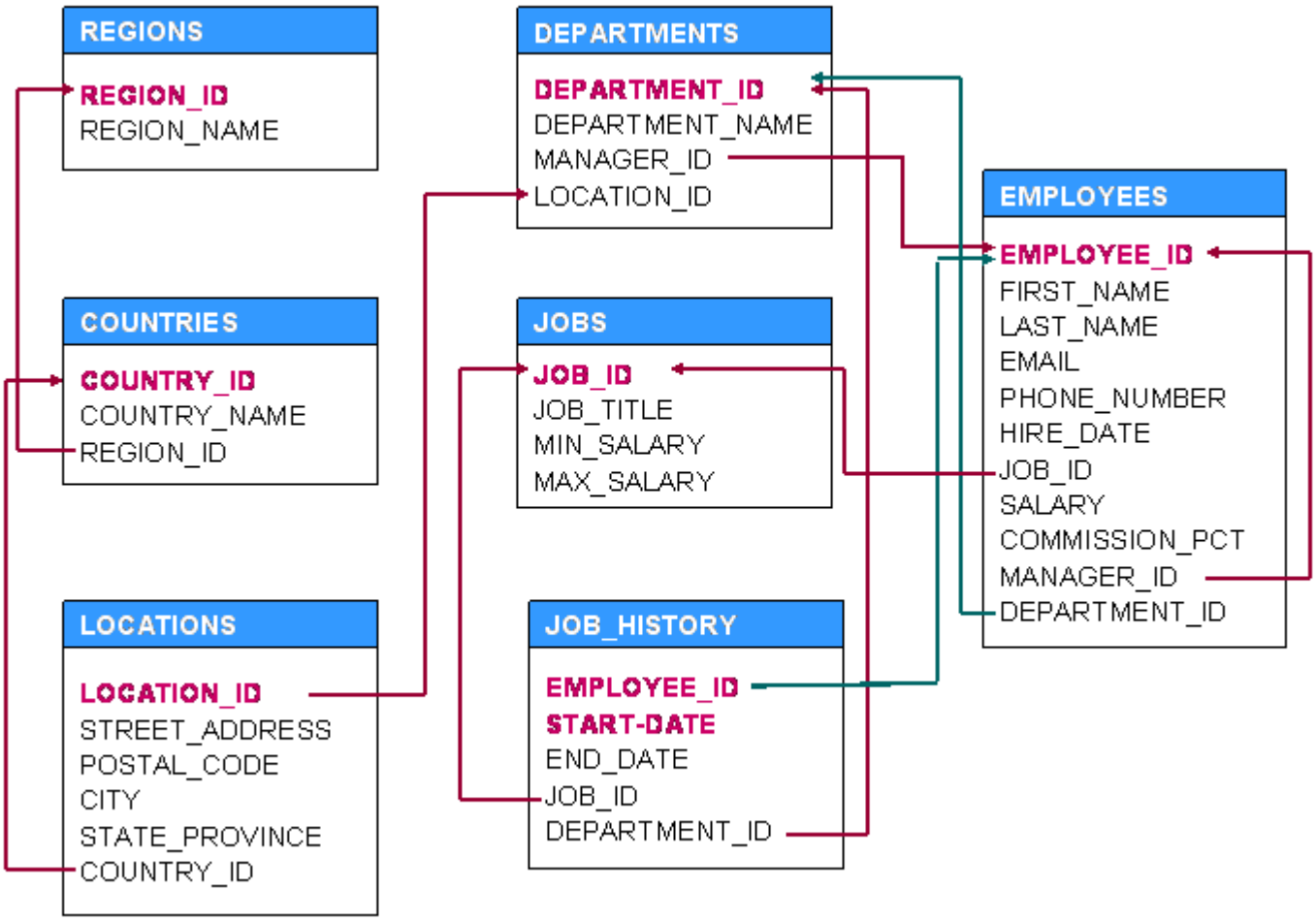
SQL Queries

Exercises related to HR Schema in Oracle Database 11g

Here are the excercises related to queries given in Oracle Database 11g.

The following is the structure of the tables provided by Oracle in Human Resource Schema (HR).



Note: Columns in RED color indicate primary key(s).

Queries

Le join vanno fatte sia con la sintassi "oracle" che con quella ANSI vedi appendice al termine del documento.
Le risposte sono state scritte con la sintassi ANSI, voi sentitevi liberi di scriverle come volete (al limite ne fate alcune ansi ed alcune "standard", non serve farle "doppie"…sarebbe solo una perdita di tempo!), l'importante è che le sappiate usare entrambe.

1. Display details of jobs where the minimum salary is greater than 10000.
2. Display the first name and join date of the employees who joined between 2002 and 2005.
3. Display first name and join date of the employees who is either IT Programmer or Sales Man.
4. Display employees who joined after 1st January 2008.
5. Display details of employee with ID 150 or 160.
6. Display first name, salary, commission pct, and hire date for employees with salary less than 10000.
7. Display job Title, the difference between minimum and maximum salaries for jobs with max salary in the range 10000 to 20000.
8. Display first name, salary, and round the salary to thousands.
9. Display details of jobs in the descending order of the title.
10. Display employees where the first name or last name starts with S.
11. Display employees who joined in the month of May.
12. Display details of the employees where commission percentage is null and salary in the range 5000 to 10000 and department is 30.
13. Display first name and date of first salary of the employees. (DUBBIO:il primo giorno del salario in base a cosa viene calcolato? Il primo giorno del mese successivo oppure dopo 30 giorni dalla data di assunzione?)
14. Display first name and experience of the employees.
15. Display first name of employees who joined in 2001.
16. Display first name and last name after converting the first letter of each name to upper case and the rest to lower case.
17. Display the first word in job title. (AMBIGUO: In che senso la prima parola? Io ho pensato tra i job_title , la prima in ordine alfabetico)
18. Display the length of first name for employees where last name contain character 'b' after 3rd position.
19. Display first name in upper case and email address in lower case for employees where the first name and email address are same irrespective of the case.
20. Display employees who joined in the current year.
21. Display the number of days between system date and 1st January 2011.
22. Display how many employees joined in each month of the current year.
23. Display manager ID and number of employees managed by the manager.
24. Display employee ID and the date on which he ended his previous job.
25. Display number of employees joined after 15th of the month.
26. Display the country ID and number of cities we have in the country.
27. Display average salary of employees in each department who have commission percentage.
28. Display job ID, number of employees, sum of salary, and difference between highest salary and lowest salary of the employees of the job.
29. Display job ID for jobs with average salary more than 10000.
30. Display years in which more than 10 employees joined.
31. Display departments in which more than five employees have commission percentage.
32. Display employee ID for employees who did more than one job in the past.
33. Display job ID of jobs that were done by more than 3 employees for more than 100 days.

34. Display department ID, year, and Number of employees joined.
35. Display departments where any manager is managing more than 5 employees.
36. Change salary of employee 115 to 8000 if the existing salary is less than 6000.
37. Insert a new employee into employees with all the required details.
38. Delete department 20.
39. Change job ID of employee 110 to IT_PROG if the employee belongs to department 10 and the existing job ID does not start with IT.
40. Insert a row into departments table with manager ID 120 and location ID in any location ID for city Tokyo. (PROBLEMA: se non inserisco la PRIMARY KEY non riesco ad inserirei dati richiesti)
41. Display department name and number of employees in the department.
42. Display job title, employee ID, number of days between ending date and starting date for all jobs in department 30 from job history.
43. Display department name and manager first name.
44. Display department name, manager name, and city.
45. Display country name, city, and department name.
46. Display job title, department name, employee last name, starting date for all jobs from 2000 to 2005.
47. Display job title and average salary of employees
48. Display job title, employee name, and the difference between maximum salary for the job and salary of the employee.
49. Display last name, job title of employees who have commission percentage and belongs to department 30.
50. Display details of jobs that were done by any employee who is currently drawing more than 15000 of salary.
51. Display department name, manager name, and salary of the manager for all managers whose experience is more than 5 years.
52. Display employee name if the employee joined before his manager.
53. Display employee name, job title for the jobs employee did in the past where the job was done less than six months.
54. Display employee name and country in which he is working.
55. Display department name, average salary and number of employees with commission within the department.
56. Display the month in which more than 5 employees joined in any department located in Sydney.
57. Display details of departments in which the maximum salary is more than 10000.
58. Display details of departments managed by ‘Smith’.
59. Display jobs into which employees joined in the current year.
60. Display employees who did not do any job in the past. (AMBIGUO: penso che bisogna considerare solo i dipendenti che sono al primo lavoro, e che quindi in passato non hanno avuto altre esperienze)
61. Display job title and average salary for employees who did a job in the past. (AMBIGUO: penso che bisogna considerare solo i dipendenti che in precedenza hanno avuto un altro lavoro)
62. Display country name, city, and number of departments where department has more than 5 employees.
63. Display details of manager who manages more than 5 employees.
64. Display employee name, job title, start date, and end date of past jobs of all employees with commission percentage null. (AMBIGUO: “past jobs”, anche qui penso come in 61.)
65. Display the departments into which no employee joined in last two years.
66. Display the details of departments in which the max salary is greater than 10000 for employees who did a job in the past. (AMBIGUO: anche qui penso come in 61.)

67. Display details of current job for employees who worked as IT Programmers in the past.
68. Display the details of employees drawing the highest salary in the department.
69. Display the city of employee whose employee ID is 105.
70. Display third highest salary of all employees

Appendice

dalla versione 9i oracle supporta, oltre alla "sua" sintassi Standard anche la ANSI/ISO SQL. Di seguito esempi della "corrispondenza" tra le due sintassi, presi dal sito:

http://www.oracle-base.com/articles/9i/ansi-iso-sql-support.php

ANSI/ISO SQL Support In Oracle 9i

Oracle 9i now supports the ANSI/ISO SQL: 1999 standards. This allows easier product migration and a reduced learning curve when cross-training, but there is no performance increase compared to the existing syntax.

Joins

A range of new join syntax are available that comply with the ANSI/ISO SQL: 1999 standards.

CROSS JOIN

The `CROSS JOIN` produces a cartesian product.

ANSI/ISO Syntax	Existing Syntax
`SELECT first_name,` `  last_name,`	`SELECT first_name,` `  last_name,`

```
        department_name
FROM    employees
        CROSS JOIN departments;
```

```
        department_name
FROM    employees,
        departments;
```

NATURAL JOIN

The `NATURAL JOIN` performs a join for all columns with matching names in the two tables.

ANSI/ISO Syntax

```
SELECT department_name,
       city
FROM   departments
       NATURAL JOIN locations;
```

Existing Syntax

```
SELECT d.department_name,
       l.city
FROM   departments d,
       locations l
WHERE  d.location_id = l.location_id
AND    d.country     = l.country;
```

JOIN ... USING

The `USING` clause is used if several columns share the same name, but you do not wish to join using all of these common columns. The columns listed in the USING clause cannot have any qualifiers in the statement, including the WHERE clause.

ANSI/ISO Syntax

```
SELECT d.department_name,
       l.city
FROM   departments d
       JOIN locations l USING
(location_id);
```

Existing Syntax

```
SELECT d.department_name,
       l.city
FROM   departments d,
       locations l
WHERE  d.location_id =
l.location_id;
```

JOIN ... ON

The `ON` clause is used to join tables where the column names do not match. The join conditions are removed from the filter conditions in the where clause.

ANSI/ISO Syntax

```
SELECT d.department_name,
       l.city
FROM   departments d
       JOIN locations l ON (d.location_id = l.id);
```

Existing Syntax

```
SELECT d.department_name,
       l.city
FROM   departments d,
       locations l
WHERE  d.location_id = l.id;
```

Multiple Joins

Multiple Joins are those where more than two tables are joined. The SQL: 1999 standard assumes the tables are joined from the left to the right, with the join conditions only being able to reference columns relating to the current join and any previous joins to the left.

ANSI/ISO Syntax

```
SELECT employee_id,
```

Existing Syntax

```
SELECT employee_id,
```

```
       city,
       department_name
FROM   locations l
       JOIN departments d ON (d.location_id =
l.location_id)
       JOIN employees e ON (d.department_id =
e.department_id);
```

```
       city,
       department_name
FROM   locations l,
       departments d,
       employees e
WHERE  d.location_id    =
l.location_id
AND    d.department_id =
e.department_id;
```

OUTER JOIN

There are three variations on the outer join. The `LEFT OUTER JOIN` returns all the rows from the table on the left side of the join, along with the values from the right hand side, or NULLs if a matching row doesn't exist. The `RIGHT OUTER JOIN` does the reverse of this. Finally, the `FULL OUTER JOIN` returns all rows from both tables, filling in any blanks with nulls.

ANSI/ISO Syntax	Existing Syntax
`SELECT e.last_name, d.department_name FROM employees e LEFT OUTER JOIN departments d ON (e.department_id = d.department_id);`	`SELECT e.last_name, d.department_name FROM employees e, departments d WHERE e.department_id = d.department_id(+);`
`SELECT e.last_name, d.department_name FROM employees e RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id);`	`SELECT e.last_name, d.department_name FROM employees e, departments d WHERE e.department_id(+) = d.department_id;`
`SELECT e.last_name, d.department_name FROM employees e FULL OUTER JOIN departments d ON (e.department_id = d.department_id);`	`No Equivalent!`

Extra filter conditions can be added to the join to using AND to form a complex join. These are often necessary when filter conditions are required to restrict an outer join. If these filter conditions are placed in the WHERE clause and the outer join returns a NULL value for the filter column the row would be thrown away. If the filter condition is coded as part of the join the situation can be avoided.

CASE Expressions

The case expression is a more flexible extension of the `DECODE` statement. In its simplest form it is used to return a value when a match is found.

```
SELECT last_name, commission_pct,
  (CASE commission_pct
     WHEN 0.1  THEN 'Low'
     WHEN 0.15 THEN 'Average'
     WHEN 0.2  THEN 'High'
     ELSE 'N/A'
   END) Commission
FROM employees
ORDER BY last_name;
```

A more complex version is the Searched case expression where a comparison expression is used to find a match.

```
SELECT last_name, job_id, salary,
  (CASE
     WHEN job_id LIKE 'SA_MAN' AND salary < 12000 THEN '10%'
     WHEN job_id LIKE 'SA_MAN' AND salary >= 12000 THEN '15%'
     WHEN job_id LIKE 'IT_PROG' AND salary < 9000 THEN '8%'
     WHEN job_id LIKE 'IT_PROG' AND salary >= 9000 THEN '12%'
     ELSE 'NOT APPLICABLE'
  END) pay_raise
FROM employees;
```

Return values cannot be the literal NULL.

NULLIF Function

The NULLIF function returns a NULL value if both parameters are equal in value. If the parameters are not equal, it returns the value of the first parameter. The following query would return NULL.

```
SELECT NULLIF(1,1) FROM dual;
```

COALESCE Function

The COALESCE function returns the first non-NULL value in an expression list. If all expressions are null it returns NULL. The following query would return '3'.

```
SELECT COALESCE(NULL, NULL, '3') FROM dual;
```

Scalar Subqueries

Scalar subqueries return a single value. They could be used in previous versions of Oracle in some parts of an SQL statement, but Oracle9i extends their use to almost any place where an expression can be used, including:

- CASE expressions
- SELECT statement
- VALUES clause of an INSERT statement
- WHERE clause
- ORDER BY clause
- As a parameter of a function

For example.

```
INSERT INTO my_table VALUES ((SELECT 1 FROM dual), NULL);

SELECT Substr((SELECT 'ABC' FROM dual), 1, 1) FROM dual;
```