SQL joins and nested queries

Agenda

- Joining data
- Nested SELECT queries

Joining data

Joining data

- Sometimes you need data from more than one table:

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
Fay	20

DEPARTMENT_ID	DEPARTMENT_NAME
90	Executive
20	Marketing
10	Administration

LAST_NAME	DEPARTMENT_NAME
King	Executive
Fay	Marketing
Kochhar	Executive

Joining data

- This will produce a Cartesian product:

```
SELECT LAST_NAME, DEPARTMENT_NAME
FROM EMPLOYEES, DEPARTMENTS
```

- The result:

LAST_NAME	DEPARTMENT_NAME
King	Executive
King	Marketing
King	Administration
Kochhar	Executive
Kochhar	Marketing
..	..

Joining data

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition

Joining data

- Different types of joins include:
  - Natural joins
  - Join with `USING` clause
  - Inner joins with `ON` clause
  - Left, right and full outer joins
  - Self joins
  - Cross joins

Joining data

- The `NATURAL JOIN` combines the rows from two tables that have equal values in all matched by name columns

```
SELECT DEPARTMENT_ID, DEPARTMENT_NAME,
  LOCATION_ID, CITY
FROM DEPARTMENTS NATURAL JOIN LOCATIONS
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
...	...	...	...

Joining data

- If several columns have the same names we can limit the `NATURAL JOIN` to only one of them by the `USING` clause:

```
SELECT E.EMPLOYEE_ID, E.LAST_NAME,
  D.LOCATION_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E JOIN DEPARTMENTS D
  USING (DEPARTMENT_ID)
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_NAME
102	De Haan	1700	Executive
103	Hunold	1400	IT
104	Ernst	1400	IT
...	...	...	...

Joining data

- To specify arbitrary conditions or specify columns to join, the `ON` clause is used
  - Such `JOIN` is called also `INNER JOIN`

```
SELECT E.EMPLOYEE_ID, E.LAST_NAME,
  E.DEPARTMENT_ID, D.DEPARTMENT_ID, D.LOCATION_ID
FROM EMPLOYEES E JOIN DEPARTMENTS D
  ON (E.DEPARTMENT_ID = D.DEPARTMENT_ID)
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800

Joining data

- The join of two tables returning only matched rows is an **inner join**

- A join between two tables that returns the results of the inner join as well as unmatched rows from the left (or right) table is a **left** (or **right**) **outer join**

- A join between two tables that returns the results of an inner join as well as the results of a left and right join is a ***full outer join***

Joining data

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS
  MANAGER_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E INNER JOIN DEPARTMENTS D
  ON E.EMPLOYEE_ID=D.MANAGER_ID
```

MANAGER_NAME	DEPARTMENT_ID	DEPARTMENT_ NAME
Jennifer Whalen	10	Administration
Michael Hartstein	20	Marketing
Den Raphaely	30	Purchasing
Susan Mavris	40	Human Resources
Adam Fripp	50	Shipping
Alexander Hunold	60	IT
Hermann Baer	70	Public Relations
...	...	...

Joining data

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS
  MANAGER_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E LEFT OUTER JOIN DEPARTMENTS D
  ON E.EMPLOYEE_ID=D.MANAGER_ID
```

MANAGER_NAME	DEPARTMENT_ID	DEPARTMENT_ NAME
Jennifer Whalen	10	Administration
Michael Hartstein	20	Marketing
Den Raphaely	30	Purchasing
Clara Vishney	(null)	(null)
Jason Mallin	(null)	(null)
Hazel Philtanker	(null)	(null)
Nanette Cambrault	(null)	(null)
…	…	…

Joining data

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS
  MANAGER_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E RIGHT OUTER JOIN DEPARTMENTS D
  ON E.EMPLOYEE_ID=D.MANAGER_ID
```

MANAGER_NAME	DEPARTMENT_ID	DEPARTMENT_ NAME
Jennifer Whalen	10	Administration
Michael Hartstein	20	Marketing
Den Raphaely	30	Purchasing
(null)	120	Treasury
(null)	130	Corporate Tax
(null)	140	Control And Credit
(null)	150	Shareholder Services
...	...	...

Joining data

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS
  MANAGER_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E FULL OUTER JOIN DEPARTMENTS D
  ON E.EMPLOYEE_ID=D.MANAGER_ID
```

MANAGER_NAME	DEPARTMENT_ID	DEPARTMENT_ NAME
Jennifer Whalen	10	Administration
Michael Hartstein	20	Marketing
…	…	…
Clara Vishney	(null)	(null)
Jason Mallin	(null)	(null)
…	…	…
(null)	150	Shareholder Services
…	…	…

Joining data

- Several ways to express outer join:
  - ANSI SQL notation:

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS
  MANAGER_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E LEFT OUTER JOIN DEPARTMENTS D
  ON E.EMPLOYEE_ID=D.MANAGER_ID
```

  - Oracle-specific outer join syntax with (+)

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS
  MANAGER_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E JOIN DEPARTMENTS D
  ON E.EMPLOYEE_ID=D.MANAGER_ID(+)
```

Joining data

- A three-way join is a join of three tables

```
SELECT E.EMPLOYEE_ID, CITY, DEPARTMENT_NAME
FROM EMPLOYEES E
JOIN DEPARTMENTS D
  ON D.DEPARTMENT_ID = E.DEPARTMENT_ID
JOIN LOCATIONS L
  ON D.LOCATION_ID = L.LOCATION_ID
```

EMPLOYEE_ID	CITY	DEPARTMENT_ NAME
103	Southlake	IT
104	Southlake	IT
124	San Francisco	Administration
...	...	...

Joining data

- Self join means to join a table to itself
  - Always used with table aliases

```
SELECT CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME,
 ' is managed by ', M.LAST_NAME) as MSG
FROM EMPLOYEES E JOIN EMPLOYEES M
ON (E.MANAGER_ID = M.EMPLOYEE_ID)
```

MSG
Neena Kochhar is managed by King
Lex De Haan is managed by King
Alexander Hunold is managed by De Haan
Bruce Ernst is managed by Hunold
..

Joining data

- The CROSS JOIN clause produces the cross-product of two tables
  - Same as a Cartesian product
  - Not used often

```
SELECT LAST_NAME, DEPARTMENT_NAME
FROM EMPLOYEES CROSS JOIN DEPARTMENTS
```

LAST_NAME	DEPARTMENT_NAME
King	Executive
King	Marketing
King	Administration
Kochhar	Executive
..	..

Joining data

- You can apply additional conditions in the `WHERE` clause:

```
SELECT E.EMPLOYEE_ID,
  CONCAT(E.FIRST_NAME, ' ', E.LAST_NAME) AS NAME,
  E.MANAGER_ID, E.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E JOIN DEPARTMENTS D ON
  (E.DEPARTMENT_ID = D.DEPARTMENT_ID)
WHERE E.MANAGER_ID = 149
```

EMPLOYEE_ID	NAME	MANAGER_ID	DEPARTMENT_ID	DEPARTMENT_NAME
174	Ellen Abel	149	80	Sales
175	Alyssa Hutton	149	80	Sales
...	...	...	...	...

Joining data

- Joins can apply any Boolean expression in the `ON` clause:

```
SELECT E.FIRST_NAME, E.LAST_NAME,D.DEPARTMENT_NAME
FROM EMPLOYEES E
  INNER JOIN DEPARTMENTS D
  ON (E.DEPARTMENT_ID = D.DEPARTMENT_ID
  AND E.HIRE_DATE > TO_DATE('1/1/1999', 'DD/MM/YYYY')
  AND D.DEPARTMENT_NAME in ('Sales', 'Finance'))
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
148	Gerald	Cambrault	Sales
149	Eleni	Zlotkey	Sales
113	Luis	Popp	Finance
...	...	...	...

Nested SELECT queries

Nested SELECT queries

- SELECT statements can be nested in the where clause

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM EMPLOYEES
WHERE SALARY =
   (SELECT MAX(SALARY) FROM EMPLOYEES)

SELECT FIRST_NAME, LAST_NAME, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN
   (SELECT DEPARTMENT_ID FROM DEPARTMENTS
    WHERE DEPARTMENT_NAME='Accounting')
```

- Note: Always prefer joins to nested SELECT statements (better performance)

Nested SELECT queries

- Using the `EXISTS` operator in `SELECT` statements
  - Find all employees that have worked in the past in the department #110

```
SELECT FIRST_NAME, LAST_NAME
FROM EMPLOYEES E
WHERE EXISTS
   (SELECT EMPLOYEE_ID FROM JOB_HISTORY JH
    WHERE DEPARTMENT_ID = 110 AND
    JH.EMPLOYEE_ID=E.EMPLOYEE_ID)
```

Nested SELECT queries

- MySQL does not support `SELECT TOP X`
  - We can use the `LIMIT`:

```
SELECT * FROM
   (SELECT LAST_NAME, SALARY
    FROM EMPLOYEES
    ORDER BY SALARY DESC) AS INNER_TBL
LIMIT 5
```

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Russell	14000
Partners	13500

Nested SELECT queries

- Oracle also does not support `SELECT TOP X`
  - We can use the `ROWNUM` pseudo column

```
SELECT * FROM
   (SELECT LAST_NAME, SALARY
    FROM EMPLOYEES
    ORDER BY SALARY DESC)
WHERE ROWNUM <= 5
```

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Russell	14000
Partners	13500

Nested SELECT queries

- Oracle provides hierarchy traversal operator – `CONNECT BY PRIOR`

```
SELECT SYS_CONNECT_BY_PATH(
  CONCAT(FIRST_NAME, ' ', LAST_NAME), '/') as PATH
FROM EMPLOYEES
START WITH FIRST_NAME='Steven' AND LAST_NAME = 'King'
CONNECT BY PRIOR EMPLOYEE_ID = MANAGER_ID;
```

PATH
/Steven King
/Steven King/Neena Kochhar
/Steven King/Neena Kochhar/Nancy Greenberg
/Steven King/Neena Kochhar/Nancy Greenberg/Daniel Faviet
…

Questions ?

Exercises (1)

1. Write an SQL query to find all employees that are paid more than 10 000. Order them in decreasing order by salary. Use `ORDER BY`.
2. Write an SQL query to find the first 10 employees joined the company (most senior people).
3. Write an SQL query to find all skills and certificates of an employee. Use join with `USING` clause.
4. Write an SQL query to find all skills and certificates of an employee. Use inner join with `ON` clause.

Exercises (2)

5. Write a SQL query to find all departments with their employees. User right outer join.
6. Rewrite the last SQL query to use left outer join.
7. Write a SQL query to find the manager name of each department (use alternative HRM schema).
8. Modify the last SQL query to find also the department of each department manager (use alternative HRM schema).
9. Write a SQL query to find the names of all employees from the departments "Sales" and "Finance" whose hire year is between 1995 and 2000 (use alternative HRM schema)..

Exercises (3)

10. Write an SQL query to display all employees (first and last name) along with their corresponding manager (first and last name). Use self-join (use alternative HRM schema).
11. Write an SQL query to display all employees, along with their job title, department, title, skills and certificates. Use multiple joins (use alternative HRM schema).
12. Find last 5 hired employees in the company. Use `LIMIT` (alternative HRM schema).
13. Write a hierarchical SQL query to print the organizational structure of the “Sales” department. For each employee from the “Sales” department print all his/her managers as path to the big boss (use alternative HRM schema).

Exercises (4)

14. Write an SQL query to find the average salary in the "Human Resources" department. Use `AVG()`.
15. Write an SQL query to find the number of employees in the "Human Resources" department. Use `COUNT(*)`.
16. Write an SQL query to find the number of all locations where the company has an office (use alternative HRM schema).
17. Write an SQL query to find the number of all departments that have manager (use alternative HRM schema).
18. Write an SQL query to find all department names and the average salary for each of them (use alternative HRM schema)..

Exercises (5)

19. Write an SQL query to find the count of all employees in each department. Display the name and number of employees for each department.
20. *Write an SQL query to find for each department and for each manager the count of all corresponding employees.*
21. Write an SQL query to find all managers that have exactly 3 employees. Display their names and the name and their department.
22. Write an SQL query to find the total number of employees that have more than two certificates.
23. Write an SQL query to find for each department and for each job title the total number of employees.