

CSc 484

Database Management Systems

Ken Gamradt

Spring 2024

Introduction to SQL (III)

SQL DML Statements

- **SELECT** – to query data in the database
- **INSERT** – to insert data into a table
- **UPDATE** – to update data in a table
- **DELETE** – to delete data from a table

SELECT Statement

- **Purpose:** retrieve and display data from one or more database tables
- Able to perform the equivalent of relational algebra's
 - **Selection**
 - **Projection**
 - **Join**operations in a single statement
- Most frequently used SQL command

SELECT Statement

(general form)

// Specify which columns appear in the output

// Column name or expression

select [**distinct** | **all**] { * | [columnExpression [**as** newName]] [,...] }

// Specify the table(s) to be used

from TableName [alias] [,...]

// Filter rows

[**where** condition]

// Forms groups of rows with common column value

// Filter groups subject to condition

[**group by** columnList] [**having** condition]

// Specify the output order

[**order by** columnList]

Symbols in the general form

- vertical bar (|) indicates a **choice** among alternatives
 - E.g., $a \mid b \mid c$
- curly braces indicate a **required element**
 - E.g., $\{a\}$
- square brackets indicate an **optional element**
 - E.g., $[a]$
- ellipsis (...) is used to indicate **optional repetition** of an item zero or more times

University schema

classroom (building, room_number, capacity)

department (dept_name, building, budget)

course (course_id, title, dept_name, credits)

instructor (ID, name, dept_name, salary)

section (course_id, sec_id, semester, year, building, room_number, time_slot_id)

teaches (ID, course_id, sec_id, semester, year)

student (ID, name, dept_name, tot_cred)

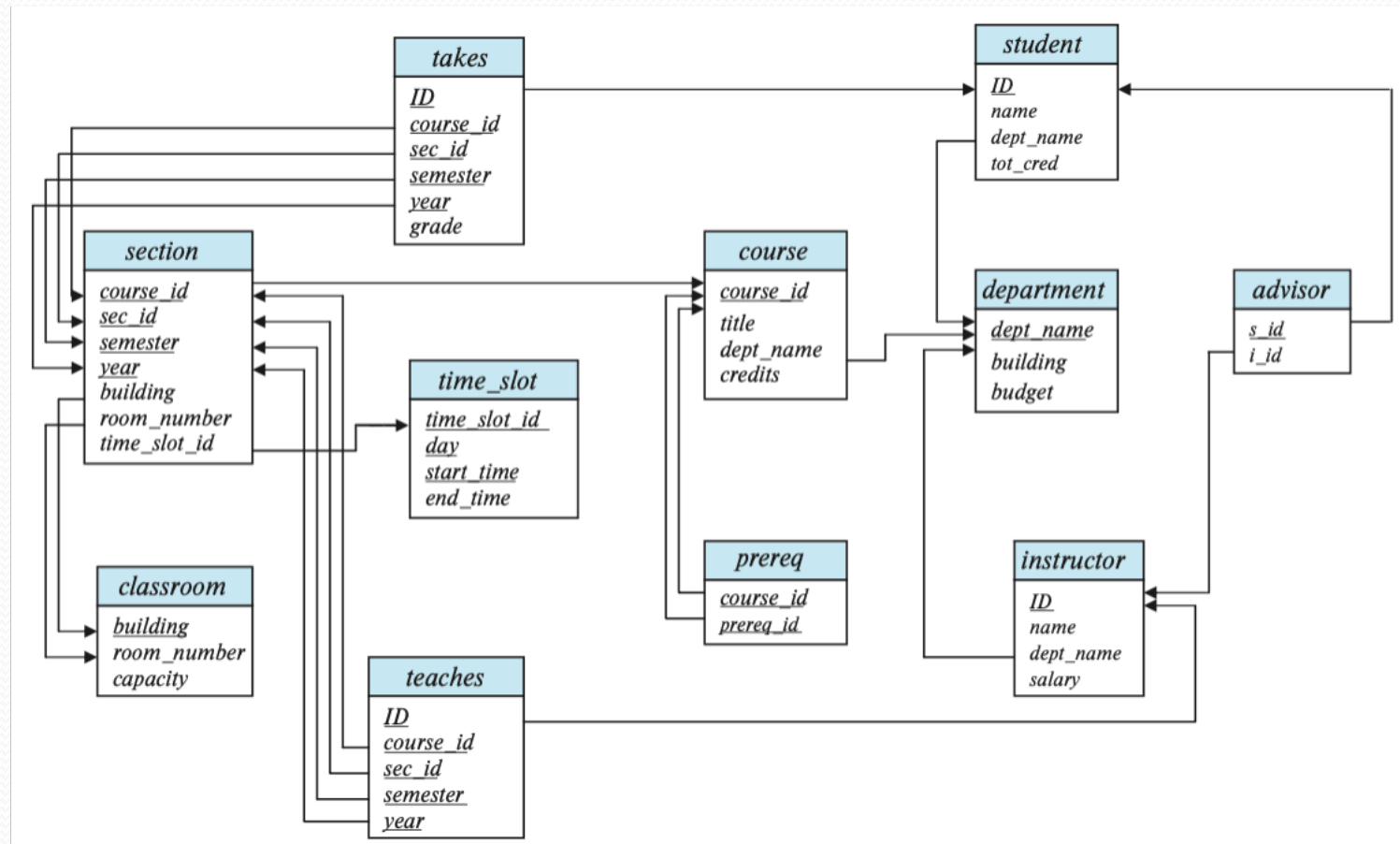
takes (ID, course_id, sec_id, semester, year, grade)

advisor (s_ID, i_ID)

time_slot (time_slot_id, day, start_time, end_time)

prereq (course_id, prereq_id)

University schema



SELECT Statement

(retrieve all columns, all rows)

- Example: list all details of all instructors

// List of all column names

```
select ID, name, dept_name, salary  
      from instructor;
```

// A quick way to express “all columns”

// in SQL using an asterisk (*)

```
select *  
      from instructor;
```

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

SELECT Statement

(retrieve specific columns, all rows)

- Example: list for all instructors showing only: ID, name, salary

***select** ID, name, salary
from instructor;*

id	name	salary
abc Filter...	abc Filter...	abc Filter...
10101	Srinivasan	65000.00
12121	Wu	90000.00
15151	Mozart	40000.00
22222	Einstein	95000.00
32343	El Said	60000.00
33456	Gold	87000.00
45565	Katz	75000.00
58583	Califieri	62000.00
76543	Singh	80000.00
76766	Crick	72000.00
83821	Brandt	92000.00
98345	Kim	80000.00


SELECT Statement

(use of DISTINCT)

- DISTINCT: to eliminate the duplicates of the result relation
 - E.g., list the department names that all instructors work in
 - Select doesn't eliminate duplicates when it projects over one or more columns
 - Use distinct to eliminate duplicates

select distinct dept_name
from instructor;

select dept_name
from instructor;



	dept_name
1	Biology
2	Comp. Sci.
3	Elec. Eng.
4	Finance
5	History
6	Music
7	Physics

	dept_name
1	Comp. Sci.
2	Finance
3	Music
4	Physics
5	History
6	Physics
7	Comp. Sci.
8	History
9	Finance
10	Biology
11	Comp. Sci.
12	Elec. Eng.

Fact: in practice, database systems don't check the duplicates of tuples in a relation without a primary key

SELECT Statement (use of DISTINCT)

- List all instructor's id along with the course id they have taught

*select distinct ID, course_id
from teaches;*

ID	course_id
10101	CS-101
10101	CS-315
10101	CS-347
12121	FIN-201
15151	MU-199
22222	PHY-101
32343	HIS-351
45565	CS-101
45565	CS-319
76766	BIO-101
76766	BIO-301
83821	CS-190
83821	CS-319
98345	EE-181

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

SELECT Statement

(use of ALL)

- **ALL**: specifies explicitly that duplicates are not removed
 - In select statement, duplicates remained by default

```
select all ID, course_id  
from teaches;
```

```
select ID, course_id  
from teaches;
```

id	course_id
abc Filter...	abc Filter...
10101	CS-101
10101	CS-315
10101	CS-347
12121	FIN-201
15151	MU-199
22222	PHY-101
32343	HIS-351
45565	CS-101
45565	CS-319
76766	BIO-101
76766	BIO-301
83821	CS-190
83821	CS-190
83821	CS-319
98345	EE-181

SELECT Statement

(row selection: WHERE clause)

- **WHERE**: a predicate (or search condition) involving attributes of the relation in the from clause, to specify the rows to be retrieved
- Five basic search conditions
 - **Comparison**
 - Compare the value of one expression to the value of another expression
 - **Range**
 - Test whether the value of an expression falls within a specified range of values
 - **Set membership**
 - Test whether the value of an expression equals one of a set of values
 - **Pattern match**
 - Test whether a string matches a specified pattern
 - **Null**
 - Test whether a column has a null (unknown) value

SELECT Statement

(WHERE – comparison search condition)

- Example: list all instructors with a salary greater than \$60,000

select ID, name, dept_name, salary
from instructor
where salary > 60000;

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
22222	Einstein	Physics	95000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

SELECT Statement

(WHERE – comparison search condition)

- In SQL, the following simple comparison operators are available:

Operator	Description
=	equals
<>	not equal to (ISO standard)
!=	not equal to (allowed in some dialects)
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

SELECT Statement

(WHERE – comparison search condition)

- Use logical operators to generate more complex predicates
 - AND, OR, NOT
 - Parentheses can be used to show the order of evaluation
- The rules for evaluating a conditional expression
 - An expression is evaluated *from left to right*
 - Subexpressions in *brackets are evaluated first*
 - *Evaluate NOT, then evaluate AND, then evaluate OR*

SELECT Statement

(WHERE – comparison search condition)

- Example: list all instructors from physics and music dept

*select **

from instructor

where dept_name = 'Physics' or dept_name = 'Music';

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
33456	Gold	Physics	87000.00

SELECT Statement

(WHERE – range search condition)

- Example: list all instructors with a salary between 80,000 and 100,000

```
select *  
from instructor  
// BETWEEN test includes the endpoints  
where salary between 80000 and 100000;
```

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
12121	Wu	Finance	90000.00
22222	Einstein	Physics	95000.00
33456	Gold	Physics	87000.00
76543	Singh	Finance	80000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

SELECT Statement

(WHERE – range search condition)

- A negated version of the range test: NOT BETWEEN
 - E.g., list all instructors with a salary out of the range of 80,000 and 100,000

*select **
from instructor
where salary not between 80000 and 100000;

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
10101	Srinivasan	Comp. Sci.	65000.00
15151	Mozart	Music	40000.00
32343	El Said	History	60000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76766	Crick	Biology	72000.00

SELECT Statement

(WHERE – range search condition)

- BETWEEN test can be expressed equally well using two comparison tests

```
select *  
  from instructor  
  where salary >= 80000 and salary <= 100000;
```

```
select *  
  from instructor  
  where salary < 80000 or salary > 100000;
```


SELECT Statement

(WHERE – set membership search condition)

- **IN**: tests whether a data value matches one of a list of values
- E.g., list all instructors from Physics and Music dept

```
select *  
from instructor  
where dept_name in ('Physics', 'Music');
```

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
33456	Gold	Physics	87000.00

SELECT Statement

(WHERE – set membership search condition)

- **NOT IN**: be used to check for the data values that do not from the list
- E.g., list all instructors except those from Physics and Music dept

*select **

from instructor

where dept_name not in ('Physics', 'Music');

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
32343	El Said	History	60000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

SELECT Statement

(WHERE – set membership search condition)

- Set membership expression test can be expressed equally well using a comparison test

```
select *  
  from instructor  
  where dept_name = 'Physics' or dept_name = 'Music';
```

```
select *  
  from instructor  
  where dept_name <> 'Physics' and dept_name != 'Music';
```

SELECT Statement

(WHERE – pattern match search condition)

- Key words: **LIKE**, **NOT LIKE**
- Special pattern-matching symbols
 - % : represents any sequence of zero or more characters
 - _ : represents any single character
- All other characters in the pattern represent themselves

SELECT Statement

(WHERE – pattern match search condition)

- Pattern matching examples:
 - **LIKE 'Intro%'** : matches any string beginning with 'Intro'
 - **LIKE '%Comp%'** : matches any string containing 'Comp' as a substring
 - 'Intro to Computer Science Lab', 'Computer Science I'
 - **LIKE '___'** : matches any string of exactly three characters
 - **LIKE '___%'** : matches any string of at least three characters
 - **NOT LIKE 'H%'** : matches any string that doesn't start with character 'H'

SELECT Statement

(WHERE – pattern match search condition)

- Example: find the names of all departments whose building name includes the substring 'Wat'

```
select dept_name, building  
from department  
where building like '%Wat%';
```

dept_name	building
abc Filter...	abc Filter...
Biology	Watson
Physics	Watson

SELECT Statement

(WHERE – pattern match search condition)

- **ESCAPE character**: used for patterns to include the special pattern characters (% , _)
- Use a backslash (\) as the escape character
- Examples patterns:
 - **LIKE 'ab\%cd%' ESCAPE '\'**: matches all strings beginning with 'ab%cd'

Tips on strings

- Strings are enclosed in single quotes
- A single quote that is part of a string can be specified by using two single quotes
 - "it's right" can be specified by 'it"s right'

insert into instructor

values ('10001', 'don"t know', 'Music', 100000);

id	name	dept_name	salary
abc Filter...	abc Filter...	abc Filter...	abc Filter...
10001	don't know	Music	100000.00
10101	Srinivasan	Comp. Sci.	65000.00
11111	Hanks	Music	NULL
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00

Tips on strings

- SQL standard specifies that the equality operation on strings is case sensitive
- However, some database systems (such as MySQL, SQL Server) treat strings case-insensitive when matching strings

```
select dept_name  
from department  
where (dept_name = 'biology');
```

	dept_name
1	Biology

SQL Server

	department (1r x 1c)
	dept_name
	Biology

MySQL

Data Output	Explain	Messages
dept_name [PK] character varying (20)		

PostgreSQL

Comments in SQL programming

- `/*` multiple-line comment `*/`
- `--` (two hyphens): for single line comment – to the end of the current line

```
/*  
    Text_of_comment  
*/
```

```
select dept_name      -- department name selected  
...
```

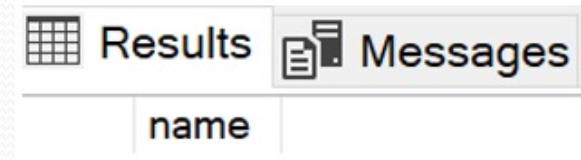
- MySQL single-line comment
 - `#` : for single line comment – to the end of the current line

SELECT Statement

(WHERE – NULL search condition)

- IS NULL / IS NOT NULL
 - E.g., list all the instructors' names whose salary hasn't been recorded

```
select name  
from instructor  
where salary IS NULL;
```



The screenshot shows a database interface with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with a single column labeled 'name'.

name

Acknowledgements

- WIKIPEDIA
 - <https://en.wikipedia.org/wiki/SQL>