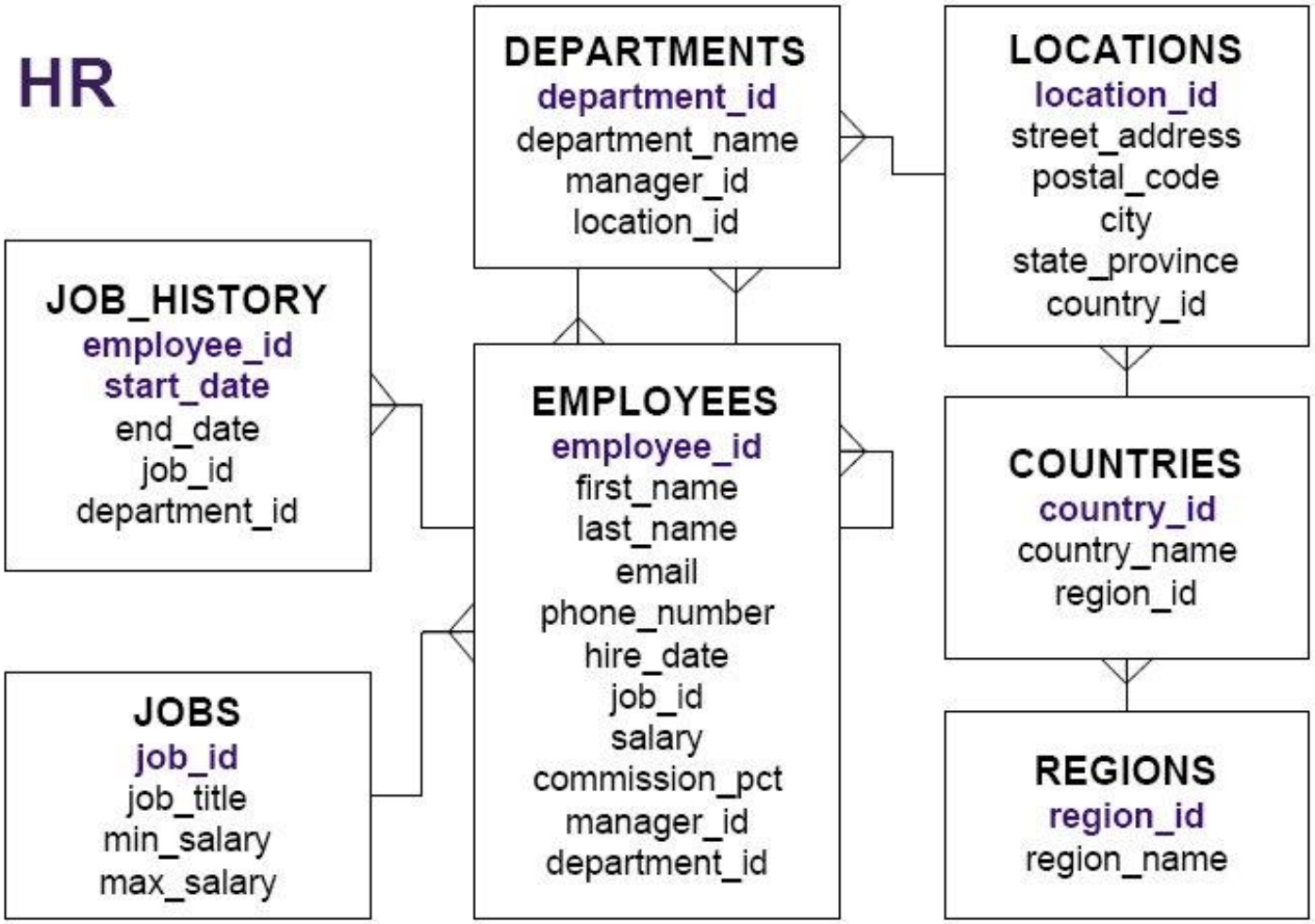
SQL

Prof. Edson Kaneshima

Esquema HR


HR
DEPARTMENTS
department_id
department_name
manager_id
location_id
LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id
JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id
EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id
COUNTRIES
country_id
country_name
region_id
JOBS
job_id
job_title
min_salary
max_salary
REGIONS
region_id
region_name

Instruções Básicas de SQL

- O SQL é a linguagem padrão ANSI para acesso a banco de dados relacionais e engloba tanto uma DDL quanto uma DML.

- É uma linguagem baseada em conjunto, isto é, com um simples comando recuperamos um conjunto de registros, sem precisarmos efetuar leituras registro a registro.

Consultas em múltiplas tabelas

- Para consultar dados de mais de uma tabela, você precisa identificar colunas comuns entre as duas tabelas.
  - Um exemplo seria a coluna DEPARTMENT_ID na tabela de EMPREGADO que corresponde a coluna de mesmo nome (DEPARTAMENT_ID) na tabela de DEPARTAMENTO.

- Na cláusula WHERE, você define a relação entre as tabelas listadas na cláusula FROM usando operadores de comparação.
  - Normalmente o operador de igualdade (=) é o utilizado.

- Veremos que ao invés de utilizarmos o WHERE, podemos indicar a correspondência entre duas tabelas utilizando a cláusula JOIN.
  - A consulta entre múltiplas tabelas sem correspondência ou colunas em comum é conhecida como produto cartesiano.

Junções Simples

- Uma junção simples é conhecida como *inner join*, pois ela retorna somente as linhas que satisfazem as condições de junção.
  - Um exemplo seria a busca do código e nome dos departamentos e suas respectivas cidades. Esta informação não consta em uma única tabela, bastando uma junção simples.
    - select DEPARTMENTS.DEPARTMENT_ID,
      DEPARTMENTS.DEPARTMENT_NAME, LOCATIONS.CITY
      from DEPARTMENTS, LOCATIONS
      where DEPARTMENTS.LOCATION_ID = LOCATIONS.LOCATION_ID
  - O resultado é algo como abaixo:

```
ID_DEPARTAMENTO  NOME_DEPARTAMENTO              CIDADE
---------------  ------------------------------ ------------------------------
            10  Administration                 Seattle
            20  Marketing                      Toronto
           ...  ...                            ...
           270  Payroll                        Seattle
```

Junções Complexa

- Uma junção complexa é uma junção simples que contém na cláusula WHERE, além da condição de junção, uma outra condição para filtrar as linhas retornadas.
  - Um exemplo seria a busca do código e nome dos departamentos e suas respectivas cidades fora dos Estados Unidos.
    - select DEPARTMENTS.DEPARTMENT_ID,
      DEPARTMENTS.DEPARTMENT_NAME, LOCATIONS.CITY
      from DEPARTMENTS, LOCATIONS
      where DEPARTMENTS.LOCATION_ID = LOCATIONS.LOCATION_ID
      and LOCATIONS.COUNTRY_ID <> 'US'
  - O resultado é algo como abaixo:

```
ID_DEPARTAMENTO NOME_DEPARTAMENTO              CIDADE
--------------- ------------------------------ ------------------------------
             20 Marketing                      Toronto
            ... ...                            ...
             80 Sales                          Oxford
```

Criando *aliases* para tabelas

- Observe a consulta do *slide* anterior reescrita usando “apelidos”, como o seguinte:
  - select **d**.DEPARTMENT_ID, **d**.DEPARTMENT_NAME, **l**.CITY
    from DEPARTMENTS **d**, LOCATIONS **l**
    where **d**.LOCATION_ID = **l**. LOCATION_ID
    and **l**.COUNTRY_ID <> 'US'

- Apelidos de tabela aumentam a legibilidade da consulta. Eles são muito utilizados para reduzir grandes nomes de tabelas em curtos apelidos.

NATURAL JOIN

- No *natural join*, a junção é realizada por todas as colunas que possuem o mesmo nome em ambas as tabelas.

- Neste tipo de junção não se deve qualificar o nome das colunas com o nome da tabela ou com o “apelido” da tabela.

- Sintaxe:
  - <nome-da-tabela> NATURAL [INNER] JOIN <nome-da-tabela>

- Vamos refazer o exemplo da busca do código e nome dos departamentos e suas respectivas cidades.
  - select DEPARTMENT_ID, DEPARTMENT_NAME, CITY
    from DEPARTMENTS **natural join** LOCATIONS

JOIN ... USING

- Neste tipo de junção, a junção é realizada pelas colunas indicadas e que possuem o mesmo nome em ambas as tabelas.
- A cláusula USING especifica os nomes das colunas que devem ser utilizadas para realizar a junção das tabelas.
- Também neste tipo de junção não se deve qualificar o nome das colunas com o nome da tabela ou com o “apelido” da tabela.
- Sintaxe:
  - <nome-da-tabela> [INNER] JOIN <nome-da-tabela> USING (<colunas>)

JOIN ... USING

- Vamos refazer o exemplo da busca do código e nome dos departamentos e suas respectivas cidades.
  - select DEPARTMENT_ID, DEPARTMENT_NAME, CITY
    from DEPARTMENTS **join** LOCATIONS
    using (LOCATION_ID)

JOIN ... ON

- Quando não se tem nomes de colunas comuns entre tabelas para fazer uma junção ou deseja-se especificar condições de junções arbitrárias, este é o tipo de junção a ser utilizado.

- Neste tipo de junção pode-se e deve-se qualificar o nome das colunas com o nome da tabela ou com o “apelido” da tabela.

- Sintaxe:
  - <nome-da-tabela> [INNER] JOIN <nome-da-tabela> ON <condição>

JOIN ... ON

- Vamos refazer o exemplo da busca do código e nome dos departamentos e suas respectivas cidades.
  - select d.DEPARTMENT_ID, d.DEPARTMENT_ID, l.CITY
    from DEPARTMENTS d **join** LOCATIONS l
    on (d.LOCATION_ID = l.LOCATION_ID)

Junções em múltiplas tabelas

- Vamos fazer o exemplo da busca do primeiro nome do empregado, nome do seu departamento, nome da cidade do seu departamento.
  - select e.FIRST_NAME, d.DEPARTMENT_NAME, l.CITY
    from EMPLOYEES e
    JOIN DEPARTMENTS d
    on (e.DEPARTMENT_ID = d.DEPARTMENT_ID)
    JOIN HR.LOCATIONS l
    on (d.LOCATION_ID = l. LOCATION_ID)

OUTER JOINS

- Até agora, vimos *inner joins*, que retornam somente as linhas correspondentes entre ambas as tabelas, e produtos cartesianos, que retornam uma combinação de todas as linhas de ambas as tabelas.

- Em alguns casos, deseja-se ver os dados de uma tabela, mesmo que não haja uma linha correspondente na tabela de junção.

- Para o problema acima, existe a junção do tipo *outer join*. Ele retorna valores baseados nas condições de *inner join*, como também linhas não relacionadas de um ou dos dois lados das tabelas.

OUTER JOINS

- Na sintaxe do Oracle, o símbolo + envolvido por parênteses, é uma notação de *outer join* na consulta.

- Digite (+) próximo a coluna da tabela onde pode não haver linhas correspondentes.

- Um exemplo é buscar a lista de nomes de todos os países e cidades em que possuem departamentos localizados. Caso o país não tenha nenhum departamento localizado, deseja-se exibi-lo mesmo assim.
  - select p.COUNTRY_NAME, l.CITY from COUNTRIES p, LOCATIONS l
    where p. COUNTRY_ID = l.COUNTRY_ID **(+)**

OUTER JOINS

- O exemplo da página anterior pode ser reescrito das seguintes formas:
  - select p.COUNTRY_NAME, l.CITY
    from COUNTRIES p **left outer join** LOCATIONS l
    on p.COUNTRY_ID = l. COUNTRY_ID

  - select p.COUNTRY_NAME, l.CITY
    from COUNTRIES p **left outer join** LOCATIONS l
    using (COUNTRY_ID)

  - select p. COUNTRY_NAME, l.CITY
    from COUNTRIES p **natural left outer join** LOCATIONS l

OUTER JOINS

- Um exemplo inverso é buscar a lista de todas as cidades que possuem departamentos e os seus países. Caso a cidade não esteja vinculada a um país, deseja-se exibí-la mesmo assim.
  - select p.COUNTRY_NAME, l.CITY from COUNTRIES p, LOCATIONS l

    where p.COUNTRY_ID **(+)** = l. COUNTRY_ID

OUTER JOINS

- O exemplo da página anterior pode ser reescrito das seguintes formas:
  - select p.COUNTRY_NAME, l.CITY
    from COUNTRIES p **right outer join** LOCATIONS l
    on p.COUNTRY_ID = l. COUNTRY_ID

  - select p.COUNTRY_NAME, l.CITY
    from COUNTRIES p **right outer join** LOCATIONS l
    using (COUNTRY_ID)

  - select p.COUNTRY_NAME, l.CITY
    from COUNTRIES p **natural right outer join** LOCATIONS l

SUBQUERIES

- Uma *subquery* é um consulta dentro de uma consulta.
- Quando você tem várias *subqueries*, a *subquery* mais interna é avaliada primeiro.
- Podem ser usadas com todas as instruções DML.

SUBQUERIES

- Buscar o primeiro e o último nome e o salário do empregado com o maior salário.
  - select e.FIRST_NAME, e.LAST_NAME, e.SALARY
    from EMPLOYEES e
    where e.SALARY =
    (select max(SALARY) from EMPLOYEES)

  - Resultado:

```
PRIMEIRO_NOME        ULTIMO_NOME                    SALARIO
-------------------- ------------------------- ----------
Steven               King                            24000
```

SUBQUERIES

- Buscar o primeiro e o último nome dos empregados e o código do departamento cujo nome do departamento comece com a letra 'A'.
  - select e.FIRST_NAME, e.LAST_NAME, e.DEPARTMENT_ID
    from EMPLOYEES e
    where e.DEPARTMENT_ID IN
    (select DEPARTMENT_ID from DEPARTMENTS
    where DEPARTMENT_NAME like 'A%')
  - Resultado:

```
PRIMEIRO_NOME        ULTIMO_NOME                    ID_DEP
-------------------- ------------------------- ----------
Jennifer             Whalen                             10
Shelley              Higgins                           110
William              Gietz                             110
```

SUBQUERIES

- Buscar o código do departamento, o primeiro e o último nome e o salário do empregado com o maior salário do seu departamento.
  - select e.DEPARTMENT_ID,
    e.FIRST_NAME, e.LAST_NAME, e.SALARY
    from EMPLOYEES e
    where e.SALARY =
    (select max(ei.SALARY) from EMPLOYEES ei
    where ei.DEPARTMENT_ID = e.DEPARTMENT_ID)

SUBQUERIES

- Resultado:

ID_DEP	PRIMEIRO_NOME	ULTIMO_NOME	SALARIO
90	Steven	King	24000
60	Alexander	Hunold	9000
100	Nancy	Greenberg	12000
30	Den	Raphaely	11000
50	Adam	Fripp	8200
80	John	Russell	14000
10	Jennifer	Whalen	4400
20	Michael	Hartstein	13000
40	Susan	Mavris	6500
70	Hermann	Baer	10000
110	Shelley	Higgins	12000