Unidade 3

Bancos de Dados Relacionais e Linguagem SQL

3.2 Junções e Funções de Grupo

Junções Externas

- Observamos então possibilidades de recuperação de dados tendo em vista dados que correspondem à condição de junção.
- Existem situações onde necessita-se recuperar os dados correspondentes à condição de junção e os que não são correspondentes.
- O ANSI-99 SQL nos fornece o recurso das junções externas (conforme vimos na álgebra relacional) para solucionar este problema.

Junções Externas

- A operação que realiza a junção interna é o INNER JOIN, que une as tabelas e retorna somente as linhas correspondentes.
- OUTER JOIN é uma operação que representa a junção externa e retorna as linhas correspondentes e as não correspondentes.
- A ordem da relação (tabelas) apresentada na cláusula FROM está diretamente relacionada.
- Serão consideradas as tabelas citadas à direita e à esquerda a cláusula.

Junções Externas - LEFT OUTER JOIN

```
USE employees;

SELECT e.last_name, e.gender, de.dept_no
FROM employees e LEFT OUTER JOIN  dept_manager de
     ON (e.emp_no=de.emp_no)
WHERE e.gender = 'F';

INSERT INTO departments(dept_no,dept_name) VALUES ('d010', 'IT Governance'),('d011', 'Presidence'),('d012', 'FINOPS');

-- Trago o registro da tabela da esquerda mesmo que não exista a correspondência na tabela da direita!
SELECT dt.dept_name,de.from_date, de.to_date
FROM    departments dt LEFT OUTER JOIN  dept_manager  de
        ON (dt.dept_no=de.dept_no);

SELECT dt.dept_name,de.from_date, de.to_date
FROM    departments dt LEFT OUTER JOIN  dept_manager  de
        ON (dt.dept_no=de.dept_no)
ORDER BY de.from_date DESC;
```

Junções Externas - RIGHT OUTER JOIN

```
SELECT  e.first_name, dt.dept_name
FROM    employees e INNER JOIN  dept_emp de
        ON (e.emp_no=de.emp_no)
        RIGHT OUTER JOIN departments dt
        ON (dt.dept_no=de.dept_no)
ORDER BY de.dept_no ASC;

USE sakila;

INSERT INTO category (name) VALUES ('FINANCIAL'),('RESEARCH'),('IT SERIES'),('DIFF');

SELECT *
FROM film_category fc RIGHT JOIN  category c
    ON (c.category_id = fc.category_id)
```

Junções Externas - FULL OUTER JOIN

- Você poderá observar em algumas distribuições de SGBDs que encontramos também o FULL OUTER JOIN.
- No caso do MYSQL utiliza-se o UNION que veremos avançando no nosso curso.

Funções NULL - NULLIF

- As vezes precisamos comparar alguns valores para compor as tabelas e no retorno podemos substituir os valores.
- A função NULLIF compara duas expressões e, caso elas sejam iguais, a função retorna um valor nulo.
- Se forem diferentes, a função retorna a primeira expressão;

```
SELECT *, nullif(address_id, 4)
FROM staff;
```

```
-- Expressão NULLIF compara e retorna valor nulo caso o resultado da comparação seja verdadeiro.
SELECT first_name, LENGTH(first_name) AS "Tamanho FN", last_name,
LENGTH(last_name) AS "Tamanho LN", NULLIF(LENGTH(first_name),
LENGTH(last_name)) AS "Comparação de tamanho"
FROM actor;
```

Funções NULL - COALESCE

- A tradução da palavra COALESCE é unir e este é o objetivo desta função.
- Se a primeira expressão for nula, a função pesquisa a lista de parâmetros até encontrar uma expressão não nula.
- Bastante utilizada para comparar cadastros e se temos os dados disponíveis para atender a algum requisito.

```
SELECT COALESCE("DADO 1",NULL, "Faltam dados")
FROM DUAL;
```

```
SELECT COALESCE(NULL, "DADO 2", "Faltam dados")
FROM DUAL;
```

```
SELECT COALESCE(NULL, NULL, "Faltam dados")
FROM DUAL;
```

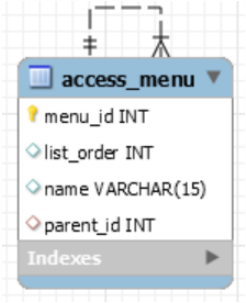
Expressões de condição - CASE

- A expressão de condição mais utilizada e a CASE pois pertence ao conjunto ofertado no padrão ANSI/ISO SQL 99.
- CASE te possibilita fazer a comparação observando-se individualmente os campos e entregar resultados específicos para os usuários.

```
SELECT amount,
(CASE
WHEN amount <= 2 THEN "Volume de pagamento baixo"
WHEN amount <= 4 THEN "Volume de pagamento Medio"
WHEN amount > 5 THEN "Cliente Preferencial"
ELSE "Volume de pagamento alto"
END) as "Volume Locações"
FROM payment
WHERE amount > 5;
```

AUTOJUNÇÃO E CONSULTA HIERÁRQUICA

- Na modelagem de dados, às vezes será necessário mostrar uma entidade com uma relação consigo mesma, ou seja, auto relacionamento.
- Em uma organização de menu por exemplo, um item de menu pode estar vinculado a outro, dependendo de sua disposição em uma página.
- Na modelagem esta é a representação do auto relacionamento (*self join*).

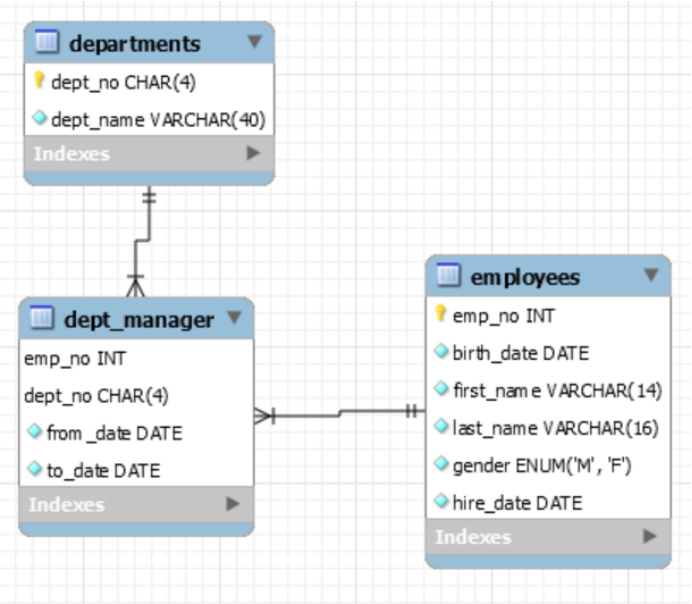


AUTOJUNÇÃO E CONSULTA HIERÁRQUICA

- Na consulta hierárquica com auto relacionamento, deve-se utilizar dois alias para a mesma tabela.
- Na prática serão tratados como duas tabelas diferentes.

```
-- Fazendo a consulta hierárquica com auto relacionamento
SELECT m.name as "Menu pai", f.name  as "Filhote"
FROM access_menu m, access_menu f
WHERE m.menu_id = f.parent_id;
```

AUTOJUNÇÃO E CONSULTA HIERÁRQUICA



- A consulta hierárquica, pode ser tratada também com uma relação NxN.
- Nos dois casos a utilização da junção para montagem do resultado será obrigatória.
- Precisamos observar simplificação do modelo juntamente com a operação dos dados.

AUTOJUNÇÃO E CONSULTA HIERÁRQUICA

- Podemos através destas consultas podemos criar um organograma mostrando a estrutura de uma empresa ou departamento, por exemplo.
- Outro bom exemplo é uma árvore genealógica, que pode ter diversas ramificações em níveis superiores ou inferiores dado um certo ponto de vista.
- Utilizando-se consultas hierárquicas, pode-se recuperar dados com base na relação hierárquica natural entre as linhas de uma tabela.

AUTOJUNÇÃO E CONSULTA HIERÁRQUICA

- Um banco de dados relacional não armazena registros de modo hierárquico e quando existe uma relação hierárquica entre as linhas de uma única tabela, um processo chamado de *tree walking* permite que a hierarquia seja construída
- Uma consulta hierárquica é um método para reportar as ramificações de uma árvore em uma ordem específica.

```
-- Este código não funciona no MySQL mas vejam como seria bem mais simples
SELECT m.id, m.name FROM access_menu m
CONNECT BY prioir id = menu_id START WITH parentId is null
```

AUTOJUNÇÃO E CONSULTA HIERÁRQUICA

- Através das junções conseguimos montar todas uma resposta hierárquica porém sem formatação. É possível inclusive utilizarmos a mesma tabela mais de uma vez na junção para ampliar o retorno de dados diminuindo o número de tratativas a serem feitas para apresentar a resposta.

```
SELECT  CONCAT(e.first_name," é gerenciado por ", em.first_name)
FROM employees e INNER JOIN dept_emp de ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN dept_manager dm ON (d.dept_no=dm.dept_no)
                 INNER JOIN employees em ON (em.emp_no=dm.emp_no)
WHERE de.to_date = '9999-01-01' and dm.to_date = '9999-01-01'
AND d.dept_no='d009'
ORDER BY e.first_name;
```

```
SELECT LPAD(e.last_name, LENGTH(em.last_name)+(2*2)-2,'_') AS "Org_Chart"
FROM employees e INNER JOIN dept_emp de ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN dept_manager dm ON (d.dept_no=dm.dept_no)
                 INNER JOIN employees em ON (em.emp_no=dm.emp_no)
WHERE de.to_date = '9999-01-01' and dm.to_date = '9999-01-01'
AND d.dept_no='d009'
ORDER BY e.first_name;
```

Equijunção e Não Equijunção

- Às vezes chamada de junção "simples" ou "interna", uma **equijunção** é uma junção de tabelas que combina linhas que tenhamos mesmos valores para as colunas especificadas.
- No ANSI, **equijunção** é equivalente a NATURAL JOIN, JOIN USING, JOIN ON.
- Uma **equijunção** usa o operador de igual a para especificar a condição de junção.
- Quando não há a cláusula WHERE com o operador igual gera-se o produto cartesiano.

Equijunção e Não Equijunção

- As junções que vimos até agora retornam linhas com um valor correspondente em duas tabelas.
- As linhas que não satisfaziam à condição eram simplesmente deixadas de fora.
- Pode ser que seja desejável que todos os dados de uma das tabelas sejam retornados, mesmo sem correspondências na outra tabela.
- A **não-equijunção** nos permite unir duas tabelas mesmo sem possuir a coluna

Equijunção e Não Equijunção

- Como não existem correspondências exatas entre as duas colunas em cada tabela, o operador de igualdade = não pode ser usado.
- Embora condições de comparação como < = e > = possam ser usadas, BETWEEN...AND é uma maneira mais eficaz de executar a não-equijunção.
- A não-equijunção é equivalente ao JOIN ON do ANSI (em que se usa uma condição que não seja "igual a").

Referências bibliográficas

ELMASRI, R., NAVATHE, S. B., Sistemas de Banco de Dados: Fundamentos e Aplicações. 3ª Ed., Editora LTC, 2002.

MANNINO, Michael V. Projeto, Desenvolvimento de Aplicações e Administração de Banco de Dados. 3ª. Ed. Porto Alegre. Bookman. 2008.