CSC 365

Introduction to Database Systems

SELECT - Other JOIN Types

In addition to `[INNER] JOIN` (theta join in relational algebra) SQL supports another type of join (`OUTER`) that has three variations:

```
LEFT [OUTER] JOIN

RIGHT [OUTER] JOIN

FULL [OUTER] JOIN
```

LEFT OUTER JOIN

```
SELECT *
FROM A
  LEFT OUTER JOIN B ON (A.id = B.id)
```

Produces a result containing *all* records from A, paired with matching records from B. If a record from A has no match in B, the record from A is listed along with empty (`null`-padded) columns from B.

How does this differ from a cross product?

Can we arrive at the same result using our five "primitive" relational algebra operators?

LEFT OUTER JOIN Example

```
-- List every airplane along with any associated flight details
-- For planes that are not in use (no flights), list airplane
-- details with "null" flight information
SELECT *
FROM Airplane AS A
  LEFT OUTER JOIN Flight AS F ON A.TailNum = F.TailNum
ORDER BY A.TailNum
```

RIGHT OUTER JOIN

```
SELECT *
FROM A
  RIGHT OUTER JOIN B ON (A.id = B.id)
```

Produces a result containing *all* records from **B**, paired with matching records from A. If a record from B has no match in A, the record from B is listed , preceded by `null`-padded columns from A.

RIGHT OUTER JOIN

```
SELECT A.id, A.a_val, B.b_val
FROM A RIGHT OUTER JOIN B ON (A.id = B.id)
```

...is equivalent to...

```
SELECT A.id, A.a_val, B.b_val
FROM B LEFT OUTER JOIN A ON (A.id = B.id)
```

Here we switched A & B,
changed `RIGHT` to `LEFT`

RIGHT OUTER JOIN - Why?

Why do we need both `LEFT` and `RIGHT OUTER JOIN`?

One example:

```
SELECT *
FROM (A
   LEFT JOIN B ON (A.id = B.id))
   RIGHT JOIN C ON (B.id = C.id)
```

Note: Join order / precedence becomes important when using `OUTER` joins!

FULL OUTER JOIN

```
SELECT *
FROM A
  FULL OUTER JOIN B ON (A.id = B.id)
```

The result includes *all* records from *both* A and B, matching them where possible. If no match, the unmatched side is padded with `null`.

Current versions of MySQL do not support `FULL OUTER JOIN`.

FULL OUTER JOIN

MySQL does not support `FULL OUTER JOIN`. In MySQL, the syntax may be emulated as follows:

```
(SELECT * FROM A LEFT OUTER JOIN B ON (A.id = B.id))
 UNION ALL
(SELECT * FROM A RIGHT OUTER JOIN B ON (A.id = B.id)
 WHERE A.id IS NULL)
```

FULL OUTER JOIN Example

```
(SELECT *
 FROM Student AS s
   LEFT OUTER JOIN Department AS d ON (s.MajorCode = d.Code))
UNION ALL
(SELECT *
 FROM Student AS s
   RIGHT OUTER JOIN Department AS d ON (s.MajorCode = d.Code)
 WHERE s.StudentID IS NULL)
ORDER BY StudentID, Code
```

How does the result of `FULL OUTER JOIN` differ from cartesian product?

SQL Joins

A single SQL SELECT statement may include multiple JOINs. Also, multiple JOIN types may be combined in a single SELECT.

```
SELECT P.Name AS PilotName, A.TailNum, Make, Model, Runway, Date
FROM (Airplane AS A
  INNER JOIN Flight AS F ON (A.TailNum = F.TailNum) )
  RIGHT OUTER JOIN Pilot P ON (F.PilotID = P.PilotID)
ORDER BY PilotName
```

Joins are evaluated left-to-right, unless parentheses are present

OUTER Joins - Relational Algebra Notation

`LEFT OUTER JOIN`	⟕	$\overset{\circ}{\bowtie}_{L}$
`RIGHT OUTER JOIN`	⟖	$\overset{\circ}{\bowtie}_{R}$
`FULL OUTER JOIN`	⟗	$\overset{\circ}{\bowtie}$

Also possible to express a *theta outer join* (⟕$_{\theta}$) using subscript notation we saw with regular theta (AKA inner) joins. Additional notation notes:

- Our textbook uses the ("up tack") symbol ⊥ to represent `null` values.
- Other resources represent `null` using lower-case omega (ω)

Outer Joins - Examples

Suppliers/Parts 7 - parts not listed in the catalog

HW4 Exercise 9 - Bands in which Irmin Schmidt did not play

List all days of the week, along with the number of Twists purchased on that day. Show a zero if no twists were purchased on particular day.

SQL NULL - Truth Table

x	y	x AND y	x OR y	NOT x
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	UNKNOWN	UNKNOWN	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
UNKNOWN	TRUE	UNKNOWN	TRUE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	UNKNOWN
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	UNKNOWN	FALSE	UNKNOWN	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE