Banco de Dados

AULA 4 – CONSULTAS SQL

Revisão – `INSERT INTO`

```
INSERT INTO departments (department_id, name)
VALUES (100, 'Purchasing');
```

```
INSERT INTO departments
VALUES (101, 'Finance', NULL, '2015-12-31');
```

Revisão – `UPDATE`

```
UPDATE departments

SET location = 'West';
```

```
UPDATE departments

SET department_id = 104

WHERE department_id = 103;
```

Revisão – `DELETE` e `TRUNCATE`

```
DELETE FROM departments

WHERE name = 'Technology';
```

```
DELETE FROM departments;
```

```
TRUNCATE departments CASCADE;
```

Revisão – COMMIT, SAVEPOINT, ROLLBACK e ROLLBACK TO SAVEPOINT

```
COMMIT;
…
ROLLBACK;
```

```
BEGIN;
    …
    SAVEPOINT nome;
    …
    ROLLBACK TO SAVEPOINT nome;
COMMIT;
```

Introdução

- Uma instrução de consulta SQL permite a recuperação de dados de uma ou mais relações (tabelas, visões) existentes em uma base de dados.
- Não é necessário especificar a forma de recuperação e a ordem na qual os dados serão recuperados.
- SQL é uma linguagem não procedural;
- Instruções de consulta podem ser extremamente complexas.
- Para melhor entendimento inicia-se por instruções simples, chegando progressivamente às mais complexas.

Forma da instrução **SELECT**

- O conhecimento da estrutura e operadores da álgebra relacional é importante no entendimento do processo de execução das consultas.
- A cláusula WHERE não é obrigatória, mas é usada com muita frequência.

```
SELECT (operação de projeção)
FROM (produto cartesiano)
WHERE (seleção)
```

Modelo de execução

- Modelo canônico de execução de uma consulta SQL:

  1 - É feito um produto cartesiano de todas as tabelas, relações envolvidas, citadas na cláusula **`FROM`**.

  2 - São selecionadas todas as linhas que obedecem aos critérios definidos na cláusula **`WHERE`**.

  3 - É feita a projeção das colunas que vão ao resultado, colunas definidas na cláusula **`SELECT`**.

Modelo de execução

- Exemplos de consultas SQL:

```
SELECT *
FROM department;
```

```
SELECT department_id, department_name
FROM department;
```

Modelo de execução

- A cláusula **`FROM`** apresenta somente uma tabela, portanto não há produto cartesiano.
- São projetados quatro campos na primeira consulta, no qual o “*” indica o retorno de todos os campos da tabela “department”, e dois campos no segundo caso.

Álgebra relacional

- π department_id, department_name (department).
- π: operação de projeção dos campos que serão mostrados.

- Inserindo uma operação de seleção para mostrar somente códigos iguais a 10: π department_id, department_name (π department_id = 10 (department)).
- A primeira operação resulta na produção de uma relação com os registros resultantes da seleção (π).

Produto cartesiano

- Na matemática, produto cartesiano de dois conjuntos é o conjunto de todos os pares ordenados, tais que, em cada par, o primeiro elemento vem do primeiro conjunto e o segundo vem do segundo conjunto.
- Em SQL, o produto cartesiano é um combinação de todos os registros de uma tabela com todos da outra, sem considerar a relação existente entre estas tabela.

Produto cartesiano

- O número final de linhas é o resultado da multiplicação do número de linhas da primeira tabela pelo número de linhas da segunda.
- O número de colunas (caso não seja definido na cláusula **SELECT**) é a soma das colunas das duas tabelas.

```
SELECT *
FROM department, region;
```

Renomear tabelas

- É possível atribuir um apelido para qualquer tabela referenciada na cláusula **`FROM`**.
- Em alguns casos, pode ser necessário renomear uma tabela, principalmente quando a tabela é utilizada mais de uma vez na cláusula **`FROM`**.
- Em casos de haver mais de uma chave estrangeira associada a mesma tabela ou ainda um auto-relacionamento que force o uso da mesma tabela duas vezes.

```
SELECT d.department_name, r.region_name

FROM department d, region r;
```

Junção e produto cartesiano

- Normalmente é desnecessário mostrar a combinação de todos os registros de uma tabela com os de outra.
- Quando existe um atributo comum em ambas as relações é possível estabelecer a junção. O atributo em questão é chave estrangeira.
- Para tanto, é incluída uma cláusula **`WHERE`** para a seleção dos registros associados.

Junção

- A operação de junção é usada para produzir a lista dos registros de duas tabelas que estão relacionados através de atributos comuns.
- O caso mais comum de junção é a chamada junção natural, na qual existe igualdade de valores, porém as junções podem envolver outros critérios de comparação, não apenas igualdade.

```
SELECT c.country_name, r.region_name
FROM country c, region r
WHERE c.region_id = r.region_id;
```

Junção usando a álgebra relacional

- π country.region_id = region.region_id (country x region)
- É realizada uma seleção para mostrar somente os registros que atendam ao critério de seleção, o que equivale a uma junção.
- Na álgebra existe o operador “x” que faz a junção natural, desde que os campos associados tenham nomes iguais.

Junção natural utilizando a cláusula **`WHERE`**

- **`WHERE country.region_id = region.region_id`**
- É necessário colocar o nome da tabela em frente ao nome do campo porque ambas as tabelas possuem campos com o mesmo nome (region_id).
- Na linguagem SQL, diferentemente da álgebra relacional, é necessário especificar os critérios de junção, mesmo que os nomes das colunas sejam os mesmos.

Tipos de Junção

- **`INNER`**: Retorna o número de linhas que satisfazem o predicado de junção. A condição pode ser qualquer comparação, igualdade ou outra.
- A junção pode ser feita em duas operações (**`FROM e WHERE`**) ou somente na cláusula **`FROM tabela1 INNER JOIN tabela2 ON <condição>`**.

Junções externas

- **`LEFT`**: Mostra todos os registros da primeira tabela, mesmo que não estejam associados a nenhum da segunda.
- **`RIGHT`**: Todos os registros da segunda tabela são exibidos, mesmo não estando associados a nenhum da primeira.
- **`FULL`**: Combina os resultados do dois outros **`JOIN`**s externos, mostra no mínimo uma vez os registros de cada tabela, mesmo que não exista nenhum associado na outra.

Exemplo de junções externas

```
SELECT *
FROM department RIGHT JOIN location
ON department.location_id = location.location_id
ORDER BY department.department_id;
```

- A junção pode ser **LEFT**, onde todos os registros da primeira tabela são exibidos.

Eliminação de duplicatas

- Exemplo de consulta com valores repetidos: Mostrar o código de todos os departamentos já associados a algum funcionário.

```
SELECT department_id
FROM employee;
```

- É necessária somente a tabela **employee** porque a mesma possui o código do departamento.

Eliminação de duplicatas

- No SQL mostrado, se um mesmo departamento tiver sido usado para mais de um funcionário, o código correspondente irá aparecer no resultado igual ao número de vezes.
- Na álgebra relacional, valores não são mostrados mais de uma vez, mas em SQL este é o padrão;
- Para não haver valores duplicados, é necessário usar a cláusula **`DISTINCT`**.

```
SELECT DISTINCT department_id

FROM employee;
```

Eliminação de duplicatas

- A eliminação de valores duplicados envolve toda a cláusula **`SELECT`**, isto significa que se forem projetados vários campos só haverá a eliminação se ocorrer a repetição do resultado completo.

```
SELECT DISTINCT department_id, employee_first_name
FROM employee;
```

- Neste exemplo, se houverem resultados com valores iguais em ambos os campos, haverá eliminação, mas se apenas o código de departamento for o mesmo de outro registro a eliminação não ocorre.

Elaboração de consultas

- Passos básicos para elaboração de uma consulta (**`SELECT`**, **`FROM`**, **`WHERE`**):

1. Definir quais as tabelas que deverão constar na cláusula **`FROM`**.
2. Efetuar a junção, selecionando apenas os registros relacionados.
3. Projetar os campos que devem aparecer no resultado.
4. Definir outros critérios de seleção.
5. Definir agrupamentos, operações sobre campos, ordenação, etc.

Operadores para seleção (`WHERE`)

- Comparações: `=, >, <, >=, <=`
- Negação: **`NOT`** ou `!`
  - **`department_id != 1`**
- Conectores lógicos: **`AND`** e **`OR`**
- Entre dois valores: **`BETWEEN`**
  - **`BETWEEN 10 AND 15`**

Operadores para seleção (**`WHERE`**)

- Conteúdo interno em um campo: **`LIKE`**
  - **`'%Rua%'`**: Qualquer valor que contenha a palavra Rua
  - **`'Rua%'`**: Rua no início
  - **`'_%'`**: Pelo menos um caractere
- No operador **`LIKE`** os caracteres `%` e `_` significam um conjunto qualquer de caracteres e um caractere respectivamente.
- Operador **`LIKE`** possui alto custo de processamento.

Ordenação

- Ordenação de resultados: Qualquer combinação de colunas, mesmo os que não são projetados pela cláusula **`SELECT`**.
- A cláusula **`ORDER BY`** permite indicar uma lista de campos que definem a ordem de exibição dos dados.

```
ORDER BY employee.first_name;
```

- Quando houver mais de um campo, os mesmos serão separados por “,”.
- **`ORDER`** é sempre a última cláusula.

Auto-relacionamento

- Considerando a existência de um campo que indique o chefe de um funcionário (manager_id).
- O campo indica o código do funcionário chefe na mesma tabela.
- Neste caso, para mostrar o nome do funcionário e de seu chefe, a tabela é associada a ela mesma.

```
SELECT emp.employee_first_name, man.employee_first_name

FROM employee emp, employee man

WHERE emp.manager_id = man.employee_id;
```

Auto-relacionamento

- Inicialmente a tabela será renomeada.
- A junção será realizada com as relações **`emp`** e **`man`**.
- Embora seja uma junção com dados da mesma tabela, para a instrução **`FROM`** será um produto cartesiano normal, como se fossem tabelas diferentes.

União

- Operação equivalente ao operador π da álgebra relacional.
- De acordo com as características deste operador:
  - As tabelas ou relações resultantes de uma consulta devem ter o mesmo número de campos.
  - O domínio da i-ésima coluna da primeira deve igual ao da i-ésima da segunda.

União

- Mostrar nome, código e departamento dos funcionários dos departamentos 1 e 6:

```
SELECT employee_first_name, employee_id, department_id
FROM employee
WHERE department_id = 1
UNION
SELECT employee_first_name, employee_id, department_id
FROM employee
WHERE department_id = 6
ORDER BY department_id;
```

Projeção de dados calculados

- Na operação de projeção, é possível efetuar um cálculo usando valores dos campos.

```
SELECT job_name, job_max_salary * 1.1 AS
job_new_max_salary

FROM job
```

- O salário máximo é acrescido de 10% e o campo é renomeado para **`job_new_max_salary`**.

Projeção de dados calculados

- Campos calculados: é possível utilizar qualquer operador aritmético (*, /, -,+).
- Mais de um campo pode ser usado na expressão de cálculo.
- Na projeção podem aparecer textos fixos, por exemplo **`SELECT employee_first_name AS "Nome"`**
- Algumas funções podem ser aplicadas a valores unitários de campos, por exemplo **`UPPER (name)`**.

Leitura recomendada

- Dump e Backup, disponível em:

http://savepoint.blog.br/dump-nao-e-backup/

http://www.postgresql.org/docs/8.1/static/backup.html

Exercícios

Ver Lista 2