



Database Management SQL Joins

By: Meyer Tanuan (2022), Glenn Paulley (2015), John Mckay (2011)

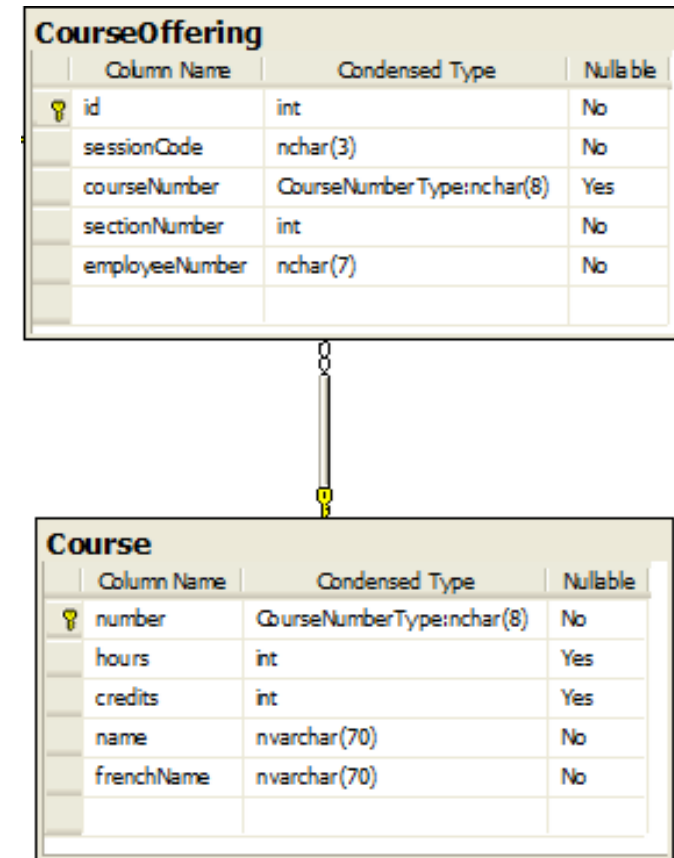
What does it mean to *join* two tables?

- In any database we typically have different tables to represent different types of “things” – abstract objects or concepts, or physical entities such as people, or car parts, or
- In a well-designed database, rows in every table are uniquely identified by a value called a ***primary key (PK)***
 - *Examples: student number, invoice number, course number*
- In a relational database, we represent relationships through the existence of primary key values in ***other*** tables
 - When these values exist, they are termed ***foreign keys (FK)***



Representing relationships

- In a well-designed relational database, all tables have primary keys (unique identifiers)
- A table that has a primary key involved in a relationship is sometimes called a “*master*”, or “*parent*” table
- A table that contains a reference to a primary key of the other table is sometimes called a “*detail*” or “*child*” table
- Zero or more rows in the detail table can refer to the same row in the master table




Inner joins: Course and CourseOffering tables

Number	Name	Credits
CDEV1020	Co-op Prep	1
INFO2080	SD: Design	3

parent table with PK

CourseNumber	SessionCode
CDEV1020	F08
CDEV1020	F09
INFO2080	W09

child table with FK



CourseNumber	Name	SessionCode
CDEV1020	Co-op Prep	F08
CDEV1020	Co-op Prep	F09
INFO2080	SD: Design	W09

Joins

- When you query the database, you often need to join the data from more than one table
 - You need to relate information from one table to another, either
 - Looking for matching values (typically matching primary and foreign key values), or
 - Looking for the absence of any matches
 - You need more detailed information about an instance of an “object”, such as it’s name – not merely its primary key (unique identifier)



Joins: What are the available courses?

```
SELECT courseNumber  
  ,Course.name  
  ,sessionCode  
  ,capacity  
  ,enrollment
```

Specify table.column (mandatory if there is ambiguity)

List the tables to be joined

```
FROM CourseOffering, Course
```

```
WHERE Course.number = CourseOffering.courseNumber
```

Join condition; in this case retrieve rows where the values in the course number columns match

Partial result of join query above

courseNumber	name	sessionCode	capacity	enrollment
CDEV1020	Co-op and Career Preparation	F08	30	14
CDEV1020	Co-op and Career Preparation	F09	30	18
CDEV1020	Co-op and Career Preparation	F10	30	20
CDEV1020	Co-op and Career Preparation	F11	30	23
CDEV1020	Co-op and Career Preparation	F12	30	21
CDEV1020	Co-op and Career Preparation	F13	30	22
CDEV1020	Co-op and Career Preparation	F14	30	19
CDEV1020	Co-op and Career Preparation	F15	30	25
CDEV1020	Co-op and Career Preparation	F16	30	24
CDEV1020	Co-op and Career Preparation	F17	30	23
CDEV1020	Co-op and Career Preparation	F18	30	25
CDEV1020	Co-op and Career Preparation	F19	30	22

Inner joins: Query Designer with explicit join syntax

Query Designer

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...
name		Course	<input checked="" type="checkbox"/>				
courseNumber		CourseOffering	<input checked="" type="checkbox"/>				
sessionCode		CourseOffering	<input checked="" type="checkbox"/>				
capacity		CourseOffering	<input checked="" type="checkbox"/>				
enrollment		CourseOffering	<input checked="" type="checkbox"/>				

SELECT Course.name, CourseOffering.courseNumber, CourseOffering.sessionCode, CourseOffering.capacity, CourseOffering.enrollment
FROM Course INNER JOIN
CourseOffering ON Course.number = CourseOffering.courseNumber

OK Cancel

- Query ->
Design in Editor...
- Select Course
 - Click Add
- Select CourseOffering
 - Click Add
- Check 5 columns
- Explicit join syntax:
 - **INNER JOIN**
 - **ON**

Qualifying column names

- If the same column name appears in more than one table in a join, it is *ambiguous*
- Column names must be qualified in order to resolve the ambiguity:

SELECT table.column

Correlation names (table alias)

- You can qualify each table name with an alias (typically one or two letters) to simplify the SQL syntax in your queries:

```
SELECT co.courseNumber  
       ,c.name  
       ,co.sessionCode  
       ,co.capacity  
       ,co.enrollment  
FROM CourseOffering AS co  
     , Course AS c  
WHERE c.number = co.courseNumber
```

More complex joins

- You can join as many tables as you like – but complex queries can often require additional execution time and computer resources such as memory

```
SELECT co.courseNumber, c.name, co.sessionCode  
      , p.name AS 'program', co.capacity, co.enrollment  
FROM CourseOffering co, Course c, ProgramCourse pc, Program p  
WHERE c.number = co.courseNumber AND c.number = pc.courseNumber  
      AND pc.programCode = p.code  
ORDER BY co.courseNumber
```

Complex joins: Query Designer with explicit join syntax

Query Designer

CourseOffering

- * (All Columns)
- ☒ id
- ☒ sessionCode
- ☒ courseNumber
- ☐ sectionNumber
- ☐ employeeNumber
- ☒ capacity
- ☒ enrollment

Program

- * (All Columns)
- ☐ code
- ☐ acronym
- ☐ campusCode
- ☐ schoolCode
- ☐ credentialCode
- ☒ name
- ☐ frenchName

Course

- * (All Columns)
- ☐ number
- ☐ hours
- ☐ credits
- ☒ name

ProgramCourse

- * (All Columns)
- ☐ programCode
- ☒ courseNumber
- ☐ semester

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
courseNumber		CourseOff...	<input checked="" type="checkbox"/>						
name		Course	<input checked="" type="checkbox"/>						
sessionCode		CourseOff...	<input checked="" type="checkbox"/>						
name	Expr1	Program	<input checked="" type="checkbox"/>						
capacity		CourseOff...	<input checked="" type="checkbox"/>						
enrollment		CourseOff...	<input checked="" type="checkbox"/>						

SELECT CourseOffering.courseNumber, Course.name, CourseOffering.sessionCode, Program.name AS Expr1, CourseOffering.capacity, CourseOffering.enrollment
FROM CourseOffering INNER JOIN
Course ON CourseOffering.courseNumber = Course.number INNER JOIN
ProgramCourse ON Course.number = ProgramCourse.courseNumber INNER JOIN
Program ON ProgramCourse.programCode = Program.code

OK Cancel

- Query -> **Design in Editor...**
 - Add tables 1-at-a-time:
 - CourseOffering, Course, ProgramCourse, Program
- Check 6 columns
- Explicit join syntax:
 - **INNER JOIN, ON**
- Next steps:
 - Modify column alias:
 - Expr1 to 'program'
 - Add ORDER BY ...

Joins and WHERE

- The WHERE clause in a select statement can include join conditions and other conditions

```
WHERE c.number = co.courseNumber  
      AND c.number = pc.courseNumber  
      AND pc.programCode = p.code  
      AND p.acronym = 'ITID'
```

Join conditions

Join condition

Restriction condition

Cross Joins

If you don't specify a join condition, *every* row in the first table is joined with *every* row in the second table

```
SELECT p.lastName, p.firstName, e.businessPhone  
FROM Person p, Employee e
```

Note: This query returns 6076 meaningless rows

- Also known as Cartesian product

```
SELECT Person.lastName, Person.firstName, Employee.businessPhone  
FROM Person CROSS JOIN Employee
```

Self-Joins

- A self join is simply a table joined to itself
- You must create two distinct table aliases and qualify their usage with each column reference
- Self joins are often required when a table has a foreign key to itself, as when:
 - A table of Employees has a column with the ID of each employee's manager
 - A table of Parts has a column with the ID of the part assembly that part is in
 - A table of Departments has a column denoting the division of the firm that department is part of



Self-Join query example

Question: What are the IDs and phone extensions of all employees and their immediate managers?

```
SELECT e.number, e.extension, m.number, m.extension  
FROM Employee e, Employee m  
WHERE e.reportsTo = m.number
```

Explicit join syntax:

```
SELECT e.number, e.extension, m.number, m.extension  
FROM Employee e INNER JOIN Employee m  
ON e.reportsTo = m.number;
```


Result of self-join query above

number	extension	number	extension
2117745	3578	4427732	4298
3214350	8529	4427732	4298
3455730	7693	4427732	4298
3514239	8371	4427732	4298
3566163	2378	4427732	4298
5512736	6183	6860242	1795
6886352	6742	8436780	7361
7572373	1478	4427732	4298
8135170	8529	4427732	4298
8383259	5162	6434468	9832

Join table expressions

- So far, we have considered
 - SQL queries with two or more tables
 - Queries that contain a FROM clause that lists all of tables to be queried, with (join) conditions in the WHERE clause
- In SQL, you can create more complex joins within a query's FROM clause by using table expressions and explicit use of the JOIN keyword
 - MS SQL Server supports INNER joins. It also supports LEFT, RIGHT, and FULL outer join table expressions, and CROSS join table expressions

Important: semantically, the FROM clause is computed before the WHERE clause is applied



Join table expressions

- Inner joins and Cross joins are both associative and commutative
 - $(L \text{ JOIN } R) \text{ JOIN } T$ is equivalent to $(R \text{ JOIN } T) \text{ JOIN } L$
 - $(L \text{ CROSS JOIN } R) \text{ CROSS JOIN } T$ is equivalent to $(R \text{ CROSS JOIN } T) \text{ CROSS JOIN } L$
- FULL OUTER JOIN is commutative
- LEFT OUTER JOIN does not commute and is also not associative
- RIGHT OUTER JOIN is simply the reverse of LEFT OUTER JOIN
 - RIGHT OUTER JOINS are rarely specified in practice

Outer Joins

- Outer joins are similar to inner joins
- In an outer join, the system retrieves rows from the two tables that match on the join condition, but
 - In addition, return rows from the “preserved” table that do not match with the other table
 - Values from the other table where rows do not match are returned as NULL
 - Hence one of the tables, or both in the case of FULL OUTER JOIN, is termed the “null-supplying” table
- Outer joins can only be specified as table expressions
 - An outer join cannot be computed from the restriction (WHERE) of a Cartesian product



Why do we need outer joins?

- Many reporting applications require the use of left outer joins because of the need to display all objects (rows) from a table, even when they don't match anything else
- Examples:
 - List all students, and the number of courses they are enrolled in. If a student is not taking any courses, show 0 (zero)
 - List all parts and their warehouse locations; if a part is not in stock, list it anyway
 - List all employees and their immediate manager, including any employees who don't have managers
- LEFT OUTER JOINS are very common in practice

LEFT outer joins

- Consider the previous question: What are the ids and phone extensions of all employees and their immediate managers?

```
SELECT e.number, e.extension, m.number, m.extension  
FROM Employee e, Employee m  
WHERE e.reportsTo = m.number
```

Note: The result contains only ten employees – but there are 14 employees in the Employee table!

LEFT outer joins vs. inner joins

- The problem: performing an inner join of the Employee table with itself, matching on the “reportsTo” column, eliminates from the result any employee that **does not have a manager**
- Left outer joins solve this problem by ensuring that one half of the join is returned in its entirety
 - We say that that table is preserved in the result



Using a LEFT outer join

- Consider the previous question: What are the ids and titles of all employees and their immediate managers?

```
SELECT e.number, e.extension, m.number, m.extension  
FROM Employee AS e LEFT OUTER JOIN Employee AS m  
ON (e.reportsTo = m.number)
```



LEFT outer join

- `SELECT * FROM L LEFT OUTER JOIN R ON (L.X = R.X)`
- Result consists of:
 - The combination of rows from L and R that match on their X-values (same as inner join), and
 - In addition, those rows of L that do not match any row of R on their X-values
 - In this case, such output rows have the values from the row of L, along with NULL attributes as placeholders for the non-existent matching row from table R
 - L is termed the *preserved* table, as all rows of L are preserved in the result (whether they match a row from R or not)
 - R is termed the *null-supplying* table



Another LEFT OUTER JOIN example

- Question: What are the prerequisites of 'COMP' courses?

Specify table.column
(mandatory if there is
ambiguity)

```
SELECT c.number, c.name, cp.prerequisiteNumber  
FROM Course c LEFT OUTER JOIN CoursePrerequisiteAnd cp  
ON( c.number = cp.courseNumber )  
WHERE c.number LIKE 'COMP%';
```

Null-supplying
table

Preserved table

Left Outer Join
table
expression

ON condition; in this case
retrieve rows where the
values in the course
number columns match

LEFT OUTER JOIN: how do these work?

number	name	credits	courseNumber	prerequisiteNumber
COMP1230	Database Design	4	COMP1230	INFO1570
COMP2050	Applied SQL	3	COMP2050	COMP1230
			COMP2160	COMP1380

Number	Name	prerequisiteNumber
COMP1230	Database Design	INFO1570
COMP1380	Advanced User Applications	NULL

No prerequisite for COMP1380

RIGHT OUTER JOIN

- Identical to LEFT OUTER JOIN simply by commuting the two sides, for example
 - FROM S LEFT OUTER JOIN T ON (S.X = T.X) is equivalent to
 - FROM T RIGHT OUTER JOIN S ON (S.X = T.X)
- RIGHT OUTER JOIN is rarely seen in practice
 - Since it is equivalent to LEFT OUTER JOIN
- DBMS systems may rewrite RIGHT OUTER JOINS to LEFT OUTER JOINS as part of query execution



FULL OUTER JOIN

- In a FULL OUTER JOIN, both sides of the join are both *preserved* and *null-supplied*
- A FULL OUTER JOIN such as `L FULL OUTER JOIN R ON (L.X = R.X)` is equivalent to the concatenation of three results:
 - The INNER JOIN of L and R using the predicate `L.X = R.X`; and
 - The LEFT OUTER JOIN of L and R, with L preserved and R null-supplied, returning only the non-matching rows; and
 - The RIGHT OUTER JOIN of L and R, with R preserved and L null-supplied, again returning only the non-matching rows



FULL OUTER JOIN Example – Step 1 (no join)

- Step 1: List the schools (no join)

```
USE SIS;
```

```
SELECT code, name  
FROM dbo.School  
ORDER BY code;
```

FULL OUTER JOIN – Step 1 Result

code	name
BUS	Business and Hospitality
CAA	Career and Academic Access
CLI	Conestoga Language Institute
EIT	Engineering and Information Technology
HLC	Health and Life Sciences and Community Services
LS	Liberal Studies
MD	Media and Design
TAP	Trades and Apprenticeship

FULL OUTER JOIN – Step 2

- Step 2: List the schools and the employees who work in those schools (**ordinary inner join**) – this retrieves (only) the rows that match – note the rows that are missing in the result

```
SELECT School.code, School.name  
       , Employee.number, Employee.extension  
FROM School INNER JOIN Employee  
       ON (School.code = Employee.schoolCode)  
ORDER BY School.code;
```


FULL OUTER JOIN – Step 2 Result

code	name	number	extension
BUS	Business and Hospitality	6434468	9832
BUS	Business and Hospitality	8383259	5162
EIT	Engineering and Information Technology	7572373	1478
EIT	Engineering and Information Technology	8135170	8529
EIT	Engineering and Information Technology	2117745	3578
EIT	Engineering and Information Technology	3214350	8529
EIT	Engineering and Information Technology	3455730	7693
EIT	Engineering and Information Technology	3514239	8371
EIT	Engineering and Information Technology	3566163	2378
EIT	Engineering and Information Technology	4427732	4298
TAP	Trades and Apprenticeship	6860242	1795
TAP	Trades and Apprenticeship	6886352	6742
TAP	Trades and Apprenticeship	8436780	7361

FULL OUTER JOIN – Step 3

- Step 3: Convert the INNER JOIN into a **LEFT OUTER JOIN**

```
SELECT School.code
      , School.name
      , Employee.number
      , Employee.extension
FROM School LEFT OUTER JOIN Employee
      ON (School.code = Employee.schoolCode)
ORDER BY School.code;
```

FULL OUTER JOIN – Step 3 Result

code	name	number	extension
BUS	Business and Hospitality	6434468	9832
BUS	Business and Hospitality	8383259	5162
CAA	Career and Academic Access	NULL	NULL
CLI	Conestoga Language Institute	NULL	NULL
EIT	Engineering and Information Technology	2117745	3578
EIT	Engineering and Information Technology	3214350	8529
EIT	Engineering and Information Technology	3455730	7693
EIT	Engineering and Information Technology	3514239	8371
EIT	Engineering and Information Technology	3566163	2378
EIT	Engineering and Information Technology	4427732	4298
EIT	Engineering and Information Technology	7572373	1478
EIT	Engineering and Information Technology	8135170	8529
HLC	Health and Life Sciences and Community Services	NULL	NULL
LS	Liberal Studies	NULL	NULL
MD	Media and Design	NULL	NULL
TAP	Trades and Apprenticeship	6860242	1795
TAP	Trades and Apprenticeship	6886352	6742
TAP	Trades and Apprenticeship	8436780	7361

- What do the NULL values signify?

FULL OUTER JOIN – Step 4

- Step 4: Execute a **RIGHT OUTER JOIN** instead

```
SELECT School.code, School.name  
       , Employee.number, Employee.extension  
FROM School RIGHT OUTER JOIN Employee  
       ON (School.code = Employee.schoolCode)  
ORDER BY School.code;
```

FULL OUTER JOIN – Step 4 Result

code	name	number	extension
NULL	NULL	5512736	6183
BUS	Business and Hospitality	6434468	9832
BUS	Business and Hospitality	8383259	5162
EIT	Engineering and Information Technology	7572373	1478
EIT	Engineering and Information Technology	8135170	8529
EIT	Engineering and Information Technology	2117745	3578
EIT	Engineering and Information Technology	3214350	8529
EIT	Engineering and Information Technology	3455730	7693
EIT	Engineering and Information Technology	3514239	8371
EIT	Engineering and Information Technology	3566163	2378
EIT	Engineering and Information Technology	4427732	4298
TAP	Trades and Apprenticeship	6860242	1795
TAP	Trades and Apprenticeship	6886352	6742
TAP	Trades and Apprenticeship	8436780	7361

- What do the NULL values signify?

FULL OUTER JOIN – Step 5

- Step 5: Compute the **FULL OUTER JOIN**

```
SELECT School.code, School.name  
       , Employee.number, Employee.extension  
FROM School FULL OUTER JOIN Employee  
       ON (School.code = Employee.schoolCode)  
ORDER BY School.code;
```



FULL OUTER JOIN – Step 5 Result

code name	number	extension
NULL NULL	5512736	6183
BUS Business and Hospitality	6434468	9832
BUS Business and Hospitality	8383259	5162
CAA Career and Academic Access	NULL	NULL
CLI Conestoga Language Institute	NULL	NULL
EIT Engineering and Information Technology	2117745	3578
EIT Engineering and Information Technology	3214350	8529
EIT Engineering and Information Technology	3455730	7693
EIT Engineering and Information Technology	3514239	8371
EIT Engineering and Information Technology	3566163	2378
EIT Engineering and Information Technology	4427732	4298
EIT Engineering and Information Technology	7572373	1478
EIT Engineering and Information Technology	8135170	8529
HLC Health and Life Sciences and Community Services	NULL	NULL
LS Liberal Studies	NULL	NULL
MD Media and Design	NULL	NULL
TAP Trades and Apprenticeship	6860242	1795
TAP Trades and Apprenticeship	6886352	6742
TAP Trades and Apprenticeship	8436780	7361

- What do the NULL values signify in this case?