION_SCHEMA.CARDINAL_NUMBER,
    NUMERIC_SCALE                 INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DATETIME_PRECISION            INFORMATION_SCHEMA.CARDINAL_NUMBER,
    INTERVAL_TYPE                  INFORMATION_SCHEMA.CHARACTER_DATA,
    INTERVAL_PRECISION            INFORMATION_SCHEMA.CARDINAL_NUMBER,
    USER_DEFINED_TYPE_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER,
```

6.21 DATA_TYPE_DESCRIPTOR base table

USER_DEFINED_TYPE_SCHEMA	INFORMATION_SCHEMA.SQL_IDENTIFIER,
USER_DEFINED_TYPE_NAME	INFORMATION_SCHEMA.SQL_IDENTIFIER,
SCOPE_CATALOG	INFORMATION_SCHEMA.SQL_IDENTIFIER,
SCOPE_SCHEMA	INFORMATION_SCHEMA.SQL_IDENTIFIER,
SCOPE_NAME	INFORMATION_SCHEMA.SQL_IDENTIFIER,
MAXIMUM_CARDINALITY	INFORMATION_SCHEMA.CARDINAL_NUMBER,


```

CONSTRAINT DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS
  CHECK ( ( DATA_TYPE IN
            ( 'CHARACTER', 'CHARACTER VARYING', 'CHARACTER LARGE OBJECT' )
          AND
            ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
          AND
            ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
              NUMERIC_SCALE, DATETIME_PRECISION,
              USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME ) IS NULL
          AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
          AND
            ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
            MAXIMUM_CARDINALITY IS NULL )
    OR
    ( DATA_TYPE IN
      ( 'BINARY LARGE OBJECT' )
    AND
      ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH )
        IS NOT NULL
    AND
      ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
        NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE, DATETIME_PRECISION,
        USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
    AND
      ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
      MAXIMUM_CARDINALITY IS NULL )
    OR
    ( DATA_TYPE IN
      ( 'SMALLINT', 'INTEGER', 'BIGINT' )
    AND
      ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
        COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
      NUMERIC_PRECISION_RADIX IN
        ( 2, 10 )
    AND
      NUMERIC_PRECISION IS NOT NULL
    AND
      NUMERIC_SCALE = 0
    AND
      DATETIME_PRECISION IS NULL
  )

```

```
AND
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
  MAXIMUM_CARDINALITY IS NULL )
OR
  ( DATA_TYPE IN
    ( 'NUMERIC', 'DECIMAL' )
  AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
  AND
    NUMERIC_PRECISION_RADIX = 10
  AND
    ( NUMERIC_PRECISION, NUMERIC_SCALE ) IS NOT NULL
  AND
    DATETIME_PRECISION IS NULL
  AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
  AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
    MAXIMUM_CARDINALITY IS NULL )
OR
  ( DATA_TYPE IN
    ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
  AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
  AND
    NUMERIC_PRECISION IS NOT NULL
  AND
    NUMERIC_PRECISION_RADIX = 2
  AND
    NUMERIC_SCALE IS NULL
  AND
    DATETIME_PRECISION IS NULL
  AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
  AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
  AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
    MAXIMUM_CARDINALITY IS NULL )
OR
  ( DATA_TYPE IN
    ( 'DATE', 'TIME', 'TIMESTAMP',
      'TIME WITH TIME ZONE', 'TIMESTAMP WITH TIME ZONE' )
  AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
  AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NULL
```

```

AND
    NUMERIC_SCALE IS NULL
AND
    DATETIME_PRECISION IS NOT NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'INTERVAL'
    AND
        ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
        ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NULL
    AND
        NUMERIC_SCALE IS NULL
    AND
        DATETIME_PRECISION IS NOT NULL
    AND
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME ) IS NULL
    AND
        INTERVAL_TYPE IN
        ( 'YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'SECOND',
          'YEAR TO MONTH', 'DAY TO HOUR', 'DAY TO MINUTE',
          'DAY TO SECOND', 'HOUR TO MINUTE',
          'HOUR TO SECOND', 'MINUTE TO SECOND' )
    AND
        INTERVAL_PRECISION IS NOT NULL
    AND
        ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
        MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'BOOLEAN'
    AND
        ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
        ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NULL
    AND
        NUMERIC_SCALE IS NULL
    AND
        DATETIME_PRECISION IS NULL
    AND
        ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME ) IS NULL
    AND

```



```
( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE = 'USER-DEFINED'
AND
( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
  CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE, INTERVAL_PRECISION,
  SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
  USER_DEFINED_TYPE_NAME ) IS NOT NULL
AND
MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE = 'REF'
AND
( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
AND
( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION,
  INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
  USER_DEFINED_TYPE_NAME ) IS NOT NULL
AND
MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE = 'ARRAY'
AND
( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
  CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE, INTERVAL_PRECISION )
  IS NULL
AND
( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
  USER_DEFINED_TYPE_NAME ) IS NULL
AND
( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
MAXIMUM_CARDINALITY IS NOT NULL )
OR
( DATA_TYPE = 'MULTISET'
AND
( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
  NUMERIC_SCALE, DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
  CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE, INTERVAL_PRECISION )
  IS NULL
AND
( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
  USER_DEFINED_TYPE_NAME ) IS NULL
AND
( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
MAXIMUM_CARDINALITY IS NULL )
```

6.21 DATA_TYPE_DESCRIPTOR base table

```

OR
  ( DATA_TYPE = 'ROW'
    AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE, DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
        CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE, INTERVAL_PRECISION )
        IS NULL
      AND
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME ) IS NULL
      AND
        ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
      AND
        MAXIMUM_CARDINALITY IS NULL )
OR
  ( DATA_TYPE NOT IN
    ( 'CHARACTER', 'CHARACTER VARYING', 'CHARACTER LARGE OBJECT',
      'BINARY LARGE OBJECT',
      'NUMERIC', 'DECIMAL', 'SMALLINT', 'INTEGER', 'BIGINT',
      'FLOAT', 'REAL', 'DOUBLE PRECISION',
      'DATE', 'TIME', 'TIMESTAMP',
      'INTERVAL', 'BOOLEAN', 'USER-DEFINED',
      'REF', 'ROW', 'ARRAY', 'MULTISET' ) ) ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_UDT
  CHECK ( USER_DEFINED_TYPE_CATALOG <>
    ANY ( SELECT CATALOG_NAME
          FROM SCHEMATA )
    OR
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IN
        ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME
          FROM USER_DEFINED_TYPES ) ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_PRIMARY_KEY
  PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, DTD_IDENTIFIER ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_COLLATION_CHARACTER_SET_APPLICABILITY
  CHECK ( CHARACTER_SET_CATALOG NOT IN
    ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      COLLATION_CATALOG NOT IN
        ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
        COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IN
        ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
          FROM COLLATION_CHARACTER_SET_APPLICABILITY ) ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA )
    REFERENCES SCHEMATA

```

)

Description

- 1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, and OBJECT_NAME are the fully qualified name of the object (table, domain, SQL-invoked routine, user-defined type, or sequence generator) whose descriptor includes the data type descriptor, and OBJECT_TYPE is 'TABLE', 'DOMAIN', 'ROUTINE', 'USER-DEFINED TYPE', or 'SEQUENCE', as the case may be.
- 2) The value of DTD_IDENTIFIER is the implementation-dependent value that uniquely identifies the data type descriptor among all data type descriptors of the schema object identified by OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and OBJECT_TYPE.
- 3) The values of DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are, respectively, the data type, the maximum length in characters if it is a character string type, the maximum length in octets if it is a character string type, the qualified name of the character set and applicable collation, if any, if it is a character string type, the precision and radix of the precision if it is a numeric type, the scale if it is a numeric type, the fractional seconds precision if it is a datetime or interval type, the fully qualified name of the user-defined type or the referenced structured type if it is a reference type, and the fully qualified name of a referenceable table, if specified, of the data type being described.
- 4) If DATA_TYPE is 'INTERVAL', then the values of INTERVAL_TYPE are the value for <interval qualifier> (as specified in Table 27, “Codes used for <interval qualifier>s in Dynamic SQL”, in ISO/IEC 9075-2) for the data type being described; otherwise, INTERVAL_TYPE is the null value.
- 5) If DATA_TYPE is 'INTERVAL', then the values of INTERVAL_PRECISION are the interval leading field precision of the data type being described; otherwise, INTERVAL_PRECISION is the null value.
- 6) Case:
 - a) If DATA_TYPE is 'USER-DEFINED', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the user-defined type being described.
 - b) If the DATA_TYPE is 'REF', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the referenced structured type of the reference type being described.
 - c) Otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.
- 7) If DATA_TYPE is 'REF', then the values of SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are the qualified name of the referenceable table, if any; otherwise, the values of SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are the null value.

- 8) If DATA_TYPE is the name of some character string type and OBJECT_SCHEMA is 'INFORMATION_SCHEMA', then the values for CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are implementation-defined.
- 9) If DATA_TYPE is 'ARRAY', then the value of MAXIMUM_CARDINALITY is the maximum cardinality of the array type being described. Otherwise, the value of MAXIMUM_CARDINALITY is the null value.
- 10) If DATA_TYPE is 'ROW' then the data type being described is a row type.

6.22 DIRECT_SUPERTABLES base table

Function

The DIRECT_SUPERTABLES base table contains one row for each direct subtable-supertable relationship.

Definition

```
CREATE TABLE DIRECT_SUPERTABLES (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT DIRECT_SUPERTABLES_PRIMARY_KEY
        PRIMARY KEY (TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, SUPERTABLE_NAME ),

    CONSTRAINT DIRECT_SUPERTABLES_FOREIGN_KEY_TABLE_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT DIRECT_SUPERTABLES_FOREIGN_KEY_SUPERTABLE_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, SUPERTABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT DIRECT_SUPERTABLES_CHECK_NOT_SAME_TABLES
        CHECK ( TABLE_NAME <> SUPERTABLE_NAME ),

    CONSTRAINT DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY
        CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA,
                    SUPERTABLE_NAME, TABLE_NAME ) NOT IN
                ( SELECT TABLE_CATALOG, TABLE_SCHEMA,
                      TABLE_NAME, SUPERTABLE_NAME
                    FROM DIRECT_SUPERTABLES ) ),

    CONSTRAINT DIRECT_SUPERTABLES_CHECK_NOT_ALSO_INDIRECT
        CHECK (
            NOT EXISTS (
                WITH RECURSIVE SUPER
                ( TYPE, SUBTABLE_CATALOG, SUBTABLE_SCHEMA, SUBTABLE_NAME,
                  SUPERTABLE_NAME )
                AS
                ( SELECT 0, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      SUPERTABLE_NAME
                  FROM DIRECT_SUPERTABLES
                UNION
                  SELECT 1, S.SUBTABLE_CATALOG, S.SUBTABLE_SCHEMA, S.SUBTABLE_NAME,
                        D.SUPERTABLE_NAME
                  FROM SUPER AS S
                JOIN
                  DIRECT_SUPERTABLES AS D
                ON ( S.SUBTABLE_CATALOG, S.SUBTABLE_SCHEMA, S.SUBTABLE_NAME )
```

6.22 DIRECT_SUPERTABLES base table

```
        = ( D.TABLE_CATALOG, D.TABLE_SCHEMA, D.TABLE_NAME ) )  
SELECT SUBTABLE_CATALOG, SUBTABLE_SCHEMA, SUBTABLE_NAME, SUPERTABLE_NAME  
FROM SUPER  
WHERE TYPE = 1  
INTERSECT  
SELECT *  
FROM DIRECT_SUPERTABLES ) )  
)
```

Description

- 1) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the fully qualified name of the subtable.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, and SUPERTABLE_NAME are the fully qualified name of the direct supertable.

6.23 DIRECT_SUPERTYPES base table

Function

The DIRECT_SUPERTYPES base table contains one row for each direct subtype-supertype relationship.

Definition

```
CREATE TABLE DIRECT_SUPERTYPES (
    USER_DEFINED_TYPE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTYPE_CATALOG             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTYPE_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTYPE_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT DIRECT_SUPERTYPES_PRIMARY_KEY
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME,
                      SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ),

    CONSTRAINT DIRECT_SUPERTYPES_FOREIGN_KEY_USER_DEFINED_TYPES_1
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT DIRECT_SUPERTYPES_FOREIGN_KEY_USER_DEFINED_TYPES_2
        FOREIGN KEY ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME )
        REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT DIRECT_SUPERTYPES_CHECK_NOT_SAME_TYPES
        CHECK ( ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME ) <>
              ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ) ),

    CONSTRAINT DIRECT_SUPERTYPES_CHECK_NO_REFLEXIVITY
        CHECK ( ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME,
                  USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME ) NOT IN
              ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME,
                      SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA,
                      SUPERTYPE_NAME
                FROM DIRECT_SUPERTYPES ) ),

    CONSTRAINT DIRECT_SUPERTYPES_CHECK_NOT_ALSO_INDIRECT
        CHECK (
            NOT EXISTS (
                WITH RECURSIVE SUPER
                ( TYPE, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME,
                  SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ) AS
```

6.23 DIRECT_SUPERTYPES base table

```

( SELECT 0, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
    USER_DEFINED_TYPE_NAME,
    SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
  FROM DIRECT_SUPERTYPES
 UNION
  SELECT 1, S.USER_DEFINED_TYPE_CATALOG, S.USER_DEFINED_TYPE_SCHEMA,
    S.USER_DEFINED_TYPE_NAME,
    D.SUPERTYPE_CATALOG, D.SUPERTYPE_SCHEMA, D.SUPERTYPE_NAME
  FROM SUPER AS S
    JOIN
      DIRECT_TPERTYPES AS D
    ON ( D.USER_DEFINED_TYPE_CATALOG, D.USER_DEFINED_TYPE_SCHEMA,
        D.USER_DEFINED_TYPE_NAME ) =
      ( S.SUPERTYPE_CATALOG, S.SUPERTYPE_SCHEMA,
        S.SUPERTYPE_NAME ) )
 SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
    USER_DEFINED_TYPE_NAME,
    SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
  FROM SUPER
 WHERE TYPE = 1
 INTERSECT
  SELECT *
  FROM DIRECT_SUPERTYPES ) )

```

Description

- 1) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the fully qualified name of the user-defined type that is a direct subtype.
- 2) The values of SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, and SUPERTYPE_NAME are the fully qualified name of the user-defined type that is the direct supertype.

6.24 DOMAIN_CONSTRAINTS base table

Function

The DOMAIN_CONSTRAINTS table has one row for each domain constraint associated with a domain. It effectively contains a representation of the domain constraint descriptors.

Definition

```
CREATE TABLE DOMAIN_CONSTRAINTS (  
    CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    DOMAIN_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT DOMAIN_CATALOG_NOT_NULL  
            NOT NULL,  
    DOMAIN_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT DOMAIN_SCHEMA_NOT_NULL  
            NOT NULL,  
    DOMAIN_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT DOMAIN_NAME_NOT_NULL  
            NOT NULL,  
    IS_DEFERRABLE          INFORMATION_SCHEMA.CHARACTER_DATA  
        CONSTRAINT DOMAIN_CONSTRAINTS_DEFERRABLE_NOT_NULL  
            NOT NULL,  
    INITIALLY_DEFERRED     INFORMATION_SCHEMA.CHARACTER_DATA  
        CONSTRAINT DOMAIN_CONSTRAINTS_INITIALLY_DEFERRED_NOT_NULL  
            NOT NULL,  
  
    CONSTRAINT DOMAIN_CONSTRAINTS_PRIMARY_KEY  
        PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),  
  
    CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_SCHEMATA  
        FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )  
            REFERENCES SCHEMATA,  
  
    CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_CHECK_CONSTRAINTS  
        FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )  
            REFERENCES CHECK_CONSTRAINTS,  
  
    CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_DOMAINS  
        FOREIGN KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )  
            REFERENCES DOMAINS,  
  
    CONSTRAINT DOMAIN_CONSTRAINTS_CHECK_DEFERRABLE  
        CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED ) IN  
            ( VALUES ( 'NO', 'NO' ),  
              ( 'YES', 'NO' ),  
              ( 'YES', 'YES' ) ) ),  
  
    CONSTRAINT DOMAIN_CONSTRAINTS_CHECK_SCHEMA_IDENTITY
```

ISO/IEC 9075-11:2003 (E)
6.24 DOMAIN_CONSTRAINTS base table

```
CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
        = ( DOMAIN_CATALOG, DOMAIN_SCHEMA ) )
)
```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the fully qualified name of the domain constraint.
- 2) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA and DOMAIN_NAME are the fully qualified name of the domain in which the domain constraint is defined.
- 3) The values of IS_DEFERRABLE have the following meanings:

YES	The domain constraint is deferrable.
NO	The domain constraint is not deferrable.

- 4) The values of INITIALLY_DEFERRED have the following meanings:

YES	The domain constraint is initially deferred.
NO	The domain constraint is initially immediate.

6.25 DOMAINS base table

Function

The DOMAINS table has one row for each domain. It effectively contains a representation of the domain descriptors.

Definition

```
CREATE TABLE DOMAINS (
    DOMAIN_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_DEFAULT           INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT DOMAINS_PRIMARY_KEY
        PRIMARY KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ),

    CONSTRAINT DOMAINS_FOREIGN_KEY_SCHEMATA
        FOREIGN KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA ) REFERENCES SCHEMATA,

    CONSTRAINT DOMAIN_CHECK_DATA_TYPE
        CHECK ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
            'DOMAIN', DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR ) )
)
```

Description

- 1) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are the fully qualified name of the domain.
- 2) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, and DTD_IDENTIFIER are the values of DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the domain.
- 3) The value of DOMAIN_DEFAULT is null if the domain being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of DOMAIN_DEFAULT is “TRUNCATED”. Otherwise, the value of DOMAIN_DEFAULT is a character representation of the default value for the domain that obeys the rules specified for <default option> in [Subclause 11.5, “<default clause>”](#).

NOTE 10 — “TRUNCATED” is different from other values like CURRENT_USER or CURRENT_TIMESTAMP in that it is not an SQL <key word> and does not correspond to a defined value in SQL.

6.26 ELEMENT_TYPES base table

Function

The ELEMENT_TYPES table has one row for each collection type. It effectively contains a representation of the element descriptor of the collection type.

Definition

```
CREATE TABLE ELEMENT_TYPES (
    OBJECT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLECTION_TYPE_IDENTIFIER INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ELEMENT_TYPES_PRIMARY_KEY
        PRIMARY KEY (OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER ),

    CONSTRAINT ELEMENT_TYPES_CHECK_COLLECTION_TYPE
        CHECK (
            ( OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                  OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR
              WHERE DATA_TYPE IN ( 'ARRAY', 'MULTISET' ) ) ),

    CONSTRAINT ELEMENT_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT ELEMENT_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR
)
```

Description

- 1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and COLLECTION_TYPE_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the collection type whose element type is being described.

- 2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the element type of the collection type.
- 3) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROOT_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the root data type of the element type.

6.27 FIELDS base table

Function

The FIELDS table has one row for each field of each row type. It effectively contains a representation of the field descriptors.

Definition

```
CREATE TABLE FIELDS (
    OBJECT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROW_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FIELD_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION        INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT FIELDS_ORDINAL_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT FIELDS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT FIELDS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM FIELDS
                          GROUP BY OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                                OBJECT_TYPE, ROW_IDENTIFIER ) ),
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_NULLABLE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT FIELDS_IS_NULLABLE_NOT_NULL
        NOT NULL
    CONSTRAINT FIELDS_IS_NULLABLE_CHECK
        CHECK ( IS_NULLABLE IN ( 'YES', 'NO' ) ),

    CONSTRAINT FIELDS_PRIMARY_KEY
        PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                     OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME ),

    CONSTRAINT FIELDS_UNIQUE
        UNIQUE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, ROW_IDENTIFIER, ORDINAL_POSITION ),

    CONSTRAINT FIELDS_CHECK_ROW_TYPE
        CHECK (
            ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, ROW_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR
              WHERE DATA_TYPE = 'ROW' ) ),
```

```

CONSTRAINT FIELDS_REFERENCED_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
REFERENCES DATA_TYPE_DESCRIPTOR,

CONSTRAINT FIELDS_REFERENCED_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
REFERENCES DATA_TYPE_DESCRIPTOR
)

```

Description

- 1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROW_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the row type containing the field being described.
- 2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROOT_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the root data type of the field type.
- 3) The value of FIELD_NAME is the name of the field being described.
- 4) The value of ORDINAL_POSITION is the ordinal position of the field in the row type.
- 5) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the field being described.
- 6) The values of IS_NULLABLE have the following meanings:

YES	The field is possibly nullable.
NO	The field is known not nullable.

6.28 KEY_COLUMN_USAGE base table

Function

The KEY_COLUMN_USAGE table has one or more rows for each row in the TABLE_CONSTRAINTS table that has a CONSTRAINT_TYPE of “UNIQUE”, “PRIMARY KEY”, or “FOREIGN KEY”. The rows list the columns that constitute each unique constraint, and the referencing columns in each foreign key constraint.

Definition

```
CREATE TABLE KEY_COLUMN_USAGE (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT KEY_COLUMN_TABLE_CATALOG_NOT_NULL
    NOT NULL,
  TABLE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT KEY_COLUMN_TABLE_SCHEMA_NOT_NULL
    NOT NULL,
  TABLE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT KEY_COLUMN_TABLE_NAME_NOT_NULL
    NOT NULL,
  COLUMN_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
  ORDINAL_POSITION       INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT KEY_COLUMN_ORDINAL_POSITION_NOT_NULL
    NOT NULL
    CONSTRAINT KEY_COLUMN_USAGE_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
    CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT KEY_COLUMN_USAGE_ORDINAL_POSITION_CONTIGUOUS_CHECK
    CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                      FROM KEY_COLUMN_USAGE
                      GROUP BY CONSTRAINT_CATALOG,
                             CONSTRAINT_SCHEMA,
                             CONSTRAINT_NAME ) ),
  POSITION_IN_UNIQUE_CONSTRAINT INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT KEY_COLUMN_USAGE_POSITION_IN_UNIQUE_CONSTRAINT_GREATER_THAN_ZERO_CHECK
    CHECK ( POSITION_IN_UNIQUE_CONSTRAINT > 0 ),

  CONSTRAINT KEY_COLUMN_USAGE_UNIQUE_POSITION_IN_UNIQUE_CONSTRAINT
    UNIQUE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
             CONSTRAINT_NAME, POSITION_IN_UNIQUE_CONSTRAINT ),

  CONSTRAINT KEY_COLUMN_USAGE_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                 COLUMN_NAME ),

  CONSTRAINT KEY_COLUMN_USAGE_UNIQUE
    UNIQUE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
            CONSTRAINT_NAME, ORDINAL_POSITION ),
```



```
CONSTRAINT KEY_COLUMN_USAGE_FOREIGN_KEY_COLUMNS
  FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
    REFERENCES COLUMNS,

CONSTRAINT KEY_COLUMN_CONSTRAINT_TYPE_CHECK
CHECK (
  ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
  ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
    FROM TABLE_CONSTRAINTS
    WHERE CONSTRAINT_TYPE IN
      ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) ),

CONSTRAINT KEY_COLUMN_USAGE_POSITION_IN_UNIQUE_CONSTRAINT_CHECK
CHECK ( ( POSITION_IN_UNIQUE_CONSTRAINT IS NULL
  AND
    NOT ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
      ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
        FROM REFERENTIAL_CONSTRAINTS ) )
  OR
  ( POSITION_IN_UNIQUE_CONSTRAINT IS NOT NULL
  AND
    ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
    ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
      FROM REFERENTIAL_CONSTRAINTS )
  AND
    NOT EXISTS ( SELECT *
                  FROM KEY_COLUMN_USAGE
                  GROUP BY CONSTRAINT_CATALOG,
                           CONSTRAINT_SCHEMA,
                           CONSTRAINT_NAME
                  HAVING COUNT ( POSITION_IN_UNIQUE_CONSTRAINT )
                        <> MAX ( POSITION_IN_UNIQUE_CONSTRAINT ) ) ) )
)
```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier of the table name, and the column name of the column that participates in the unique, primary key, or foreign key constraint being described.
- 3) The value of ORDINAL_POSITION is the ordinal position of the specific column in the constraint being described. If the constraint described is a key of cardinality 1 (one), then the value of ORDINAL_POSITION is always 1 (one).
- 4) Case:
 - a) If the constraint being described is a foreign key constraint, then the value of POSITION_IN_UNIQUE_CONSTRAINT is the ordinal position of the referenced column corresponding to the referencing column being described, in the corresponding unique key constraint.

- b) Otherwise, the value of POSITION_IN_UNIQUE_CONSTRAINT is the null value.

6.29 METHOD_SPECIFICATION_PARAMETERS base table

Function

The METHOD_SPECIFICATION_PARAMETERS base table has one row for each SQL parameter of each method specification described in the METHOD_SPECIFICATIONS base table.

Definition

```
CREATE TABLE METHOD_SPECIFICATION_PARAMETERS (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION          INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT METHOD_SPECIFICATION_PARAMETER_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                           FROM METHOD_SPECIFICATION_PARAMETERS
                           GROUP BY SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) ),
    DTD_IDENTIFIER            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_MODE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATION_PARAMETER_MODE_CHECK
        CHECK ( PARAMETER_MODE IN
              ( 'IN' ) ),
    IS_RESULT                 INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT METHOD_SPECIFICATION_PARAMETER_IS_RESULT_CHECK
        CHECK ( IS_RESULT IN ( 'YES', 'NO' ) ),
    AS_LOCATOR                INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT METHOD_SPECIFICATION_PARAMETER_AS_LOCATOR_CHECK
        CHECK ( AS_LOCATOR IN ( 'YES', 'NO' ) ),
    PARAMETER_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      ORDINAL_POSITION ),

    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES METHOD_SPECIFICATIONS,

    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_CHECK_DATA_TYPE
        CHECK (
            ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
              'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
```

6.29 METHOD_SPECIFICATION_PARAMETERS base table

```

        OBJECT_TYPE, DTD_IDENTIFIER
FROM DATA_TYPE_DESCRIPTOR ) )
)

```

Description

- 1) The values of **SPECIFIC_CATALOG**, **SPECIFIC_SCHEMA**, and **SPECIFIC_NAME** are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked method whose parameters are being described.
- 2) The values of **USER_DEFINED_TYPE_CATALOG**, **USER_DEFINED_TYPE_SCHEMA**, and **USER_DEFINED_TYPE_NAME** are the fully qualified name of the user-defined type with which the SQL-invoked method is associated.
- 3) The value of **ORDINAL_POSITION** is the ordinal position of the SQL parameter in the SQL-invoked method.
- 4) The values of **PARAMETER_MODE** have the following meanings:

IN	The SQL parameter being described is an input parameter.
----	--

- 5) The values of **SPECIFIC_CATALOG**, **SPECIFIC_SCHEMA**, **SPECIFIC_NAME**, and **DTD_IDENTIFIER** are the values of **OBJECT_CATALOG**, **OBJECT_SCHEMA**, **OBJECT_NAME**, and **DTD_IDENTIFIER**, respectively, of the row in **DATA_TYPE_DESCRIPTOR** that describes the data type of the parameter being described.
- 6) The values of **IS_RESULT** have the following meanings:

YES	The parameter is the RESULT parameter of a type-preserving function.
NO	The parameter is not the RESULT parameter of a type-preserving function.

- 7) The values of **AS_LOCATOR** have the following meanings:

YES	The parameter is passed AS LOCATOR .
NO	The parameter is not passed AS LOCATOR .

- 8) Case:

- a) If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of **PARAMETER_NAME** is that <SQL parameter name>.
- b) Otherwise, the value of **PARAMETER_NAME** is the null value.

- 9) **FROM_SQL_SPECIFIC_CATALOG**, **FROM_SQL_SPECIFIC_SCHEMA**, and **FROM_SQL_SPECIFIC_NAME** are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the from-sql routine for the parameter being described.

6.30 METHOD_SPECIFICATIONS base table

Function

The METHOD_SPECIFICATIONS base table has one row for each method specification.

Definition

```
CREATE TABLE METHOD_SPECIFICATIONS (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    METHOD_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_STATIC                 INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATION_IS_STATIC_CHECK
    CHECK ( IS_STATIC IN ( 'YES', 'NO' ) ),
    IS_OVERRIDING             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_IS_OVERRIDING_NOT_NULL
    NOT NULL
    CONSTRAINT METHOD_SPECIFICATION_IS_OVERRIDING_CHECK
    CHECK ( IS_OVERRIDING IN ( 'YES', 'NO' ) ),
    IS_CONSTRUCTOR            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_IS_CONSTRUCTOR_NOT_NULL
    NOT NULL
    CONSTRAINT METHOD_SPECIFICATION_IS_CONSTRUCTOR_CHECK
    CHECK ( IS_CONSTRUCTOR IN ( 'YES', 'NO' ) ),
    METHOD_LANGUAGE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_LANGUAGE_CHECK
    CHECK ( METHOD_LANGUAGE IN
        ( 'SQL', 'ADA', 'C',
          'COBOL', 'FORTRAN', 'MUMPS', 'PASCAL', 'PLI' ) ),
    PARAMETER_STYLE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_PARAMETER_STYLE_CHECK
    CHECK ( PARAMETER_STYLE IN
        ( 'SQL', 'GENERAL' ) ),
    DTD_IDENTIFIER             INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_DETERMINISTIC           INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_IS_DETERMINISTIC_CHECK
    CHECK ( IS_DETERMINISTIC IN ( 'YES', 'NO' ) ),
    SQL_DATA_ACCESS            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_NOT_NULL
    NOT NULL
    CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
        'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
    IS_NULL_CALL               INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_IS_NULL_CALL_CHECK
    CHECK ( IS_NULL_CALL IN ( 'YES', 'NO' ) ),
    TO_SQL_SPECIFIC_CATALOG   INFORMATION_SCHEMA.SQL_IDENTIFIER,
```

6.30 METHOD_SPECIFICATIONS base table

```

TO_SQL_SPECIFIC_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
AS_LOCATOR                      INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_AS_LOCATOR_CHECK
        CHECK ( AS_LOCATOR IN ( 'YES', 'NO' ) ),
CREATED                        INFORMATION_SCHEMA.TIME_STAMP,
LAST_ALTERED                   INFORMATION_SCHEMA.TIME_STAMP,
RESULT_CAST_FROM_DTD_IDENTIFIER INFORMATION_SCHEMA.SQL_IDENTIFIER,
RESULT_CAST_AS_LOCATOR         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_RESULT_CAST_AS_LOCATOR_CHECK
        CHECK ( RESULT_CAST_AS_LOCATOR IN ( 'YES', 'NO' ) ),

CONSTRAINT METHOD_SPECIFICATIONS_PRIMARY_KEY
    PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

CONSTRAINT METHOD_SPECIFICATIONS_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
        REFERENCES SCHEMATA,

CONSTRAINT METHOD_SPECIFICATIONS_FOREIGN_KEY_USER_DEFINED_TYPES
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES MATCH FULL,

CONSTRAINT METHOD_SPECIFICATIONS_CHECK_DATA_TYPE
    CHECK (
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) ),

CONSTRAINT METHOD_SPECIFICATIONS_COMBINATIONS
    CHECK (
        ( ( METHOD_LANGUAGE = 'SQL'
          AND
            IS_DETERMINISTIC IS NULL )
        OR
          ( METHOD_LANGUAGE <> 'SQL'
          AND
            IS_DETERMINISTIC IS NOT NULL ) ) ),

CONSTRAINT METHOD_SPECIFICATIONS_IS_CONSTRUCTOR_COMBINATION_CHECK
    CHECK ( IS_CONSTRUCTOR = 'NO' OR IS_OVERRIDING = 'NO' ),

CONSTRAINT METHOD_SPECIFICATIONS_SAME_SCHEMA
    CHECK ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA ) ),
CONSTRAINT METHOD_SPECIFICATIONS_CHECK_RESULT_CAST
    CHECK ( ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NULL
        AND
          RESULT_CAST_AS_LOCATOR IS NULL )
        OR
        ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NOT NULL
        AND
          RESULT_CAST_AS_LOCATOR IS NOT NULL

```

```

AND
( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
  'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
  ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, RESULT_CAST_FROM_DTD_IDENTIFIER
    FROM DATA_TYPE_DESCRIPTOR )
  )
)
)

```

Description

- 1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked method being described.
- 2) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type name of the user-defined type with which the SQL-invoked method is associated.
- 3) The values of METHOD_NAME is the identifier of the method name of the SQL-invoked method being described.
- 4) The values of IS_STATIC have the following meanings:

YES	The SQL-invoked routine is a static method.
NO	The SQL-invoked routine is not a static method.

- 5) The values of IS_OVERRIDING have the following meanings:

YES	The SQL-invoked method is an overriding method.
NO	The SQL-invoked method is an original method.

- 6) The values of IS_CONSTRUCTOR have the following meanings:

YES	The SQL-invoked method is an SQL-invoked constructor method.
NO	The SQL-invoked method is not an SQL-invoked constructor method.

- 7) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the result type of the method.

6.30 METHOD_SPECIFICATIONS base table

8) The values of IS_NULL_CALL have the following meanings:

YES	The SQL-invoked routine is a null-call function.
NO	The SQL-invoked routine is not a null-call function.

9) The value of METHOD_LANGUAGE is the explicit or implicit <language name> contained in the method specification being described.

10) Case:

a) If the method being defined specifies LANGUAGE SQL, then the values of IS_DETERMINISTIC, PARAMETER_STYLE, TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPECIFIC_NAME are the null value.

b) Otherwise:

i) The values of IS_DETERMINISTIC have the following meanings:

YES	The method is deterministic.
NO	The method is possibly not deterministic.

ii) The values of PARAMETER_STYLE have the following meanings:

SQL	The method specification specified PARAMETER STYLE SQL.
GENERAL	The method specification specified PARAMETER STYLE GENERAL.

iii) TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPECIFIC_NAME are catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the result type of the SQL-invoked method being described.

11) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

12) The values of AS_LOCATOR have the following meanings:

YES	The return value is passed AS LOCATOR.
NO	The return value is not passed AS LOCATOR.

- 13) The value of **CREATED** is the value of **CURRENT_TIMESTAMP** at the time when the SQL-invoked method specification being described was created.
- 14) The value of **LAST_ALTERED** is the value of **CURRENT_TIMESTAMP** at the time that the SQL-invoked method specification being described was last altered. This value is identical to the value of **CREATED** for SQL-invoked routines that have never been altered.
- 15) Case:
- a) If the method specification descriptor of the SQL-invoked method being described does not include an indication that the SQL-invoked method specifies a <result cast>, then the values of **RESULT_CAST_FROM_DTD_IDENTIFIER** and **RESULT_CAST_AS_LOCATOR** are the null value.
 - b) Otherwise, **SPECIFIC_CATALOG**, **SPECIFIC_SCHEMA**, **SPECIFIC_NAME**, and **RESULT_CAST_FROM_DTD_IDENTIFIER** are the values of **OBJECT_CATALOG**, **OBJECT_SCHEMA**, **OBJECT_NAME**, and **DTD_IDENTIFIER**, respectively, of the row in **DATA_TYPE_DESCRIPTOR** that describes the <data type> specified in the <result cast> of the SQL-invoked method being described.

Case:

- i) If the method specification descriptor of the SQL-invoked method being described does not include an indication that the <data type> specified in the <result cast> has a locator indication, then the value of **RESULT_CAST_AS_LOCATOR** is 'NO'.
- ii) Otherwise, the value of **RESULT_CAST_AS_LOCATOR** is 'YES'.

6.31 PARAMETERS base table

Function

The PARAMETERS table has one row for each SQL parameter of each SQL-invoked routine described in the ROUTINES base table.

Definition

```
CREATE TABLE PARAMETERS (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION          INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT PARAMETERS_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                           FROM PARAMETERS
                           GROUP BY SPECIFIC_CATALOG,
                                   SPECIFIC_SCHEMA,
                                   SPECIFIC_NAME ) ),
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_MODE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PARAMETER_MODE_NOT_NULL
        NOT NULL
    CONSTRAINT PARAMETER_MODE_CHECK
        CHECK (
            PARAMETER_MODE IN
            ( 'IN', 'OUT', 'INOUT' ) ),
    IS_RESULT                   INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PARAMETERS_IS_RESULT_CHECK
        CHECK (
            IS_RESULT IN
            ( 'YES', 'NO' ) ),
    AS_LOCATOR                  INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PARAMETERS_AS_LOCATOR_CHECK
        CHECK (
            AS_LOCATOR IN
            ( 'YES', 'NO' ) ),
    PARAMETER_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_SCHEMA    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT PARAMETERS_PRIMARY_KEY
```

```

PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
              SPECIFIC_NAME, ORDINAL_POSITION ),

CONSTRAINT PARAMETERS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
    REFERENCES SCHEMATA,
CONSTRAINT PARAMETERS_CHECK_DATA_TYPE
  CHECK (
    ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
      SPECIFIC_NAME, 'ROUTINE', DTD_IDENTIFIER ) IN
    ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
          OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
      FROM DATA_TYPE_DESCRIPTOR ) )
)

```

Description

- 1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine that contains the SQL parameter being described.
- 2) The value of ORDINAL_POSITION is the ordinal position of the SQL parameter in the SQL-invoked routine.
- 3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the parameter.
- 4) The values of PARAMETER_MODE have the following meanings:

IN	The SQL parameter being described is an input parameter.
OUT	The SQL parameter being described is an output parameter.
INOUT	The SQL parameter being described is an input parameter and an output parameter.

- 5) The values of IS_RESULT have the following meanings:

YES	The parameter is the RESULT parameter of a type-preserving function.
NO	The parameter is not the RESULT parameter of a type-preserving function.

- 6) The values of AS_LOCATOR have the following meanings:

YES	The parameter is passed AS LOCATOR.
NO	The parameter is not passed AS LOCATOR.

7) Case:

- a) If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of `PARAMETER_NAME` is that <SQL parameter name>.
 - b) Otherwise, the value of `PARAMETER_NAME` is the null value.
- 8) `FROM_SQL_SPECIFIC_CATALOG`, `FROM_SQL_SPECIFIC_SCHEMA`, and `FROM_SQL_SPECIFIC_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the from-sql routine for the input parameter being described.
- 9) `TO_SQL_SPECIFIC_CATALOG`, `TO_SQL_SPECIFIC_SCHEMA`, and `TO_SQL_SPECIFIC_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the output parameter being described.

6.32 REFERENCED_TYPES base table

Function

The REFERENCED_TYPES table has one row for each reference type. It effectively contains a representation of the referenced type descriptors.

Definition

```
CREATE TABLE REFERENCED_TYPES (
    OBJECT_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_TYPE_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT REFERENCED_TYPES_PRIMARY_KEY
        PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                      OBJECT_NAME, OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER ),

    CONSTRAINT REFERENCED_TYPES_CHECK_REFERENCE_TYPE
        CHECK ( ( OBJECT_CATALOG, OBJECT_SCHEMA,
                  OBJECT_NAME, OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR
                WHERE DATA_TYPE = 'REF' ) ),

    CONSTRAINT REFERENCED_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                      OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT REFERENCED_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                      OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR
)
```

Description

- 1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and REFERENCE_TYPE_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the reference type whose referenced type is being described.

- 2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the referenced type of the reference type.
- 3) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROOT_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the root data type of the reference type.

6.33 REFERENTIAL_CONSTRAINTS base table

Function

The REFERENTIAL_CONSTRAINTS table has one row for each row in the TABLE_CONSTRAINTS table that has a CONSTRAINT_TYPE value of “FOREIGN KEY”.

Definition

```
CREATE TABLE REFERENTIAL_CONSTRAINTS (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  UNIQUE_CONSTRAINT_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT UNIQUE_CONSTRAINT_CATALOG_NOT_NULL
    NOT NULL,
  UNIQUE_CONSTRAINT_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT UNIQUE_CONSTRAINT_SCHEMA_NOT_NULL
    NOT NULL,
  UNIQUE_CONSTRAINT_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT UNIQUE_CONSTRAINT_NAME_NOT_NULL
    NOT NULL,
  MATCH_OPTION             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENTIAL_MATCH_OPTION_NOT_NULL
    NOT NULL
    CONSTRAINT REFERENTIAL_MATCH_OPTION_CHECK
    CHECK ( MATCH_OPTION IN
      ( 'NONE', 'PARTIAL', 'FULL' ) ),
  UPDATE_RULE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENTIAL_UPDATE_RULE_NOT_NULL
    NOT NULL
    CONSTRAINT REFERENTIAL_UPDATE_RULE_CHECK
    CHECK ( UPDATE_RULE IN
      ( 'CASCADE',
        'SET NULL',
        'SET DEFAULT',
        'RESTRICT',
        'NO ACTION' ) ),
  DELETE_RULE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENTIAL_DELETE_RULE_NOT_NULL
    NOT NULL
    CONSTRAINT REFERENTIAL_DELETE_RULE_CHECK
    CHECK ( DELETE_RULE IN
      ( 'CASCADE',
        'SET NULL',
        'SET DEFAULT',
        'RESTRICT',
        'NO ACTION' ) ),

  CONSTRAINT REFERENTIAL_CONSTRAINTS_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),
```

6.33 REFERENTIAL_CONSTRAINTS base table

```

CONSTRAINT REFERENTIAL_CONSTRAINTS_CONSTRAINT_TYPE_CHECK
CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
        ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM TABLE_CONSTRAINTS
          WHERE CONSTRAINT_TYPE = 'FOREIGN KEY' ) ),

CONSTRAINT UNIQUE_CONSTRAINT_CHECK_REFERENCES_UNIQUE_CONSTRAINT
CHECK ( UNIQUE_CONSTRAINT_CATALOG NOT IN
        ( SELECT CATALOG_NAME
          FROM SCHEMATA )
      OR
        ( ( UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA,
            UNIQUE_CONSTRAINT_NAME ) IN
          ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
            FROM TABLE_CONSTRAINTS
            WHERE CONSTRAINT_TYPE IN
              ( 'UNIQUE', 'PRIMARY KEY' ) ) ) )
)

```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA, and UNIQUE_CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the unique or primary key constraint applied to the referenced column list being described.
- 3) The values of MATCH_OPTION have the following meanings:

NONE	No <match type> was specified.
PARTIAL	A <match type> of PARTIAL was specified.
FULL	A <match type> of FULL was specified.

- 4) The values of UPDATE_RULE have the following meanings for a referential constraint that has an <update rule>:

NO ACTION	A <referential action> of NO ACTION was specified.
RESTRICT	A <referential action> of RESTRICT was specified.
CASCADE	A <referential action> of CASCADE was specified.
SET NULL	A <referential action> of SET NULL was specified.

SET DEFAULT	A <referential action> of SET DEFAULT was specified.
----------------	--

- 5) The values of DELETE_RULE have the following meanings for a referential constraint that has a <delete rule>:

NO ACTION	A <referential action> of NO ACTION was specified.
RESTRICT	A <referential action> of RESTRICT was specified.
CASCADE	A <referential action> of CASCADE was specified.
SET NULL	A <referential action> of SET NULL was specified.
SET DEFAULT	A <referential action> of SET DEFAULT was specified.

6.34 ROLE_AUTHORIZATION_DESCRIPTORs base table

Function

Contains a representation of the role authorization descriptors.

Definition

```
CREATE TABLE ROLE_AUTHORIZATION_DESCRIPTORs (
  ROLE_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTOR                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  IS_GRANTABLE             INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_IS_GRANTABLE_CHECK
    CHECK ( IS_GRANTABLE IN
      ( 'YES', 'NO' ) ),

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_PRIMARY_KEY
    PRIMARY KEY ( ROLE_NAME, GRANTEE ),

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_CHECK_ROLE_NAME
    CHECK ( ROLE_NAME IN
      ( SELECT AUTHORIZATION_NAME
        FROM AUTHORIZATIONS
        WHERE AUTHORIZATION_TYPE = 'ROLE' ) ),

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
    FOREIGN KEY ( GRANTOR )
      REFERENCES AUTHORIZATIONS,

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_FOREIGN_KEY_AUTHORIZATIONS_GRANTEE
    FOREIGN KEY ( GRANTEE )
      REFERENCES AUTHORIZATIONS
)
```

Description

- 1) The value of ROLE_NAME is the <role name> of some <role granted> by the <grant role statement> or the <role name> of a <role definition>.
- 2) The value of GRANTEE is an <authorization identifier>, possibly PUBLIC, or <role name> specified as a <grantee> contained in a <grant role statement>, or the <authorization identifier> of the current SQL-session when the <role definition> is executed.
- 3) The value of GRANTOR is the <authorization identifier> of the user or role who granted the role identified by ROLE_NAME to the user or role identified by the value of GRANTEE.
- 4) The values of IS_GRANTABLE have the following meanings:

6.34 ROLE_AUTHORIZATION_DESCRIPTOR base table

YES	The described role is grantable.
NO	The described role is not grantable.

A role is grantable if the WITH ADMIN OPTION is specified in the <grant role statement> or a <role definition> is executed.

6.35 ROUTINE_COLUMN_USAGE base table

Function

The ROUTINE_COLUMN_USAGE table has one row for each column identified in an SQL routine.

Definition

```
CREATE TABLE ROUTINE_COLUMN_USAGE (
    SPECIFIC_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_COLUMN_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

    CONSTRAINT ROUTINE_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
        CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
              OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
                FROM COLUMNS ) ),

    CONSTRAINT ROUTINE_COLUMN_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

Description

- 1) The ROUTINE_COLUMN_USAGE table has one row for each table identified by at least one of:
 - a) A <column reference> contained in a <search condition> contained in a <delete statement: searched> or an <update statement: searched> contained in the <SQL routine body> of an SQL-invoked routine.
 - b) A <column reference> contained in a <value expression> simply contained in an <update source> or an <assigned row> contained in the <SQL routine body> of an SQL-invoked routine.
 - c) A <column name> contained in an <insert column list> of an <insert statement> contained in the <SQL routine body> of an SQL-invoked routine.

- d) A <column name> is contained in an <object column> contained in either an <update statement: positioned> or an <update statement: searched> contained in the <SQL routine body> of an SQL-invoked routine.
- 2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine being described.
- 3) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and identifier respectively, of a column that is referenced in the SQL-invoked routine being described.

6.36 ROUTINE_PRIVILEGES base table

Function

The ROUTINE_PRIVILEGES table has one row for each execute privilege descriptor for an SQL-invoked routine. It effectively contains a representation of the execute privilege descriptors.

Definition

```
CREATE TABLE ROUTINE_PRIVILEGES (
    GRANTOR                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE          INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ROUTINE_PRIVILEGES_TYPE_CHECK
        CHECK ( PRIVILEGE_TYPE IN
            ( 'EXECUTE' ) ),
    IS_GRANTABLE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ROUTINE_PRIVILEGES_GRANTABLE_NOT_NULL
        NOT NULL
    CONSTRAINT ROUTINE_PRIVILEGES_GRANTABLE_CHECK
        CHECK ( IS_GRANTABLE IN
            ( 'YES', 'NO' ) ),

    CONSTRAINT ROUTINE_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
            SPECIFIC_NAME, PRIVILEGE_TYPE ),

    CONSTRAINT ROUTINE_PRIVILEGES_FOREIGN_KEY_TABLES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES,

    CONSTRAINT ROUTINE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
        FOREIGN KEY ( GRANTOR )
        REFERENCES AUTHORIZATIONS,

    CONSTRAINT ROUTINE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS GRANTEE
        FOREIGN KEY ( GRANTEE )
        REFERENCES AUTHORIZATIONS
)
```

Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted execute privileges, on the SQL-invoked routine identified by SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, to the user or role identified by the value of GRANTEE for the privilege being described.

- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or “PUBLIC” to indicate all users, to whom the privilege being described is granted.
- 3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine on which the privilege being described has been granted.
- 4) The values of PRIVILEGE_TYPE have the following meanings:

EXECUTE	The user has EXECUTE privilege on the SQL-invoked routine identified by SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME.
---------	---

- 5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.37 ROUTINE_ROUTINE_USAGE base table

Function

The ROUTINE_ROUTINE_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in an SQL routine.

Definition

```
CREATE TABLE ROUTINE_ROUTINE_USAGE (
    SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_ROUTINE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                     ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME ),

    CONSTRAINT ROUTINE_ROUTINE_USAGE_CHECK_REFERENCES_ROUTINES
        CHECK ( ROUTINE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME ) IN
              ( SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME
                FROM ROUTINES ) ),

    CONSTRAINT ROUTINE_ROUTINE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

Description

- 1) The ROUTINE_ROUTINE_USAGE table has one row for each SQL-invoked routine *R1* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <SQL routine body> of an SQL routine *R2*.
- 2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R2*.
- 3) The values of ROUTINE_CATALOG, ROUTINE_SCHEMA, and ROUTINE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R1*.

6.38 ROUTINE_SEQUENCE_USAGE base table

Function

The ROUTINE_SEQUENCE_USAGE table has one row for each external sequence generator identified in an SQL routine.

Definition

```
CREATE TABLE ROUTINE_SEQUENCE_USAGE (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                     SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ),

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_CHECK_REFERENCES_SEQUENCES
        CHECK ( SEQUENCE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ) IN
              ( SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
                FROM SEQUENCES ) ),

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

Description

- 1) The ROUTINE_SEQUENCE_USAGE table has one row for each sequence generator *SEQ* identified by a <sequence generator name> contained in the <SQL routine body> of an SQL-invoked routine *R* being described.
- 2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.
- 3) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, and SEQUENCE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SEQ*.

6.39 ROUTINE_TABLE_USAGE base table

Function

The ROUTINE_TABLE_USAGE table has one row for each table identified in an SQL routine.

Definition

```
CREATE TABLE ROUTINE_TABLE_USAGE (
    SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_TABLE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT ROUTINE_TABLE_USAGE_CHECK_REFERENCES_TABLES
        CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
        OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM TABLES ) ),

    CONSTRAINT ROUTINE_TABLE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

Description

- 1) The ROUTINE_TABLE_USAGE table has one row for each table identified by at least one of:
 - a) A <table reference> contained in a <query expression> simply contained in a <cursor specification> or an <insert statement> contained in the <SQL routine body> of an SQL-invoked routine.
 - b) A <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <SQL routine body> of an SQL-invoked routine.
 - c) A <table reference> contained in a <search condition> contained in a <delete statement: searched> or an <update statement: searched> contained in the <SQL routine body> of an SQL-invoked routine.
 - d) A <table reference> contained in a <value expression> simply contained in an <update source> or an <assigned row> contained in the <SQL routine body> of an SQL-invoked routine.

- e) A <table name> contained in either a <delete statement: positioned> or a <delete statement: searched> contained in the <SQL routine body> of an SQL-invoked routine.
 - f) A <table name> immediately contained in an <insert statement> that does not contain an <insert column list> contained in the <SQL routine body> of an SQL-invoked routine.
- 2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine being described.
 - 3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table that is referenced in the SQL-invoked routine being described.

6.40 ROUTINES base table

Function

The ROUTINES base table has one row for each SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINES (  
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SPECIFIC_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SPECIFIC_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    ROUTINE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    ROUTINE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    ROUTINE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    MODULE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    MODULE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    MODULE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    USER_DEFINED_TYPE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    USER_DEFINED_TYPE_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    ROUTINE_TYPE              INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT ROUTINE_TYPE_NOT_NULL  
        NOT NULL  
    CONSTRAINT ROUTINE_TYPE_CHECK  
        CHECK ( ROUTINE_TYPE IN  
            ( 'PROCEDURE', 'FUNCTION',  
              'INSTANCE METHOD', 'STATIC METHOD', 'CONSTRUCTOR METHOD' ) ),  
    DTD_IDENTIFIER            INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    ROUTINE_BODY              INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT ROUTINE_BODY_NOT_NULL  
        NOT NULL  
    CONSTRAINT ROUTINE_BODY_CHECK  
        CHECK ( ROUTINE_BODY IN  
            ( 'SQL', 'EXTERNAL' ) ),  
    ROUTINE_DEFINITION        INFORMATION_SCHEMA.CHARACTER_DATA,  
    EXTERNAL_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    EXTERNAL_LANGUAGE         INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT EXTERNAL_LANGUAGE_CHECK  
        CHECK ( EXTERNAL_LANGUAGE IN  
            ( 'ADA', 'C', 'COBOL',  
              'FORTRAN', 'MUMPS', 'PASCAL', 'PLI' ) ),  
    PARAMETER_STYLE           INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT PARAMETER_STYLE_CHECK  
        CHECK ( PARAMETER_STYLE IN  
            ( 'SQL', 'GENERAL' ) ),  
    IS_DETERMINISTIC          INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT IS_DETERMINISTIC_CHECK  
        CHECK ( IS_DETERMINISTIC  
            IN ( 'YES', 'NO' ) ),  
    SQL_DATA_ACCESS           INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT ROUTINES_SQL_DATA_ACCESS_NOT_NULL
```

```

NOT NULL
CONSTRAINT      ROUTINES_SQL_DATA_ACCESS_CHECK
CHECK ( SQL_DATA_ACCESS IN
        ( 'NO SQL', 'CONTAINS SQL',
          'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
IS_NULL_CALL      INFORMATION_SCHEMA.CHARACTER_DATA,
CONSTRAINT ROUTINES_IS_NULL_CALL_CHECK
CHECK ( IS_NULL_CALL IN
        ( 'YES', 'NO' ) ),
SQL_PATH      INFORMATION_SCHEMA.CHARACTER_DATA,
SCHEMA_LEVEL_ROUTINE      INFORMATION_SCHEMA.CHARACTER_DATA,
MAX_DYNAMIC_RESULT_SETS      INFORMATION_SCHEMA.CARDINAL_NUMBER,
IS_USER_DEFINED_CAST      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_IS_USER_DEFINED_CAST_CHECK
CHECK ( IS_USER_DEFINED_CAST IN
        ( 'YES', 'NO' ) ),
IS_IMPLICITLY_INVOCABLE      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_IS_IMPLICITLY_INVOCABLE_CHECK
CHECK ( IS_IMPLICITLY_INVOCABLE IN
        ( 'YES', 'NO' ) ),
SECURITY_TYPE      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_SECURITY_TYPE_CHECK
CHECK ( SECURITY_TYPE IN
        ( 'DEFINER', 'INVOKER', 'IMPLEMENTATION DEFINED' ) ),
TO_SQL_SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
AS_LOCATOR      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_AS_LOCATOR_CHECK
CHECK ( AS_LOCATOR IN
        ( 'YES', 'NO' ) ),
CREATED      INFORMATION_SCHEMA.TIME_STAMP,
LAST_ALTERED      INFORMATION_SCHEMA.TIME_STAMP,
NEW_SAVEPOINT_LEVEL      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_NEW_SAVEPOINT_LEVEL_CHECK
CHECK ( NEW_SAVEPOINT_LEVEL IN
        ( 'YES', 'NO' ) ),
IS_UDT_DEPENDENT      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_IS_UDT_DEPENDENT_NOT_NULL
NOT NULL
CONSTRAINT ROUTINES_IS_UDT_DEPENDENT_CHECK
CHECK ( IS_UDT_DEPENDENT IN
        ( 'YES', 'NO' ) ),
RESULT_CAST_FROM_DTD_IDENTIFIER      INFORMATION_SCHEMA.SQL_IDENTIFIER,
RESULT_CAST_AS_LOCATOR      INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_RESULT_CAST_AS_LOCATOR_CHECK
CHECK ( RESULT_CAST_AS_LOCATOR IN
        ( 'YES', 'NO' ) ),

CONSTRAINT ROUTINES_PRIMARY_KEY
PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

CONSTRAINT ROUTINES_FOREIGN_KEY_SCHEMATA
FOREIGN KEY ( ROUTINE_CATALOG, ROUTINE_SCHEMA )
REFERENCES SCHEMATA,

```

ISO/IEC 9075-11:2003 (E)
6.40 ROUTINES base table

```
CONSTRAINT ROUTINES_FOREIGN_KEY_USER_DEFINED_TYPES
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME )
    REFERENCES USER_DEFINED_TYPES
    MATCH FULL,

CONSTRAINT ROUTINES_COMBINATIONS
    CHECK ( ( ROUTINE_BODY = 'SQL'
              AND
              ( EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE ) IS NULL )
            OR
            ( ROUTINE_BODY = 'EXTERNAL'
              AND
              ( EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE ) IS NOT NULL ) ),

CONSTRAINT ROUTINES_SAME_SCHEMA
    CHECK ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
              ( ROUTINE_CATALOG, ROUTINE_SCHEMA )
            OR ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
              ( MODULE_CATALOG, MODULE_SCHEMA )
            OR ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA ) ),

CONSTRAINT ROUTINES_CHECK_RESULT_TYPE
    CHECK ( ( ROUTINE_TYPE = 'PROCEDURE'
              AND
              DTD_IDENTIFIER IS NULL )
            OR
            ( ROUTINE_TYPE <> 'PROCEDURE'
              AND
              ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', DTD_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR ) ) ),

CONSTRAINT ROUTINES_CHECK_RESULT_CAST
    CHECK ( ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NULL
              AND
              RESULT_CAST_AS_LOCATOR IS NULL )
            OR
            ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NOT NULL
              AND
              RESULT_CAST_AS_LOCATOR IS NOT NULL
              AND
              ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', RESULT_CAST_FROM_DTD_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR ) ) )
)
```

Description

- 1) The values of `SPECIFIC_CATALOG`, `SPECIFIC_SCHEMA`, and `SPECIFIC_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine being described.
- 2) The values of `ROUTINE_CATALOG`, `ROUTINE_SCHEMA`, and `ROUTINE_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the routine name of the SQL-invoked routine being described.
- 3) The values of `MODULE_CATALOG`, `MODULE_SCHEMA`, and `MODULE_NAME` are the null value.
- 4) Case:
 - a) If the SQL-invoked routine being described was defined as a method of a user-defined type, then the values of `USER_DEFINED_TYPE_CATALOG`, `USER_DEFINED_TYPE_SCHEMA`, and `USER_DEFINED_TYPE_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type name of this user-defined type.
 - b) Otherwise, the values of `USER_DEFINED_TYPE_CATALOG`, `USER_DEFINED_TYPE_SCHEMA`, and `USER_DEFINED_TYPE_NAME` are the null value.
- 5) The values of `ROUTINE_TYPE` have the following meanings:

PROCEDURE	The SQL-invoked routine being described is an SQL-invoked procedure.
FUNCTION	The SQL-invoked routine being described is an SQL-invoked function that is not an SQL-invoked method.
INSTANCE METHOD	The SQL-invoked routine being described is an SQL-invoked method that is neither a static SQL-invoked method nor an SQL-invoked constructor method.
STATIC METHOD	The SQL-invoked routine being described is a static SQL-invoked method.
CONSTRUCTOR METHOD	The SQL-invoked routine being described is an SQL-invoked constructor method.

- 6) If the SQL-invoked routine being described is an SQL-invoked procedure, then `DTD_IDENTIFIER` is the null value; otherwise, `SPECIFIC_CATALOG`, `SPECIFIC_SCHEMA`, and `SPECIFIC_NAME`, and `DTD_IDENTIFIER` are the values of `OBJECT_CATALOG`, `OBJECT_SCHEMA`, `OBJECT_NAME`, and `DTD_IDENTIFIER`, respectively, of the row in `DATA_TYPE_DESCRIPTOR` that describes the result type of the SQL-invoked routine being described.
- 7) The values of `ROUTINE_BODY` have the following meanings:

SQL	The SQL-invoked routine being described is an SQL routine.
-----	--

EXTERNAL	The SQL-invoked routine being described is an external routine.
----------	---

8) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

9) The value of IS_DETERMINISTIC indicates whether DETERMINISTIC was specified when the SQL-invoked routine was defined.

10) The values of IS_NULL_CALL have the following meanings:

YES	The SQL-invoked routine is a function and returns null if any of its parameters are null.
NO	The SQL-invoked routine is a function and its return value is determined by invoking the routine.
<i>null</i>	The routine being described is a procedure.

11) Case:

- a) If the SQL-invoked routine being described is an SQL routine, and the SQL-invoked routine is not contained in an SQL-server module definition, and the character representation of the <routine body> that defined the SQL-invoked routine can be represented without truncation, then the value of ROUTINE_DEFINITION is that character representation.
- b) Otherwise, the value of ROUTINE_DEFINITION is the null value.

12) Case:

- a) If the SQL-invoked routine being described is an external routine, then:
 - i) The value of EXTERNAL_NAME is the external name of the external routine.
 - ii) The value of EXTERNAL_LANGUAGE is the language of the external routine.
 - iii) The value of PARAMETER_STYLE is the SQL parameter passing style of the external routine.
- b) Otherwise, the values of EXTERNAL_NAME, EXTERNAL_LANGUAGE and PARAMETER_STYLE are the null value.

13) Case:

- a) If the routine being described is an SQL routine, then the value of SQL_PATH is the SQL-path of the routine being described.
- b) Otherwise, the value of SQL_PATH is the null value.

14) Case:

- a) If the SQL-invoked routine is a schema-level routine, then the value of SCHEMA_LEVEL_ROUTINE is 'YES'.
- b) Otherwise, the value of SCHEMA_LEVEL_ROUTINE is 'NO'.

15) The value of MAX_DYNAMIC_RESULT_SETS is

Case:

- a) If the routine being described is an SQL-invoked procedure defined by an <SQL-invoked routine> for which <maximum dynamic result sets> was specified, then the value of <maximum dynamic result sets>.
- b) Otherwise, 0 (zero).

16) The values of IS_USER_DEFINED_CAST have the following meanings:

YES	The SQL-invoked routine is a function that is a user-defined cast function.
NO	The function SQL-invoked routine is a function that is not a user-defined cast function.
<i>null</i>	The SQL-invoked routine is a procedure.

17) The values of IS_IMPLICITLY_INVOCABLE have the following meanings:

YES	The user-defined cast function is implicitly invocable.
NO	The user-defined cast function is not implicitly invocable.
<i>null</i>	The routine is not a user-defined cast function.

18) The values of SECURITY_TYPE have the following meanings:

DEFINER	The routine has the security characteristic DEFINER.
INVOKER	The routine has the security characteristic INVOKER.
IMPLEMENTATION DEFINED	The external routine has the security characteristic IMPLEMENTATION DEFINED.
<i>null</i>	The SQL-invoked routine is not an external routine and Feature T324, "Explicit security for SQL routines" is not implemented.

- 19) TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA and TO_SQL_SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the result type of the SQL-invoked routine being described.

- 20) The values of AS_LOCATOR have the following meanings:

YES	The return value of the SQL-invoked routine being described is passed AS LOCATOR.
NO	The return value of the SQL-invoked routine being described is not passed AS LOCATOR.

- 21) If Feature T272, “Enhanced savepoint management”, is not implemented, then the value of NEW_SAVEPOINT_LEVEL is null; otherwise, the values of NEW_SAVEPOINT_LEVEL have the following meanings:

YES	The SQL-invoked routine is an SQL-invoked function or is an SQL-invoked procedure that specifies NEW SAVEPOINT LEVEL.
NO	The SQL-invoked routine is an SQL-invoked procedure that does not specify NEW SAVEPOINT LEVEL or specifies OLD SAVEPOINT LEVEL.

- 22) The values of IS_UDT_DEPENDENT have the following meanings:

YES	The SQL-invoked routine being described is dependent on a user-defined type.
NO	The SQL-invoked routine being described is not dependent on a user-defined type.

- 23) Case:

- a) If the routine descriptor of the SQL-invoked routine being described does not include an indication that the SQL-invoked routine specifies a <result cast>, then the values of RESULT_CAST_FROM_DTD_IDENTIFIER and RESULT_CAST_AS_LOCATOR are the null value.
- b) Otherwise, SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, and RESULT_CAST_FROM_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the <data type> specified in the <result cast> of the SQL-invoked routine being described.

Case:

- i) If the routine descriptor of the SQL-invoked routine being described does not include an indication that the <data type> specified in the <result cast> has a locator indication, then the value of RESULT_CAST_AS_LOCATOR is 'NO'.
- ii) Otherwise, the value of RESULT_CAST_AS_LOCATOR is 'YES'.

- 24) The value of CREATED is the value of CURRENT_TIMESTAMP at the time when the SQL-invoked routine being described was created.

- 25) The value of `LAST_ALTERED` is the value of `CURRENT_TIMESTAMP` at the time that the SQL-invoked routine being described was last altered. This value is identical to the value of `CREATED` for SQL-invoked routines that have never been altered.

6.41 SCHEMATA base table

Function

The SCHEMATA table has one row for each schema.

Definition

```
CREATE TABLE SCHEMATA (  
    CATALOG_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SCHEMA_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SCHEMA_OWNER                  INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT SCHEMA_OWNER_NOT_NULL  
            NOT NULL,  
    DEFAULT_CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL  
            NOT NULL,  
    DEFAULT_CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL  
            NOT NULL,  
    DEFAULT_CHARACTER_SET_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT DEFAULT_CHARACTER_SET_NAME_NOT_NULL  
            NOT NULL,  
    SQL_PATH                      INFORMATION_SCHEMA.CHARACTER_DATA,  
  
    CONSTRAINT SCHEMATA_PRIMARY_KEY  
        PRIMARY KEY ( CATALOG_NAME, SCHEMA_NAME ),  
  
    CONSTRAINT SCHEMATA_FOREIGN_KEY_AUTHORIZATIONS  
        FOREIGN KEY ( SCHEMA_OWNER )  
        REFERENCES AUTHORIZATIONS,  
  
    CONSTRAINT SCHEMATA_FOREIGN_KEY_CATALOG_NAMES  
        FOREIGN KEY ( CATALOG_NAME )  
        REFERENCES CATALOG_NAMES  
)
```

Description

- 1) The value of CATALOG_NAME is the name of the catalog of the schema described by this row.
- 2) The value of SCHEMA_NAME is the unqualified schema name of the schema described by this row.
- 3) The values of SCHEMA_OWNER are the authorization identifiers that own the schemata.
- 4) The values of DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA, and DEFAULT_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the default character set for columns and domains in the schemata.
- 5) Case:

- a) If <schema path specification> was specified in the <schema definition> that defined the schema described by this row and the character representation of the <schema path specification> can be represented without truncation, then the value of SQL_PATH is that character representation.
- b) Otherwise, the value of SQL_PATH is the null value.

6.42 SEQUENCES base table

Function

The SEQUENCES base table has one row for each external sequence generator.

Definition

```
CREATE TABLE SEQUENCES (
    SEQUENCE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MAXIMUM_VALUE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_MAXIMUM_VALUE_NOT_NULL NOT NULL,
    MINIMUM_VALUE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_MINIMUM_VALUE_NOT_NULL NOT NULL,
    INCREMENT                  INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_INCREMENT_NOT_NULL NOT NULL,
    CYCLE_OPTION               INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_CYCLE_OPTION_NOT_NULL NOT NULL,

    CONSTRAINT SEQUENCES_PRIMARY_KEY
        PRIMARY KEY (SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME),

    CONSTRAINT SEQUENCES_FOREIGN_KEY_SCHEMATA
        FOREIGN KEY ( SEQUENCES_CATALOG, SEQUENCES_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT SEQUENCES_CYCLE_OPTION_CHECK
        CHECK ( CYCLE_OPTION IN ( 'YES', 'NO' ) ),

    CONSTRAINT SEQUENCES_CHECK_DATA_TYPE
        CHECK ( ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA,
                  SEQUENCE_NAME, 'SEQUENCE', DTD_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR ) )
)
```

Description

- 1) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, and SEQUENCE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the sequence generator being described.
- 2) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and

DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the sequence generator.

- 3) The values of MAXIMUM_VALUE, MINIMUM_VALUE, and INCREMENT are the character representations of maximum value, minimum value, and increment, respectively, of the sequence generator being described.
- 4) The values of CYCLE_OPTION have the following meanings:

YES	The cycle option of the sequence generator is CYCLE.
NO	The cycle option of the sequence generator is NO CYCLE.

6.43 SQL_CONFORMANCE base table

Function

The SQL_CONFORMANCE base table has one row for each conformance element (part, package, feature, and subfeature) identified by ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_CONFORMANCE
(
    TYPE                                INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_CONFORMANCE_TYPE_CHECK
    CHECK ( TYPE IN ( 'PART', 'FEATURE', 'SUBFEATURE', 'PACKAGE' ) ),
    ID                                  INFORMATION_SCHEMA.CHARACTER_DATA,
    SUB_ID                             INFORMATION_SCHEMA.CHARACTER_DATA,
    NAME                               INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_CONFORMANCE_NAME_NOT_NULL
    NOT NULL,
    SUB_NAME                           INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_SUPPORTED                       INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_CONFORMANCE_IS_SUPPORTED_NOT_NULL
    NOT NULL
    CONSTRAINT SQL_CONFORMANCE_IS_SUPPORTED_CHECK
    CHECK ( IS_SUPPORTED IN
            ( 'YES', 'NO' ) ),
    IS_VERIFIED_BY                     INFORMATION_SCHEMA.CHARACTER_DATA,
    COMMENTS                           INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_CONFORMANCE_PRIMARY_KEY
    PRIMARY KEY ( TYPE, ID, SUB_ID ),

    CONSTRAINT SQL_CONFORMANCE_CHECK_SUPPORTED_VERIFIED
    CHECK ( IS_SUPPORTED = 'YES'
    OR
    IS_VERIFIED_BY IS NULL )
)
```

Description

- 1) The SQL_CONFORMANCE table consists of exactly one row for each SQL part, package, feature, and subfeature defined in ISO/IEC 9075.
- 2) The values of TYPE have the following meanings:

PART	the conformance element described is a Part of ISO/IEC 9075.
FEATURE	the conformance element described is an optional feature of ISO/IEC 9075.

SUBFEA- TURE	the conformance element described is a subfeature of an optional feature of ISO/IEC 9075.
PACKAGE	the conformance element described is a package of features of ISO/IEC 9075.

- 3) The ID and NAME columns identify the conformance element described.
- 4) If the conformance element is a subfeature, then the SUB_ID and SUB_NAME columns identify the subfeature by the subfeature identifier and name assigned to it. Otherwise, the values of SUB_ID and SUB_NAME are each a character string consisting of a single space.
- 5) The IS_SUPPORTED column is 'YES' if an SQL-implementation fully supports that conformance element described when SQL-data in the identified catalog is accessed through that implementation and is 'NO' if the SQL-implementation does not fully support that conformance element described when accessing SQL-data in that catalog.
- 6) If full support for the conformance element described has been verified by testing, then the IS_VERIFIED_BY column shall contain information identifying the conformance test used to verify the conformance claim; otherwise, IS_VERIFIED_BY shall be the null value.
- 7) If the value of the IS_SUPPORTED column for a feature is 'YES' and if that feature has subfeatures, then the value of the IS_SUPPORTED column in every row identifying subfeatures of the feature shall also be 'YES'.
- 8) The COMMENTS column contains any implementer comments pertinent to the identified SQL part, package, feature, or subfeature.

6.44 SQL_IMPLEMENTATION_INFO base table

Function

The SQL_IMPLEMENTATION_INFO base table has one row for each SQL-implementation information item defined by ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_IMPLEMENTATION_INFO (
    IMPLEMENTATION_INFO_ID          INFORMATION_SCHEMA.CHARACTER_DATA,
    IMPLEMENTATION_INFO_NAME        INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT SQL_IMPLEMENTATION_INFO_NAME_NOT_NULL
            NOT NULL,
    INTEGER_VALUE                   INFORMATION_SCHEMA.CARDINAL_NUMBER,
    CHARACTER_VALUE                 INFORMATION_SCHEMA.CHARACTER_DATA,
    COMMENTS                       INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_IMPLEMENTATION_INFO_PRIMARY_KEY
        PRIMARY KEY ( IMPLEMENTATION_INFO_ID ),

    CONSTRAINT SQL_IMPLEMENTATION_INFO_CHECK_INTEGER_EXCLUDES_CHARACTER
        CHECK ( INTEGER_VALUE IS NULL
            OR
                CHARACTER_VALUE IS NULL )
)
```

Description

- 1) The SQL_IMPLEMENTATION_INFO table consists of exactly one row for each SQL-implementation information item defined in ISO/IEC 9075.
- 2) The IMPLEMENTATION_INFO_ID and IMPLEMENTATION_INFO_NAME columns identify the SQL-implementation information item by the integer and name assigned to it.
- 3) Depending on the type of information, the value is present in either INTEGER_VALUE or CHARACTER_VALUE; the other column is the null value.
- 4) The COMMENTS column is intended for any implementer comments pertinent to the identified item.

6.45 SQL_LANGUAGES base table

Function

The SQL_LANGUAGES table has one row for each ISO and implementation-defined SQL language binding and programming language for which conformance is claimed.

NOTE 11 — The SQL_LANGUAGES base table provides, among other information, the same information provided by the SQL object identifier specified in Subclause 6.4, “Object identifier for Database Language SQL”, in ISO/IEC 9075-1.

Definition

```
CREATE TABLE SQL_LANGUAGES (
    SQL_LANGUAGE_SOURCE          INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_LANGUAGES_SOURCE_NOT_NULL
    NOT NULL,
    SQL_LANGUAGE_YEAR            INFORMATION_SCHEMA.CHARACTER_DATA,
    SQL_LANGUAGE_CONFORMANCE     INFORMATION_SCHEMA.CHARACTER_DATA,
    SQL_LANGUAGE_INTEGRITY       INFORMATION_SCHEMA.CHARACTER_DATA,
    SQL_LANGUAGE_IMPLEMENTATION  INFORMATION_SCHEMA.CHARACTER_DATA,
    SQL_LANGUAGE_BINDING_STYLE   INFORMATION_SCHEMA.CHARACTER_DATA,
    SQL_LANGUAGE_PROGRAMMING_LANGUAGE  INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_LANGUAGES_YEAR_ISO
    CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
    OR
        ( SQL_LANGUAGE_YEAR IS NOT NULL
        AND
            SQL_LANGUAGE_YEAR IN
                ( '1987', '1989', '1992', '1999', '2003' ) ) ),

    CONSTRAINT SQL_LANGUAGES_CONFORMANCE_ISO_2003
    CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
    OR
        ( SQL_LANGUAGE_YEAR <> '2003'
        OR
            ( SQL_LANGUAGE_CONFORMANCE IS NOT NULL
            AND
                SQL_LANGUAGE_CONFORMANCE IN
                    ( 'CORE' ) ) ) ),

    CONSTRAINT SQL_LANGUAGES_IMPLEMENTATION_ISO
    CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
    OR
        SQL_LANGUAGE_IMPLEMENTATION IS NULL ),

    CONSTRAINT SQL_LANGUAGES_BINDING_STYLE_ISO_2003
    CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
    OR
        ( SQL_LANGUAGE_YEAR <> '2003'
        OR
            ( SQL_LANGUAGE_BINDING_STYLE IS NOT NULL
```

ISO/IEC 9075-11:2003 (E)
6.45 SQL_LANGUAGES base table

```
AND
    SQL_LANGUAGE_BINDING_STYLE IN
        ( 'DIRECT', 'EMBEDDED', 'MODULE' ) ) ) ),

CONSTRAINT SQL_LANGUAGES_STANDARD_VALID_CHECK_ISO_2003
    CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
        OR
            ( SQL_LANGUAGE_YEAR <> '2003'
                OR
                    ( SQL_LANGUAGE_BINDING_STYLE = 'DIRECT'
                        AND
                            SQL_LANGUAGE_PROGRAMMING_LANGUAGE IS NULL )
                OR
                    ( ( SQL_LANGUAGE_BINDING_STYLE IN
                        ( 'EMBEDDED', 'MODULE' )
                            AND
                                SQL_LANGUAGE_PROGRAMMING_LANGUAGE IN
                                    ( 'ADA', 'C', 'COBOL', 'FORTRAN',
                                        'MUMPS', 'PASCAL', 'PLI' ) )
                        OR
                            ( SQL_LANGUAGE_BINDING_STYLE IN
                                ( 'EMBEDDED' )
                                    AND
                                        SQL_LANGUAGE_PROGRAMMING_LANGUAGE IN
                                            ( 'JAVA' ) ) ) ) ) ) ) ) )
```

Description

- 1) Each row represents one binding of an ISO or implementation-defined SQL language.
- 2) The value of SQL_LANGUAGE_SOURCE is the name of the source of the language definition. The source of standard SQL language is the value 'ISO 9075', while the source of an implementation-defined version of SQL is implementation-defined.
- 3) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_YEAR is the year that the ISO standard was approved. Otherwise, the value of SQL_LANGUAGE_YEAR is implementation-defined.

NOTE 12 — As each new ISO SQL standard revision is approved, a new valid value of SQL_LANGUAGE_YEAR will be added to the CHECK constraint for SQL_LANGUAGE_YEAR_ISO.

- 4) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_CONFORMANCE is the conformance level to which conformance is claimed for the ISO standard. Otherwise, the value of SQL_LANGUAGE_CONFORMANCE is implementation-defined.
- 5) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_IMPLEMENTATION is null. Otherwise, the value of SQL_LANGUAGE_IMPLEMENTATION is an implementation-defined character string value.
- 6) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_BINDING_STYLE is the style of binding of the SQL language. If the value of SQL_LANGUAGE_BINDING_STYLE is 'MODULE', then the binding style of <SQL-client module definition> is supported. If the

value of SQL_LANGUAGE_BINDING_STYLE is 'EMBEDDED', then the binding style of <embedded SQL host program> is supported. If the value of SQL_LANGUAGE_BINDING_STYLE is 'DIRECT', then the binding style of <direct SQL statement> is supported. Otherwise, the value of SQL_LANGUAGE_BINDING_STYLE is implementation-defined.

- 7) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_PROGRAMMING_LANGUAGE is the programming language supported by the binding style indicated by the value of SQL_LANGUAGE_BINDING_STYLE.

If the value of SQL_LANGUAGE_BINDING_STYLE is 'DIRECT', then SQL_LANGUAGE_PROGRAMMING_LANGUAGE is the null value. If the value of SQL_LANGUAGE_BINDING_STYLE is 'MODULE', then SQL_LANGUAGE_PROGRAMMING_LANGUAGE has the value 'ADA', 'C', 'COBOL', 'FORTRAN', 'MUMPS', 'PASCAL', or 'PLI'. If the value of SQL_LANGUAGE_BINDING_STYLE is 'EMBEDDED', then SQL_LANGUAGE_PROGRAMMING_LANGUAGE has the value 'ADA', 'C', 'COBOL', 'FORTRAN', 'JAVA', 'MUMPS', 'PASCAL', or 'PLI'.

Case:

- a) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'ADA', then Ada is supported with the given binding style.
- b) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'C', then C is supported with the given binding style.
- c) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'COBOL', then COBOL is supported with the given binding style.
- d) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'FORTRAN', then Fortran is supported with the given binding style.
- e) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'JAVA', then Java is supported with the given binding style.
- f) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'MUMPS', then M is supported with the given binding style.
- g) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'PASCAL', then Pascal is supported with the given binding style.
- h) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'PLI', then PL/I is supported with the given binding style.
- i) Otherwise, the value of SQL_LANGUAGE_PROGRAMMING_LANGUAGE is implementation-defined.

6.46 SQL_SIZING base table

Function

The SQL_SIZING base table has one row for each sizing item defined in ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_SIZING (
    SIZING_ID                INFORMATION_SCHEMA.CARDINAL_NUMBER,
    SIZING_NAME              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL
        NOT NULL,
    /* If SUPPORTED_VALUE is the null value, that means that the item      */
    /* being described is not applicable in the implementation.            */
    /* If SUPPORTED_VALUE is 0 (zero), that means that the item            */
    /* being described has no limit or the limit cannot be determined.*/
    SUPPORTED_VALUE          INFORMATION_SCHEMA.CARDINAL_NUMBER,
    COMMENTS                 INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_SIZING_PRIMARY_KEY
        PRIMARY KEY (SIZING_ID ),

    CONSTRAINT SQL_SIZING_SIZING_NAME_UNIQUE
        UNIQUE ( SIZING_NAME )
)
```

Description

- 1) The SQL_SIZING table shall consist of exactly one row for each SQL sizing item defined in ISO/IEC 9075.
- 2) The SIZING_ID and SIZING_NAME columns identify the sizing item by the integer and description assigned to it.
- 3) The values of the SUPPORTED_VALUE column are:

0	The SQL-implementation either places no limit on this sizing item or the SQL-implementation cannot determine the limit.
<i>null</i>	The SQL-implementation does not support any features for which this sizing item is applicable.
Any other value	The maximum size supported by the SQL-implementation for this sizing item.

- 4) The COMMENTS column is intended for any implementor comments pertinent to the identified SQL sizing item.

6.47 SQL_SIZING_PROFILES base table

Function

The SQL_SIZING base table has one row for each sizing item defined in ISO/IEC 9075 for each defined profile.

Definition

```
CREATE TABLE SQL_SIZING_PROFILES (  
    SIZING_ID                INFORMATION_SCHEMA.CARDINAL_NUMBER,  
    PROFILE_ID               INFORMATION_SCHEMA.CHARACTER_DATA,  
    PROFILE_NAME             INFORMATION_SCHEMA.CHARACTER_DATA  
        CONSTRAINT SQL_SIZING_PROFILE_NAME_NOT_NULL  
            NOT NULL,  
    /* If REQUIRED_VALUE is the null value, that means that the item      */  
    /* being described is not applicable in the profile.                  */  
    /* If REQUIRED_VALUE is 0 (zero), that means that the profile          */  
    /* does not set a limit for this sizing item.                         */  
    REQUIRED_VALUE            INFORMATION_SCHEMA.CARDINAL_NUMBER,  
    COMMENTS                 INFORMATION_SCHEMA.CHARACTER_DATA,  
  
    CONSTRAINT SQL_SIZING_PROFILE_PRIMARY_KEY  
        PRIMARY KEY (SIZING_ID, PROFILE_ID ),  
  
    CONSTRAINT SQL_SIZING_PROFILE_FOREIGN_KEY_SQL_SIZING  
        FOREIGN KEY ( SIZING_ID )  
            REFERENCES SQL_SIZING  
)
```

Description

- 1) The SQL_SIZING_PROFILE table shall consist of exactly one row for each SQL sizing item defined in ISO/IEC 9075 for each profile that is defined in the table.
- 2) The SIZING_ID column identifies the sizing item by the integer assigned to it.
- 3) The PROFILE_ID column shall contain information identifying a profile.
- 4) The values of the REQUIRED_VALUE column are:

0	The profile places no limit on this sizing item.
<i>null</i>	The profile does not require any features for which this sizing item is applicable.
Any other value	The minimum size required by the profile for this sizing item.

- 5) The COMMENTS column is intended for any implementor comments pertinent to the identified SQL sizing item within the profile.

6.48 TABLE_CONSTRAINTS base table

Function

The TABLE_CONSTRAINTS table has one row for each table constraint associated with a table. It effectively contains a representation of the table constraint descriptors.

Definition

```
CREATE TABLE TABLE_CONSTRAINTS (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_TYPE          INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT CONSTRAINT_TYPE_NOT_NULL
    NOT NULL
  CONSTRAINT CONSTRAINT_TYPE_CHECK
    CHECK ( CONSTRAINT_TYPE IN
      ( 'UNIQUE', 'PRIMARY KEY',
        'FOREIGN KEY', 'CHECK' ) ),
  TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TABLE_CONSTRAINTS_TABLE_CATALOG_NOT_NULL
    NOT NULL,
  TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TABLE_CONSTRAINTS_TABLE_SCHEMA_NOT_NULL
    NOT NULL,
  TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TABLE_CONSTRAINTS_TABLE_NAME_NOT_NULL
    NOT NULL,
  IS_DEFERRABLE            INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TABLE_CONSTRAINTS_IS_DEFERRABLE_NOT_NULL
    NOT NULL,
  INITIALLY_DEFERRED       INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TABLE_CONSTRAINTS_INITIALLY_DEFERRED_NOT_NULL
    NOT NULL,

  CONSTRAINT TABLE_CONSTRAINTS_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),

  CONSTRAINT TABLE_CONSTRAINTS_DEFERRED_CHECK
    CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED ) IN
      ( VALUES ( 'NO', 'NO' ),
        ( 'YES', 'NO' ),
        ( 'YES', 'YES' ) ) ),

  CONSTRAINT TABLE_CONSTRAINTS_CHECK_VIEWS
    CHECK ( TABLE_CATALOG NOT IN
      ( SELECT CATALOG_NAME
        FROM SCHEMATA )
    OR
      ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
```

```

        ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
          FROM TABLES
          WHERE TABLE_TYPE <> 'VIEW' ) ) ),

CONSTRAINT TABLE_CONSTRAINTS_UNIQUE_CHECK
CHECK ( 1 =
  ( SELECT COUNT ( *)
    FROM ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM TABLE_CONSTRAINTS
          WHERE CONSTRAINT_TYPE IN
            ( 'UNIQUE', 'PRIMARY KEY' )
        UNION ALL
          SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM REFERENTIAL_CONSTRAINTS
        UNION ALL
          SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM CHECK_CONSTRAINTS ) AS X
    WHERE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) =
      ( X.CONSTRAINT_CATALOG, X.CONSTRAINT_SCHEMA, X.CONSTRAINT_NAME ) ) ),

CONSTRAINT UNIQUE_TABLE_PRIMARY_KEY_CHECK
CHECK ( UNIQUE ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLE_CONSTRAINTS
                  WHERE CONSTRAINT_TYPE = 'PRIMARY KEY' ) )
)

```

Description

- 1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described. If the <table constraint definition> or <add table constraint definition> that defined the constraint did not specify a <constraint name>, then the values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are implementation-defined.
- 2) The values of CONSTRAINT_TYPE have the following meanings:

FOREIGN KEY	The constraint being described is a foreign key constraint.
UNIQUE	The constraint being described is a unique constraint.
PRIMARY KEY	The constraint being described is a primary key constraint.
CHECK	The constraint being described is a check constraint.

- 3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, the unqualified schema name, and the qualified identifier of the name of the table to which the table constraint being described applies.
- 4) The values of IS_DEFERRABLE have the following meanings:

YES	The table constraint is deferrable.
NO	The table constraint is not deferrable.

5) The values of INITIALLY_DEFERRED have the following meanings:

YES	The table constraint is initially deferred.
NO	The table constraint is initially immediate.

6.49 TABLE_METHOD_PRIVILEGES base table

Function

The TABLE_METHOD_PRIVILEGES base table has one row for each table/method privilege descriptor. It effectively contains a representation of the table/method privilege descriptors.

Definition

```
CREATE TABLE TABLE_METHOD_PRIVILEGES (
    GRANTOR                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_GRANTABLE          INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TABLE_METHOD_PRIVILEGES_IS_GRANTABLE_NOT_NULL
        NOT NULL
    CONSTRAINT TABLE_METHOD_PRIVILEGES_IS_GRANTABLE_CHECK
        CHECK ( IS_GRANTABLE IN
            ( 'YES', 'NO' ) ),

    CONSTRAINT TABLE_METHOD_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

    CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
            REFERENCES ROUTINES,

    CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
        FOREIGN KEY ( GRANTOR )
            REFERENCES AUTHORIZATIONS,

    CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTEE
        FOREIGN KEY ( GRANTEE )
            REFERENCES AUTHORIZATIONS
)
```

Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted a table/method privilege on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME, and the method of the identified table's structured type identified by the SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, to the user or role identified by the value of GRANTEE for the table/method privilege being described.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the table/method privilege being described is granted.
- 3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table on which the privilege being described was granted.
- 4) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the method on which the privilege being described was granted.
- 5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.50 TABLE_PRIVILEGES base table

Function

The TABLE_PRIVILEGES table has one row for each table privilege descriptor. It effectively contains a representation of the table privilege descriptors.

Definition

```
CREATE TABLE TABLE_PRIVILEGES (
  GRANTOR              INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE              INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PRIVILEGE_TYPE        INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TABLE_PRIVILEGES_TYPE_CHECK
    CHECK ( PRIVILEGE_TYPE IN
      ( 'SELECT', 'INSERT', 'DELETE', 'UPDATE',
        'TRIGGER', 'REFERENCES' ) ),
  IS_GRANTABLE          INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TABLE_PRIVILEGES_GRANTABLE_NOT_NULL
    NOT NULL
  CONSTRAINT TABLE_PRIVILEGES_GRANTABLE_CHECK
    CHECK ( IS_GRANTABLE IN
      ( 'YES', 'NO' ) ),
  WITH_HIERARCHY        INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TABLE_PRIVILEGES_WITH_HIERARCHY_NOT_NULL
    NOT NULL
  CONSTRAINT TABLE_PRIVILEGES_WITH_HIERARCHY_CHECK
    CHECK ( WITH_HIERARCHY IN
      ( 'YES', 'NO' ) ),

  CONSTRAINT TABLE_PRIVILEGES_PRIMARY_KEY
    PRIMARY KEY ( GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      PRIVILEGE_TYPE ),

  CONSTRAINT TABLE_PRIVILEGES_FOREIGN_KEY_TABLES
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
      REFERENCES TABLES,

  CONSTRAINT TABLE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
    FOREIGN KEY ( GRANTOR )
      REFERENCES AUTHORIZATIONS,

  CONSTRAINT TABLE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS GRANTEE
    FOREIGN KEY ( GRANTEE )
      REFERENCES AUTHORIZATIONS
)
```

Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted table privileges, on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME, to the user or role identified by the value of GRANTEE for the table privilege being described.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or “PUBLIC” to indicate all users, to whom the table privilege being described is granted.
- 3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table on which the privilege being described has been granted.
- 4) The values of PRIVILEGE_TYPE have the following meanings:

SELECT	The user has SELECT privileges on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.
DELETE	The user has DELETE privileges on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.
INSERT	The user will automatically be granted INSERT privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
UPDATE	The user will automatically be granted UPDATE privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
REFER- ENCES	The user will automatically be granted REFERENCES privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
TRIGGER	The user has TRIGGER privilege on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.

- 5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

- 6) The values of WITH_HIERARCHY have the following meanings:

YES	The privilege being described was granted WITH HIERARCHY OPTION and is thus grantable.
-----	--

NO	The privilege being described was not granted WITH HIERARCHY OPTION and is thus not grantable.
----	--

6.51 TABLES base table

Function

The TABLES table contains one row for each table including views. It effectively contains a representation of the table descriptors.

Definition

```
CREATE TABLE TABLES (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_TYPE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TABLE_TYPE_NOT_NULL
    NOT NULL
    CONSTRAINT TABLE_TYPE_CHECK
    CHECK ( TABLE_TYPE IN
        ( 'BASE TABLE', 'VIEW', 'GLOBAL TEMPORARY', 'LOCAL TEMPORARY' ) ),
    SELF_REFERENCING_COLUMN_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_GENERATION    INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENCE_GENERATION_CHECK
    CHECK ( REFERENCE_GENERATION IN
        ( 'SYSTEM GENERATED', 'USER GENERATED', 'DERIVED' ) ),
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_INSERTABLE_INTO      INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT IS_INSERTABLE_INTO_NOT_NULL
    NOT NULL
    CONSTRAINT IS_INSERTABLE_INTO_CHECK
    CHECK ( IS_INSERTABLE_INTO IN
        ( 'YES', 'NO' ) ),
    IS_TYPED                INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT IS_TYPED_NOT_NULL
    NOT NULL
    CONSTRAINT IS_TYPED_CHECK
    CHECK ( IS_TYPED IN
        ( 'YES', 'NO' ) ),
    COMMIT_ACTION           INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COMMIT_ACTIONCHECK
    CHECK ( COMMIT_ACTION IN
        ( 'DELETE', 'PRESERVE' ) ),

    CONSTRAINT TABLES_PRIMARY_KEY
    PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT TABLES_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA )
    REFERENCES SCHEMATA,
```

```

CONSTRAINT TABLES_CHECK_TABLE_IN_COLUMNS
CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
        ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
          FROM COLUMNS ) ),

CONSTRAINT TABLES_FOREIGN_KEY_USER_DEFINED_TYPES
FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME )
REFERENCES USER_DEFINED_TYPES MATCH FULL,

CONSTRAINT TABLES_TYPED_TABLE_CHECK
CHECK ( ( IS_TYPED = 'YES'
        AND
        ( ( USER_DEFINED_TYPE_CATALOG,
              USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME,
              SELF_REFERENCING_COLUMN_NAME,
              REFERENCE_GENERATION ) IS NOT NULL
        AND
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          SELF_REFERENCING_COLUMN_NAME ) IN
        ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
              COLUMN_NAME
          FROM COLUMNS
          WHERE IS_SELF_REFERENCING = 'YES' ) ) )
OR
        ( IS_TYPED = 'NO'
        AND
        ( USER_DEFINED_TYPE_CATALOG,
          USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME,
          SELF_REFERENCING_COLUMN_NAME,
          REFERENCE_GENERATION ) IS NULL ) ),

CONSTRAINT TABLES_SELF_REFERENCING_COLUMN_CHECK
CHECK ( ( SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION ) IS NULL
        OR ( SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION ) IS NOT NULL ),

CONSTRAINT TABLES_TEMPORARY_TABLE_CHECK
CHECK ( ( TABLE_TYPE IN ( 'GLOBAL TEMPORARY', 'LOCAL TEMPORARY' ) )
        = ( COMMIT_ACTION IS NOT NULL ) ),

CONSTRAINT TABLES_CHECK_NOT_VIEW
CHECK ( NOT EXISTS (
        SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM TABLES
        WHERE TABLE_TYPE = 'VIEW'
    EXCEPT
        SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM VIEWS ) )
)

```

Description

- 1) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the fully qualified name of the table.
- 2) The values of TABLE_TYPE have the following meanings:

BASE TABLE	The table being described is a persistent base table.
VIEW	The table being described is a viewed table.
GLOBAL TEMPORARY	The table being described is a global temporary table.
LOCAL TEMPORARY	The table being described is a created local temporary table.

- 3) The value of SELF_REFERENCING_COLUMN_NAME is the name of the self-referencing column of the table, if the table is a typed table. Otherwise, the value of SELF_REFERENCING_COLUMN_NAME is the null value.
- 4) The values of COMMIT_ACTION have the following meanings:

DELETE	A <table commit action> of DELETE was specified.
PRESERVE	A <table commit action> of PRESERVE was specified.
<i>null</i>	The table being described is not a temporary table.

- 5) The values of REFERENCE_GENERATION have the following meanings:

SYSTEM GENERATED	The values of the self-referencing column of the table are generated by the SQL-server.
USER GENERATED	The values of the self-referencing column of the table are generated by the user.
DERIVED	The values of the self-referencing column of the table are generated from columns of the table.
<i>null</i>	The table being described does not have a self-referencing column.

- 6) If the table being described is a table of a structured type *TY*, then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and

USER_DEFINED_TYPE_NAME are the fully qualified name of *TY*; otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.

7) The values of IS_INSERTABLE_INTO have the following meanings:

a) If the SQL-implementation supports Feature T111, “Updatable joins, unions, and columns”, then:

YES	The table being described is insertable-into.
NO	The table being described is not insertable-into.

b) Otherwise:

YES	The table being described is insertable-into and simply updatable.
NO	The table being described is not insertable-into or not simply updatable.

8) The values of IS_TYPED have the following meanings:

YES	The table being described is a typed table.
NO	The table being described is not a typed table.

6.52 TRANSFORMS base table

Function

The TRANSFORMS base table has one row for each transform.

Definition

```
CREATE TABLE TRANSFORMS (  
    USER_DEFINED_TYPE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    USER_DEFINED_TYPE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    USER_DEFINED_TYPE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SPECIFIC_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SPECIFIC_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    SPECIFIC_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    GROUP_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    TRANSFORM_TYPE                INFORMATION_SCHEMA.CHARACTER_DATA  
    CONSTRAINT TRANSFORM_TYPE_NOT_NULL  
        NOT NULL  
    CONSTRAINT TRANSFORM_TYPE_CHECK  
        CHECK ( TRANSFORM_TYPE IN  
            ( 'TO SQL', 'FROM SQL' ) ),  
  
    CONSTRAINT TRANSFORMS_PRIMARY_KEY  
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,  
            USER_DEFINED_TYPE_NAME, GROUP_NAME, TRANSFORM_TYPE ),  
  
    CONSTRAINT TRANSFORMS_TYPES_FOREIGN_KEY  
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,  
            USER_DEFINED_TYPE_NAME )  
            REFERENCES USER_DEFINED_TYPES,  
  
    CONSTRAINT TRANSFORMS_ROUTINES_FOREIGN_KEY  
        FOREIGN KEY (SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME)  
            REFERENCES ROUTINES  
)
```

Description

- 1) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type for which the transform being described applies.
- 2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-invoked routine that acts as the transform function for the transform being described. The value of GROUP_NAME is the identifier that acts as the name of a transform group.
- 3) The values of TRANSFORM_TYPE have the following meanings:

TO SQL	The transform being described identifies a to-sql function
FROM SQL	The transform being described identifies a from-sql function

6.53 TRANSLATIONS base table

Function

The TRANSLATIONS table has one row for each character transliteration descriptor.

Definition

```
CREATE TABLE TRANSLATIONS (
  TRANSLATION_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TRANSLATION_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TRANSLATION_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SOURCE_CHARACTER_SET_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_CATALOG_NOT_NULL
    NOT NULL,
  SOURCE_CHARACTER_SET_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_SCHEMA_NOT_NULL
    NOT NULL,
  SOURCE_CHARACTER_SET_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_NAME_NOT_NULL
    NOT NULL,
  TARGET_CHARACTER_SET_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_CATALOG_NOT_NULL
    NOT NULL,
  TARGET_CHARACTER_SET_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_SCHEMA_NOT_NULL
    NOT NULL,
  TARGET_CHARACTER_SET_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_NAME_NOT_NULL
    NOT NULL,
  TRANSLATION_SOURCE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_CATALOG_NOT_NULL
    NOT NULL,
  TRANSLATION_SOURCE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_SCHEMA_NOT_NULL
    NOT NULL,
  TRANSLATION_SOURCE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_NAME_NOT_NULL
    NOT NULL,

  CONSTRAINT TRANSLATIONS_PRIMARY_KEY
    PRIMARY KEY ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME ),

  CONSTRAINT TRANSLATIONS_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA )
      REFERENCES SCHEMATA,

  CONSTRAINT TRANSLATIONS_FOREIGN_KEY_ROUTINES
    FOREIGN KEY ( TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
                  TRANSLATION_SOURCE_NAME )
      REFERENCES ROUTINES,
```



```
CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_SOURCE
CHECK ( SOURCE_CHARACTER_SET_CATALOG NOT IN
      ( SELECT CATALOG_NAME
        FROM SCHEMATA )
      OR
      ( SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
        SOURCE_CHARACTER_SET_NAME ) IN
      ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME
        FROM CHARACTER_SETS ) ),

CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_TARGET
CHECK ( TARGET_CHARACTER_SET_CATALOG NOT IN
      ( SELECT CATALOG_NAME
        FROM SCHEMATA )
      OR
      ( TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
        TARGET_CHARACTER_SET_NAME ) IN
      ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME
        FROM CHARACTER_SETS ) )
)
```

Description

- 1) The values of TRANSLATION_CATALOG, TRANSLATION_SCHEMA, and TRANSLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the transliteration being described.
- 2) The values of SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA, and SOURCE_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set specified as the source for the transliteration.
- 3) The values of TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA, and TARGET_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set specified as the target for the transliteration.
- 4) The values of TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA, and TRANSLATION_SOURCE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine used for the transliteration.

6.54 TRIGGERED_UPDATE_COLUMNS base table

Function

The TRIGGERED_UPDATE_COLUMNS base table has one row for each column identified by a <column name> in a <trigger column list> of a trigger definition.

Definition

```
CREATE TABLE TRIGGERED_UPDATE_COLUMNS (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    EVENT_OBJECT_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER
        CONSTRAINT EVENT_OBJECT_CATALOG_NOT_NULL
        NOT NULL,
    EVENT_OBJECT_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
        CONSTRAINT EVENT_OBJECT_SCHEMA_NOT_NULL
        NOT NULL,
    EVENT_OBJECT_TABLE       INFORMATION_SCHEMA.SQL_IDENTIFIER
        CONSTRAINT EVENT_OBJECT_TABLE_NOT_NULL
        NOT NULL,
    EVENT_OBJECT_COLUMN      INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_PRIMARY_KEY
        PRIMARY KEY
        ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME, EVENT_OBJECT_COLUMN ),

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_EVENT_MANIPULATION_CHECK
        CHECK ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME ) IN
            ( SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME
              FROM TRIGGERS
              WHERE EVENT_MANIPULATION = 'UPDATE' ) ),

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS,

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_FOREIGN_KEY_COLUMNS
        FOREIGN KEY ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
                      EVENT_OBJECT_TABLE, EVENT_OBJECT_COLUMN )
        REFERENCES COLUMNS
)
```

Description

- 1) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, schema name, and trigger name of the trigger being described.

6.54 TRIGGERED_UPDATE_COLUMNS base table

- 2) The values of EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, and EVENT_OBJECT_TABLE are the catalog name, schema name, and table name of the table containing the column being described. The TRIGGERED_UPDATE_COLUMNS base table has one row for each column contained in an explicitly specified <trigger column list> of a trigger whose trigger event is UPDATE.

6.55 TRIGGER_COLUMN_USAGE base table

Function

The TRIGGER_COLUMN_USAGE base table has one row for each column of a table identified by a <table name> contained in a <table reference> that is contained in the <search condition> of a <triggered action> or explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

Definition

```
CREATE TABLE TRIGGER_COLUMN_USAGE (
    TRIGGER_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_COLUMN_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

    CONSTRAINT TRIGGER_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
        FOREIGN KEY
            ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
            REFERENCES COLUMNS,

    CONSTRAINT TRIGGER_COLUMN_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
            REFERENCES TRIGGERS
)
```

Description

- 1) The TRIGGER_COLUMN_USAGE base table has one row for each column identified by at least one of:
 - a) A <column reference> contained in a <search condition> contained in a <trigger definition> of a trigger.
 - b) A <column reference> contained in a <search condition> contained in a <delete statement: searched> or an <update statement: searched> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.
 - c) A <column reference> contained in a <value expression> simply contained in a <update source> or an <assigned row> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.

- d) A <column name> contained in an <insert column list> of an <insert statement> contained in the <triggered SQL statement> contained in a <trigger definition> of a trigger.
 - e) A <column name> contained in an <object column> contained in either an <update statement: positioned> or an <update statement: searched> contained in the <triggered SQL statement> contained in a <trigger definition> of a trigger.
- 2) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the trigger being described.
 - 3) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column of a table identified by a <table name> contained in the <search condition> of a <triggered action>, or explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

6.56 TRIGGER_ROUTINE_USAGE base table

Function

The TRIGGER_ROUTINE_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in a <trigger definition>.

Definition

```
CREATE TABLE TRIGGER_ROUTINE_USAGE (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_ROUTINE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

    CONSTRAINT TRIGGER_ROUTINE_USAGE_CHECK_REFERENCES_ROUTINES
        CHECK ( SPECIFIC_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
              ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
                FROM ROUTINES ) ),

    CONSTRAINT TRIGGER_ROUTINE_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS
)
```

Description

- 1) The TRIGGER_ROUTINE_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in a <trigger definition>.
- 2) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the trigger being described.
- 3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

6.57 TRIGGER_SEQUENCE_USAGE base table

Function

The TRIGGER_SEQUENCE_USAGE table has one row for each external sequence generator identified in a trigger.

Definition

```
CREATE TABLE TRIGGER_SEQUENCE_USAGE (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_SEQUENCE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ),

    CONSTRAINT TRIGGER_SEQUENCE_USAGE_CHECK_REFERENCES_SEQUENCES
        CHECK ( SEQUENCE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ) IN
              ( SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
                FROM SEQUENCES ) ),

    CONSTRAINT TRIGGER_SEQUENCE_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS
)
```

Description

- 1) The TRIGGER_SEQUENCE_USAGE table has one row for each sequence generator *SEQ* identified by a <sequence generator name> contained in the <triggered action> of a trigger *TR* being described.
- 2) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TR*.
- 3) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, and SEQUENCE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SEQ*.

6.58 TRIGGER_TABLE_USAGE base table

Function

The TRIGGER_TABLE_USAGE base table has one row for each table identified by a <table name> contained in the <search condition> of a <triggered action> or referenced in a <triggered SQL statement> of a <trigger definition>.

Definition

```
CREATE TABLE TRIGGER_TABLE_USAGE      (
    TRIGGER_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_TABLE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT TRIGGER_TABLE_USAGE_CHECK_REFERENCES_TABLES
        CHECK ( TABLE_CATALOG NOT IN
                ( SELECT CATALOG_NAME FROM SCHEMATA )
              OR
                ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
                ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLES ) ),

    CONSTRAINT TRIGGER_TABLE_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS
)
```

Description

- 1) The TRIGGER_TABLE_USAGE base table has one row for each table identified by at least one of:
 - a) A <table reference> contained in a <query expression> simply contained in a <cursor specification> or an <insert statement> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.
 - b) A <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.

- c) A <table reference> contained in a <search condition> contained in a <delete statement: searched> or an <update statement: searched> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.
 - d) A <table reference> contained in a <value expression> simply contained in a <update source> or an <assigned row> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.
 - e) A <table name> contained in either a <delete statement: positioned> or a <delete statement: searched> contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.
 - f) A <table name> immediately contained in an <insert statement> that does not contain an <insert column list> and that is contained in a <triggered SQL statement> contained in a <trigger definition> of a trigger.
- 2) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the trigger being described.
 - 3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> contained in the <search condition> of a <triggered action>, or referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

6.59 TRIGGERS base table

Function

The TRIGGERS base table has one row for each trigger. It effectively contains a representation of the trigger descriptors.

Definition

```
CREATE TABLE TRIGGERS (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    EVENT_MANIPULATION       INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_EVENT_MANIPULATION_CHECK
    CHECK ( EVENT_MANIPULATION IN
        ( 'INSERT', 'DELETE', 'UPDATE' ) ),
    EVENT_OBJECT_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRIGGERS_EVENT_OBJECT_CATALOG_NOT_NULL
    NOT NULL,
    EVENT_OBJECT_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRIGGERS_EVENT_OBJECT_SCHEMA_NOT_NULL
    NOT NULL,
    EVENT_OBJECT_TABLE       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRIGGERS_EVENT_OBJECT_TABLE_NOT_NULL
    NOT NULL,
    ACTION_ORDER             INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT TRIGGERS_ACTION_ORDER_NOT_NULL
    NOT NULL,
    ACTION_CONDITION         INFORMATION_SCHEMA.CHARACTER_DATA,
    ACTION_STATEMENT         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_ACTION_STATEMENT_NOT_NULL
    NOT NULL,
    ACTION_ORIENTATION       INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_ACTION_ORIENTATION_CHECK
    CHECK ( ACTION_ORIENTATION IN
        ( 'ROW', 'STATEMENT' ) ),
    ACTION_TIMING            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_ACTION_TIMING_CHECK
    CHECK ( ACTION_TIMING IN
        ( 'BEFORE', 'AFTER' ) ),
    ACTION_REFERENCE_OLD_TABLE INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ACTION_REFERENCE_NEW_TABLE INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ACTION_REFERENCE_OLD_ROW  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ACTION_REFERENCE_NEW_ROW  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CREATED                   INFORMATION_SCHEMA.TIME_STAMP,

    CONSTRAINT TRIGGERS_PRIMARY_KEY
    PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME ),

    CONSTRAINT TRIGGERS_FOREIGN_KEY_SCHEMATA
```

```

FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
REFERENCES SCHEMATA,

CONSTRAINT TRIGGERS_REFERENCES_TABLES
CHECK ( EVENT_OBJECT_CATALOG <>
      ANY ( SELECT CATALOG_NAME
            FROM SCHEMATA )
OR
      ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE ) IN
      ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM TABLES ) )
)

```

Description

- 1) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the fully qualified name of the trigger being described.
- 2) The values of EVENT_MANIPULATION have the following meaning:

INSERT	The <trigger event> is INSERT.
DELETE	The <trigger event> is DELETE.
UPDATE	The <trigger event> is UPDATE.

- 3) The values of EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, and EVENT_OBJECT_TABLE are the fully qualified name of the subject table of the trigger being described.
- 4) The values of ACTION_TIMING have the following meaning:

BEFORE	The <trigger action time> is BEFORE.
AFTER	The <trigger action time> is AFTER.

- 5) The value of ACTION_REFERENCE_OLD_TABLE is the <old values table alias> of the trigger being described.
- 6) The value of ACTION_REFERENCE_NEW_TABLE is the <new values table alias> of the trigger being described.
- 7) The value of ACTION_REFERENCE_OLD_ROW is the <old values correlation name> of the trigger being described.
- 8) The value of ACTION_REFERENCE_NEW_ROW is the <new values correlation name> of the trigger being described.
- 9) The value of ACTION_ORDER is the ordinal position of the trigger in the list of triggers with the same EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE, EVENT_MANIPULATION, CONDITION_TIMING, and ACTION_ORIENTATION.

- 10) The value of ACTION_CONDITION is a character representation of the <search condition> in the <triggered action> of the trigger being described.
- 11) ACTION_STATEMENT is a character representation of the <triggered SQL statement> in the <triggered action> of the trigger being described.
- 12) The values of ACTION_ORIENTATION have the following meanings:

ROW	The <triggered action> specifies FOR EACH ROW.
STATE- MENT	The <triggered action> specifies FOR EACH STATEMENT.

- 13) The value of CREATED is the value of CURRENT_TIMESTAMP at the time when the trigger being described was created.

6.60 USAGE_PRIVILEGES base table

Function

The USAGE_PRIVILEGES table has one row for each usage privilege descriptor. It effectively contains a representation of the usage privilege descriptors.

Definition

```
CREATE TABLE USAGE_PRIVILEGES (
  GRANTOR          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  OBJECT_CATALOG   INFORMATION_SCHEMA.SQL_IDENTIFIER,
  OBJECT_SCHEMA    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  OBJECT_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  OBJECT_TYPE      INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT USAGE_PRIVILEGES_OBJECT_TYPE_CHECK
    CHECK ( OBJECT_TYPE IN
      ( 'DOMAIN', 'CHARACTER SET',
        'COLLATION', 'TRANSLATION', 'SEQUENCE' ) ),
  IS_GRANTABLE     INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT USAGE_PRIVILEGES_IS_GRANTABLE_NOT_NULL
    NOT NULL
  CONSTRAINT USAGE_PRIVILEGES_IS_GRANTABLE_CHECK
    CHECK ( IS_GRANTABLE IN
      ( 'YES', 'NO' ) ),

  CONSTRAINT USAGE_PRIVILEGES_PRIMARY_KEY
    PRIMARY KEY ( GRANTOR, GRANTEE, OBJECT_CATALOG, OBJECT_SCHEMA,
      OBJECT_NAME, OBJECT_TYPE ),

  CONSTRAINT USAGE_PRIVILEGES_CHECK_REFERENCES_OBJECT
    CHECK ( ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE ) IN
      ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN'
        FROM DOMAINS
      UNION
        SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
          'CHARACTER SET'
        FROM CHARACTER_SETS
      UNION
        SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME, 'COLLATION'
        FROM COLLATIONS
      UNION
        SELECT TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME,
          'TRANSLATION'
        FROM TRANSLATIONS
      UNION
        SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME,
          'SEQUENCE'
        FROM SEQUENCES ) ),
```

6.60 USAGE_PRIVILEGES base table

```

CONSTRAINT USAGE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
  FOREIGN KEY ( GRANTOR )
    REFERENCES AUTHORIZATIONS,

CONSTRAINT USAGE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS GRANTEE
  FOREIGN KEY ( GRANTEE )
    REFERENCES AUTHORIZATIONS
)

```

Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted usage privileges, on the object of the type identified by OBJECT_TYPE that is identified by OBJECT_CATALOG, OBJECT_SCHEMA, and OBJECT_NAME, to the user or role identified by the value of GRANTEE for the usage privilege being described.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or “PUBLIC” to indicate all users, to whom the usage privilege being described is granted.
- 3) The values of OBJECT_CATALOG, OBJECT_SCHEMA, and OBJECT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the object to which the privilege applies.
- 4) The values of OBJECT_TYPE have the following meanings:

DOMAIN	The object to which the privilege applies is a domain.
CHARACTER SET	The object to which the privilege applies is a character set.
COLLATION	The object to which the privilege applies is a collation.
TRANSLATION	The object to which the privilege applies is a transliteration.
SEQUENCE	The object to which the privilege applies is a sequence generator.

- 5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.61 USER_DEFINED_TYPE_PRIVILEGES base table

Function

The USER_DEFINED_TYPE_PRIVILEGES table has one row for each user-defined type privilege descriptor. It effectively contains a representation of the privilege descriptors.

Definition

```
CREATE TABLE USER_DEFINED_TYPE_PRIVILEGES (
    GRANTOR                                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE                         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PRIVILEGE_TYPE_CHECK
        CHECK ( PRIVILEGE_TYPE = 'TYPE USAGE' ),
    IS_GRANTABLE                           INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_IS_GRANTABLE_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_IS_GRANTABLE_CHECK
        CHECK ( IS_GRANTABLE IN
            ( 'YES', 'NO' ) ),

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY( GRANTOR, GRANTEE,
            USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME, PRIVILEGE_TYPE ),

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_FOREIGN_KEY_USER_DEFINED_TYPE
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS_GRANTOR
        FOREIGN KEY ( GRANTOR )
        REFERENCES AUTHORIZATIONS,

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_FOREIGN_KEY_AUTHORIZATIONS GRANTEE
        FOREIGN KEY ( GRANTEE )
        REFERENCES AUTHORIZATIONS
)
```

Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted access privileges on the TYPE USAGE privilege being described to the user or role identified by the value of GRANTEE.

6.61 USER_DEFINED_TYPE_PRIVILEGES base table

- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or “PUBLIC” to indicate all users, to whom the user-defined type privilege being described is granted.
- 3) The value of PRIVILEGE_TYPE has the following meaning:

TYPE USAGE	The user has TYPE USAGE privilege on this user-defined type.
---------------	--

- 4) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable

6.62 USER_DEFINED_TYPES base table

Function

The USER_DEFINED_TYPES table has one row for each user-defined type.

Definition

```
CREATE TABLE USER_DEFINED_TYPES (
    USER_DEFINED_TYPE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATEGORY          INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_USER_DEFINED_TYPE_CATEGORY_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPES_USER_DEFINED_TYPE_CATEGORY_CHECK
        CHECK ( USER_DEFINED_TYPE_CATEGORY IN
            ( 'STRUCTURED', 'DISTINCT' ) ),
    SOURCE_DTD_IDENTIFIER                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_INSTANTIABLE                     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_IS_INSTANTIABLE_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPES_IS_INSTANTIABLE_CHECK
        CHECK ( IS_INSTANTIABLE IN
            ( 'YES', 'NO' ) ),
    IS_FINAL                            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_IS_FINAL_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPES_IS_FINAL_CHECK
        CHECK ( IS_FINAL IN
            ( 'YES', 'NO' ) ),
    ORDERING_FORM                       INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_CHECK
        CHECK ( ORDERING_FORM IN
            ( 'NONE', 'FULL', 'EQUALS' ) ),
    ORDERING_CATEGORY                   INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_ORDERING_CATEGORY_CHECK
        CHECK ( ORDERING_CATEGORY IN
            ( 'RELATIVE', 'MAP', 'STATE' ) ),
    ORDERING_ROUTINE_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDERING_ROUTINE_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDERING_ROUTINE_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_TYPE                     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_REFERENCE_TYPE_CHECK
        CHECK ( REFERENCE_TYPE IN
            ( 'SYSTEM GENERATED', 'USER GENERATED', 'DERIVED' ) ),
    REF_DTD_IDENTIFIER                  INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT USER_DEFINED_TYPES_PRIMARY_KEY
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
```

6.62 USER_DEFINED_TYPES base table

```

        USER_DEFINED_TYPE_NAME ),

CONSTRAINT USER_DEFINED_TYPES_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA )
        REFERENCES SCHEMATA,

CONSTRAINT USER_DEFINED_TYPES_FOREIGN_KEY_ROUTINES
    FOREIGN KEY ( ORDERING_ROUTINE_CATALOG, ORDERING_ROUTINE_SCHEMA,
        ORDERING_ROUTINE_NAME )
        REFERENCES ROUTINES,

CONSTRAINT USER_DEFINED_TYPES_CHECK_SOURCE_TYPE
    CHECK ( ( USER_DEFINED_TYPE_CATEGORY = 'STRUCTURED'
        AND
            SOURCE_DTD_IDENTIFIER IS NULL )
    OR
        ( USER_DEFINED_TYPE_CATEGORY = 'DISTINCT'
        AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, SOURCE_DTD_IDENTIFIER ) IS NOT NULL
        AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, SOURCE_DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR
                WHERE DATA_TYPE NOT IN
                    ( 'ROW', 'ARRAY', 'MULTISET', 'REFERENCE',
                        'USER-DEFINED' ) ) ) ),

CONSTRAINT USER_DEFINED_TYPES_CHECK_USER_GENERATED_REFERENCE_TYPE
    CHECK ( ( REFERENCE_TYPE <> 'USER GENERATED'
        AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                REF_DTD_IDENTIFIER ) NOT IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR ) )
    OR
        ( REFERENCE_TYPE = 'USER GENERATED'
        AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                REF_DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR
                WHERE DATA_TYPE IN
                    ( 'CHARACTER', 'INTEGER' ) ) ) )
)

```

Description

- 1) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA and USER_DEFINED_TYPE_NAME are the fully qualified name of the user-defined type being described.
- 2) The values of USER_DEFINED_TYPE_CATEGORY have the following meanings:

STRUC-TURED	The user-defined type is a structured type.
DISTINCT	The user-defined type is a distinct type.

- 3) If USER_DEFINED_TYPE_CATEGORY is 'STRUCTURED', then the value of SOURCE_DTD_IDENTIFIER is the null value; otherwise, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, and SOURCE_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the source type of the distinct type.
- 4) The values of IS_INSTANTIABLE have the following meanings:

YES	The user-defined type is instantiable.
NO	The user-defined type is not instantiable.

- 5) The values of IS_FINAL have the following meanings:

YES	The user-defined type cannot have subtypes.
NO	The user-defined type can have subtypes.

- 6) The values of ORDERING_FORM have the following meanings:

NONE	Two values of this type may not be compared.
FULL	Two values of this type may be compared for equality or relative order.
EQUALS	Two values of this type may be compared for equality only.

- 7) The values of ORDERING_CATEGORY have the following meanings:

RELATIVE	Two values of this type can be compared with a relative routine.
MAP	Two values of this type may be compared with a map routine.
STATE	Two values of this type may be compared with a state routine.

6.62 USER_DEFINED_TYPES base table

- 8) The values of ORDER_ROUTINE_CATALOG, ORDER_ROUTINE_SCHEMA, and ORDER_ROUTINE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine used for ordering the user-defined type.
- 9) The values of REFERENCE_TYPE have the following meanings:

SYSTEM GENER- ATED	REF values for tables of this structured type are system generated.
USER GEN- ERATED	REF values for tables of this structured type are user generated of the data type specified by USER GENERATED TYPE.
DERIVED	REF values for tables of this structured type are derived from the columns corresponding to the specified attributes.

- 10) If the value of REFERENCE_TYPE is not 'USER GENERATED', then the value of REF_DTD_IDENTIFIER is the null value; otherwise, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, and REF_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the user-generated REF values of the structured type.

6.63 VIEW_COLUMN_USAGE base table

Function

The VIEW_COLUMN_USAGE table has one row for each column of a table that is explicitly or implicitly referenced in the <query expression> of the view being described.

Definition

```
CREATE TABLE VIEW_COLUMN_USAGE (
  VIEW_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT VIEW_COLUMN_USAGE_PRIMARY_KEY
    PRIMARY KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

  CONSTRAINT VIEW_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
    CHECK ( TABLE_CATALOG NOT IN
            ( SELECT CATALOG_NAME FROM SCHEMATA )
          OR
            ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
              FROM COLUMNS ) ),

  CONSTRAINT VIEW_COLUMN_USAGE_FOREIGN_KEY_VIEWS
    FOREIGN KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME )
      REFERENCES VIEWS
)
```

Description

- 1) The values of VIEW_CATALOG, VIEW_SCHEMA, and VIEW_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the view being described.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column of a table that is explicitly or implicitly referenced in the <query expression> of the view being described.

6.64 VIEW_ROUTINE_USAGE base table

Function

The VIEW_ROUTINE_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in a <view definition>.

Definition

```
CREATE TABLE VIEW_ROUTINE_USAGE (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT VIEW_ROUTINE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                     SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

    CONSTRAINT VIEW_ROUTINE_USAGE_CHECK_REFERENCES_VIEWS
        CHECK ( TABLE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM VIEWS ) ),

    CONSTRAINT VIEW_ROUTINE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

Description

- 1) The VIEW_ROUTINE_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <query expression> of the <view definition> of the view being described.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the viewed table being described.
- 3) The values of ROUTINE_CATALOG, ROUTINE_SCHEMA, and ROUTINE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

6.65 VIEW_TABLE_USAGE base table

Function

The VIEW_TABLE_USAGE table has one row for each table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of a view.

Definition

```
CREATE TABLE VIEW_TABLE_USAGE (
  VIEW_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT VIEW_TABLE_USAGE_PRIMARY_KEY
    PRIMARY KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

  CONSTRAINT VIEW_TABLE_USAGE_CHECK_REFERENCES_TABLES
    CHECK ( TABLE_CATALOG NOT IN
            ( SELECT CATALOG_NAME
              FROM SCHEMATA )
          OR
            ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES ) ),

  CONSTRAINT VIEW_TABLE_USAGE_FOREIGN_KEY_VIEWS
    FOREIGN KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME )
      REFERENCES VIEWS
)
```

Description

- 1) The values of VIEW_CATALOG, VIEW_SCHEMA, and VIEW_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the view being described.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the view being described.

6.66 VIEWS base table

Function

The VIEWS table contains one row for each row in the TABLES table with a TABLE_TYPE of 'VIEW'. Each row describes the query expression that defines a view. The table effectively contains a representation of the view descriptors.

Definition

```
CREATE TABLE VIEWS (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    VIEW_DEFINITION          INFORMATION_SCHEMA.CHARACTER_DATA,
    CHECK_OPTION            INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT CHECK_OPTION_NOT_NULL
            NOT NULL
        CONSTRAINT CHECK_OPTION_CHECK
            CHECK ( CHECK_OPTION IN
                ( 'CASCADED', 'LOCAL', 'NONE' ) ),
    IS_UPDATABLE            INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT IS_UPDATABLE_NOT_NULL
            NOT NULL
        CONSTRAINT IS_UPDATABLE_CHECK
            CHECK ( IS_UPDATABLE IN
                ( 'YES', 'NO' ) ),

    CONSTRAINT VIEWS_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT VIEWS_IN_TABLES_CHECK
        CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES
              WHERE TABLE_TYPE = 'VIEW' ) ),

    CONSTRAINT VIEWS_IS_UPDATABLE_CHECK_OPTION_CHECK
        CHECK ( ( IS_UPDATABLE, CHECK_OPTION ) NOT IN
            ( VALUES ( 'NO', 'CASCADED' ), ( 'NO', 'LOCAL' ) ) )
)
```

Description

- 1) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the fully qualified name of the viewed table.
- 2) Case:

- a) If the character representation of the <query expression> contained in the corresponding view descriptor can be represented without truncation, then the value of VIEW_DEFINITION is that character representation.
- b) Otherwise, the value of VIEW_DEFINITION is the null value.

NOTE 13 — Any implicit column references that were contained in the <query expression> associated with the <view definition> are replaced by explicit column references in VIEW_DEFINITION.

- 3) The values of CHECK_OPTION have the following meanings:

CASCADED	The corresponding view descriptor indicates that the view has the CHECK OPTION that is to be applied as CASCADED.
LOCAL	The corresponding view descriptor indicates that the view has the CHECK OPTION that is to be applied as LOCAL.
NONE	The corresponding view descriptor indicates that the view does not have the CHECK OPTION.

- 4) The values of IS_UPDATABLE have the following meanings:

YES	The view is effectively updatable.
NO	The view is not effectively updatable.

This page intentionally left blank.

7 Conformance

7.1 Claims of conformance to SQL/Schemata

No requirements in addition to the requirements of ISO/IEC 9075-1, [Clause 8](#), “Conformance”, are required for a claim of conformance to this part of ISO/IEC 9075.

7.2 Additional conformance requirements for SQL/Schemata

A claim of conformance that includes a claim of conformance to a feature in this part of ISO/IEC 9075 shall also include a claim of conformance to the same feature, if present, in ISO/IEC 9075-2.

7.3 Implied feature relationships of SQL/Schemata

Table 1 — Implied feature relationships of SQL/Schemata

Feature ID	Feature Description	Implied Feature ID	Implied Feature Description
T011	Timestamp in Information Schema	F411	Time zone specification
T011	Timestamp in Information Schema	F555	Enhanced seconds precision

This page intentionally left blank.

Annex A

(informative)

SQL Conformance Summary

This Annex modifies Annex A, “SQL Conformance Summary”, in ISO/IEC 9075-2.

The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

- 1) Specifications for Feature F231, “Privilege tables”:
 - a) Subclause 5.18, “COLUMN_PRIVILEGES view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_PRIVILEGES.
 - b) Subclause 5.23, “DATA_TYPE_PRIVILEGES view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES.
 - c) Subclause 5.37, “ROLE_COLUMN_GRANTS view”:
 - i) Without Feature F231, “Privilege tables”, and Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_COLUMN_GRANTS.
 - d) Subclause 5.38, “ROLE_ROUTINE_GRANTS view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS.
 - e) Subclause 5.39, “ROLE_TABLE_GRANTS view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_GRANTS.
 - f) Subclause 5.42, “ROLE_UDT_GRANTS view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_UDT_GRANTS.
 - g) Subclause 5.44, “ROUTINE_PRIVILEGES view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_PRIVILEGES.
 - h) Subclause 5.60, “TABLE_PRIVILEGES view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_PRIVILEGES.

- i) Subclause 5.70, “UDT_PRIVILEGES view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.UDT_PRIVILEGES.
 - j) Subclause 5.71, “USAGE_PRIVILEGES view”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.USAGE_PRIVILEGES.
 - k) Subclause 5.77, “Short name views”:
 - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUT_GRANTS.
- 2) Specifications for Feature F251, “Domain support”:
- a) Subclause 5.3, “CARDINAL_NUMBER domain”:
 - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.CARDINAL_NUMBER.
 - b) Subclause 5.4, “CHARACTER_DATA domain”:
 - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHARACTER_DATA.
 - c) Subclause 5.5, “SQL_IDENTIFIER domain”:
 - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IDENTIFIER.
 - d) Subclause 5.6, “TIME_STAMP domain”:
 - i) Without Feature F251, “Domain support”, and Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TIME_STAMP.
 - e) Subclause 5.26, “DOMAIN_CONSTRAINTS view”:
 - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAIN_CONSTRAINTS.
 - f) Subclause 5.27, “DOMAINS view”:
 - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.
 - g) Subclause 5.77, “Short name views”:
 - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS_S.
- 3) Specifications for Feature F341, “Usage tables”:
- a) Subclause 5.12, “CHECK_CONSTRAINT_ROUTINE_USAGE view”:

- i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.
- b) Subclause 5.16, “COLUMN_COLUMN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.
- c) Subclause 5.17, “COLUMN_DOMAIN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE.
- d) Subclause 5.19, “COLUMN_UDT_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_UDT_USAGE.
- e) Subclause 5.21, “CONSTRAINT_COLUMN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE.
- f) Subclause 5.22, “CONSTRAINT_TABLE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.
- g) Subclause 5.31, “KEY_COLUMN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE.
- h) Subclause 5.41, “ROLE_USAGE_GRANTS view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_USAGE_GRANTS.
- i) Subclause 5.43, “ROUTINE_COLUMN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COLUMN_USAGE.
- j) Subclause 5.45, “ROUTINE_ROUTINE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_ROUTINE_USAGE.
- k) Subclause 5.46, “ROUTINE_SEQUENCE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.
- l) Subclause 5.47, “ROUTINE_TABLE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_TABLE_USAGE.

- m) Subclause 5.65, “TRIGGER_COLUMN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_COLUMN_USAGE.
- n) Subclause 5.66, “TRIGGER_ROUTINE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE.
- o) Subclause 5.67, “TRIGGER_SEQUENCE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.
- p) Subclause 5.68, “TRIGGER_TABLE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE.
- q) Subclause 5.73, “VIEW_COLUMN_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_COLUMN_USAGE.
- r) Subclause 5.74, “VIEW_ROUTINE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_ROUTINE_USAGE.
- s) Subclause 5.75, “VIEW_TABLE_USAGE view”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_TABLE_USAGE.
- t) Subclause 5.77, “Short name views”:
 - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference the INFORMATION_SCHEMA.TRIG_TABLE_USAGE view.
 - ii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_UPDATE_COLS
 - iii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COL_DOMAIN_USAGE.
 - iv) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONST_COL_USAGE.
 - v) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONST_TABLE_USAGE.
 - vi) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE_S.
 - vii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COL_USAGE.

- viii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_TABLE_USAGE.
- ix) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_ROUT_USAGE_S.
- x) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTR_ROUT_USE_S.
- xi) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_ROUT_USAGE_S.
- xii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_SEQ_USAGE_S.
- xiii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_COLUMN_USAGE.
- xiv) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE.

4) Specifications for Feature F391, “Long identifiers”:

- a) Subclause 5.2, “INFORMATION_SCHEMA.CATALOG_NAME base table”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.
- b) Subclause 5.7, “ADMINISTRABLE_ROLE_AUTHORIZATIONS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.
- c) Subclause 5.10, “ATTRIBUTES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.
- d) Subclause 5.11, “CHARACTER_SETS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHARACTER_SETS.
- e) Subclause 5.12, “CHECK_CONSTRAINT_ROUTINE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.
- f) Subclause 5.14, “COLLATIONS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS.
- g) Subclause 5.15, “COLLATION_CHARACTER_SET_APPLICABILITY view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.

- h) Subclause 5.16, “COLUMN_COLUMN_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.
- i) Subclause 5.17, “COLUMN_DOMAIN_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE.
- j) Subclause 5.20, “COLUMNS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.
- k) Subclause 5.21, “CONSTRAINT_COLUMN_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE.
- l) Subclause 5.22, “CONSTRAINT_TABLE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.
- m) Subclause 5.27, “DOMAINS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.
- n) Subclause 5.28, “ELEMENT_TYPES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.
- o) Subclause 5.30, “FIELDS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.
- p) Subclause 5.31, “KEY_COLUMN_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE.
- q) Subclause 5.32, “METHOD_SPECIFICATION_PARAMETERS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.
- r) Subclause 5.33, “METHOD_SPECIFICATIONS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.
- s) Subclause 5.34, “PARAMETERS view”:

- i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.PARAMETERS.
- t) Subclause 5.35, “REFERENCED_TYPES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.REFERENCED_TYPES.
- u) Subclause 5.36, “REFERENTIAL_CONSTRAINTS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS.
- v) Subclause 5.38, “ROLE_ROUTINE_GRANTS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS.
- w) Subclause 5.40, “ROLE_TABLE_METHOD_GRANTS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.
- x) Subclause 5.43, “ROUTINE_COLUMN_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COLUMN_USAGE.
- y) Subclause 5.45, “ROUTINE_ROUTINE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_ROUTINE_USAGE.
- z) Subclause 5.46, “ROUTINE_SEQUENCE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.
- aa) Subclause 5.47, “ROUTINE_TABLE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_TABLE_USAGE.
- ab) Subclause 5.48, “ROUTINES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.
- ac) Subclause 5.49, “SCHEMATA view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SCHEMATA.
- ad) Subclause 5.50, “SEQUENCES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES.

- ae) Subclause 5.52, “SQL_IMPLEMENTATION_INFO view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO.
- af) Subclause 5.53, “SQL_LANGUAGES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_LANGUAGES.
- ag) Subclause 5.57, “SQL_SIZING_PROFILES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_SIZING_PROFILES.
- ah) Subclause 5.59, “TABLE_METHOD_PRIVILEGES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES.
- ai) Subclause 5.61, “TABLES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLES.
- aj) Subclause 5.63, “TRANSLATIONS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSLATIONS.
- ak) Subclause 5.64, “TRIGGERED_UPDATE_COLUMNS view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS.
- al) Subclause 5.65, “TRIGGER_COLUMN_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERED_COLUMN_USAGE.
- am) Subclause 5.66, “TRIGGER_ROUTINE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE.
- an) Subclause 5.67, “TRIGGER_SEQUENCE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.
- ao) Subclause 5.68, “TRIGGER_TABLE_USAGE view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE.
- ap) Subclause 5.69, “TRIGGERS view”:

- i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.
- aq) Subclause 5.72, “USER_DEFINED_TYPES view”:
 - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.USER_DEFINED_TYPES.
- 5) Specifications for Feature F502, “Enhanced documentation tables”:
 - a) Subclause 5.52, “SQL_IMPLEMENTATION_INFO view”:
 - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO.
 - b) Subclause 5.54, “SQL_PACKAGES view”:
 - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_PACKAGES.
 - c) Subclause 5.55, “SQL_PARTS view”:
 - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_PARTS.
 - d) Subclause 5.57, “SQL_SIZING_PROFILES view”:
 - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_SIZING_PROFILES.
 - e) Subclause 5.77, “Short name views”:
 - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPL_INFO.
 - ii) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_SIZING_PROFS.
- 6) Specifications for Feature F521, “Assertions”:
 - a) Subclause 5.9, “ASSERTIONS view”:
 - i) Without Feature F521, “Assertions”, conforming SQL language shall not reference INFORMATION_SCHEMA.ASSERTIONS.
- 7) Specifications for Feature F651, “Catalog name qualifiers”:
 - a) Subclause 5.2, “INFORMATION_SCHEMA_CATALOG_NAME base table”:
 - i) Without Feature F651, “Catalog name qualifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.
- 8) Specifications for Feature F690, “Collation support ”:
 - a) Subclause 5.15, “COLLATION_CHARACTER_SET_APPLICABILITY view”:

- i) Without Feature F690, “Collation support”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.
- 9) Specifications for Feature F691, “Collation and translation”:
 - a) Subclause 5.14, “COLLATIONS view”:
 - i) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS.
 - b) Subclause 5.63, “TRANSLATIONS view”:
 - i) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSLATIONS.
 - c) Subclause 5.77, “Short name views”:
 - i) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS_S.
 - ii) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSLATIONS_S.
- 10) Specifications for Feature F696, “Additional translation documentation”:
 - a) Subclause 5.63, “TRANSLATIONS view”:
 - i) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA, or TRANSLATION_SOURCE_NAME.
 - b) Subclause 5.77, “Short name views”:
 - i) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANS_SRC_CATALOG, TRANS_SRC_SCHEMA, or TRANS_SRC_NAME.
- 11) Specifications for Feature S023, “Basic structured types”:
 - a) Subclause 5.10, “ATTRIBUTES view”:
 - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.
 - b) Subclause 5.32, “METHOD_SPECIFICATION_PARAMETERS view”:
 - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.
 - c) Subclause 5.33, “METHOD_SPECIFICATIONS view”:
 - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.
 - d) Subclause 5.77, “Short name views”:
 - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES_S.

- ii) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECS.
- iii) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC_PARAMS.

12) Specifications for Feature S024, “Enhanced structured types”:

- a) Subclause 5.25, “DIRECT_SUPERTYPES view”:
 - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.DIRECT_SUPERTYPES.
- b) Subclause 5.40, “ROLE_TABLE_METHOD_GRANTS view”:
 - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.
- c) Subclause 5.59, “TABLE_METHOD_PRIVILEGES view”:
 - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES.
- d) Subclause 5.77, “Short name views”:
 - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVS.

13) Specifications for Feature S041, “Basic reference types”:

- a) Subclause 5.35, “REFERENCED_TYPES view”:
 - i) Without Feature S041, “Basic reference types”, conforming SQL language shall not reference INFORMATION_SCHEMA.REFERENCED_TYPES.

14) Specifications for Feature S081, “Subtables”:

- a) Subclause 5.24, “DIRECT_SUPERTABLES view”:
 - i) Without Feature S081, “Subtables”, conforming SQL language shall not reference INFORMATION_SCHEMA.DIRECT_SUPERTABLES.

15) Specifications for Feature S091, “Basic array support”:

- a) Subclause 5.28, “ELEMENT_TYPES view”:
 - i) Without Feature S091, “Basic array support”, or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.

16) Specifications for Feature S241, “Transform functions”:

- a) Subclause 5.62, “TRANSFORMS view”:
 - i) Without Feature S241, “Transform functions”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRANSFORMS.

17) Specifications for Feature S271, “Basic multiset support”:

a) Subclause 5.28, “ELEMENT_TYPES view”:

- i) Without Feature S091, “Basic array support”, or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.

18) Specifications for Feature T011, “Timestamp in Information Schema”:

a) Subclause 5.6, “TIME_STAMP domain”:

- i) Without Feature F251, “Domain support”, and Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TIME_STAMP.

b) Subclause 5.33, “METHOD_SPECIFICATIONS view”:

- i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.CREATED.
- ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.LAST_ALTERED.

c) Subclause 5.48, “ROUTINES view”:

- i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.CREATED.
- ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.LAST_ALTERED.

d) Subclause 5.69, “TRIGGERS view”:

- i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.TRIGGER_CREATED.

e) Subclause 5.77, “Short name views”:

- i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC.CREATED.
- ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC.LAST_ALTERED.
- iii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.CREATED.
- iv) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.LAST_ALTERED.
- v) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.CREATED.

19) Specifications for Feature T051, “Row types”:

a) Subclause 5.30, “FIELDS view”:

- i) Without Feature T051, “Row types”, conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.

20) Specifications for Feature T111, “Updatable joins, unions, and columns”:

a) Subclause 5.20, “COLUMNS view”:

- i) Without Feature T111, “Updatable joins, unions, and columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.IS_UPDATABLE.

b) Subclause 5.77, “Short name views”:

- i) Without Feature T111, “Updatable joins, unions, and columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.S.IS_UPDATABLE.

21) Specifications for Feature T175, “Generated columns”:

a) Subclause 5.16, “COLUMN_COLUMN_USAGE view”:

- i) Without Feature T175, “Generated columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.

b) Subclause 5.20, “COLUMNS view”:

- i) Without Feature T175, “Generated columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.IS_GENERATED.
- ii) Without Feature T175, “Generated columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.GENERATION_EXPRESSION.

c) Subclause 5.77, “Short name views”:

- i) Without Feature T175, “Generated columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COL_COL_USAGE.
- ii) Without Feature T175, “Generated columns”, confirming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.S.GENERATION_EXPR.

22) Specifications for Feature T176, “Sequence generator support”:

a) Subclause 5.46, “ROUTINE_SEQUENCE_USAGE view”:

- i) Without Feature T176, “Sequence generator support”, confirming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.

b) Subclause 5.50, “SEQUENCES view”:

- i) Without Feature T176, “Sequence generator support”, confirming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES.

c) Subclause 5.67, “TRIGGER_SEQUENCE_USAGE view”:

- i) Without Feature T176, “Sequence generator support”, confirming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

d) Subclause 5.77, “Short name views”:

- i) Without Feature T176, “Sequence generator support”, confirming SQL language shall not reference INFORMATION_SCHEMA.ROUT_SEQ_USAGE_S.

- ii) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES_S.

23) Specifications for Feature T211, “Basic trigger capability”:

a) Subclause 5.64, “TRIGGERED_UPDATE_COLUMNS view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS.

b) Subclause 5.65, “TRIGGER_COLUMN_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_COLUMN_USAGE.

c) Subclause 5.66, “TRIGGER_ROUTINE_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE.

d) Subclause 5.67, “TRIGGER_SEQUENCE_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

e) Subclause 5.68, “TRIGGER_TABLE_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE.

f) Subclause 5.69, “TRIGGERS view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.

g) Subclause 5.77, “Short name views”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_UPDATE_COLS
- ii) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference the INFORMATION_SCHEMA.TRIG_TABLE_USAGE view.
- iii) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_ROUT_USAGE_S.
- iv) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_SEQ_USAGE_S.
- v) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.CREATED.
- vi) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS_S.
- vii) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_COLUMN_USAGE.

24) Specifications for Feature T272, “Enhanced savepoint management”:

a) Subclause 5.48, “ROUTINES view”:

- i) Without Feature T272, “Enhanced savepoint management”, conforming SQL-language shall not reference INFORMATION_SCHEMA.ROUTINES.NEW_SAVEPOINT_LEVEL.

25) Specifications for Feature T331, “Basic roles”:

a) Subclause 5.7, “ADMINISTRABLE_ROLE_AUTHORIZATIONS view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.

b) Subclause 5.8, “APPLICABLE_ROLES view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.APPLICABLE_ROLES.

c) Subclause 5.29, “ENABLED_ROLES view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ENABLED_ROLES.

d) Subclause 5.37, “ROLE_COLUMN_GRANTS view”:

- i) Without Feature F231, “Privilege tables”, and Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_COLUMN_GRANTS.

e) Subclause 5.38, “ROLE_ROUTINE_GRANTS view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS.

f) Subclause 5.39, “ROLE_TABLE_GRANTS view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_GRANTS.

g) Subclause 5.40, “ROLE_TABLE_METHOD_GRANTS view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.

h) Subclause 5.41, “ROLE_USAGE_GRANTS view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_USAGE_GRANTS.

i) Subclause 5.42, “ROLE_UDT_GRANTS view”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_UDT_GRANTS.

j) Subclause 5.77, “Short name views”:

- i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMIN_ROLE_AUTHS.
- ii) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUT_GRANTS.

Annex B

(informative)

Implementation-defined elements

This Annex modifies Annex B, “Implementation-defined elements”, in ISO/IEC 9075-2.

This Annex references those features that are identified in the body of this part of ISO/IEC 9075 as implementation-defined.

- 1) Subclause 6.45, “SQL_LANGUAGES base table”:
 - a) The value of SQL_LANGUAGE_IMPLEMENTATION is implementation-defined. If the value of SQL_LANGUAGE_SOURCE is not 'ISO 9075', then the values of all other columns are implementation-defined.

This page intentionally left blank.

Annex C

(informative)

Deprecated features

This Annex modifies Annex D, “Deprecated features”, in ISO/IEC 9075-2.

It is intended that the following features will be removed at a later date from a revised version of this part of ISO/IEC 9075:

- 1) The column NUMBER_OF_CHARACTERS of the CHARACTER_SETS view has been deprecated.
- 2) The column NUMBER_OF_CHARS of the CHARACTER_SETS_S view has been deprecated.
- 3) The columns COLLATION_DEFINITION, COLLATION_DICTIONARY, and COLLATION_TYPE of the COLLATIONS view have been deprecated.
- 4) The columns COLLATION_DEFN, COLLATION_DICT, and COLLATION_TYPE of the COLLATIONS_S view have been deprecated.
- 5) The SQL_LANGUAGES view has been deprecated.
- 6) The columns FEATURE_ID and FEATURE_NAME of the SQL_PACKAGES view have been deprecated.

This page intentionally left blank.

Annex D

(informative)

Incompatibilities with ISO/IEC 9075:1999

This Annex modifies Annex E, “Incompatibilities with ISO/IEC 9075:1999”, in ISO/IEC 9075-2.

This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075-2:1999.

Except as specified in this Annex, features and capabilities of Database Language SQL are compatible with ISO/IEC 9075-2:1999.

- 1) In ISO/IEC 9075-2:1999, Subclause 20.27, "ELEMENT_TYPES view", the column named ARRAY_TYPE_IDENTIFIER has been renamed to COLLECTION_TYPE_IDENTIFIER in this edition of ISO/IEC 9075-11. Similarly, in ISO/IEC 9075-2:1999, Subclause 20.69, "Short name views", the column named ARRAY_TYPE_ID (in the view ELEMENT_TYPES_S) has been renamed to COLLECTION_TYPE_ID in this edition of ISO/IEC 9075-11.
- 2) In ISO/IEC 9075-2:1999, the KEY_COLUMN_USAGE view did not provide access to columns that were part of a unique key referenceable by a referential constraint, making it impossible to reconstruct a full table definition from the Information Schema information when such a reference had been made. In this edition of ISO/IEC 9075-11, that problem is corrected.
- 3) In ISO/IEC 9075:1999, the ATTRIBUTES view, the ATTRIBUTES_S short name view, and the ATTRIBUTES base table contained a column named CHECK_REFERENCES, related to <reference scope check>. That functionality was not satisfactorily specified and has been removed from this edition of ISO/IEC 9075, as has that column.

This page intentionally left blank.

Annex E

(informative)

SQL feature taxonomy

This Annex describes a taxonomy of features and packages defined in this part of ISO/IEC 9075.

Table 2, “Feature taxonomy and definition for mandatory features” contains a taxonomy of the features of the SQL language that are specified in this part of ISO/IEC 9075. Table 3, “Feature taxonomy for optional features” contains a taxonomy of the features of the SQL language that are specified in this part of ISO/IEC 9075.

Table 36, “Feature taxonomy for optional features”, in ISO/IEC 9075-2 contains a taxonomy of the features of the SQL language not in Core SQL that are specified in this part of ISO/IEC 9075 and in ISO/IEC 9075-2.

In these tables, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

The “Feature ID” column of Table 2, “Feature taxonomy and definition for mandatory features”, and of Table 2, “Feature taxonomy and definition for mandatory features”, specifies the formal identification of each feature and each subfeature contained in the table.

The “Feature Name” column of Table 2, “Feature taxonomy and definition for mandatory features”, and or Table 2, “Feature taxonomy and definition for mandatory features”, contains a brief description of the feature or subfeature associated with the Feature ID value.

Table 2, “Feature taxonomy and definition for mandatory features”, provides the only definition of the mandatory features of this part of ISO/IEC 9075. This definition consists of indications of specific language elements supported in each feature, subject to the constraints of all Syntax Rules, Access Rules, and Conformance Rules.

Table 2 — Feature taxonomy and definition for mandatory features

	Feature ID	Feature Name	Feature Description
1	F021	Basic information schema	— Subclause 5.1, “ INFORMATION_SCHEMA Schema ”: (Support of the COLUMNS, TABLES, VIEWS, TABLE_CONSTRAINTS, REFERENTIAL_CONSTRAINTS, and CHECK_CONSTRAINTS views in the INFORMATION_SCHEMA)
2	F021-01	COLUMNS view	— Subclause 5.20, “COLUMNS view”
3	F021-02	TABLES view	— Subclause 5.61, “TABLES view”

	Feature ID	Feature Name	Feature Description
4	F021-03	VIEWS view	— Subclause 5.76, “VIEWS view”
5	F021-04	TABLE_CONSTRAINTS view	— Subclause 5.58, “TABLE_CONSTRAINTS view”
6	F021-05	REFERENTIAL_CONSTRAINTS view	— Subclause 5.36, “REFERENTIAL_CONSTRAINTS view”
7	F021-06	CHECK_CONSTRAINTS view	— Subclause 5.13, “CHECK_CONSTRAINTS view”
8	F501	Features and conformance views	— Clause 5, “Information Schema”: SQL_FEATURES, SQL_SIZING, and SQL_LANGUAGE views
9	F501-01	SQL_FEATURES view	— Subclause 5.51, “SQL_FEATURES view”
10	F501-02	SQL_SIZING view	— Subclause 5.56, “SQL_SIZING view”
11	F501-03	SQL_LANGUAGES view	— Subclause 5.53, “SQL_LANGUAGES view”
12	S011	Distinct data types	— Subclause 11.41, “<user-defined type definition>”: When <representation> is <predefined type>
13	S011-01	USER_DEFINED_TYPES view	— Subclause 5.72, “USER_DEFINED_TYPES view”
14	T321	Basic SQL-invoked routines	<p>— Subclause 11.50, “<SQL-invoked routine>” — If Feature T041, “Basic LOB data type support”, is supported, then the <locator indication> clause shall also be supported</p> <p>NOTE 14 — “Routine” is the collective term for functions, methods, and procedures. This feature requires a conforming SQL-implementation to support both user-defined functions and user-defined procedures. An SQL-implementation that conforms to Core SQL shall support at least one language for writing routines; that language may be SQL. If the language is SQL, then the basic specification capability in Core SQL is the ability to specify a one-statement routine. Support for overloaded functions and procedures is not part of Core SQL.</p>
15	T321-06	ROUTINES view	— Subclause 5.48, “ROUTINES view”
16	T321-07	PARAMETERS view	— Subclause 5.34, “PARAMETERS view”

Table 3 — Feature taxonomy for optional features

	Feature ID	Feature Name
1	F231	Privilege tables
2	F341	Usage tables
3	F391	Long identifiers
4	F502	Enhanced documentation tables
5	S023	Basic structured types
6	S024	Enhanced structured types
7	T011	Timestamp in Information Schema
8	T175	Generated columns
9	T176	Sequence generator support
10	T211	Basic trigger capability
11	T272	Enhanced savepoint management
12	T331	Basic Roles

Table 3, “Feature taxonomy for optional features”, does not provide definitions of the features; the definition of those features is found in the Conformance Rules that are further summarized in [Annex A, “SQL Conformance Summary”](#).

This page intentionally left blank.

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— A —

ACTION • 179
 ADA • 169, 192, 208, 209
 ADMIN • 15, 183
 AFTER • 238
 ALL • 120, 123, 140, 162, 164, 167, 174, 215
 AND • 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 37, 38, 39, 40, 45, 47, 50, 51, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 118, 119, 141, 146, 147, 148, 149, 150, 165, 170, 171, 194, 207, 208, 223, 246
 ANY • 150, 184, 190, 239
 ARRAY • 149, 150, 152, 160, 246
 AS • 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 45, 47, 49, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 118, 120, 153, 155, 156, 215
 ASSERTION • 118, 119, 120
 ATTRIBUTES • 18, 19, 35, 99, 123, 261, 266, 277
 AUTHORIZATION • 9, 117

— B —

BEFORE • 238
 BIGINT • 146, 150
 BINARY • 146, 150
 BOOLEAN • 148, 150
 BY • 118, 120, 123, 140, 162, 164, 165, 167, 174

— C —

C • 169, 192, 208, 209
 CASCADE • 179
 CASCADED • 252
 CASE • 66, 89, 98

CATALOG_NAME • 7, 10, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 37, 38, 39, 40, 45, 47, 50, 51, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 109, 126, 129, 131, 132, 134, 136, 141, 150, 180, 184, 188, 189, 190, 200, 214, 229, 234, 235, 236, 239, 249, 250, 251
 CHARACTER • 12, 13, 20, 146, 150, 241, 246
 CHARACTER_LENGTH • 142
 CHARACTER_SET_CATALOG • 18, 20, 24, 29, 40, 42, 44, 47, 49, 51, 53, 66, 93, 99, 100, 101, 102, 103, 104, 105, 106, 108, 113, 129, 130, 136, 145, 150, 151, 152, 229, 241
 CHARACTER_SET_NAME • 18, 20, 24, 29, 40, 42, 44, 47, 49, 51, 53, 66, 93, 99, 100, 101, 102, 103, 104, 105, 106, 108, 112, 113, 129, 130, 136, 145, 150, 151, 152, 229, 241
 CHARACTER_SET_SCHEMA • 18, 20, 24, 29, 40, 42, 44, 47, 49, 51, 53, 66, 93, 99, 100, 101, 102, 103, 104, 105, 106, 108, 113, 129, 130, 136, 145, 150, 151, 152, 229, 241
 CHECK • 7, 10, 11, 118, 119, 120, 121, 123, 125, 129, 131, 132, 133, 134, 135, 136, 138, 140, 141, 145, 146, 150, 153, 155, 157, 158, 159, 160, 162, 164, 165, 167, 169, 170, 174, 175, 177, 179, 180, 182, 184, 186, 188, 189, 190, 192, 193, 194, 202, 204, 206, 207, 208, 214, 215, 217, 219, 222, 223, 226, 229, 230, 234, 235, 236, 238, 239, 241, 243, 245, 246, 249, 250, 251, 252
 COALESCE • 29, 30
 COBOL • 169, 192, 208, 209
 COLLATION • 23, 241
 COLLATION_CATALOG • 18, 23, 24, 29, 40, 42, 44, 47, 49, 51, 53, 66, 67, 93, 99, 100, 101, 102, 103, 104, 105, 106, 108, 113, 129, 135, 136, 145, 146, 147, 148, 150, 151, 152, 241
 COLLATION_NAME • 18, 23, 24, 29, 40, 42, 44, 47, 49, 51, 53, 66, 67, 93, 99, 100, 101, 102, 103, 104, 105, 106, 108, 113, 129, 135, 136, 145, 146, 147, 148, 150, 151, 152, 241
 COLLATION_SCHEMA • 18, 23, 24, 29, 40, 42, 44, 47, 49, 51, 53, 66, 67, 93, 99, 100, 101, 102, 103, 104, 105,

106, 108, 113, 129, 135, 136, 145, 146, 147, 148, 150, 151, 152, 241

COLUMN_NAME • 25, 26, 27, 28, 29, 30, 31, 45, 55, 61, 84, 85, 95, 100, 101, 102, 103, 107, 112, 118, 119, 131, 137, 138, 139, 140, 141, 142, 143, 164, 165, 184, 185, 223, 232, 233, 249

CONSTRAINT • 7, 10, 11, 121, 123, 125, 126, 127, 128, 129, 131, 132, 133, 134, 135, 136, 137, 138, 140, 141, 145, 146, 150, 153, 155, 157, 159, 160, 162, 163, 164, 165, 167, 169, 170, 174, 175, 177, 179, 180, 182, 184, 186, 188, 189, 190, 192, 193, 194, 200, 202, 204, 206, 207, 208, 210, 212, 214, 215, 217, 219, 222, 223, 226, 228, 229, 230, 232, 234, 235, 236, 238, 239, 241, 242, 243, 245, 246, 249, 250, 251, 252

CONSTRAINT_CATALOG • 17, 21, 22, 31, 33, 39, 45, 54, 78, 101, 102, 103, 106, 118, 119, 120, 121, 131, 132, 133, 134, 157, 158, 164, 165, 179, 180, 214, 215

CONSTRAINT_NAME • 17, 21, 22, 31, 33, 39, 45, 54, 78, 101, 102, 103, 106, 118, 119, 120, 121, 131, 132, 133, 134, 157, 158, 164, 165, 179, 180, 214, 215

CONSTRAINT_SCHEMA • 17, 21, 22, 31, 33, 39, 45, 54, 78, 101, 102, 103, 106, 118, 119, 120, 121, 131, 132, 133, 134, 157, 158, 164, 165, 179, 180, 214, 215

CONSTRUCTOR • 192

CONTAINS • 169, 193

COUNT • 10, 45, 118, 120, 123, 140, 162, 164, 165, 167, 174, 215

CREATE • 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 45, 47, 49, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 117, 118, 119, 120, 121, 123, 125, 126, 127, 128, 129, 131, 132, 133, 134, 135, 136, 137, 138, 140, 145, 153, 155, 157, 159, 160, 162, 164, 167, 169, 174, 177, 179, 182, 184, 186, 188, 189, 190, 192, 200, 202, 204, 206, 207, 210, 212, 214, 217, 219, 222, 226, 228, 230, 232, 234, 235, 236, 238, 241, 243, 245, 249, 250, 251, 252

CURRENT_ROLE • 43

CURRENT_TIMESTAMP • 14, 124, 159, 173, 198, 199, 240

CURRENT_USER • 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 37, 38, 39, 40, 45, 47, 50, 51, 61, 62, 63, 64, 65, 66, 67, 69, 70, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 124, 159

— D —

DATA • 169, 193

DATE • 147, 150

DAY • 148

DECIMAL • 147, 150

DEFAULT • 14, 179

DEFINED • 18, 28, 35, 47, 50, 93, 123, 145, 149, 150, 151, 167, 170, 193, 246

DEFINER • 193

DELETE • 219, 222, 238

DERIVED • 222, 245

DETERMINISTIC • 196

DISTINCT • 18, 29, 39, 40, 42, 44, 53, 54, 118, 245, 246

DOMAIN • 11, 12, 13, 14, 30, 35, 39, 40, 145, 151, 159, 241

DOUBLE • 147, 150

— E —

ELSE • 66, 89, 98

END • 66, 89, 98

EQUALS • 245

EXCEPT • 223

EXECUTE • 186

EXISTS • 66, 89, 98, 118, 119, 153, 155, 165, 223

EXTERNAL • 192, 194

— F —

Feature F231, "Privilege tables" • 27, 36, 55, 56, 57, 60, 62, 80, 91, 92, 113, 257, 258, 271

Feature F251, "Domain support" • 11, 12, 13, 14, 39, 40, 113, 258, 268

Feature F341, "Usage tables" • 21, 25, 26, 28, 32, 34, 46, 59, 61, 63, 64, 65, 85, 86, 87, 88, 95, 96, 97, 113, 114, 258, 259, 260, 261

Feature F391, "Long identifiers" • 8, 10, 15, 19, 20, 21, 23, 24, 25, 26, 30, 32, 34, 41, 42, 44, 46, 48, 50, 52, 53, 54, 56, 58, 61, 63, 64, 65, 68, 69, 70, 72, 73, 77, 79, 81, 83, 84, 85, 86, 87, 88, 90, 94, 99, 261, 262, 263, 264, 265

Feature F502, "Enhanced documentation tables" • 72, 74, 75, 77, 114, 265

Feature F521, "Assertions" • 17, 265

Feature F651, "Catalog name qualifiers" • 10, 265

Feature F690, "Collation support" • 24, 265, 266

Feature F691, "Collation and translation" • 23, 83, 114, 266

Feature F696, "Additional translation documentation" • 83, 114, 266

FLOAT • 147, 150

FOREIGN • 10, 119, 121, 127, 128, 129, 131, 132, 134, 135, 136, 137, 138, 141, 150, 153, 155, 157, 159, 160, 163, 164, 165, 167, 170, 175, 177, 179, 180, 182, 184, 186, 188, 189, 190, 193, 194, 200, 202, 214, 217, 219,

222, 223, 226, 228, 230, 232, 234, 235, 236, 239, 242, 243, 246, 249, 250, 251

FORTRAN • 169, 192, 208, 209

FROM • 7, 10, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 118, 119, 120, 123, 129, 131, 132, 133, 134, 136, 140, 141, 150, 153, 154, 155, 156, 159, 160, 162, 164, 165, 167, 168, 170, 171, 174, 175, 177, 180, 182, 184, 188, 189, 190, 194, 202, 214, 215, 223, 226, 229, 230, 234, 235, 236, 239, 241, 246, 249, 250, 251, 252

FULL • 119, 170, 179, 194, 223, 245

FUNCTION • 192

— G —

GENERAL • 169, 192

GENERATED • 222, 245, 246, 248

GLOBAL • 222, 223

GRANT • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 42, 43, 44, 46, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113

GROUP • 118, 120, 123, 140, 162, 164, 165, 167, 174

— H —

HAVING • 165

HOURLY • 148

— I —

IMPLEMENTATION • 110, 193

IN • 7, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 37, 38, 39, 40, 42, 44, 45, 47, 50, 51, 53, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 119, 121, 123, 125, 129, 131, 132, 133, 134, 135, 136, 138, 140, 141, 145, 146, 147, 148, 150, 153, 155, 157, 159, 160, 162, 165, 167, 169, 170, 171, 174, 175, 177, 179, 180, 182, 184, 186, 188, 189, 190, 192, 193, 194, 202, 204, 207, 208, 214, 215, 217, 219, 222, 223, 226, 229, 230, 234, 235, 236, 238, 239, 241, 243, 245, 246, 249, 250, 251, 252

INCREMENT • 70, 109, 202, 203

INOUT • 174

INSERT • 138, 219, 238

INSTANCE • 192

INTEGER • 11, 146, 150, 246

INTEGRITY • 110

INTERSECT • 154, 156

INTERVAL • 148, 150, 151

INVOKER • 193

IS • 26, 35, 51, 67, 119, 141, 146, 147, 148, 149, 150, 165, 170, 194, 204, 206, 207, 208, 223, 246

— J —

JAVA • 208, 209

JOIN • 16, 17, 18, 21, 22, 24, 25, 26, 28, 30, 31, 33, 37, 38, 40, 42, 43, 44, 45, 47, 50, 51, 53, 54, 56, 61, 62, 63, 64, 65, 67, 70, 77, 85, 86, 87, 88, 93, 95, 96, 97, 119, 153, 156

— K —

KEY • 10, 31, 33, 119, 121, 123, 125, 126, 127, 128, 129, 131, 132, 133, 134, 135, 136, 137, 138, 141, 150, 153, 155, 157, 159, 160, 162, 163, 164, 165, 167, 170, 175, 177, 179, 180, 182, 184, 186, 188, 189, 190, 193, 194, 200, 202, 204, 206, 210, 212, 214, 215, 217, 219, 222, 223, 226, 228, 230, 232, 234, 235, 236, 238, 239, 241, 242, 243, 245, 246, 249, 250, 251, 252

— L —

LANGUAGE • 172

LARGE • 146, 150

LEFT • 18, 30, 50, 51, 54, 67, 93

LOCAL • 222, 223, 252

— M —

MAP • 245

MATCH • 170, 194, 223

MAX • 45, 120, 123, 140, 162, 164, 165, 167, 174

METHOD • 192

MINUTE • 148

MODIFIES • 169, 193

MODULE • 208, 209

MONTH • 148

MULTISET • 149, 150, 160, 246

MUMPS • 169, 192, 208, 209

— N —

NAME • 71, 74, 75, 204, 205

NO • 121, 123, 135, 138, 140, 141, 143, 157, 162, 167, 169, 170, 174, 179, 182, 186, 192, 193, 197, 198, 202, 204, 205, 214, 217, 219, 222, 223, 241, 243, 245, 252

NONE • 179, 245, 252
 NOT • 7, 26, 35, 118, 119, 121, 123, 125, 127, 128, 129, 131, 132, 134, 135, 136, 138, 140, 141, 145, 146, 147, 148, 149, 150, 153, 155, 157, 162, 164, 165, 167, 169, 170, 174, 179, 180, 186, 188, 189, 192, 193, 194, 200, 202, 204, 206, 207, 210, 212, 214, 217, 219, 222, 223, 226, 228, 229, 230, 234, 235, 236, 238, 241, 243, 245, 246, 249, 250, 251, 252
 NULL • 26, 35, 51, 66, 67, 89, 98, 119, 121, 123, 125, 127, 128, 129, 135, 138, 140, 141, 145, 146, 147, 148, 149, 150, 157, 162, 164, 165, 167, 169, 170, 174, 179, 186, 192, 193, 194, 200, 202, 204, 206, 207, 208, 210, 212, 214, 217, 219, 222, 223, 226, 228, 230, 238, 241, 243, 245, 246, 252
 NUMERIC • 147, 150

— O —

OBJECT • 146, 150
 ON • 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 46, 47, 48, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 153, 156
 OPTION • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36, 37, 38, 39, 40, 42, 43, 44, 46, 48, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 183
 OR • 7, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 37, 38, 39, 40, 45, 47, 50, 51, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 110, 129, 131, 132, 134, 136, 141, 146, 147, 148, 149, 150, 165, 170, 180, 184, 188, 189, 190, 194, 204, 206, 207, 208, 214, 223, 229, 234, 235, 236, 239, 246, 249, 250, 251
 OUT • 174
 OUTER • 119

— P —

PAD • 135
 PARAMETER_MODE • 47, 51, 105, 167, 168, 174, 175
 PARAMETER_NAME • 47, 51, 105, 167, 168, 174, 176
 PARTIAL • 179
 PASCAL • 169, 192, 208, 209
 PLI • 169, 192, 208, 209
 PRECISION • 147, 150
 PRESERVE • 222

PRIMARY • 10, 31, 33, 119, 121, 123, 125, 126, 127, 128, 129, 131, 132, 133, 134, 135, 136, 137, 138, 141, 150, 153, 155, 157, 159, 160, 162, 164, 165, 167, 170, 175, 177, 179, 180, 182, 184, 186, 188, 189, 190, 193, 200, 202, 204, 206, 210, 212, 214, 215, 217, 219, 222, 226, 228, 230, 232, 234, 235, 236, 238, 241, 243, 245, 249, 250, 251, 252
 PROCEDURE • 192, 194
 PUBLIC • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 138, 182, 187, 218, 220, 242, 244

— R —

READS • 169, 193
 REAL • 147, 150
 RECURSIVE • 16, 43, 153, 155
 REF • 149, 150, 151, 177, 248
 REFERENCE • 246
 REFERENCES • 10, 121, 127, 128, 129, 131, 132, 134, 135, 136, 137, 138, 141, 150, 153, 155, 157, 159, 160, 163, 165, 167, 170, 175, 177, 182, 184, 186, 188, 189, 190, 193, 194, 200, 202, 212, 217, 219, 222, 223, 226, 228, 230, 232, 234, 235, 236, 239, 242, 243, 246, 249, 250, 251
 RELATIVE • 245
 RESTRICT • 179
 ROLE • 125, 182
 ROUTINE • 35, 51, 67, 145, 151, 175, 194
 ROUTINE_CATALOG • 56, 61, 62, 63, 65, 66, 106, 107, 108, 188, 192, 193, 194, 195, 250
 ROUTINE_NAME • 56, 61, 62, 63, 65, 66, 106, 107, 108, 188, 192, 195, 250
 ROUTINE_SCHEMA • 56, 61, 62, 63, 65, 66, 106, 107, 108, 188, 192, 193, 194, 195, 250
 ROW • 150, 152, 162, 238, 246

— S —

Feature S023, “Basic structured types” • 19, 48, 50, 114, 266, 267
 Feature S024, “Enhanced structured types” • 38, 58, 79, 114, 267
 Feature S041, “Basic reference types” • 53, 267
 Feature S081, “Subtables” • 37, 267
 Feature S091, “Basic array support” • 42, 267, 268
 Feature S241, “Transform functions” • 82, 267

Feature S271, “Basic multiset support” • 42, 267, 268
 SCHEMA • 9, 117
 SCHEMA_NAME • 17, 21, 22, 24, 25, 26, 28, 31, 33, 37, 38, 61, 63, 64, 65, 66, 69, 85, 86, 87, 88, 89, 95, 96, 97, 98, 109, 200
 SCOPE_CATALOG • 18, 29, 42, 44, 47, 49, 51, 53, 66, 67, 99, 101, 102, 103, 104, 105, 106, 108, 146, 147, 148, 149, 150, 151
 SCOPE_NAME • 18, 29, 42, 44, 47, 49, 51, 53, 66, 67, 99, 101, 102, 103, 104, 105, 106, 108, 109, 146, 147, 148, 149, 150, 151
 SCOPE_SCHEMA • 18, 29, 42, 44, 47, 49, 51, 53, 66, 67, 99, 101, 102, 103, 104, 105, 106, 108, 146, 147, 148, 149, 150, 151
 SECOND • 148
 SELECT • 7, 8, 10, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 118, 119, 120, 123, 129, 131, 132, 133, 134, 136, 138, 140, 141, 150, 153, 154, 155, 156, 159, 160, 162, 164, 165, 167, 170, 171, 174, 175, 177, 180, 182, 184, 188, 189, 190, 194, 202, 214, 215, 219, 223, 229, 230, 234, 235, 236, 239, 241, 246, 249, 250, 251, 252
 SEQUENCE • 70, 145, 151, 202, 241
 SEQUENCES_CATALOG • 202
 SEQUENCES_SCHEMA • 202
 SET • 12, 13, 20, 179, 241
 SMALLINT • 146, 150
 SOURCE • 110
 SPACE • 135
 SPECIFIC_NAME • 21, 35, 47, 49, 51, 56, 58, 61, 62, 63, 64, 65, 66, 67, 79, 82, 86, 96, 101, 104, 105, 106, 107, 108, 110, 111, 132, 167, 168, 169, 170, 171, 174, 175, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 198, 217, 218, 226, 234, 250
 SQL • 169, 170, 172, 192, 194, 195, 196, 226, 227
 STATE • 245
 STATEMENT • 238
 STATIC • 192
 SYSTEM • 222, 245

— T —

Feature T011, “Timestamp in Information Schema” • 14, 50, 68, 90, 114, 258, 268
 Feature T051, “Row types” • 44, 268
 Feature T111, “Updatable joins, unions, and columns” • 30, 115, 225, 269

Feature T175, “Generated columns” • 25, 30, 115, 269
 Feature T176, “Sequence generator support” • 64, 70, 87, 115, 269, 270
 Feature T211, “Basic trigger capability” • 84, 85, 86, 87, 88, 90, 115, 270
 Feature T272, “Enhanced savepoint management” • 68, 198, 271
 Feature T331, “Basic roles” • 15, 16, 43, 55, 56, 57, 58, 59, 60, 115, 257, 271, 272
 TABLE • 10, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 46, 48, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 121, 123, 125, 126, 127, 128, 129, 131, 132, 133, 134, 135, 136, 137, 138, 140, 141, 145, 151, 153, 155, 157, 159, 160, 162, 164, 167, 169, 174, 177, 179, 182, 184, 186, 188, 189, 190, 192, 200, 202, 204, 206, 207, 210, 212, 214, 217, 219, 222, 226, 228, 230, 232, 234, 235, 236, 238, 241, 243, 245, 249, 250, 251, 252
 TABLE_NAME • 7, 25, 26, 27, 28, 29, 30, 31, 33, 35, 37, 45, 55, 57, 58, 61, 65, 78, 79, 80, 81, 84, 85, 88, 89, 90, 95, 96, 97, 98, 100, 101, 102, 103, 107, 108, 110, 111, 112, 131, 134, 137, 138, 139, 140, 141, 142, 153, 154, 164, 165, 184, 185, 190, 191, 214, 215, 217, 218, 219, 220, 222, 223, 224, 232, 233, 236, 237, 239, 249, 250, 251, 252
 TEMPORARY • 222, 223
 THEN • 66, 89, 98
 TIME • 14, 147, 150
 TIMESTAMP • 14, 147, 150
 TO • 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36, 37, 38, 39, 40, 42, 43, 44, 46, 48, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 148, 226
 TRANSFORMS • 82, 226, 267
 TRANSLATION • 83, 241
 TRIGGER • 219
 TRIGGER_CATALOG • 84, 85, 86, 87, 88, 89, 90, 111, 112, 230, 232, 233, 234, 235, 236, 237, 238, 239
 TRIGGER_NAME • 84, 85, 86, 87, 88, 89, 111, 112, 230, 232, 233, 234, 235, 236, 237, 238, 239
 TRIGGER_SCHEMA • 84, 85, 86, 87, 88, 89, 111, 112, 230, 232, 233, 234, 235, 236, 237, 238, 239
 TYPE • 18, 35, 47, 50, 71, 74, 75, 93, 123, 145, 151, 153, 154, 155, 156, 167, 170, 204, 243, 246

— U —

UNION • 16, 31, 33, 35, 43, 78, 81, 90, 120, 133, 153, 156, 215, 241
 UNIQUE • 31, 33, 119, 123, 141, 162, 164, 165, 180, 210, 214, 215
 UPDATE • 84, 138, 219, 230, 231, 238
 USAGE • 8, 11, 12, 13, 14, 59, 92, 243
 USER • 125, 222, 245, 246, 248
 USER_DEFINED_TYPE_CATALOG • 18, 28, 29, 35, 38, 42, 44, 47, 49, 50, 51, 53, 60, 66, 67, 81, 82, 91, 93, 104, 111, 113, 123, 145, 146, 147, 148, 149, 150, 151, 155, 156, 168, 169, 170, 171, 192, 194, 195, 222, 223, 224, 225, 226, 243, 245, 246, 247, 248
 USER_DEFINED_TYPE_NAME • 18, 28, 29, 35, 38, 42, 44, 47, 49, 50, 51, 53, 60, 66, 67, 81, 82, 91, 93, 104, 111, 113, 123, 146, 147, 148, 149, 150, 151, 155, 156, 168, 169, 170, 171, 192, 194, 195, 222, 223, 225, 226, 243, 245, 246, 247, 248
 USER_DEFINED_TYPE_SCHEMA • 18, 28, 29, 35, 38, 42, 44, 47, 49, 50, 51, 53, 60, 66, 67, 81, 82, 91, 93, 104, 111, 113, 123, 146, 147, 148, 149, 150, 151, 155, 156, 168, 169, 170, 171, 192, 194, 195, 222, 223, 224, 225, 226, 243, 245, 246, 247, 248
 USING • 26, 31, 42, 44, 45, 47, 51, 53, 54, 56, 61, 62, 65, 77, 119

— V —

VALUE • 11
 VALUES • 43, 121, 157, 214, 252
 VARYING • 12, 13, 146, 150
 VIEW • 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 38, 39, 40, 42, 43, 44, 45, 47, 49, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 215, 222, 223, 252

— W —

WHEN • 66, 89, 98
 WHERE • 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 33, 35, 37, 38, 39, 40, 42, 44, 45, 47, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 74, 75, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 118, 119, 123, 154, 156, 160, 162, 165, 177, 180, 182, 215, 223, 230, 246, 252
 WITH • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36, 37, 38, 39, 40, 42, 43, 44, 46, 48, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 94, 95,

96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 147, 153, 155, 183

— Y —

YEAR • 148

— Z —

ZONE • 14, 147

1 Possible problems with SQL/Schemata

I observe some possible problems with SQL/Schemata as defined in this document. These are noted below. Further contributions to this list are welcome. Deletions from the list (resulting from change proposals that correct the problems or from research indicating that the problems do not, in fact, exist) are even more welcome. Other comments may appear in the same list.

Because of the highly dynamic nature of this list (problems being removed because they are solved, new problems being added), it has become rather confusing to have the problem numbers automatically assigned by the document production facility. In order to reduce this confusion, I have instead assigned "fixed" numbers to each possible problem. These numbers will not change from printing to printing, but will instead develop "gaps" between numbers as problems are solved.

Possible problems related to SQL/Schemata

Significant Possible Problems:

999 In the body of the Working Draft, I have occasionally highlighted a point that requires urgent attention thus:

Editor's Note
Text of the problem.

These items are indexed under "***Editor's Note***".

SCHEM-016 The following Possible Problem has been noted:

Severity: Major Technical

Reference: P11, SQL/Schemata, Subclause 5.11, "CHARACTER_SETS view"

Note at: None.

Source: CD1/2001, USA-P11-002

Possible Problem:

Information schema columns that are now constant because the data is no longer maintained should be deprecated in SQL:200x and dropped in the following progression. These include CHARACTER_SETS view, columns FORM_OF_USE and NUMBER_OF_CHARACTERS; COLLATIONS view, columns COLLATION_TYPE, COLLATION_DICTIONARY and COLLATION_DEFINITION; and TRANSLATIONS view, TRANSLATION_DEFINITION column.

Proposed Solution:

None provided with comment.

Minor Problems and Wordsmithing Candidates: SCHEM-020 The following Possible Problem has been noted:

Severity: Minor Technical

Reference: P02, SQL/Foundation, Subclause 6.6, "ATTRIBUTES base table"

Note at: None.

Source: WG3:DRS-091

Possible Problem:

There is an indicator missing from the ATTRIBUTES base table, which reflects the content of the <reference scope check action> contained in a <reference scope check> contained in an <attribute definition>.

Proposed Solution:

None provided with comment.

Language Opportunities: **[SCHEM-002]** The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11-21.08, SQL/Schemata, Subclause 6.11, "CHARACTER_SETS base table"

Note at: None.

Source: DCOR 2000, SWE-STC-030

Possible Problem:

This base table contains a bare minimum of information. It could be enhanced to indicate relationships among character sets, for example whether the character set is standard, implementation-defined, or user-defined, and what character set a user-defined character set is based on.

Proposed Solution:

None provided with comment.

[SCHEM-005] The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, No particular location

Note at: None.

Source: Email from Fred Zemke, 2000-10-24, based on HEL discussions

Possible Problem:

It would be useful to have a matrix that records all the possible kinds of dependency that are possible between schema objects. Such a matrix could be placed in the standing document, "Miscellaneous facts about the SQL standard". The rows and columns of the matrix would be the various kinds of schema objects (currently I count the following that need tracking: assertion, attribute, character set, collation, column, domain, domain constraint, unique constraint, referential constraint, check constraint, original method, overriding method, privilege SQL-invoked routine, translation, trigger, transform, user-defined cast, user-defined ordering, user-defined type, view). The matrix should indicate in each cell at intersection A,B whether an object of type A can directly depend on an object of type B, and how.

Once the matrix has been constructed, it can be used to verify two things:

- that the rules of DROP and REVOKE have identified all dependencies
- that the views in the Information Schema are adequate for the user to determine all relevant dependencies.

Proposed Solution:

None provided with comment.

[SCHEM-008] The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 6.44, "SQL_IMPLEMENTATION_INFO base table"

Note at: None.

Source: WG3:PER-118/H2-2001-???

Language Opportunity:

Subclause 6.44, "SQL_IMPLEMENTATION_INFO base table", is defined to contain information about SQL-implementation information items (identified by name and number) but these items are not defined in the other parts of the standard.

Proposed Solution:

None provided with comment.

SCHEM-009 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 6.46, "SQL_SIZING base table"

Note at: None.

Source: WG3:PER-118/H2-2001-???

Language Opportunity:

Subclause 6.46, "SQL_SIZING base table", is defined to contain information about SQL sizing items (identified by name and number) but these items are not defined in the other parts of the standard. (Subclause 6.47, "SQL_SIZING_PROFILES base table", has the same problem.)

Proposed Solution:

None provided with comment.

SCHEM-013 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 6.21, "DATA_TYPE_DESCRIPTOR base table"

Note at: None.

Source: WG3:PER-100R2/H2-2001-062R2

Language Opportunity:

Paper WG3:PER-100r2 noted the following Language Opportunity:

The user may wish to recover the original type declaration, rather than the equivalent type declaration that is used by the SQL-server. This concern could be met by adding columns such as ORIGINAL_DATA_TYPE and ORIGINAL_PRECISION to the DATA_TYPE_DESCRIPTOR base table, as well as all views that draw from it. These new columns should be part of a new conformance feature, to make them optional, since not every implementation will be able to display the original type declaration.

Proposed Solution:

None provided with comment.

SCHEM-014 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 6.21, "DATA_TYPE_DESCRIPTOR base table"

Note at: None.

Source: WG3:PER-100R2/H2-2001-062R2

Language Opportunity:

Paper WG3:PER-100r2 noted the following Language Opportunity:

Users might be interested to know the largest and smallest exponents accommodated by the approximate numeric types.

Proposed Solution:

None provided with comment.

SCHEM-015 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 6.21, "DATA_TYPE_DESCRIPTOR base table"

Note at: None.

Source: WG3:PER-100R2/H2-2001-062R2

Language Opportunity:

Paper WG3:PER-100r2 noted the following Language Opportunity:

Editor's Notes for WG3:HBA-008 = H2-2003-310

A capability would be a table that simply listed all the data type equivalences of the numeric data types.

Proposed Solution:

None provided with comment.

SCHEM-018 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 5.10, "ATTRIBUTES view"

Note at: None.

Source: DCOR/2002 USA-STC-048 and WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

The function statement says that this view lists the attributes of structured types that the user has access to. "Access" is ambiguous. As the view is currently defined it appears to mean, "has USAGE or UNDER privilege on". This could be solved by changing "that are accessible" to "that the user has USAGE or UNDER privilege for". However this comment will not suggest that solution. Instead, this comment will point out that there are ways to define access to an attribute other than USAGE or UNDER privilege on the attribute's type. First, there are other ways to access the type than through USAGE privilege. The type might be the parameter type of an SQL-invoked routine that the user can execute, it might be the return type of a regular function or method that the user can execute, it might be the type of a column that the user can SELECT, or the type of a selectable nested site such as the field of a row, the element type of a collection, or the attribute type of a different structured type. All of these constitute "access" to a structured type. An analogy can be drawn between user-defined types and domains. Note that the DOMAINS view shows not just the domains that the user has USAGE privilege on; it also shows domains that are the types of columns that the user can access. After defining accessible types, you have the question of what makes an attribute accessible. Is it EXECUTE privilege on the observer? Or perhaps EXECUTE on either the observer or the mutator? Or some other criterion?

Proposed Solution:

None provided with comment.

SCHEM-019 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 5.72, "USER_DEFINED_TYPES view"

Note at: None.

Source: DCOR/2002 USA-STC-059 and WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

The word "accessible" in the function is ambiguous. What is meant is those user-defined types for which the user has USAGE or UNDER privilege. However, it is questioned in a separate comment on the ATTRIBUTES view whether "accessible" should be limited to types with USAGE or UNDER privilege. Note that DIRECT_SUPERTYPES view will reveal type T's existence if T is the direct supertype of a type T2 for which the user has USAGE or UNDER privilege, even if the user does not have USAGE or UNDER privilege on T itself. This seems inconsistent. Also, COLUMNS view will display the type T if there is a column whose type is T. It is suspected, but not verified, that the type will also be visible in other views of the Information Schema, wherever the type of a site is displayed (for example, ATTRIBUTES view, FIELDS view, ROUTINE view, PARAMETERS view). Note that DOMAINS view shows a domain if either the user has USAGE privilege on the domain or the user has SELECT privilege on a column whose type is the domain; this provides a precedent that "accessible" is not limited to "has a privilege on".

Proposed Solution:

None provided with comment.

SCHEM-021 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 5.19, "COLUMN_UDT_USAGE view"

Note at: None.

Source: WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

DCOR comment USA-STC-049 pointed out that the join condition joining DEFINITION_SCHEMA.COLUMNS with DEFINITION_SCHEMA.SCHEMATA assumes that the former table has columns named USER_DEFINED_TYPE_CATALOG and USER_DEFINED_TYPE_SCHEMA, which it does not. That comment goes on to suggest that perhaps the intent was to join in DATA_TYPE_DESCRIPTORS, which does. However, if the suggestion in USA-STC-049 were followed, this would not really be sufficient to find all columns that are dependent on a user-defined type. What about columns that are row types with a field that is a user-defined type? Or collection types with an element type that is a user-defined type? See the notion of usage-dependent added to Foundation by WG3:YYJ-083r1. Note that in that paper, it is argued that the notion of usage-dependency does not need to recurse through attributes of a structured type. While this argument is sufficient for the purpose of enforcing RESTRICT or CASCADE semantics, and justifiable for Access Rule checking, does it make sense for this view? For example, if type T1 has an attribute of type T2, and column C1 is of type T1, does C1 depend on T2 in the meaning of this view? If the user is using the view to find all columns to drop before dropping type T2, then the user wants to see C1 in this view. The alternative is that the user must do his own recursion (find all UDTs that depend on T2, then find all columns that depend on any of them.)

Proposed Solution:

None provided with comment.

SCHEM-022 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, Subclause 6.44, "SQL_IMPLEMENTATION_INFO base table"

Note at: None.

Source: WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

There is no list of values for IMPLEMENTATION_INFO_ID and IMPLEMENTATION_INFO_NAME. It seems that many of these were intended to be the codes used in CLI by GetInfo (see for example CLI GetInfo GR 10) subrules b), c), p) and q).) The writer of this comment does not know if there are codes that are necessary for CLI or other parts of SQL. But see CLI subclause 7.1 SQL_IMPLEMENTATION_INFO base table.

Proposed Solution:

None provided with comment.

SCHEM-023 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, No specific location

Note at: None.

Source: WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

Editor's Notes for WG3:HBA-008 = H2-2003-310

Implementations should not be required to expose columns about optional features that they don't support. For example, in Subclause 5.22, "COLUMNS view", the IS_SELF_REFERENCING column is meaningful only if Feature S051, "Create tables of type", is implemented. If conformance to that feature is not claimed, then references to the column should be prohibited.

Proposed Solution:

None provided with comment.

SCHEM-024 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, No specific location

Note at: None.

Source: WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

There are no usage tables to display the usage of user-defined types, except COLUMN_UDT_USAGE. We also need UDT_UDT_USAGE to document when one UDT uses another (as the type of an attribute) ROUTINE_UDT_USAGE when a routine depends on a UDT, CONSTRAINT_UDT_USAGE for constraints that depend on a UDT, TRIGGER_UDT_USAGE for triggers that depend on a UDT. (But see the concurrent comment suggesting an alternative approach that would eliminate the usage tables.)

Proposed Solution:

None provided with comment.

SCHEM-025 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P11, SQL/Schemata, No specific location

Note at: None.

Source: WG3:ZSH-153R1 = H2-2002-153R1

Language Opportunity:

Instead of the present usage tables, it would be more convenient to have a single DIRECT_DEPENDENCIES table that documented all dependencies. The table would have columns to identify the independent object and columns to identify the directly dependent object, and perhaps a column to indicate the kind of dependency. This base table could be used to define a recursive view called DEPENDENCIES to display all dependencies.

Proposed Solution:

None provided with comment.

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

999 • 2

SCHEM-002	• 4
SCHEM-005	• 4
SCHEM-008	• 4
SCHEM-009	• 5
SCHEM013-	• 5
SCHEM-014	• 5

— S —

SCHEM-015	• 5
SCHEM-016	• 2
SCHEM-018	• 6
SCHEM-019	• 6
SCHEM-020	• 3
SCHEM-021	• 7
SCHEM-022	• 7
SCHEM-023	• 7
SCHEM-024	• 8
SCHEM-025	• 8

