Database Programming - Section 6
Prabhakar Pandey

6-1: Cross Joins and Natural Joins Practice Activities

Vocabulary
Identify the vocabulary word for each definition below

CROSS JOIN	Returns the Cartesian product from two tables
NATURAL JOIN	Joins two tables based on the same column name

1. Create a cross-join that displays the last name and department name from the employees and departments tables

- SELECT e.last_name, d.department_name
  FROM employees e
  NATURAL JOIN departments d;

2. Create a query that uses a natural join to join the departments table and the locations table. Display the department id, department name, location id, and city

- SELECT d.department id, d.department_name, l.location_id, l.city
  FROM departments d
  NATURAL JOIN locations l;

3. Create a query that uses a natural join to join the departments table and the locations table. Restrict the output to only department IDs of 20 and 50. Display the department id, department name, location id, and city

- SELECT d.department id, d.department_name, l.location_id, l.city
  FROM departments d
  NATURAL JOIN locations l
  WHERE d.department_id IN (20, 50);

6-2: Join Clauses Practice Activities

Vocabulary
Identify the vocabulary word for each definition below

ON	Allows a natural join based on an arbitrary condition or two columns with different names
USING	Performs an equijoin based on one specified column name

Use the Oracle database for problems 1-6

1. Join the Oracle database locations and departments table using the location_id column. Limit the results to location 1400 only

- SELECT d.department id, d.department_name, l.location_id, l.city
  FROM departments d
  JOIN locations l ON d.location_id = l.location_id
  WHERE l.location_id = 1400;

2. Join DJs on Demand d_play_list_items, d_track_listings, and d_cds tables with the JOIN USING syntax. Include the song ID, CD number, title, and comments in the output

- SELECT p.song_id, c.cd_nimber, t.title, t.comments
  FROM d_play_list_items p
  JOIN d_track_listings t USING (song_id)
  JOIN d_cds c USING (cd_number);

3. Display the city, department name, location ID, and department ID for departments 10, 20, and 30 for the city of Seattle

- SELECT d.department id, d.department_name, l.location_id, l.city
  FROM departments d
  JOIN locations l ON d.location_id = l.location_id
  WHERE d.departments_id IN (10,20,30) AND l.city = ‘Seattle’;

4. Display country name, region ID, and region name for Americas

- SELECT c.country_name, r.region_id, r.region_name

```
FROM countries c
JOIN regions r ON c.region_id = r.region_id
WHERE r.region_name = 'Americas';
```

5. Write a statement joining the employees and jobs tables. Display the first and last names, hire date, job id, job title, and maximum salary. Limit the query to those employees who are in jobs that can earn more than $12,000

- SELECT e.first_name, e.last_name, e.hire_date, j.job_id, j.job_title, j.max_salary
  FROM employees e
  JOIN jobs j ON e.job_id = j.job_id
  WHERE j.max_salary > 12000;

6. Display job title, employee first name, last name, and email for all employees who are stock clerks

- SELECT j.job_title, e.first_name, e.last_name, e.email
  FROM employees e
  JOIN jobs j ON e.job_id = j.job_id
  WHERE j.job_title = 'Stock Clerk';

The following questions use the JOIN...ON syntax:

7. Write a statement that displays the employee ID, first name, last name, manager ID, manager first name, and manager last name for every employee in the employees table. Hint: this is a self-join

- SELECT e.employee_id, e.first_name, e.last_name, e.manager_id, m.first_name AS manager_first_name, m.last_name AS manager_last_name
  FROM employees e
  JOIN employees m ON e.manager_id = m.employee_id;

8. Use JOIN ON syntax to query and display the location ID, city, and department name for all Canadian locations

- SELECT l.location_id, l.city, d.department_name
  FROM locations l
  JOIN departments d ON l.location_id = d.location_id

WHERE l.country_id = 'CAN';

9. Query and display manager ID, department ID, department name, first name, and last name for all employees in departments 80, 90, 110, and 190

- SELECT e.manager_id, d.department_id, d.department_name, e.first_name, e.last_name
  FROM employees e
  JOIN departments d ON e.department_id = d.department_id
  WHERE d.department_id IN (80, 90, 110, 190);

10. Display employee ID, last name, department ID, department name, and hire date for those employees whose hire date was June 7, 1994

- SELECT e.employee_id, e.last_name, e.department_id, d.department_name, e.hire_date
  FROM employees e
  JOIN departments d ON e.department_id = d.department_id
  WHERE e.hire_date = '1994-06-07';

6-3: Inner Versus Outer Joins Practice Activities

Vocabulary
Identify the vocabulary word for each definition below.

FULL OUTER JOIN	Performs a join on two tables, retrieves all the rows in the left table, even if there is no match in the right table. It also retrieves all the rows in the right table, even if there is no match in the left table
OUTER JOIN	A join that returns the unmatched rows as well as matched rows
LEFT OUTER JOIN	Performs a join on two tables, retrieves all the rows in the left table even if there is no match in the right table

RIGHT OUTER JOIN	Performs a join on two tables, retrieves all the rows in the Right table even if there is no match in the Left table
INNER JOIN	A join of two or more tables that returns only matched rows

1. Return the first name, last name, and department name for all employees including those employees not assigned to a department

- SELECT first_name, last_name, department_name
  FROM employees
  LEFT OUTER JOIN departments
  ON e.department_id = department_id;

2. Return the first name, last name, and department name for all employees including those departments that do not have an employee assigned to them

- SELECT first_name, last_name, department_name
  FROM employees
  RIGHT OUTER JOIN departments
  ON e.department_id = department_id;

3. Return the first name, last name, and department name for all employees including those departments that do not have an employee assigned to them and those employees not assigned to a department

- SELECT e.first_name, e.last_name, d.department_name
  FROM employees e
  FULL OUTER JOIN departments d
  ON e.department_id = d.department_id;

4. Create a query of the DJs on Demand database to return the first name, last name, event date, and description of the event the client held. Include all the clients even if they have not had an event scheduled

- SELECT c.firstname, c.lastname, e.eventdate, e.description
  FROM clients c
  LEFT JOIN events e ON c.clientid = e.clientid

ORDER BY c.lastname, c.firstname;

5. Using the Global Fast Foods database, show the shift description and shift assignment date even if there is no date assigned for each shift description

- SELECT s.shift_description, sa.shift_assignment_date
  FROM shifts s
  LEFT OUTER JOIN shift_assignments sa
  ON s.shift_id = sa.shift_id;

6-4 Self-Joins and Hierarchical Queries

Vocabulary
Identify the vocabulary word for each definition below.

Self-join	Joins a table to itself
Hierarchical Query	Retrieves data based on a natural hierarchical relationship between rows in a table
LEVEL	Determines the number of steps down from the beginning row that should be returned by a hierarchical query
START WITH	Identifies the beginning row for a hierarchical query
CONNECT BY	Specifies the relationship between parent rows and child rows of a hierarchical query

For each problem, use the Oracle database.

1. Display the employee's last name and employee number along with the manager's last name and manager number. Label the columns: Employee, Emp#, Manager, and Mgr#, respectively.

SELECT e.lastname AS “Employee”, e.employee_id AS “Emp#”, m.last_name AS “Manager”, m.employee_id AS “Mgr#”
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id;

2. Modify question 1 to display all employees and their managers, even if the employee does not have a manager. Order the list alphabetically by the last name of the employee.

SELECT e.lastname AS “Employee”, e.employee_id AS “Emp#”, m.last_name AS “Manager”, m.employee_id AS “Mgr#”
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id
ORDER BY e.last_name;

3. Display the names and hire dates for all employees who were hired before their managers, along with their managers’ names and hire dates. Label the columns Employee, Emp Hired, Manager and Mgr Hired, respectively.

SELECT e.lastname AS “Employee”, e.hire_date AS “Emp Hired”, m.last_name AS “Manager”, m.hire_date AS “Mgr Hired”
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id
WHERE e.hire_date < m.hire_date;

4. Write a report that shows the hierarchy for Lex De Haans department. Include last name, salary, and department id in the report.

SELECT last_name, salary, department_id
FROM employees
START WITH last_name = 'De Haan'
CONNECT BY PRIOR employee_id = manager_id;

5. What is wrong in the following statement?

SELECT last_name, department_id, salary
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR manager_id = employee_id;

It should state that the prior row's employee ID is the current manager ID.

```
CONNECT BY PRIOR employee_id = manager_id;
```

6. Create a report that shows the organization chart for the entire employee table. Write the report so that each level will indent each employee 2 spaces. Since Oracle Application Express cannot display the spaces in front of the column, use - (minus) instead.

```
SELECT LPAD('-', 2 * (LEVEL - 1)) || last_name AS "Org Chart", salary, department_id
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

7. Re-write the report from 6 to exclude De Haan and all the people working for him

```
SELECT LPAD('-', 2 * (LEVEL - 1)) || last_name AS "Org Chart", salary, department_id
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND employee_id NOT IN
        (SELECT employee_id
        FROM employees
        START WITH last_name = 'De Haan'
        CONNECT BY PRIOR employee_id = manager_id);
```