CS3220 Web and Internet Programming

More SQL

Chengyu Sun
California State University, Los Angeles

Employees DB

employees

id	first_name	last_name	address	supervisor_id
1	John	Doe	Street #215	null
2	Jane	Doe	Street #711	1

projects

id	name	leader_id
1	Firestone	1
2	Blue	2

project_members

project_id	employee_id
1	1
2	1
2	2

Examples: Single-Table Selection

- 1. List the last names of the employees whose ids are less than 10
  - Remove duplicates
  - Show results in alphabetic order
- 2. Find the id of Jane Doe
- 3. Find the names of the employees who do not have a supervisor
  - Concatenate first name and last name

SQL Literals

- Number: `10, 30.2`
- String: `'CPU', 'John''s Kitchen'`
- Date: `'2007-06-01'`
- Time: `'12:00:00'`
- Boolean: `1, 0, true, false`
- `NULL`

SQL Operators

- Arithmetic
  - +, -, *, /, %
- Comparison
  - <, >, <=, >=,=,<>
  - between
- Logical
  - and, or, not
- String
  - like
- Other
  - is null
  - in
  - distinct
  - order by

LIKE

- Simple pattern matching
  - **%**: any zero or more characters
  - **_**: any single character

Common Functions in Databases

- Numerical functions
- String functions
- Date and time functions
- NULL related functions
- *Aggregation functions*

Most functions are DBMS specific.

Functions in MySQL

- https://dev.mysql.com/doc/refman/8.0/en/functions.html

Examples: Join

- 4. List the employees who work on the project with id=1
- 5. List the employees who work on the project Blue
- 6. Find the name of Jane Doe’s supervisor

Cross Join

- A.K.A. Cartesian Product
- The results are *all possible combinations* of the rows from Table 1 with the rows from Table 2

table1

A	B
a_1	b_1
a_2	b_2

$\times$

table2

C	D
c_1	d_1
c_2	d_2
c_3	d_3

$=$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_1	b_1	c_3	d_3
a_2	b_2	c_1	d_1
a_2	b_2	c_2	d_2
a_2	b_2	c_3	d_3

Equi-Join

- Cross join with additional conditions

$$\texttt{select … from } \underbrace{\texttt{T1, T2}}_{\text{cross join}} \texttt{ where } \underbrace{\texttt{… … …}}_{\text{additional conditions}}$$

Inner Join

- a.k.a Join
- Combine two rows (one from each table) if they meet the join condition
- In other words, the results include the *matching rows* from the two tables

Inner Join Example

table1

A	B
1	10
2	12

table2

C	D
1	23
3	32
4	34

table1 *inner join* table2 on A=C

⇩

A	B	C	D
1	10	1	23

Examples: Outer Join

- 7. Find the employees who are not working on any project

Outer Joins

- Include the results of an Inner Join and the unmatched rows from *one or both join tables*

Left Outer Join

- a.k.a. Left Join

<u>table1</u>

A	B
1	10
2	12

<u>table2</u>

C	D
1	23
3	32
4	34

table1 *left outer join* table2 on A=C

A	B	C	D
1	10	1	23
2	12	null	null

Right Outer Join

- a.k.a. Right Join

<u>table1</u>

A	B
1	10
2	12

<u>table2</u>

C	D
1	23
3	32
4	34

table1 *right outer join* table2 on A=C

A	B	C	D
1	10	1	23
null	null	3	32
null	null	4	34

Full Outer Join

- a.k.a. Full Join

table1

A	B
1	10
2	12

table2

C	D
1	23
3	32
4	34

table1 *full outer join* table2 on A=C

A	B	C	D
1	10	1	23
2	12	null	null
null	null	3	32
null	null	4	34

Example: Aggregation Functions

- 8. Find the number of employees whose last name is Doe

Aggregation Functions

- Operate on multiple rows and return a single result
  - `sum`
  - `avg`
  - `count`
  - `max` and `min`

Be Careful with NULL

inventory

product_id	upc	quantity	price
1	1020301	20	100
2	1342193	null	200
3	null	100	null

max(price)?? min(price)?? avg(price)??

count(upc)?? count(*)??

sum(quantity) ??

Example: Aggregation Queries

- 9. List the number of employees for each project
  - Order the results by the number of employees
- 10. List the number of projects each employee works on

Understand GROUP BY …

- Without aggregation/GROUP BY

select project_id, member_id from project_members;

project_id	**member_id**
1	1
2	1
2	2
3	2

… Understand GROUP BY

- With aggregation/GROUP BY

```
select project_id, count(member_id)
    from project_members group by project_id;
```

Grouping attribute → project_id

Aggregation attribute → member_id

project_id	member_id	
1	1	count=1
2	1	count=2
2	2	
3	2	count=1

How GROUP BY Works

1. Calculate the results *without* aggregation/GROUP BY
2. Divide the result rows into groups that *share the same value in the grouping attribute(s)*
3. Apply the aggregation function(s) to the aggregation attribute(s) *for each group*

The result attributes must be either a group attribute or a aggregation attribute.

Even More SQL

- Subquery
- Set operations
- Top N query
- Transactions