CS252: Mapping Tutorial D's Relational Operators to SQL

Hugh Darwen

Some people who are comfortable with **Tutorial D** finds SQL difficult. Others, comfortable with SQL, find **Tutorial D** difficult. The following mapping might help anybody who is in either of those two camps. For each **Tutorial D** relational operator I give a general invocation of it on the left and, on the right, an SQL expression that is as near as possible equivalent to it.

The symbols *r*, *r1* and *r2* stand for arbitrary relational expressions in **Tutorial D**. The SQL counterpart of an arbitrary relation expression is anything you can write as an operand in a FROM clause, which is either a simple table name, or a table name followed by an alias, or (*query expression*) followed by an alias, where *query expression* is an arbitrary complete SQL query.

It is assumed that in the SQL expressions *r*, *r1* and *r2* stand for tables in which each column has a name, no two columns have the same name, NULL does not appear, and no row appears more than once in the same table.

1. Projection

r { *a, b, c* }	SELECT DISTINCT *a, b, c* FROM *r*
r { ALL BUT *a, b, c* }	*no counterpart*

The ALL BUT case just has to be translated into the regular case.

2. JOIN

r1 JOIN *r2*	*r1* INNER NATURAL JOIN *r2*

In the SQL the word INNER can be omitted. You can also achieve a join in SQL by longhand (and this was the only way of doing it before 1992):

SELECT *a, b, c …* FROM *r1, r2* WHERE *r1.c1 = r2.c1 AND … r1.cn = r2.cn* where *c1, …, cn* are the common columns. If the SELECT list includes each column of *r1* exactly once and each column of *r2* that is not also a column of *r1* exactly once, then the effect of a **Tutorial D** JOIN is achieved.

3. RENAME

r RENAME (*a* AS *x, b* AS *y*)	SELECT *a* AS *x, b* AS *y, c1, …, cn* FROM *r*

where *c1, …, cn* are the remaining columns of *r*. The effect of RENAME isn't required so often in SQL because of its use of column-name qualifiers to distinguish between two columns of the same name in different operands.

4. Extension

EXTEND *r* ADD (*f1* AS *x, f2* AS *y*)	SELECT *r.*, f1* AS *x, f2* AS *y* FROM *r*

where *f1* and *f2* are arbitrary formulae. Note carefully that in standard SQL and many implementations * *must* be qualified as shown here in "*r*.*".

5. SUMMARIZE

SUMMARIZE *r* BY { *a*, *b* } ADD (*f1* AS *x*, *f2* AS *y*)	SELECT *a*, *b*, *f1* AS *x*, *f2* AS *y* FROM *r* GROUP BY *a*, *b*

where *f1* and *f2* are arbitrary formulae involving aggregation.

6. UNION

r1 UNION *r2*	SELECT * FROM *r1* UNION SELECT * FROM *r2*

In the SQL the columns of *r2* have to be in the same order as those of *r1*.

7. NOT MATCHING

r1 [NOT] MATCHING *r2*	SELECT * FROM *r1* WHERE [NOT] EXISTS (SELECT * FROM *r2* WHERE *r1.c1* = *r2.c1* … AND *r1.cn* = *r2.cn*)

where *c1*, …, *cn* are the common columns of *r1* and *r2*.

8. Difference

r1 MINUS *r2*	SELECT * FROM *r1* EXCEPT SELECT * FROM *r2*

In the SQL the columns of *r2* have to be in the same order as those of *r1*.

9. Intersection

r1 INTERSECT *r2*	SELECT * FROM *r1* INTERSECT SELECT * FROM *r2*

In the SQL the columns of *r2* have to be in the same order as those of *r1*.

10. GROUP/UNGROUP

SQL has no counterparts for these **Tutorial D** operators.

See also **SQL Subqueries: Counterparts in Tutorial D**.

End