HALMSTAD UNIVERSITY

DS4001 Databases (7.5 credits)

Lecture 7 – Intermediate SQL

Yuantao Fan

yuantao.fan@hh.se

Halmstad University

Overview

- Using string patterns and ranges
- Date and time
- Output Redirection & Control
- Nested queries
- Operation on multiple tables
- Views

Example Database

Course(cid, Course_name, Course_code, Credit_hours)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3

Teacher(tid, full_name, age, nationality)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

Teaches(tid, cid, hours)

tid	cid	hours
11	1	80
11	2	100
22	4	50
33	4	50
44	3	100

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3

```
SELECT *
FROM Course
WHERE ???
```

Find all courses with 'Data' in the name

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3

```
SELECT *
FROM Course
WHERE course_name LIKE "%Data%";
```

Find all courses with 'Data' in the name

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3

```
SELECT *
FROM Course
WHERE course_name LIKE "%Data%";
```

cid	course_name	course_code	credits
2	Data Structure	CS3320	4
4	Database	CS1310	3

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE ???
```

Find all courses with the second to last character is 'o' in the name

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name LIKE "%o_";
```

Find all courses with the second to last character is 'o' in the name

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name LIKE "%o_";
```

tid	full_name	age	nationality
22	Jens Jonathon	31	Sweden
44	Kayle Persson	33	UK

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

  - REGEXP
    - ^ beginning of a string
    - | or, e.g., ^j|Kayle'
    - [], "[gim]e"

    - -- ^ beginning
    - -- $ end
    - -- | logical or
    - -- [abcd]
    - -- [a-f]

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE ???
```

Find all teachers with full name start with 'j'

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - ’%’ – matching any substrings, including the empty ones
    - ’_’ – matching any character (length of one)

  - REGEXP
    - ^ beginning of a string
    - | or, e.g., ^j|Kayle’
    - [], “[gim]e”

    - -- ^ beginning
    - -- $ end
    - -- | logical or
    - -- [abcd]
    - -- [a-f]

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name REGEXP '^j';
```

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

  - REGEXP
    - ^ beginning of a string
    - | or, e.g., ^j|Kayle'
    - [], "[gim]e"

    - -- ^ beginning
    - -- $ end
    - -- | logical or
    - -- [abcd]
    - -- [a-f]

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE ???
```

Find all teachers with full name start with 'j' or contain 'Kayle'

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - ’%’ – matching any substrings, including the empty ones
    - ’_’ – matching any character (length of one)

  - REGEXP
    - ^ beginning of a string
    - | or, e.g., ^j|Kayle’
    - [], “[gim]e”

    - -- ^ beginning
    - -- $ end
    - -- | logical or
    - -- [abcd]
    - -- [a-f]

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name REGEXP '^j|Kayle';
```

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
44	Kayle Persson	33	UK

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - ’%’ – matching any substrings, including the empty ones
    - ’_’ – matching any character (length of one)

  - REGEXP
    - ^ beginning of a string
    - | or, e.g., ^j|Kayle’
    - [], “[gim]e”

    - -- ^ beginning
    - -- $ end
    - -- | logical or
    - -- [abcd]
    - -- [a-f]

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name REGEXP '[pl]e';
```

```
SELECT *
FROM Teacher
WHERE full_name REGEXP '[pj]e';
```

String Operation

- LIKE
  - String matching
  - used in WHERE clause
  - String-matching Operators
    - '%' – matching any substrings, including the empty ones
    - '_' – matching any character (length of one)

  - REGEXP
    - ^ beginning of a string
    - | or, e.g., ^j|Kayle'
    - [], "[gim]e"

    - -- ^ beginning
    - -- $ end
    - -- | logical or
    - -- [abcd]
    - -- [a-f]

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name REGEXP '[pl]e';
```

tid	full_name	age	nationality
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE full_name REGEXP '[pj]e';
```

tid	full_name	age	nationality
22	Jens Jonathon	31	Sweden
44	Kayle Persson	33	UK

String Operation

- String operation standards
- Quotes in different database softwares

	Case Sensitive?	**Types of string Quotes**
SQL-92	Sensitive	Single quotes ‘’ only
MySQL	Insensitive	Single and double quotes
SQLite	Sensitive	Single and double quotes
Postgres	Sensitive	Single quotes
Oracle	Sensitive	Single quotes

SQL-92

```
SELECT *
FROM Teacher
WHERE UPPER(full_name) = UPPER(‘jens Jonathon’);
```

MySQL

```
SELECT *
FROM Teacher
WHERE UPPER(full_name) = "jEnS JoNAthoN";
```

tid	full_name	age	nationality
22	Jens Jonathon	31	Sweden

String Operation

- String Functions
  - CHAR_LENGTH()
    - Calculate the length of a given string (spaces excluded)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT CHAR_LENGTH(t.full_name)
FROM Teacher AS t
WHERE UPPER(full_name) = "jEnS JoNAthoN";
```

LENGTH(t.full_nam...
13

String Operation

- String Functions
  - CHAR_LENGTH()
    - Calculate the length of a given string (spaces excluded)
  - CONCAT()
    - concatenate or merge two or more strings or words

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT CONCAT(t.full_name, ' ', t.age, " ", t.nationality)
FROM Teacher AS t
WHERE UPPER(full_name) = "jEnS JoNAthoN";
```

CONCAT(t.full_name, ' ', t.age, " ", t.nation...
Jens Jonathon 31 Sweden

String Operation

- String Functions
  - CHAR_LENGTH()
    - Calculate the length of a given string (spaces excluded)
  - CONCAT()
    - concatenate or merge two or more strings or words
  - UCASE() or UPPER(), LCASE() or LOWER()
    - Given the string in upper ot lower case

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT UPPER(t.full_name)
FROM Teacher AS t
WHERE UPPER(full_name) = "jEnS JoNAthoN";
```

UPPER(t.full_name)
JENS JONATHON

String Operation

- String Functions
  - CHAR_LENGTH()
    - Calculate the length of a given string (spaces excluded)
  - CONCAT()
    - concatenate or merge two or more strings or words
  - UCASE() or UPPER(), LCASE() or LOWER()
    - Given the string in upper ot lower case
  - FIND_IN_SET()
    - find the given value in the given set of values.

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT FIND_IN_SET('Stefan Miller', 'John Smith,
Jens Jonathon,Stefan Miller,Kayle Persson') as offset;
```

offset
3

String Operation

- String Functions
  - CHAR_LENGTH()
    - Calculate the length of a given string (spaces excluded)
  - CONCAT()
    - concatenate or merge two or more strings or words
  - UCASE() or UPPER(), LCASE() or LOWER()
    - Given the string in upper ot lower case
  - FIND_IN_SET()
    - find the given value in the given set of values.
  - SUBSTRING()
    - Acquire a part of the string, given start and end index

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT SUBSTRING(t.full_name, 1, 8)
FROM Teacher AS t;
```

SUBSTRING(t.full_name, 1,...
John Smi
Jens Jon
Stefan M
Kayle Pe

Datetime Operation

- Query date/time
- Manipulate and modify date/time

```
SELECT NOW();
SELECT current_timestamp();
SELECT current_timestamp;
```

current_timestamp
2023-02-06 05:11:34

- Current time
  - Now()
  - current_timestamp()
  - And also a keyword ‘current_timestamp’
- Date of a specific day
  - DATE()
- Subtraction of dates

Datetime Operation

- Query date/time
- Manipulate and modify date/time

```
SELECT NOW();
SELECT current_timestamp();
SELECT current_timestamp;
```

current_timestamp
2023-02-06 05:11:34

- Current time
  - Now()
  - current_timestamp()
  - And also a keyword ‘current_timestamp’

```
SELECT DATE(‘2022-1-1');
```

DATE('2022-1-...
2022-01-01

- Date of a specific day
  - DATE()

```
SELECT EXTRACT(DAY
FROM DATE('2020-10-02'));
```

EXTRACT(DAY FROM DATE('2022-1-1...
1

- Subtraction of dates

Datetime Operation

- Query date/time
- Manipulate and modify date/time

```
SELECT NOW();
SELECT current_timestamp();
SELECT current_timestamp;
```

current_timestamp
2023-02-06 05:11:34

- Current time
  - Now()
  - current_timestamp()
  - And also a keyword ‘current_timestamp’

```
SELECT DATE(‘2022-1-1');
```

DATE('2022-1-...
2022-01-01

```
SELECT EXTRACT(DAY
FROM DATE('2020-10-02'));
```

EXTRACT(DAY FROM DATE('2022-1-1...
1

- Date of a specific day
  - DATE()

```
SELECT DATE('2022-2-1') - DATE('2022-1-1');
```

DATE('2022-2-1') - DATE('2022-1-1')
100

- Subtraction of dates

Datetime Operation

- Query date/time
- Manipulate and modify date/time

```
SELECT NOW();
SELECT current_timestamp();
SELECT current_timestamp;
```

current_timestamp
2023-02-06 05:11:34

- Current time
  - Now()
  - current_timestamp()
  - And also a keyword ‘current_timestamp’

```
SELECT DATE(‘2022-1-1');
```

DATE('2022-1-...
2022-01-01

```
SELECT EXTRACT(DAY
FROM DATE('2020-10-02'));
```

EXTRACT(DAY FROM DATE('2022-1-1...
1

- Date of a specific day
  - DATE()

```
SELECT DATE('2022-2-1') - DATE('2022-1-1');
```

DATE('2022-2-1') - DATE('2022-1-1')
100

- Subtraction of dates

```
SELECT DATEDIFF(DATE('2022-2-1'), DATE('2022-1-1')) AS days;
```

days
31

Output Rediection

- Create and store query results in another table
  - Table must not ready exist
  - Number of columns has to be the same

```
CREATE TABLE CIDs (SELECT
DISTINCT cid From Course);
```

cid
1
2
3
4

- Insert tuples from query into another table
  - Inner SELECT must generate the same columns as the target table

```
INSERT INTO CIDs (SELECT
DISTINCT cid From Course);
```

cid
1
2
3
4
1
2
3
4

Output Control

- ORDER BY <col> [ASC|DESC]

```
SELECT *
FROM Course
ORDER BY credits;
```

cid	course_name	course_code	credits
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4

- LIMIT <#rows> OFFSET <#rows>

```
SELECT *
FROM Course
ORDER BY credits
LIMIT 2;
```

cid	course_name	course_code	credits
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3

```
SELECT *
FROM Course
ORDER BY credits
LIMIT 2 OFFSET 1;
```

cid	course_name	course_code	credits
4	Database	CS1310	3
1	Intro to Computer Science	CS1310	4

Nested Queries

- Queries containing other queries
  - Inner queries can appear in query
- Construction
- Difficult to optimize

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

Outer query

```
SELECT *
FROM Course
WHERE cid IN (SELECT cid
              FROM Teaches AS tt, Teacher AS t
              WHERE t.full_name LIKE '%th%' AND t.tid = tt.tid);
```

Inner query

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3

Nested Queries

- Queries containing other queries
  - Inner queries can appear in query
- Construction
- Difficult to optimize

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

Outer query

```
SELECT *
FROM Course
WHERE cid IN (SELECT cid
              FROM Teaches AS tt, Teacher AS t
              WHERE t.full_name LIKE '%th%' AND t.tid = tt.tid);
```

Inner query

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3

Nested Queries

- Queries containing other queries
  - Inner queries can appear in query
- Construction
- Difficult to optimize

```
SELECT *
FROM Course
WHERE cid IN (SELECT cid
              FROM Teaches AS tt, Teacher AS t
              WHERE t.full_name LIKE '%th%' AND t.tid = tt.tid);
```

```
SELECT *
FROM Course
WHERE cid = ANY(SELECT cid
                FROM Teaches AS tt, Teacher AS t
                WHERE t.full_name LIKE '%th%' AND t.tid = tt.tid);
```

Nested Queries

- Queries containing other queries
  - Inner queries can appear in query
- Construction
- Difficult to optimize

```
SELECT *
FROM Course
WHERE cid IN (SELECT cid
              FROM Teaches AS tt, Teacher AS t
              WHERE t.full_name LIKE '%th%' AND t.tid = tt.tid);
```

- ALL
  - must satisfy expression for all rows in the sub-query
- ANY
  - must satisfy expression for at least one row in the sub-query
- IN is eqvuivalent to '=ANY()'
- EXISTS
  - at least one row is returned

```
SELECT *
FROM Course
WHERE cid = ANY(SELECT cid
                FROM Teaches AS tt, Teacher AS t
                WHERE t.full_name LIKE '%th%' AND t.tid = tt.tid);
```

Nested Queries

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

- Find the Teacher that teaches at least two courses

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

Nested Queries

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

- Find the Teacher that teaches at least two courses

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE EXISTS(SELECT 1);
```

Nested Queries

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

- Find the Teacher that teaches at least two courses

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT *
FROM Teacher
WHERE EXISTS(SELECT 1);
```

```
SELECT *
FROM Teacher as t
WHERE (SELECT COUNT(*)
        FROM Teaches AS tt
        WHERE tt.tid = t.tid) >= 2;
```

tid	full_name	age	nationality
11	John Smith	42	America

Nested Queries

- Find all courses that has not assigned any teacher to it

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3
5	Physics	PHY100	6

Nested Queries

- Find all courses that has not assigned any teacher to it

```
SELECT * FROM Course
WHERE NOT EXISTS(SELECT *
                 FROM Teaches
                 WHERE Course.cid = Teaches.cid);
```

cid	course_name	course_code	credits
5	Physics	PHY100	6

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3
5	Physics	PHY100	6

Example Database

Course(cid, Course_name, Course_code, Credit_hours)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3
5	Physics	PHY100	6

Teacher(tid, full_name, age, nationality)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

Teaches(tid, cid, hours)

tid	cid	hours
11	1	80
11	2	100
22	4	50
33	4	50
44	3	100

Recall Relational Algebra - Join

- Syntax
  - (A ⋈ B)

- Generate a relation that contains all tuples with a common value(s) of one (or more) attrubute(s)

A(Tid, Cid)

Tid	Cid
a1	10
a2	11

B(Tid, Cid)

Tid	Cid
a2	11
a3	13

(A ⋈ B)

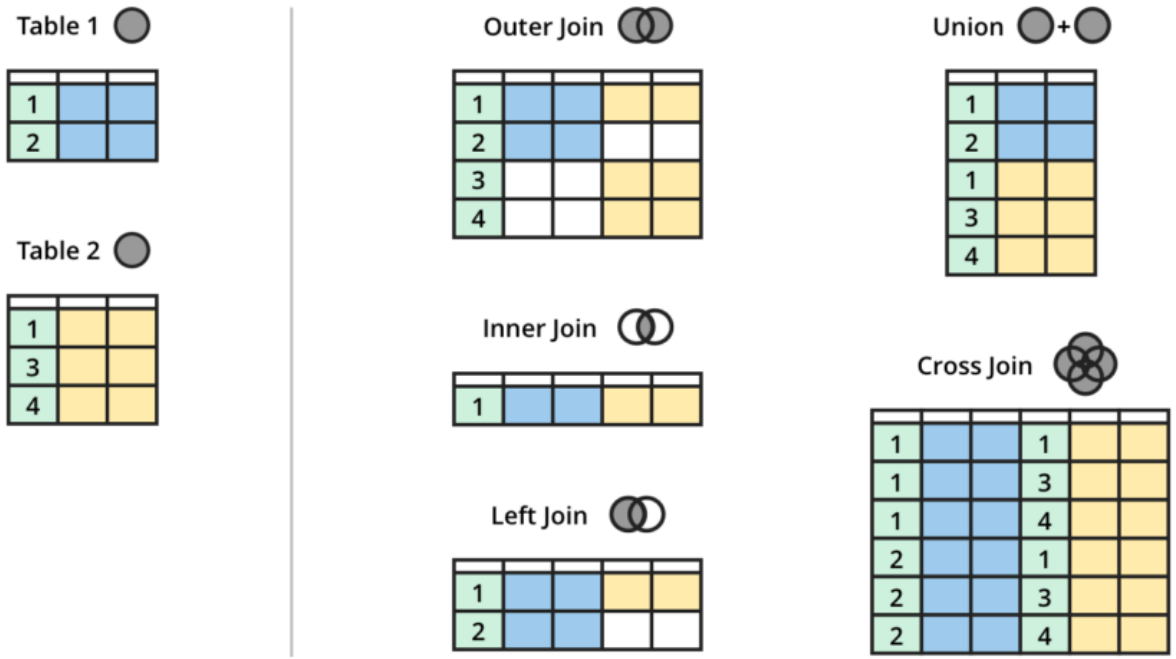
Tid	Cid
a2	11

SELECT * FROM A NATURAL JOIN B;

Operations on multiple tables

- Generate a relation that contains all tuples that are a combination of two tuples
- Merge/combine tables in different ways
  - JOIN clause is used to combine rows from two or more tables, based on a related column between them

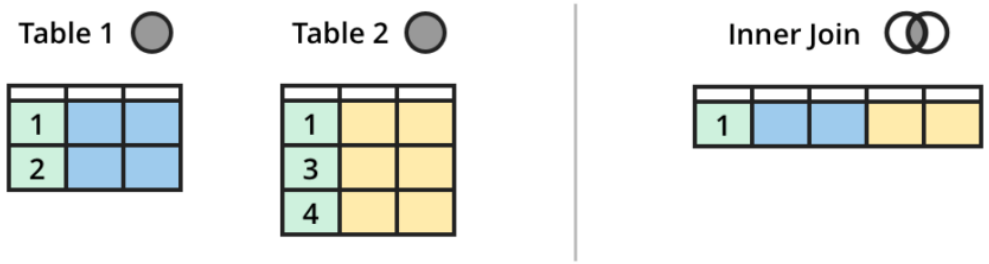
- Inner join - ⋈
- Outer join
  - Full join
  - Left join
  - Right join
- CROSS join
- Union



Source: https://dataschool.com/how-to-teach-people-sql/sql-join-types-explained-visually/

Operations on multiple tables

- Inner Join
  - "An inner join combines the columns on a common dimension (the first N columns) when possible, and only includes data for the columns that share the same values in the common N column(s)."




```
SELECT <col>
FROM Table1 as T1
JOIN Table2 as T2
ON T1.cid = T2.cid
```

Course(cid, Course_name, Course_code, Credit_hours)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3
5	Physics	PHY100	6

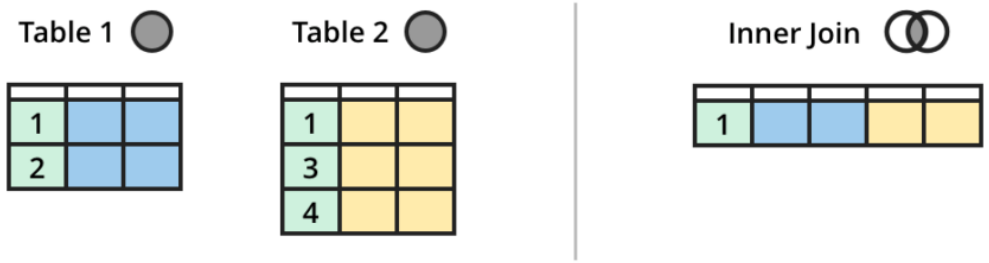
Teaches(tid, cid, hours)

tid	cid	hours
11	1	80
11	2	100
22	4	50
33	4	50
44	3	100

Operations on multiple tables

```
SELECT * FROM Courses;
SELECT * FROM Teaches;
```

- Inner Join




```
SELECT <col>
FROM Table1 as T1
JOIN Table2 as T2
ON T1.cid = T2.cid
```

Course(cid, Course_name, Course_code, Credit_hours)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

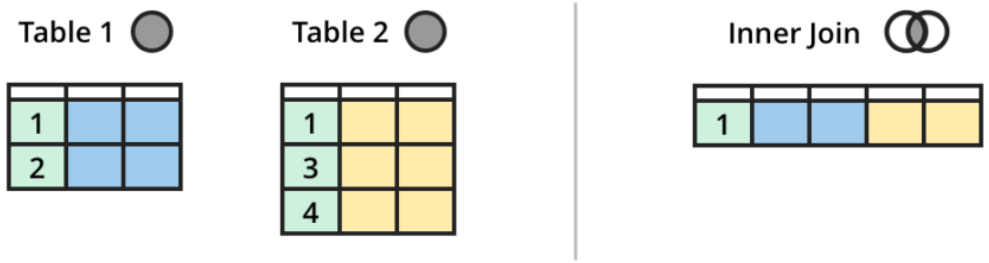
Teaches(tid, cid, hours)

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

Operations on multiple tables

```
SELECT * FROM Courses;
SELECT * FROM Teaches;
```

- Inner Join




```
SELECT *
FROM Course as c
JOIN Teaches as t
ON c.cid = t.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80

Course(cid, Course_name, Course_code, Credit_hours)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

Teaches(tid, cid, hours)

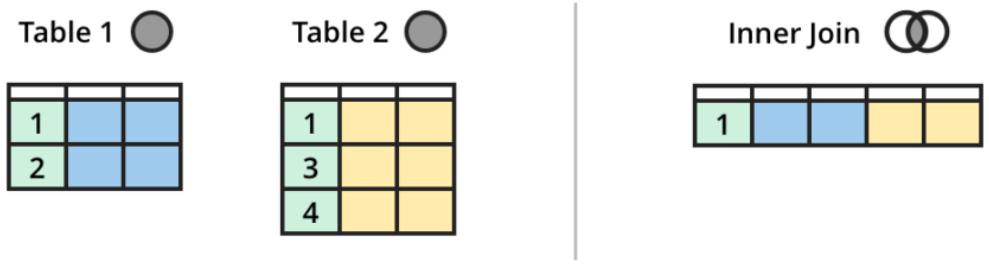
tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

Cid = 5 does not exist in Table *Teaches*

Duplicated due to one-to-many or many-to-many relationship, joined on c.cid = t.cid

Operations on multiple tables

- Inner Join




```
SELECT *
FROM Course as c
JOIN Teaches as t
ON c.cid = c.cid;
```

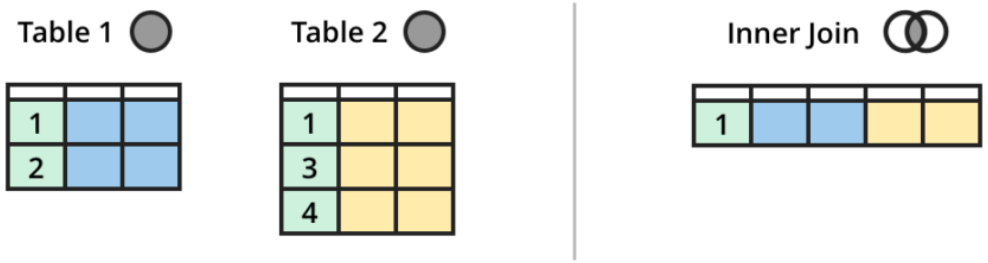
```
SELECT *
FROM Course as c, Teaches as t
WHERE c.cid = c.cid;
```

```
SELECT *
FROM Course as c
INNER JOIN Teaches as t
ON c.cid = t.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80

Operations on multiple tables

- Inner Join
  - "An inner join combines the columns on a common dimension (the first N columns) when possible, and only includes data for the columns that share the same values in the common N column(s)."



Course(cid, Course_name, Course_code, Credit_hours)

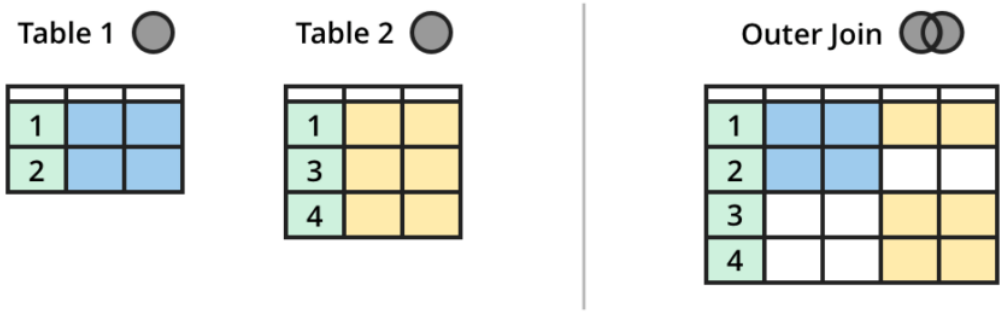
cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structures	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS3380	3
5	Physics	PHY100	6

Teaches(tid, cid, hours)

tid	cid	hours
11	1	80
11	2	100
22	4	50
33	4	50
44	3	100

Operations on multiple tables

- Outer join
  - "outer join combines the columns from all tables on one or more common dimension when possible, and includes all data from all tables."




```
SELECT *
FROM Table1 as t1
FULL OUTER JOIN Table2 as t2
ON t1.cid = t2.cid;
```

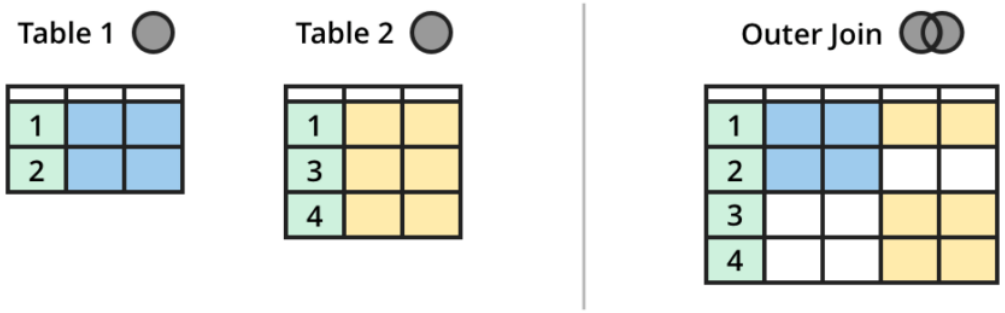
FULL OUTER JOIN not available in MySQL…

```
SELECT * FROM Table1 as t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM Table2 as t2
RIGHT JOIN t2 ON t1.id = t2.id
```

Emulating FULL OUTER JOIN in MySQL…

Operations on multiple tables

- Outer join
  - "outer join combines the columns from all tables on one or more common dimension when possible, and includes all data from all tables."




```
SELECT * FROM Table1 as t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM Table2 as t2
RIGHT JOIN t2 ON t1.id = t2.id
```

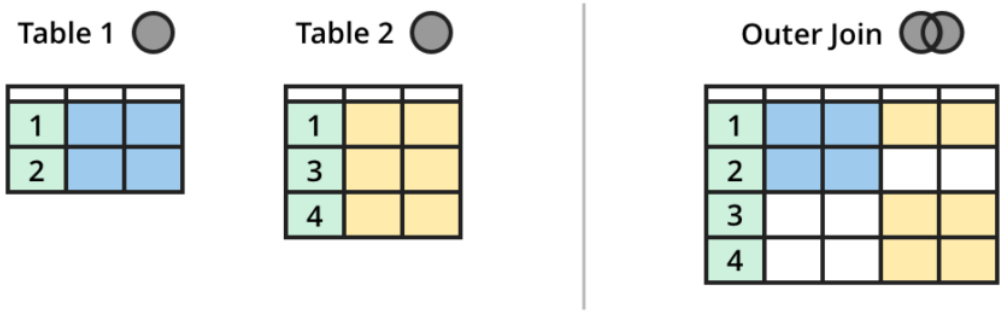
Emulating FULL OUTER JOIN in MySQL…

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80
5	Physics	PHY100	6	NULL	NULL	NULL

```
SELECT * FROM Course
LEFT JOIN Teaches ON Course.cid = Teaches.cid
UNION
SELECT * FROM Course
RIGHT JOIN Teaches ON Course.cid = Teaches.cid;
```

Operations on multiple tables

- Outer join
  - "outer join combines the columns from all tables on one or more common dimension when possible, and includes all data from all tables."




```
SELECT * FROM Table1 as t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM Table2 as t2
RIGHT JOIN t2 ON t1.id = t2.id
```

Emulating FULL OUTER JOIN in MySQL…

INNER JOIN

```
SELECT *
FROM Course as c
JOIN Teaches as t
ON c.cid = t.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80

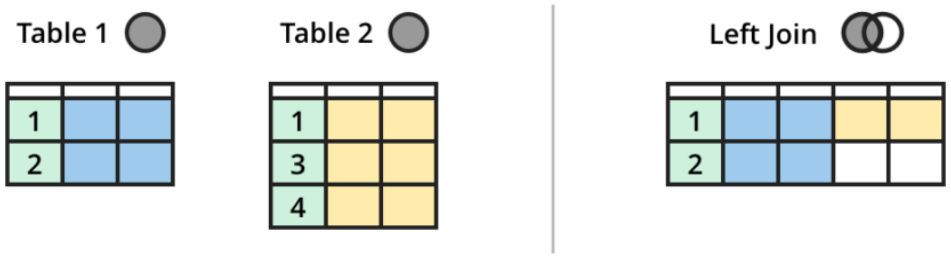
OUTER JOIN

```
SELECT * FROM Course
LEFT JOIN Teaches ON Course.cid = Teaches.cid
UNION
SELECT * FROM Course
RIGHT JOIN Teaches ON Course.cid = Teaches.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80
5	Physics	PHY100	6	NULL	NULL	NULL

Operations on multiple tables

- LEFT JOIN
  - "A left join combines the columns on a common dimension (the first N columns) when possible, returning all rows from the first table with the matching rows in the consecutive tables."




```
SELECT <col>
FROM Table1 as T1
LEFT JOIN Table2 as T2
ON T1.cid = T2.cid
```

Operations on multiple tables

```
SELECT * FROM Courses;
SELECT * FROM Teaches;
```

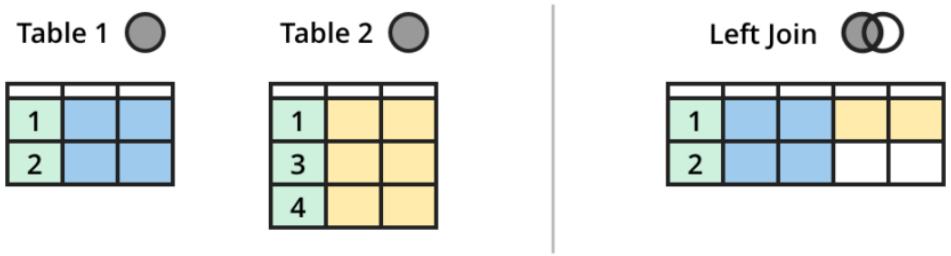
- LEFT JOIN
  - "A left join combines the columns on a common dimension (the first N columns) when possible, returning all rows from the first table with the matching rows in the consecutive tables."

Course(cid, Course_name, Course_code, Credit_hours)

cid	course_name	course_code	credits
1	Intro to Computer Science	CS1310	4
2	Data Structure	CS3320	4
3	Discrete Mathematics	MATH2410	3
4	Database	CS1310	3
5	Physics	PHY100	6

Teaches(tid, cid, hours)

tid	cid	hours
11	1	50
11	2	100
22	3	80
33	4	80
44	3	100

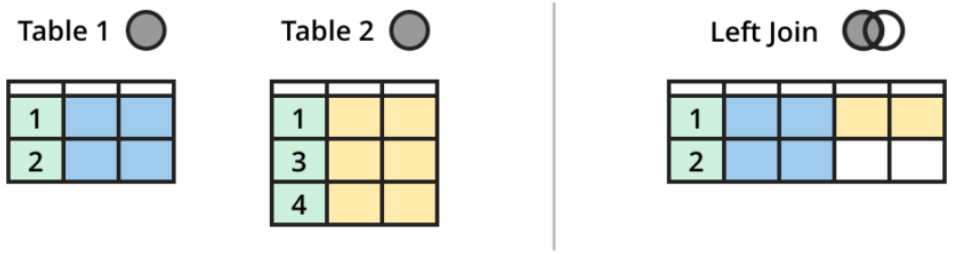



```
SELECT *
FROM Course as c
LEFT JOIN Teaches as t
ON c.cid = t.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80
5	Physics	PHY100	6	NULL	NULL	NULL

Operations on multiple tables

- LEFT JOIN
  - "A left join combines the columns on a common dimension (the first N columns) when possible, returning all rows from the first table with the matching rows in the consecutive tables."



LEFT JOIN

```
SELECT *
FROM Course as c
LEFT JOIN Teaches as t
ON c.cid = t.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
3	Discrete Mathematics	MATH2410	3	44	3	100
4	Database	CS1310	3	33	4	80
5	Physics	PHY100	6	NULL	NULL	NULL

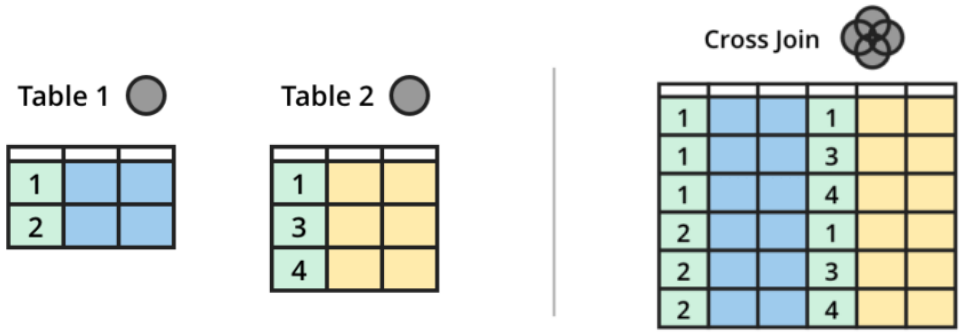
RIGHT JOIN

```
SELECT *
FROM Course as c
RIGHT JOIN Teaches as t
ON c.cid = t.cid;
```

cid	course_name	course_code	credits	tid	cid	hours
1	Intro to Computer Science	CS1310	4	11	1	50
2	Data Structure	CS3320	4	11	2	100
3	Discrete Mathematics	MATH2410	3	22	3	80
4	Database	CS1310	3	33	4	80
3	Discrete Mathematics	MATH2410	3	44	3	100

Operations on multiple tables

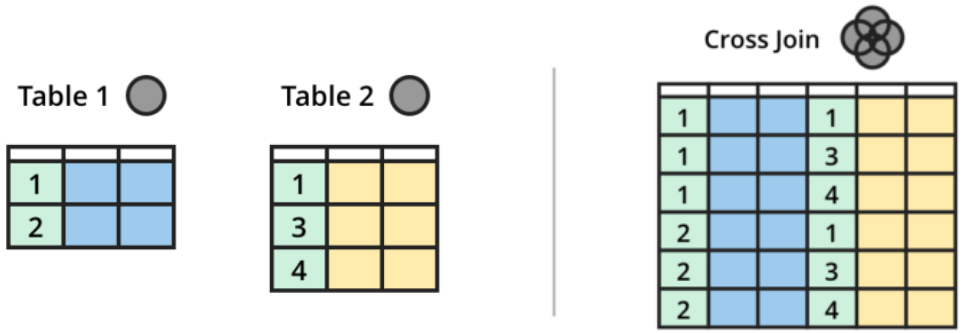
- CROSS JOIN
  - Recall Cartesian product




```
SELECT <col>
FROM Table1 as T1
CROSS JOIN Table2 as T2
```

Operations on multiple tables

- CROSS JOIN
  - Recall Cartesian product

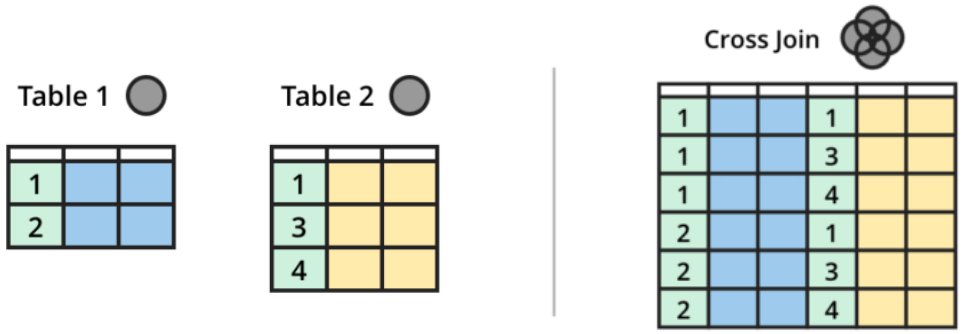



```
SELECT *
FROM Course as c
CROSS JOIN Teaches as t;
```

cid	course_name	course_code	credits	tid	cid	hours
5	Physics	PHY100	6	11	1	50
4	Database	CS1310	3	11	1	50
3	Discrete Mathematics	MATH2410	3	11	1	50
2	Data Structure	CS3320	4	11	1	50
1	Intro to Computer Science	CS1310	4	11	1	50
5	Physics	PHY100	6	11	2	100
4	Database	CS1310	3	11	2	100
3	Discrete Mathematics	MATH2410	3	11	2	100
2	Data Structure	CS3320	4	11	2	100
1	Intro to Computer Science	CS1310	4	11	2	100
5	Physics	PHY100	6	22	3	80
4	Database	CS1310	3	22	3	80
3	Discrete Mathematics	MATH2410	3	22	3	80
2	Data Structure	CS3320	4	22	3	80
1	Intro to Computer Science	CS1310	4	22	3	80
5	Physics	PHY100	6	33	4	80
4	Database	CS1310	3	33	4	80
3	Discrete Mathematics	MATH2410	3	33	4	80
2	Data Structure	CS3320	4	33	4	80
1	Intro to Computer Science	CS1310	4	33	4	80
5	Physics	PHY100	6	44	3	100
4	Database	CS1310	3	44	3	100
3	Discrete Mathematics	MATH2410	3	44	3	100
2	Data Structure	CS3320	4	44	3	100
1	Intro to Computer Science	CS1310	4	44	3	100

Operations on multiple tables

- CROSS JOIN
  - Recall Cartesian product



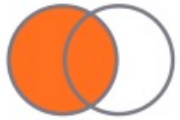

```
SELECT COUNT(*)
FROM Course as c
CROSS JOIN Teaches as t;
```

COUNT(*)
25

cid	course_name	course_code	credits	tid	cid	hours
5	Physics	PHY100	6	11	1	50
4	Database	CS1310	3	11	1	50
3	Discrete Mathematics	MATH2410	3	11	1	50
2	Data Structure	CS3320	4	11	1	50
1	Intro to Computer Science	CS1310	4	11	1	50
5	Physics	PHY100	6	11	2	100
4	Database	CS1310	3	11	2	100
3	Discrete Mathematics	MATH2410	3	11	2	100
2	Data Structure	CS3320	4	11	2	100
1	Intro to Computer Science	CS1310	4	11	2	100
5	Physics	PHY100	6	22	3	80
4	Database	CS1310	3	22	3	80
3	Discrete Mathematics	MATH2410	3	22	3	80
2	Data Structure	CS3320	4	22	3	80
1	Intro to Computer Science	CS1310	4	22	3	80
5	Physics	PHY100	6	33	4	80
4	Database	CS1310	3	33	4	80
3	Discrete Mathematics	MATH2410	3	33	4	80
2	Data Structure	CS3320	4	33	4	80
1	Intro to Computer Science	CS1310	4	33	4	80
5	Physics	PHY100	6	44	3	100
4	Database	CS1310	3	44	3	100
3	Discrete Mathematics	MATH2410	3	44	3	100
2	Data Structure	CS3320	4	44	3	100
1	Intro to Computer Science	CS1310	4	44	3	100

Operations on multiple tables

LEFT JOIN



Everything on the left
+
anything on the right that matches

```
SELECT *
FROM TABLE_1
LEFT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

ANTI LEFT JOIN

Everything on the left
that is NOT on the right

```
SELECT *
FROM TABLE_1
LEFT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
WHERE TABLE_2.KEY IS NULL
```

RIGHT JOIN

Everything on the right
+
anything on the left that matches

```
SELECT *
FROM TABLE_1
RIGHT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

ANTI RIGHT JOIN

Everything on the right
that is NOT on the left

```
SELECT *
FROM TABLE_1
RIGHT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
WHERE TABLE_1.KEY IS NULL
```

OUTER JOIN

Everything on the right
+
Everything on the left

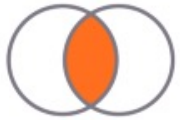
```
SELECT *
FROM TABLE_1
OUTER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

ANTI OUTER JOIN

Everything on the left and right
that is unique to each side

```
SELECT *
FROM TABLE_1
OUTER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
WHERE TABLE_1.KEY IS NULL
OR TABLE_2.KEY IS NULL
```

INNER JOIN



Only the things that match on the
left AND the right

```
SELECT *
FROM TABLE_1
INNER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

CROSS JOIN

All combination of rows from the
right and the left (cartesean
product)

```
SELECT *
FROM TABLE_1
CROSS JOIN TABLE_2
```

Source: https://towardsdatascience.com/take-your-sql-from-good-to-great-part-3-687d797d1ede

Views

- Create view statement
  - A view is a virtual table based on the result-set of an SQL statement

Syntax

```
CREATE VIEW <view_name> AS
SELECT col1, col2, ...
FROM <table_name>
WHERE <condition>;
```

Views

- Create view statement
  - A view is a virtual table based on the result-set of an SQL statement

```
CREATE VIEW CprodCourseTeacher AS
SELECT * FROM Course NATURAL JOIN Teaches;
```

cid	course_name	course_code	credits	tid	hours
1	Intro to Computer Science	CS1310	4	11	50
2	Data Structure	CS3320	4	11	100
3	Discrete Mathematics	MATH2410	3	22	80
4	Database	CS1310	3	33	80
3	Discrete Mathematics	MATH2410	3	44	100