LanguageManual Union

Union Syntax

```
select_statement UNION [ALL | DISTINCT] select_statement UNION [ALL | DISTINCT] select_statement ...
```

UNION is used to combine the result from multiple SELECT statements into a single result set.

- Hive versions prior to 1.2.0 only support UNION ALL (bag union), in which duplicate rows are not eliminated.
- In Hive 1.2.0 and later, the default behavior for UNION is that duplicate rows are removed from the result. The optional DISTINCT keyword has no effect other than the default because it also specifies duplicate-row removal. With the optional ALL keyword, duplicate-row removal does not occur and the result includes all matching rows from all the SELECT statements.

You can mix UNION ALL and UNION DISTINCT in the same query. Mixed UNION types are treated such that a DISTINCT union overrides any ALL union to its left. A DISTINCT union can be produced explicitly by using UNION DISTINCT or implicitly by using UNION with no following DISTINCT or ALL keyword.

The number and names of columns returned by each *select_statement* have to be the same. Otherwise, a schema error is thrown.

UNION within a FROM Clause

If some additional processing has to be done on the result of the UNION, the entire statement expression can be embedded in a FROM clause like below:

```
SELECT *
FROM (
  select_statement
  UNION ALL
  select_statement
) unionResult
```

For example, if we suppose there are two different tables that track which user has published a video and which user has published a comment, the following query joins the results of a UNION ALL with the user table to create a single annotated stream for all the video publishing and comment publishing events:

```
    SELECT u.id, actions.date
    FROM (
        SELECT av.uid AS uid
        FROM action_video av
        WHERE av.date = '2008-06-03'
        UNION ALL
        SELECT ac.uid AS uid
        FROM action_comment ac
        WHERE ac.date = '2008-06-03'
     ) actions JOIN users u ON (u.id = actions.uid)
```

Unions in DDL and Insert Statements

Unions can be used in views, inserts, and CTAS (create table as select) statements. A query can contain multiple UNION clauses, as shown in the syntax above.

Applying Subclauses

To apply ORDER BY, SORT BY, CLUSTER BY, DISTRIBUTE BY or LIMIT to an individual SELECT, place the clause inside the parentheses that enclose the SELECT:

```
SELECT key FROM (SELECT key FROM src ORDER BY key LIMIT 10)subq1
UNION
SELECT key FROM (SELECT key FROM src1 ORDER BY key LIMIT 10)subq2
```

To apply an ORDER BY, SORT BY, CLUSTER BY, DISTRIBUTE BY or LIMIT clause to the entire UNION result, place the ORDER BY, SORT BY, CLUSTER BY, DISTRIBUTE BY or LIMIT after the last one. The following example uses both ORDER BY and LIMIT clauses:

```
SELECT key FROM src
UNION
SELECT key FROM src1
ORDER BY key LIMIT 10
```

Column Aliases for Schema Matching

UNION expects the same schema on both sides of the expression list. As a result, the following query may fail with an error message such as "FAILED: SemanticException 4:47 Schema of both sides of union should match."

```
INSERT OVERWRITE TABLE target_table
  SELECT name, id, category FROM source_table_1
  UNION ALL
  SELECT name, id, "Category159" FROM source_table_2
```

In such cases, column aliases can be used to force equal schemas:

```
INSERT OVERWRITE TABLE target_table
  SELECT name, id, category FROM source_table_1
  UNION ALL
  SELECT name, id, "Category159" as category FROM source_table_2
```

Column Type Conversion

Before HIVE-14251 in release 2.2.0, Hive tries to perform implicit conversion across Hive type groups. With the change of HIVE-14251, Hive will only perform implicit conversion within each type group including string group, number group or date group, not across groups. In order to union the types from different groups such as a string type and a date type, an explicit cast from string to date or from date to string is needed in the query.

```
  SELECT name, id, cast('2001-01-01' as date) d FROM source_table_1
  UNION ALL
  SELECT name, id, hiredate as d FROM source_table_2
```

Version Information

Version information

In Hive 0.12.0 and earlier releases, unions can only be used within a subquery such as "SELECT * FROM (*select_statement* UNION ALL *select_statement* UNION ALL ...) *unionResult*".

As of Hive 0.13.0, unions can also be used in a top-level query: "*select_statement* UNION ALL *select_statement* UNION ALL ...". (See HIVE-6189.)

Before Hive 1.2.0, only UNION ALL (bag union) is supported. UNION (or UNION DISTINCT) is supported since Hive 1.2.0. (See HIVE-9039.)