Linquagem de consulta a banco de dados

SQL (*Structured Query Language*)

Prof.: Maiquel de Brito

BLU3024
Departamento de Engenharia de Controle, Automação e Computação
UFSC Blumenau

Sumário

Introdução

O modelo relacional (Codd, 1970)

- Baseado em *relações* (ou tabelas)

Customer

cust_id	*fname*	*lname*
1	George	Blake
2	Sue	Smith

Account

account_id	*product_cd*	*cust_id*	*balance*
103	CHK	1	$75.00
104	SAV	1	$250.00
105	CHK	2	$783.64
106	MM	2	$500.00
107	LOC	2	0

Product

product_cd	*name*
CHK	Checking
SAV	Savings
MM	Money market
LOC	Line of credit

Transaction

txn_id	*txn_type_cd*	*account_id*	*amount*	*date*
978	DBT	103	$100.00	2004-01-22
979	CDT	103	$25.00	2004-02-05
980	DBT	104	$250.00	2004-03-09
981	DBT	105	$1000.00	2004-03-25
982	CDT	105	$138.50	2004-04-02
983	CDT	105	$77.86	2004-04-04
984	DBT	106	$500.00	2004-03-27

Bancos de dados relacionais

Elementos de um banco de dados relacional:

- Entidade: elemento do mundo real que precisa ser persistido no BD
- Coluna: uma peça de dados referente a uma entidade
- Linha: um conjunto de colunas que, juntas, descrevem uma entidade
- Tabela: um conjunto de linhas

Exercício: analisar um banco de dados relacional existente. Identificar entidades, colunas, linhas, tabelas, tipos de dados etc.

Introdução

SQL (Structured Query Language)

- Desenvolvida no início dos anos 1970 pela IBM
- Padrão publicado pela ANSI em 1970
- Utilizada para manipular bancos de dados relacionais
- Linguagem não-procedural e não-imperativa

SQL I - DML

SQL - Select

select: seleciona colunas de uma tabela

sintaxe: `select <colunas> from <tabela>`

$\Pi_{Nome,Genero} Filme \equiv$ `select Nome,Genero from Filme;`

SQL - Select

Para selecionar todos os campos de uma tabela: usar `*`:

```
select * from Filme;
```

SQL - Select

É possível *renomear* as colunas que aparecem no resultado da seleção:

```
select idFilme as Codigo, Nome,
Pais_Producao as Nacionalidade,
Ano_Producao as Data
from Filme;
```

SQL - Select

A cláusula **where**: utilizada para filtrar as linhas a serem selecionadas

Usualmente, não se espera selecionar todas as linhas de uma tabela

```
select Nome, Data_Nascimento from Ator
where Data_Nascimento <= '1965-12-31'
```

$$\equiv$$

$$\Pi_{\text{Nome, Data\_Nascimento}}(\sigma_{\text{Data\_Nascimento} \leq 1965-12-31} Ator)$$

SQL - Select

A cláusula **where**: admite operadores <, <= ($\leq$), >, >= ($\geq$), = e <> ($\neq$), além dos conectivos lógicos `and` ($\wedge$), `or` ($\vee$) e `not` ($\neg$)
Exercícios:

1. Um usuário de um serviço de *streamming* considera que os bons filmes são os franceses, suspenses, as ficções lançadas após o ano 2010 e os dramas e aventuras anteriores ao ano 1995. Elabore uma consulta que recomende nomes de filmes para esse usuário.

SQL - Select

A cláusula **from** admite referenciar mais de uma tabela

```
select a.*, p.* from Ator a, Pais p;
```

$$\equiv$$

$$Ator \times Pais$$

SQL - Select

A cláusula **from** admite referenciar mais de uma tabela

```
select a.*, p.* from Ator a, Pais p
where a.idPais=p.idPais
```

$$\equiv$$

$$Ator \bowtie_{Ator.idPais=Pais.idPais} Pais$$

SQL - Select

A cláusula **from** admite referenciar mais de uma tabela

```
select a.*, p.* from Ator a, Pais p
where a.idPais=p.idPais
```

$\equiv$

```
select a.*, p.* from Ator a
join Pais p on a.idPais=p.idPais
```

$\equiv$

$$Ator \bowtie_{Ator.idPais=Pais.idPais} Pais$$

SQL - Select

Exercícios:

1. Listar os nomes dos filmes e os seus respectivos diretores
2. Listar os nomes dos diretores que dirigiram filmes na década de 1990.

SQL - Select

A cláusula **like**: permite selecionar tuplas utilizando um trecho de algum atributo.

Ex.: Selecionar os filmes cujos nomes começem com “A” ou terminem com “S”:

```
select * from Filme where Nome like "A%"or Nome Like "%s"
```

Exercício:
Selecionar todos os filmes cujo nome contenha ditongo aberto.

SQL - Select

A cláusula **order by** permite ordenar os resulados de uma consulta
Ex.: Selecionar todos os dados dos diretores por ordem alfabética de nome

```
select * from Diretor order by Nome
```

Exercícios:

1. Listar todos os atores em ordem alfabética de nome
2. Listar todos os atores, ordenados do mais novo para o mais velho
3. Listar todos os atores, ordenados do mais velho para o mais novo
4. Listar todos os filmes de aventura, ordenados do mais antigo para o mais recente

SQL - Select

Exercício:

Selecionar o código e o nome de todos os atores e de todos os diretores.

SQL - Select

A cláusula **union** realiza a operação de *união* ($\cup$) entre duas relações (semelhante à união da álgebra relacional e da teoria dos conjuntos)

Linhas duplicadas são removidas (idempotência)

diferente do que aconece com o `select`

Para manter linhas duplicadas, utiliza-se **union all**

A união só acontece entre relações com o mesmo número de colunas

SQL - Select

Exercício:
Selecionar o código e o nome de todos os atores e de todos os diretores.

```
select idAtor, Nome from Ator
union
select idDiretor , Nome from Diretor
```

SQL - Select

Exercício:
Listar o nome dos diretores e o nome dos filmes que eles dirigiram. O nome dos diretores que não dirigiram filme algum também deve aparecer listado.

SQL - Select

A cláusa **left outer join** executa o produto cartesianno entre duas relações, selecionando as tuplas que atendem à cláusula *where* e selecionando também as tuplas da relação que aparece antes (ou à esquerda) do *join* que não atendem ao *where*

SQL - Select

Exercício:
Listar o nome dos diretores e o nome dos filmes que eles dirigiram. O nome dos diretores que não dirigiram filme algum também deve aparecer listado.

```
select Diretor.Nome, Filme.Nome from Diretor
left outer join Filme on Filme.Diretor = Diretor.idDiretor
```

$$\equiv$$

$$\Pi_{Diretor.Nome, Filme.Nome}(Diretor \ ⟕_{(Filme.Diretor = Diretor.idDiretor)} \ Filme)$$

SQL - Select

A cláusa **right outer join** executa o produto cartesianno entre duas relações, selecionando as tuplas que atendem à cláusula *where* e selecionando também as tuplas da relação que aparece depois (ou à direita) do *join* que não atendem ao *where*.

SQL - Select

Exercício:
Listar o nome dos diretores e o nome dos filmes que eles dirigiram. O nome dos filmes que não têm diretor definido também deve aparecer listado.

```
select Diretor.Nome, Filme.Nome from Diretor
  right outer join Filme on Filme.Diretor =
                  Diretor.idDiretor
```

$$\equiv$$

$$\Pi_{Diretor.Nome, Filme.Nome}(Diretor \rtimes_{(Filme.Diretor = Diretor.idDiretor)} Filme)$$

SQL - Select

A cláusa **full outer join** executa o produto cartesianno entre duas relações, selecionando as tuplas que atendem à cláusula *where* e selecionando também, de ambas as relações, as tuplas da relação que não atendem ao *where*. Nem todos os SGBDs implementam *full outer join*

Pode ser implementado através do *Union*

SQL - Select

Exercício:
Listar o nome dos diretores e o nome dos filmes que eles dirigiram. O nome dos diretores que não dirigiram filme algum, bem como o nome dos filmes que não têm diretor definido, também devem aparecer listados.

```
select Diretor.Nome, Filme.Nome from Diretor
left outer join Filme on Filme.Diretor=Diretor.idDiretor
union
select Diretor.Nome, Filme.Nome from Diretor
right outer join Filme on Filme.Diretor=Diretor.idDiretor
```

$$\equiv$$

$$\Pi_{Diretor.Nome, Filme.Nome}(Diretor \rtimes_{(Filme.Diretor=Diretor.idDiretor)} Filme)$$
$$\cup$$
$$\Pi_{Diretor.Nome, Filme.Nome}(Diretor \ltimes_{(Filme.Diretor=Diretor.idDiretor)} Filme)$$

$$\equiv$$

$$\Pi_{Diretor.Nome, Filme.Nome}(Diretor \rtimes\ltimes_{(Filme.Diretor=Diretor.idDiretor)} Filme)$$

Funções de Agregação

Funções de Agregação

Funções de Agregação: reunem um conjunto de valores de entrada e retornam um único valor de saída:

- Média: avg
- Mínimo: min
- Máximo: max
- Total: sum
- Contagem: count

Funções de Agregação

Exemplo:

```
select avg (salary)
from instructor
where dept name= ‘Comp.  Sci.’
```

Funções de Agregação

Exemplo:

```
select count(*) from Filme
```

```
select count(*) from Filme
where Genero='Suspense';
```

Agregação com agrupamento

Exemplo:

```
select dept name, avg (salary)
from instructor
group by dept name
```

Filtragem por agrupamento

Exemplo:

```
select dept name, avg (salary)
from instructor
group by dept name
having avg (salary) > 42000;
```

Inserção de dados

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Exemplo

```
INSERT INTO Aluno(Nome, Idade, Telefone)
VALUES ('Bob', 21, '(47)99999-8888');
```

– Link para mais informações sobre *inserção*.

Alteração de dados

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Exemplo

```
UPDATE Aluno
SET Nome = 'Bob' WHERE idAluno = 10;
```

– Link para mais informações sobre *update*.

Exclusão de dados

```
DELETE FROM table_name
WHERE condition;
```

Exemplo

```
DELETE FROM Aluno
WHERE idAluno = 10;
```

– Link para mais informações sobre *delete*.

DML

Criação de Schema

```
create database nome_do_schema;
```

ex.: `create database universidade;`

Criação de tabelas

```
CREATE TABLE nome_da_tabela (
  coluna1 tipo_de_dado,
  coluna2 tipo_de_dado,
  coluna3 tipo_de_dado,
  ....
);
```

Exemplo

```
create table Endereco(
   idEndereco INT PRIMARY KEY,
   Rua VARCHAR(100),
   Bairro VARCHAR(30),
   Cidade VARCHAR(50)
);
```

– Link para mais informações sobre *create table*.
– Link para mais informações sobre *tipos de dados*.

Modificação de tabelas

```
ALTER TABLE nome_da_tabela (
    ADD nova_coluna tipo_de_dado,
    MODIFY coluna2 tipo_de_dado,
    DROP coluna3
   ....
);
```

Exemplo

```
alter table Aluno(
   ADD Data_Nascimento Date,
   MODIFY Nome VARCHAR(120),
   DROP Idade
);
```

– Link para mais informações sobre *alter table*.

Exclusão de tabelas

```
DROP TABLE nome_da_tabela;
```

Exemplo

```
DROP TABLE Aluno;
```

– Link para mais informações sobre *drop table*.