ISO/IEC JTC1 SC32 WG3 BCN-036R1

Title:	BCN-036R1: Select list EXCLUDE
Date:	2025-09-17
Author:	Peter Eisentraut
Status:	change proposal

References:

[Foundation IWD]	9075_9IWD15-02-Foundation_2025-06-30.pdf
[UKB-014R1]	"Minutes of ballot resolution meeting", 2005-11-28
[WLG-064R1]	"Column Selection by Exclusion", Hugh Darwen, 2005-11-17

Abstract

Add EXCLUDE clause to <select list>.

Revision history

R1: Added section "Review of previous proposal".

1 Introduction

This proposal adds a clause to the SELECT list "all columns" wildcard to exclude some columns from the wildcard expansion. For example:

```
create table t2 (foo int, bar int, baz int);
select * exclude (bar) from t2;
```

results in a result set that only has columns "foo" and "baz". Similarly, the qualified-asterisk variant

```
select t2.* exclude (bar) from t2;
```

would have the same result.

In my anecdotal experience, this is among the top three features requested for addition to SQL to improve ease of use.

There are existing implementations:

- H2[1]
- Databricks[2]
- BigQuery[3]
- Snowflake[4]
- DuckDB[5]

1 https://www.h2database.com/html/grammar.html#wildcard_expression
2 https://docs.databricks.com/aws/en/sql/language-manual/sql-ref-syntax-qry-star
3 https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#select_list
4 https://docs.snowflake.com/en/sql-reference/sql/select
5 https://duckdb.org/docs/sql/expressions/star.html#exclude-clause

It is apparent from this selection that this feature is particularly useful for analytics use cases.

There is a catch: These implementations do not agree on the keyword to use:

- H2: EXCEPT
- Databricks: EXCEPT
- BigQuery: EXCEPT
- Snowflake: EXCLUDE
- DuckDB: EXCLUDE

This proposal uses EXCLUDE. The reason is that EXCEPT is already used for another purpose, and while that does not directly conflict with the new proposed use, it is awfully close. Furthermore, many implementations allow SELECT without a FROM clause, and there is some interest in standardizing that (LO FND-A51), in which case SQL text like

```
SELECT * EXCEPT ...
```

could be very confusing and difficult to parse.

If this proposal ends up in the standard, existing implementations that use EXCEPT could easily add support for EXCLUDE as an alias. The reverse would likely be more difficult.

2 Review of previous proposal

It was brought to my attention that this change was already previously proposed by [WLG-064R1] and not adopted. The arguments in that paper were similar to mine. It additionally had some arguments on the academic side. The minutes in paper [UKB-014R1] summarize the discussion of the paper:

> Following discussion of the paper a majority position emerged that there should be no enhancements to the functionality around SELECT *, and that although the paper was technically acceptable, it should be rejected.
>
> On a vote, approval of the paper was rejected 1-4-0 (UK for, Canada, Germany, Japan, USA against; Australia absent).
>
> It was agreed that Seq#063 should be marked as resolved with no action

This does not record the arguments against the paper, but I suspect that they were similar to the arguments against WLG-022 discussed at the same meeting (and recorded in those minutes just above the quoted text).

I think the reality has shown that this feature continues to be popular and in demand. Seeing that there are a growing number of implementations and some potential questions about the details of the behavior (see below), it seems worth trying to standardize it.

Note that [WLG-064R1] proposed the use of the "EXCEPT" keyword, unlike my proposal. Other than that, it appears to agree pretty much with my proposal in terms of the semantics, except that it does not want to accept qualified exclusion elements in a <qualified asterisk> (see my example 9 below). The actual implementation is quite a bit different (and simpler) in my paper.

3 Discussion

Here are some test cases for the proposed functionality.

Use the following table definitions:

```
create table t1 (foo int);
create table t2 (foo int, bar int, baz int);
create table s1.t3 (a int, b int);
```

Then the following would happen:

Example 1:

```
select * exclude (bar) from t2;
```

Result columns are: foo, baz

Example 2:

```
select * exclude (bar) from t1;
```

This is an error: "bar" does not exist in t1. (All existing implementations agree on that. You cannot exclude non-existing columns.)

Example 3:

```
select * exclude (foo) from t1;
```

Causing * to expand to an empty select list is an error. (Implementations that allow zero-column tables might choose to allow this as an extension. (H2 allows this. Databricks crashes.)) (But see example 11 below.)

Example 4:

```
select * exclude () from t1;
```

An empty exclude list is an error. (Implementations agree.)

Example 5:

```
select * exclude (bar, bar) from t2;
```

Duplicate entries in the exclude list are an error. (Implementations agree.)

Example 6:

```
select * exclude (foo) from t1, t2;
```

This is an error because "foo" is ambiguous, just like "select foo from t1, t2" is ambiguous. (One existing implementation (DuckDB) allows this and excludes both "foo" columns. The remaining cited implementations reject this.)

Example 7:

```
select * exclude (bar) from t1, t2;
```

Result columns are "foo", "foo", "baz".

Example 8:

```
select * exclude (t2.bar) from t1, t2;
select * exclude (t1.foo) from t1, t2;
select * exclude (s1.t3.a) from s1.t3;
```

Any qualification that is allowed in the select list is also allowed in the EXCLUDE list. (Snowflake and BigQuery do not allow any qualifications in the EXCLUDE/EXCEPT list.)

The general idea is entries in the exclude list and entries in the select list are resolved in the same way, and what is valid or invalid in one should be the same in the other. So notionally,

```
select SOMETHING, * exclude (SOMETHING) ...
```

would be equivalent (up to column order) to

```
select * ...
```

Example 9:

```
select t2.* exclude (t2.foo) from t2;
```

The qualification here is pretty much redundant, but most implementations accept it, and it keeps the semantics of the EXCLUDE list the same for the qualified and the unqualified asterisk.

Example 10:

```
select t2.* exclude (t1.foo) from t1, t2;
```

This is an error.

Example 11:

```
select foo, t1.* exclude (foo) from t1;
```

The qualified asterisk resolves to an empty column list, but the overall select list is not empty, so this is allowed. (Implementations agree.)

4 Proposal for [Foundation IWD]

4.1 Changes to subclause 4.27.15, "Known functional dependencies in a <query specification>"

1. Modify the fourth paragraph as follows:

Let *S* be a set of columns of *R* such that every element of *S* arises ~~from the use of <asterisk> in *SL* or~~ by the specification of a column reference as a <value expression> simply contained in *SL*. [...]

[Note: The mention of <asterisk> can be deleted because per subclause 4.27.2, syntactic transformations are applied before the rules to determine functional dependencies are used, and therefore, no <asterisk> will remain by the time this rule is used.]

4.2 Changes to subclause 5.2, "<token> and <separator>"

1. Add "EXCLUDE" to the list of reserved words:

```
<reserved word> ::=
...
... | EVERY | EXCEPT | EXCLUDE | EXEC | EXECUTE | ...
```

4.3 Changes to subclause 6.7, "<column reference>"

1. Modify syntax rule 7) b) ii) as follows:

ii) If *QQ* is not grouped, or if *QCR* is contained in the <select list exclude> or in the <from clause> or the <where clause> simply contained in *QQ*, then *QCR* is an *ordinary column reference*.

[Note: This is necessary to allow the following:

create table t4 (a int, b int, c int);
select t4. exclude (b, c) from t4 group by a;*

Otherwise, the column references b and c would be classified as group-invariant and rejected in this context.]

4.4 Changes to subclause 7.13, "<group by clause>"

1. Modify syntax rule 7 as follows:

7) Let *SL1* be obtained from *SL* by replacing every ~~<asterisk>~~ <select list all columns> and <asterisked identifier chain> using the syntactic transformations in the Syntax Rules of Subclause 7.16, “<query specification>”.

4.5 Changes to subclause 7.16, "<query specification>"

1. Changes to the Format:

```
<query specification> ::=
  SELECT [ <set quantifier> ] <select list> <table expression>

<select list> ::=
    ~~<asterisk>~~<select list all columns>
    | <select sublist> [ { <comma> <select sublist> }... ]

<select sublist> ::= ...

<select list all columns> ::=
  <asterisk> [ <select list exclude> ]

<select list exclude> ::=
  EXCLUDE <left paren> <column reference> [ { <comma> <column
reference> }... ] <right paren>

<qualified asterisk> ::=
~~    <asterisked identifier chain> <period> <asterisk>~~
    <qualified all columns>
  | <all fields reference>

<qualified all columns> ::=
  <asterisked identifier chain> <period> <asterisk> [ <select list
exclude> ]
```

2. Add a new syntax rule:

5) If <select list exclude> is specified, then no column shall be referenced more than once by the <column reference>s contained in the <select list exclude>.

3. Change syntax rule 7 as follows:

7) If the <select list> is a <select list all columns>, then:

Case:

a) If the <select list all columns> ~~"*"~~ is simply contained in a <table subquery> that is immediately contained in an <exists predicate>, then the <select list all columns> is equivalent to a <value expression> that is an arbitrary <literal>.

b) Otherwise~~,~~: *[inject the following into the existing sentence]*

Case:

i) If the <select list all columns> contains a <select list exclude>, then:

1) Let *SLE* be the <select list exclude>.

2) Let *SLEC* be the sequence of column references contained in *SLE*.

ii) Otherwise, let *SLEC* be an empty sequence.

~~t~~The <select list all columns> ~~"*"~~ is equivalent to a <value expression> sequence in which each <value expression> is a column reference that references a column of *T* that is not a column identified in *SLEC*, and each column of *T* is referenced exactly once. The columns are referenced in the ascending sequence of their ordinal position within *T*.

4. Change syntax rule 10 as follows:

10) If ~~<asterisked identifier chain>~~ <qualified all columns> *QAC* is specified, then:

a) Let *IC* be ~~an~~ the <asterisked identifier chain> contained in *QAC*.

...

10) g) Case:

i) If the basis is a <table or query name> or <correlation name>, then let *TQ* be the table associated with the basis. *[inject the following into the existing paragraph]*

Case:

i) If *QAC* specifies <select list exclude>, then:

1) Let *SLE* be the <select list exclude>.

2) Let *SLEC* be the sequence of column references contained in *SLE*.

3) Every column reference in *SLE* shall reference a column of *TQ* that is not a common column of a <joined table>.

ii) Otherwise, let *SLEC* be an empty sequence.

The <select sublist> is equivalent to a <value expression> sequence in which each <value expression> is a column reference *CR* that references a column of *TQ* that is not a common column of a <joined table> and that is not a column identified in *SLEC*. Each column of *TQ* that is not a common column shall be referenced exactly once. The columns shall be referenced in the ascending sequence of their ordinal positions within *TQ*.

ii) Otherwise let *BL* be the length of the basis of *IC*.

QAC shall not contain <select list exclude>.

[Note: This is because <all fields references> does not support exclude lists. LO?]

Case:

...

5. Add a note after syntax rule 10:

Note *NNN* — If after the syntactic transformations specified by the above Syntax Rules, the <select list> is empty, it no longer conforms to the Format of <select list> and is therefore invalid.

6. Change syntax rule 18 as follows:

18) A column *C* of *TQS* is readily known not null if *C* is defined (after performing syntactic transformations to eliminate ~~<asterisk>~~ <select list all columns> and <qualified asterisk>). [...]

7. Add a conformance rule:

7) Without feature O*NNN*, "Select list EXCLUDE", conforming SQL language shall not contain a <select list exclude>.

4.6 Changes to subclause 8.11, "<exists predicate>"

[Note: No changes to conformance rule 1 needed. Just pointing out that it was considered.]

4.7 Changes to subclause 9.20, "Transformation of query specifications"

1. Change syntax rule 4 as follows:

4) If the <select list> of *GWQ* ~~immediately~~ simply contains ~~<asterisk>~~ <select list all columns> or simply contains <qualified asterisk>, then Syntax Rules of this Subclause are applied with *GWQ* as *QUERY SPEC IN*; let *GWQ2* be the *QUERY SPEC OUT* returned from the application of those Syntax Rules; otherwise, let *GWQ2* be *GWQ*.

[Note: I think this text has a pre-existing fault. The phrase "Syntax Rules of this Subclause" ought to refer to the syntax rules 7 and 11 of Subclause 7.16, "<query specification>", where the asterisks are expanded. This was apparently adjusted incorrectly when subclause 9.20 was factored out of subclause 7.16 (source code says W09-009). It seems to me that this refactoring might be incomplete, because here subclause 9.20 calls back into 7.16 without a proper subclause signature (which was the point that W09-009 wanted to fix). The proper fix might be to make the syntax rules that expand the asterisks into their own callable subclause. A quick fix could use wording like in subclause 7.13, "<group by clause>", shown earlier.]

5 Language opportunities

- Consider whether an EXCLUDE clause could be applied to <all fields reference>.

6 Checklist

1.	Interactions with other concurrent proposals identified and editorial assistance given	**see also BCN-035**
2.	Concepts	n/a
3.	Access Rules	n/a
4.	Conformance Rules, including the relevant Annexes	**yes**
5.	Lists of statements by category	n/a
6.	Collation coercibility determination for changes related to character strings	n/a
7.	Closing Possible Problems when a proposal resolves them	none
8.	Any new Possible Problems clearly identified	**yes**
9.	Reserved and non-reserved keywords	**yes**
10.	Information and Definition Schemas	n/a

11.	Implementation-defined and –dependent Annexes	n/a
12.	Incompatibilities Annex	n/a
1.	Table of identifiers used by diagnostics statements	n/a
2.	Embedded SQL bindings and host language implications	n/a
3.	Dynamic SQL issues: including Dynamic descriptor areas	n/a
4.	PSM impact	none
5.	Schemata impact	none
6.	XML impact	none
7.	MDA impact	none
8.	PGQ impact	none

— end of the paper —