IT 775
Database Technology
SQL-DML

Select Statement

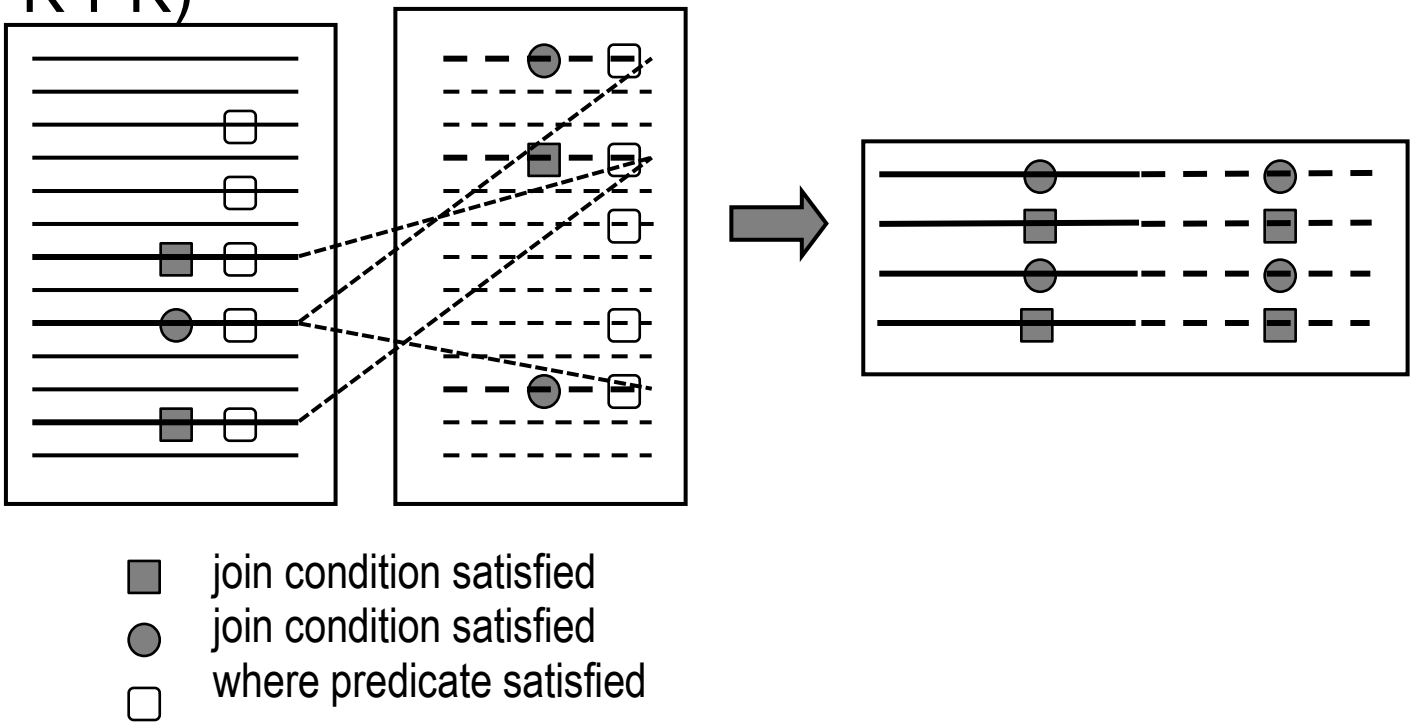
Joins

SQL Joins – The Basics

- SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

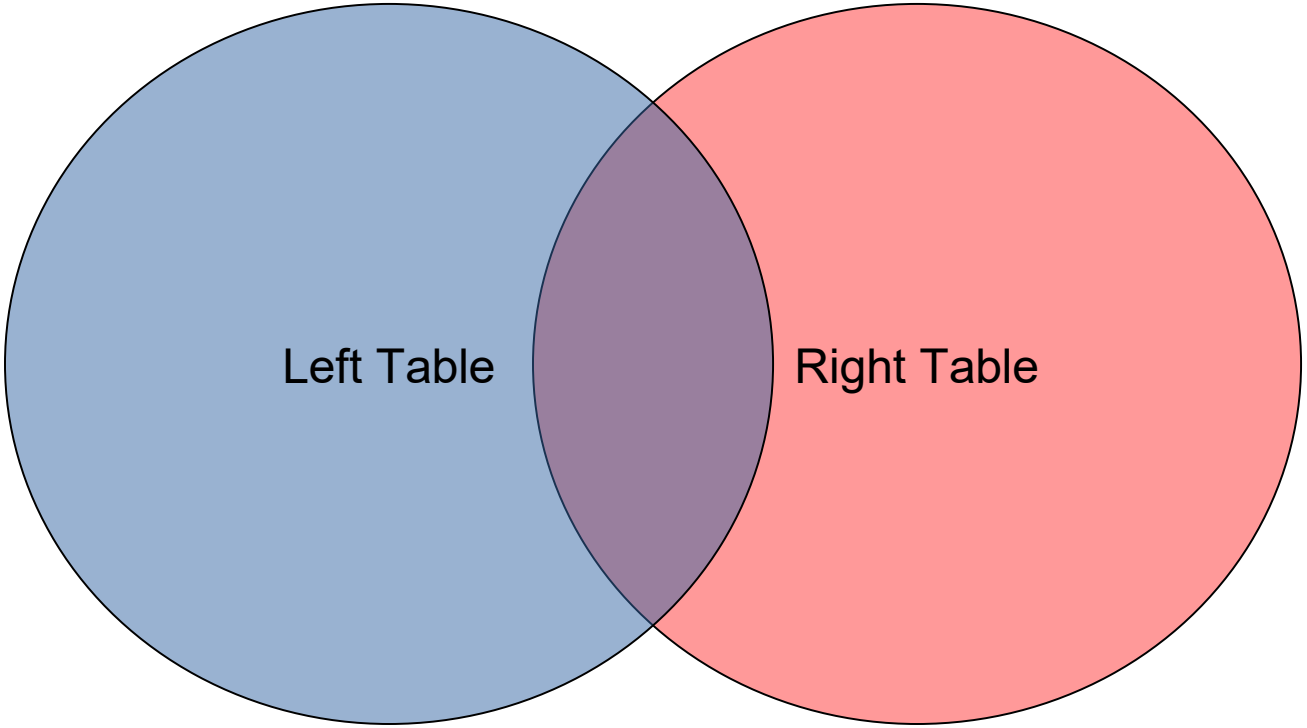
Join Queries

JOIN often matches a Foreign Key (FK) with the Primary (PK) it references (though not all JOINs are FK-PK)

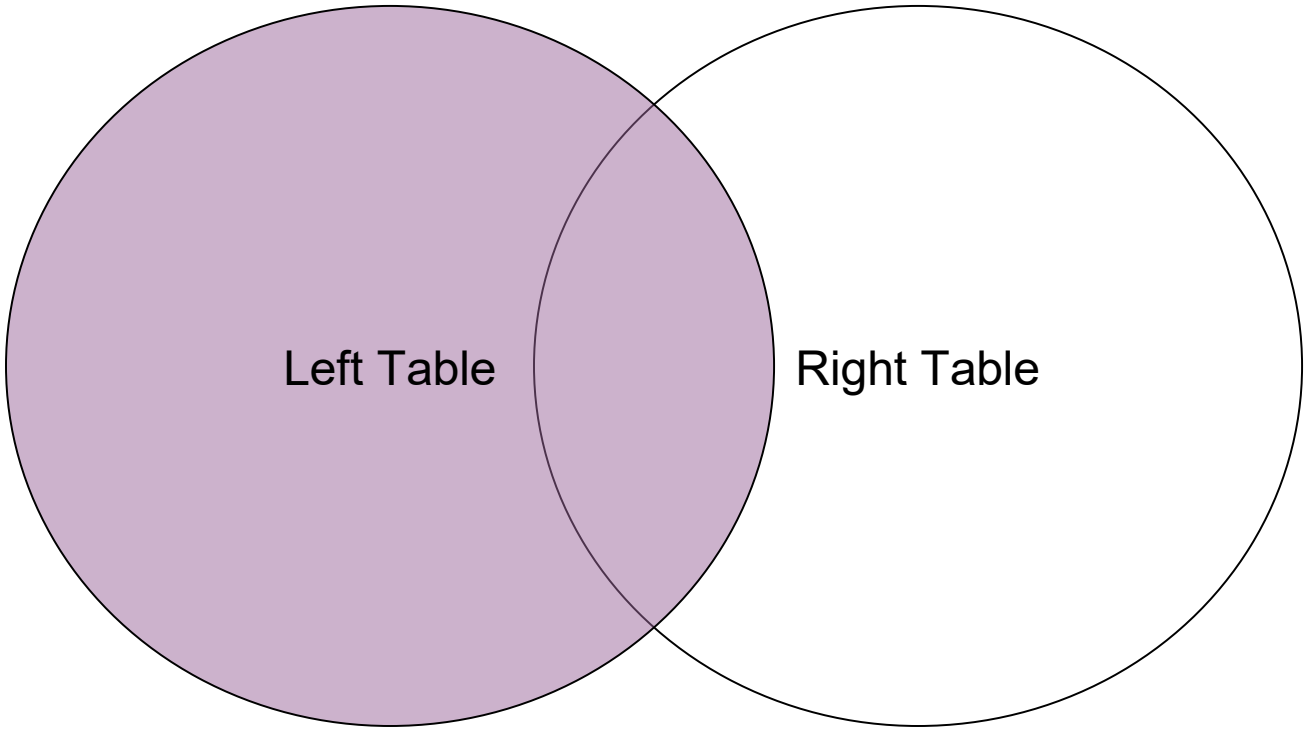


- result contains concatenated pairs of tuples, one from each relation, that satisfy the join condition

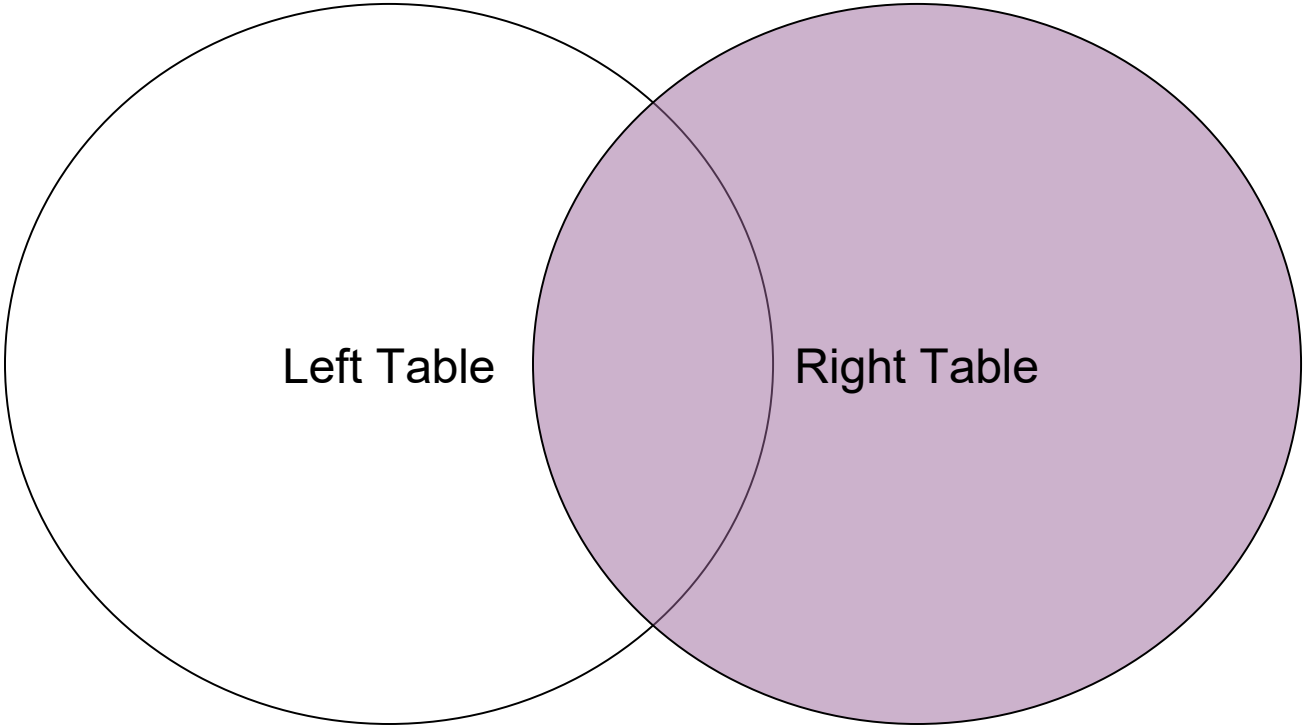
Joins Basic Concept
- Inner Join -



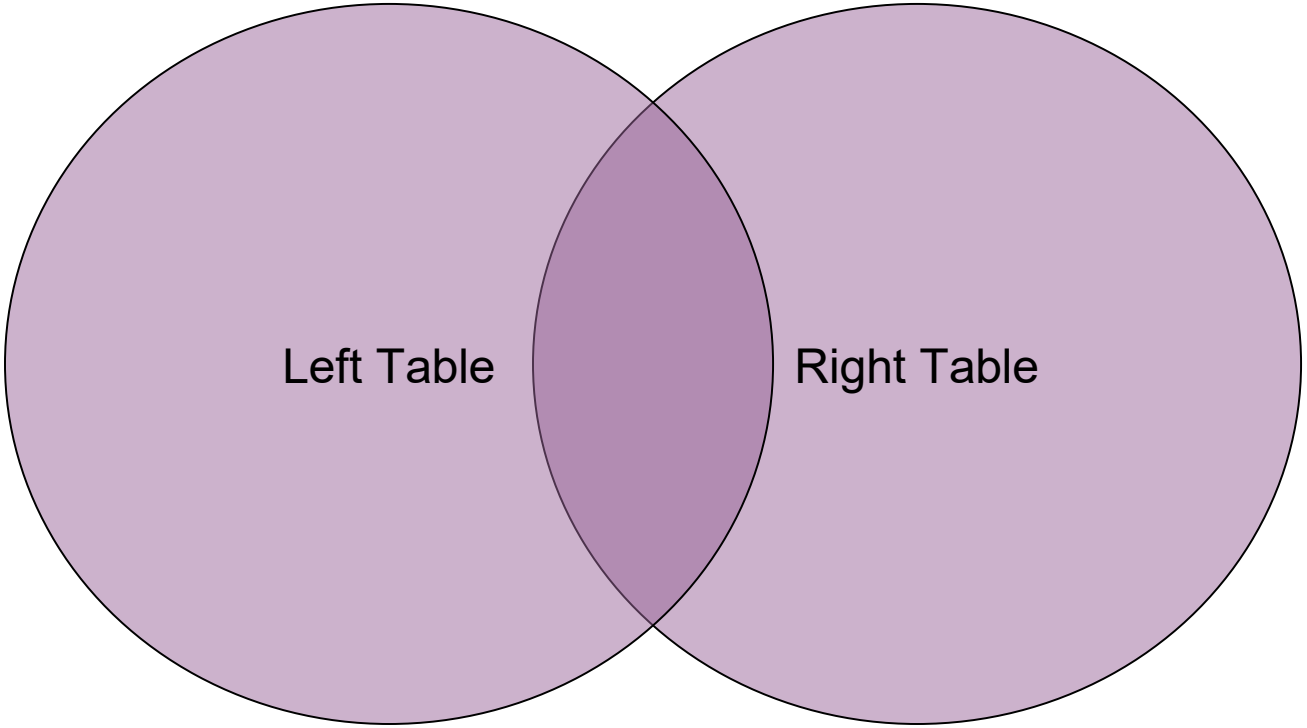
Joins Basic Concept
- Left Join -



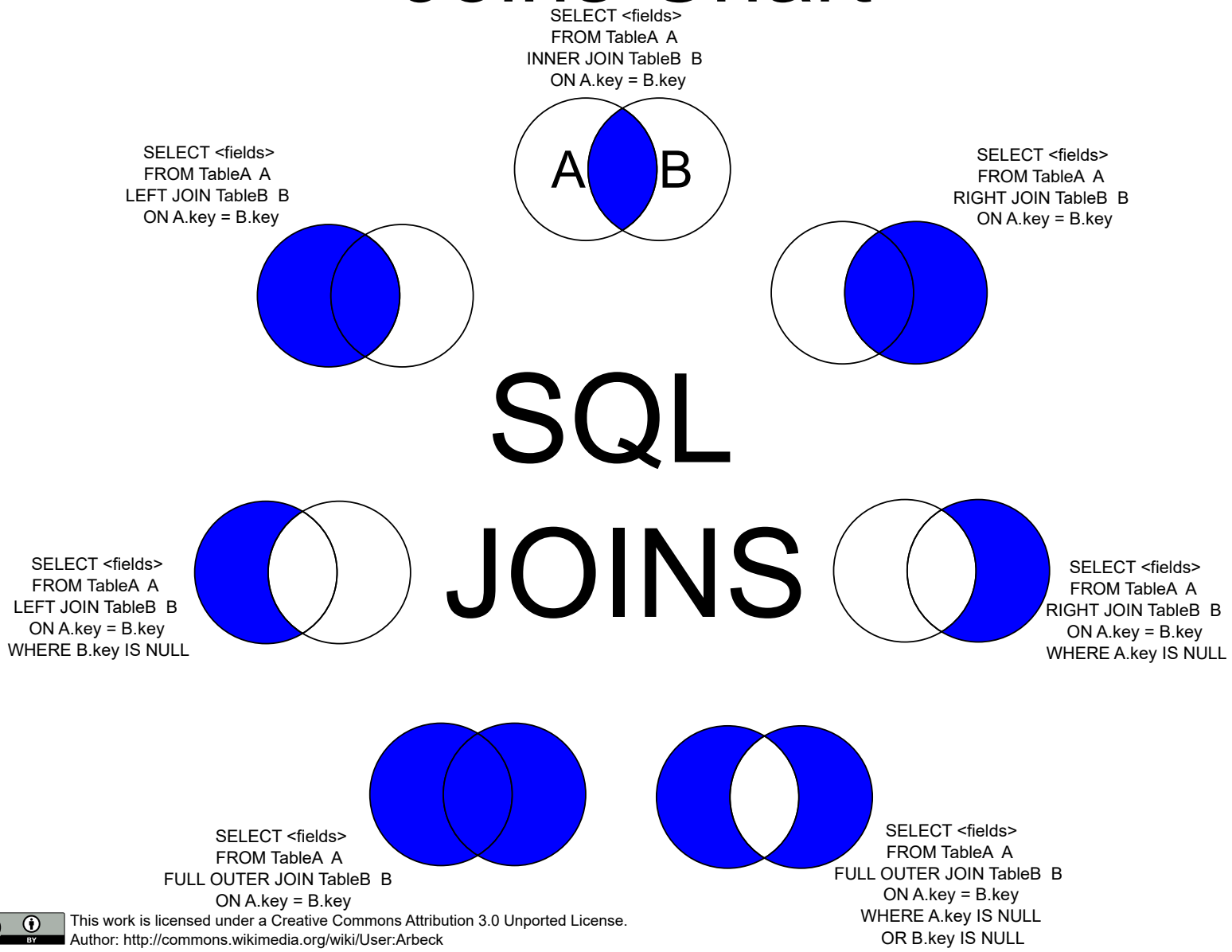
Joins Basic Concept
- Right Join -



Joins Basic Concept
- Full Outer Join -



Joins Chart



Join Syntax

- SQL join queries have two syntax variations
  - SQL-89 – WHERE clause specifies both
    - column value restrictions
    - join criteria
  - SQL-92 – WHERE clause specifies column restrictions
    - separate syntaxes for
      - inner join, including specialized equi-join and natural join
      - cross join
      - outer join, including left and right partial outer join

Join – SQL–89 Syntax

- SQL-89 Syntax

simple retrieval from multiple tables

```
SELECT      retrieval attributes in FROM clause tables
  FROM      list of source tables
WHERE       conditional expression
```

- e.g.,

```
SELECT student.sname, faculty.name, major, phone
FROM student, faculty
WHERE       major = faculty.dname
AND ( major = 'cs' OR major = 'math' );
```

Join – SQL–92 Syntax

- SQL–92 join syntax

  join–expr ::=
  table CROSS JOIN table |
  table [NATURAL] [join–type] JOIN [USING (attr–list) | ON join–cond]

  join–type ::= INNER | OUTER | (LEFT | RIGHT | FULL) [OUTER]

- general theta join – expresses any join condition
  - table JOIN table ON (join condition expression)
- equijoin – all join conditions are equality tests
  - table JOIN table
    - all attributes with matching names in both tables participate
  - table JOIN table USING (attr list)
    - only listed attributes with matching names participate
- natural join – equijoin, duplicate columns not in result
  - table NATURAL [JOIN] table
- defer OUTER and CROSS JOIN operations for now

Inner Join – General Theta Join

always valid even if

- join attributes have different names
- join conditions are not all equalities

```
SELECT    student.name, faculty.name, phone
  FROM    student JOIN faculty ON major=dname;

SELECT    student.name, faculty.name, phone
  FROM    student JOIN faculty
    ON    (major <> dname )
WHERE     ( major = 'cs' OR major = 'math' );
```

cross-table expressions move from WHERE to ON

within-table column restrictions remain in WHERE

Inner EquiJOIN

- use when
  - all join attribute pairs have the same name in both tables
  - join condition tests are all equality tests
  - Example -- given the following tables:

department(dname, office, phone, chair)
faculty(name, dname, office, phone)

  - join department & faculty over dname (only join attr)
  - do not want office & phone as join criteria

SELECT name, department.dname, chair FROM department JOIN faculty USING (dname);

Inner EquiJOIN

- can drop USING clause when
  - all join attribute pairs have the same name in both tables, and
  - no other columns in the tables have the same name
  - Example -- given the following tables:

student(name, *snbr*, major)
enroll(*snbr*, dept, cnbr)

```
SELECT * FROM student JOIN enroll WHERE major = 'cs';
```

  - schema of result

(name, student.snbr, major, enroll.snbr, dept, cnbr)

Inner Natural JOIN

- USING-free
- appropriate whenever equijoin is appropriate
- advantage: join attributes appear only once in result

Inner Natural JOIN

- Appropriate when
  - all join attributes have the same name in both tables, and
  - join condition tests are all equality tests
  - no other columns in both tables have the same name
  - e.g., given the following tables

    student(name, *snbr*, major)
    enroll(*snbr*, dept, cnbr)

    SELECT * FROM student NATURAL JOIN enroll WHERE major = ′cs′;
  - join attribute is *snbr* (only pair with same name)
  - schema of result

    (name, snbr, major, dept, cnbr)

SQL Join with Table Aliases

- aliases are a shorthand way to refer to tables
- always correct syntax to use aliases
- sometimes necessary, e.g., joining table with itself

SQL Join with Table Aliases

Aliases (or correlation variables) distinguish references to same table -- each refers to a table row (possibly the same row)

- e.g., list employees and their supervisors

  employee(name, id, title, dept, super), where super is a FK referencing the id of the employee's supervisor

```
SELECT    e.name, s.name
  FROM    employee e JOIN employee s ON ( e.super = s.id );
```

  - e refers to the employee tuple

s refers to the supervisor tuple

- for this prerequisite table

  prereq(dept cnbr pdept pcnbr)

- list courses and the prerequisites of their prerequisites

```
SELECT P.dept, P.cnbr, Q.pdept, Q.pcnbr FROM prereq P JOIN prereq Q
    ON ( P.pdept = Q.dept AND P.pcnbr = Q.cnbr );
```

N-Way Joins

- for tables:

  student (sname snbr major)
  enroll (snbr dept cnbr sect)
  course (dept cnbr cr)

instr (name dname cnbr sect)
faculty (name dname office phone)

- 3-way join
  - list students and their enrollments along with credits

```
SELECT sname, course.dept, course.cnbr, cr
  FROM student NATURAL JOIN enroll NATURAL JOIN course;
```

- 4-way join
  - list students, their instructors, and the instructors′ offices

```
SELECT student.name, faculty.name, office
  FROM ( ( student NATURAL JOIN enroll )
                  JOIN instr ON ( enroll.dept = instr.dname )
        ) JOIN faculty ON ( instr.name = faculty.name );
```

Different SQL Joins

Before we continue with examples, we will list the types of JOIN you can use, and the differences between them.

- **JOIN**: Return rows when there is at least one match in both tables
- **LEFT JOIN**: Return all rows from the left table, even if there are no matches in the right table
- **RIGHT JOIN**: Return all rows from the right table, even if there are no matches in the left table
- **FULL JOIN**: Return rows when there is a match in one of the tables

SQL INNER JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON
table_name1.column_name=table_name2
.column_name
```

INNER JOIN is the same as JOIN

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

The INNER JOIN keyword return rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

SQL LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

PS: In some databases LEFT JOIN is called LEFT OUTER JOIN.

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	

The LEFT JOIN keyword returns all the rows from the left table (Persons), even if there are no matches in the right table (Orders).

SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword Return all rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

SQL RIGHT JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

PS: In some databases RIGHT JOIN is called RIGHT OUTER JOIN.

SQL RIGHT JOIN Example

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the orders with containing persons -

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
		34764

The RIGHT JOIN keyword returns all the rows from the right table (Orders), even if there are no matches in the left table (Persons).

SQL FULL JOIN Keyword

The FULL JOIN keyword return rows when there is a match in one of the tables.

SQL FULL JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

SQL FULL JOIN Example

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the persons and their orders, and all the

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	
		34764

The FULL JOIN keyword returns all the rows from the left table (Persons), and all the rows from the right table (Orders). If there are rows in "Persons" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Persons", those rows will be listed as well.

Outer Join

SQL outer join of two tables is the union of

- the result of an inner join on the two tables
- unmatched left operand tuples extended with NULLs
- unmatched right operand tuples extended with NULLs

Outer Join has only SQL–92 syntax

```
SELECT   list of attributes from all tables
 FROM    table OUTER-JOIN-OPR table
         conditional expression for
         join conditions
WHERE    conditional expression
         for column value restrictions
```

OUTER-JOIN-OPR

- FULL | LEFT | RIGHT [OUTER] JOIN

Outer Join Conditions

- SQL–92 supports three forms of outer join
  - same ones as inner join
    - general form (ON)
    - equijoin –– when all join attributes have same names
      & all join conditions use equality (USING)
    - natural join –– when all join attributes have same names and no others do

SELECT * FROM student FULL JOIN dept ON major = dname;

N-way Outer Joins

- any or all joins in a cascaded join expression can be outer joins
- For example: list all students, their instructors, and their instructors' offices

student (*name* id major gpa advisor)
enroll (id dept course# sect#)
instr (name dept course# sect#)
faculty (*name* dname *office* phone)

N-way Outer Joins (cont)

- left outer join has all students, enrolled or not, but no empty courses
- right outer join has instructors with faculty info, all faculty included
- outer join includes all students & all instructors, active or not
- the last includes all instructors, faculty or not, and all faculty, instructional faculty or not

N-way Outer Joins (cont)

```
SELECT      student.name, faculty.name, office
  FROM      ( ( student LEFT JOIN enroll ON (student.id =
enroll.id) )
            FULL JOIN
              ( instr RIGHT JOIN faculty
                ON instr.name = faculty.name
              )
            ON  enroll.dept = instr.dept
           AND enroll.course# = instr.course#
           AND enroll.sect# = instr.sect#
            )
```

Cross Joins

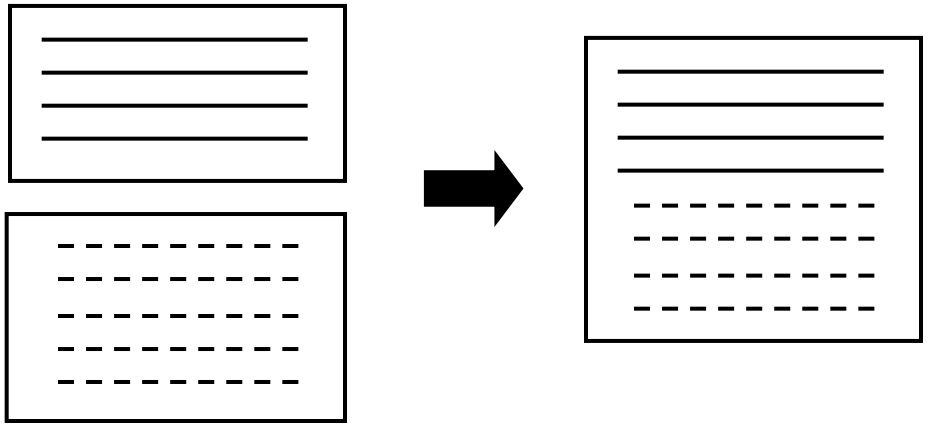
- cross join is a cartesian product of each tuple from the left side concatenated with every tuple on the right side
- there is no "join" condition because all tuples participate unconditionally
- this lists all students against every course

```
SELECT    student.name, faculty.name, office
  FROM    student CROSS JOIN faculty
```

- the result can be very large = n^2
- obviously one doesn't do this very often but it is useful for teaching certain concepts
- use another approach if possible

Set Queries

- syntax: $table_1$ set–op $table_2$ where set–op is one of:
  - UNION
  - ~~EXCEPT~~ not part of MySQL
  - ~~INTERSECTION~~ not part of MySQL
  - ~~CARTESIAN PRODUCT~~ sql CROSS JOIN is a join form
- $table_1$ and $table_2$ are
  - table names, or
  - SQL queries that return tables
- tables must be "union compatible" (but not for cross join)
- in SQL, compatibility is by position, not name



Union Compatibility

union compatibility in SQL means two tables must have corresponding columns (left–to–right order) that agree in number and type

names of corresponding columns need not match

```
SELECT firstname FROM student
UNION
SELECT lastname FROM student
```

- reliance on types rather than explicit domains means set operations can more easily be misused

consider the following meaningless query

```
SELECT name, dept, course# FROM student JOIN enroll
UNION
SELECT name, dept, course# FROM instr
```

Union Queries

combine rows from 2 separate tables into 1 table

SELECT column(s) FROM table(s) WHERE predicate
UNION
SELECT column(s) FROM table(s) WHERE predicate

selects can have any legal select form

Union Queries (cont.)

e.g., list the names and gpas of cs & math majors

```
SELECT name, snbr FROM student WHERE major = 'cs'
UNION
SELECT name, snbr FROM student WHERE major = 'math‘;
```

of course, boolean query can express this query

since both selects are from the same table

```
SELECT name, snbr FROM student WHERE major = 'cs' OR major =
    'math‘;
```

Union Examples

- list students, courses they have taken or are taking

```
SELECT sname, snbr, course.dept, course.cnbr, cr
  FROM student NATURAL JOIN enroll
                       NATURAL JOIN course
UNION
SELECT sname, snbr, course.dept, course.cnbr, cr
  FROM student NATURAL JOIN transcript
                       NATURAL JOIN course
WHERE grade >= 1.0
```

- this task requires union to do in a single query

Links

For Maria DB – Note the following

- https://mariadb.com/kb/en/basic-sql-statements
- https://mariadb.com/kb/en/joins/
- https://mariadb.com/kb/en/join-syntax/
- https://mariadb.com/kb/en/joining-tables-with-join-clauses/
- https://mariadb.com/kb/en/union/