Banco de Dados

Introdução a SQL

Fagner Leal – pantoja.ti@gmail.com

Baseado nos slides de Jaudete Daltio e André Santanchè

Structured Query Language

- Criada pela IBM research (início dos anos 70)
- Linguagem declarativa para manipulação e recuperação de dados
- Linguagem padrão para os SGBDs relacionais
- Versão estável: SQL-99

Structured Query Language

- **Dividida em 4 módulos:**
  - Linguagem de Definição de Dados (DDL)
    - Definir esquemas de relação, excluir relações e modificar esquemas
  - Linguagem de Manipulação de Dados (DML)
    - Inserir, excluir e modificar dados e linguagem de consulta
    - A linguagem de consulta é inspirada em Álgebra Relacional
  - Linguagem de Controle de Dados (DCL)
    - Gerenciar aspectos de controle de acesso entre usuários e dados
  - Linguagem de Transação de Dados(DTL)
    - Gerenciar aspectos de transações

Definição de Dados

Definição de Dados (DDL)

- **Objetos**
  - Esquemas (Banco de dados)
  - Tabela (Relação)
  - Visões(views)
  - Asserções
  - Gatilhos (triggers)
- **Paralelos com Modelo Relacional**
  - Tabela = Relação
  - Linha = Tupla
  - Coluna = Atributo

Definição de Dados (DDL)

- **CREATE**
  - Cria um objeto dentro da base de dados
- **ALTER**
  - Altera um objeto já existente
- **DROP**
  - Apaga um objeto do banco de dados

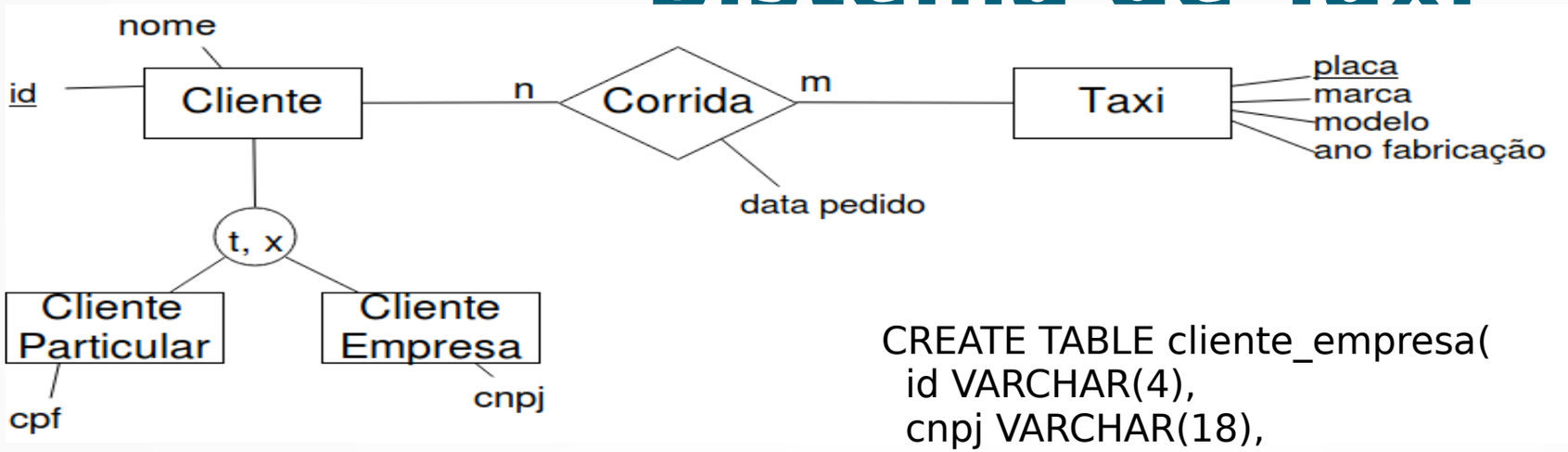
Create

```
CREATE DATABASE <nome_do_banco>
```

```
CREATE TABLE <tabela> (
    <campo1> <tipo>,
      [..., <campon> <tipo>]
    PRIMARY KEY <coluna>
    FOREIGN KEY <coluna> REFERENCES <tabela_ref>(<coluna_ref>)
    )
```

- PRIMARY KEY: Restrição de chave primária
- FOREIGN KEY**:** Restrição de chave estrangeira
- Tipos de domínios básicos: char(n), varchar(n), int, real, double, float, boolean, date, etc.
- Outras restrições: NOT NULL, UNIQUE, CHECK

Sistema de Taxi



Script para criar banco sistema_taxi

```
CREATE DATABASE sistema_taxi
USE sistema_taxi

CREATE TABLE cliente(
  id VARCHAR(4),
  nome VARCHAR(80),
  PRIMARY KEY(id)
);
CREATE TABLE cliente_particular(
  id VARCHAR(4),
  cpf VARCHAR(14),
  PRIMARY KEY(id),
  FOREIGN KEY(id) REFERENCES cliente(id)
);
CREATE TABLE cliente_empresa(
  id VARCHAR(4),
  cnpj VARCHAR(18),
  PRIMARY KEY(id),
  FOREIGN KEY(id) REFERENCES cliente(id)
);
CREATE TABLE taxi (
  placa VARCHAR(7),
  marca VARCHAR(30),
  modelo VARCHAR(30),
  anofab INTEGER,
  PRIMARY KEY(placa)
);
CREATE TABLE corrida (
  cliid VARCHAR(4),
  placa VARCHAR(7),
  dataPedido DATE,
  PRIMARY KEY(cliid, placa, dataPedido),
  FOREIGN KEY(cliid) REFERENCES cliente(id),
  FOREIGN KEY(placa) REFERENCES taxi(placa)
);
```

Create Table

- **Script para criação do banco Sistema de Taxi**

```
CREATE TABLE cliente(
  id VARCHAR(4),
  nome VARCHAR(80),
  PRIMARY KEY(id)
);
CREATE TABLE cliente_particular(
  id VARCHAR(4),
  cpf VARCHAR(14),
  PRIMARY KEY(id),
  FOREIGN KEY(id) REFERENCES cliente(id)
);
CREATE TABLE cliente_empresa(
  id VARCHAR(4),
  cnpj VARCHAR(18),
  PRIMARY KEY(id),
  FOREIGN KEY(id) REFERENCES cliente(id)
);
```

```
CREATE TABLE taxi (
  placa VARCHAR(7),
  marca VARCHAR(30),
  modelo VARCHAR(30),
  anofab INTEGER,
  PRIMARY KEY(placa)
);
CREATE TABLE corrida (
  cliid VARCHAR(4),
  placa VARCHAR(7),
  dataPedido DATE,
  PRIMARY KEY(cliid, placa, dataPedido),
  FOREIGN KEY(cliid) REFERENCES cliente(id),
  FOREIGN KEY(placa) REFERENCES taxi(placa)
);
```

Alter e Drop Table

ALTER TABLE <tabela> ADD <coluna><tipo>

ALTER TABLE <tabela> DROP <coluna>

- Adicionar/remover nova coluna
- Exemplos

ALTER TABLE cliente_particular ADD email varchar(255)

ALTER TABLE cliente_particular DROP email

DROP TABLE <tabela>

- Exluir tabela existente

Manipulação de Dados

Manipulação de Dados (DML)

- Inserir, atualizar ou remover registros
  - **INSERT**
  - **UPDATE**
  - **DELETE**

- Realizar consulta
  - **SELECT**

Insert

- **Insere linhas (tuplas) numa relação**
- **Insere todos os atributos da linha, na mesma ordem em que foI especificado no CREATE TABLE**

INSERT INTO <tabela>
VALUES (valor$_1$, valor$_2$, …, v**alor$_n$)**

- **Insere somente os atributos especificados:**

INSERT INTO <tabela> (campo$_1$, campo$_2$, …, campo$_n$)
VALUES (valor$_1$, valor$_2$, …, valor$_n$)

Insert

- **Script para popular banco Sistema de Taxi**

```
INSERT INTO cliente VALUES
    ('1755', 'Doriana'),
    ('93','DinoTech'),
    ('1532','Asdrúbal'),
    ('1780','Quincas'), ('97','Proj');
INSERT INTO cliente_particular VALUES
    ('1755', '567.387.387-44'),
    ('1532', '448.754.253-44'),
     ('1780', '576.456.123-55');
INSERT INTO cliente_empresa VALUES ('93', '58.443.828/0001-02'), ('97', '44.876.234/7789-10');
INSERT INTO taxi VALUES
    ('DAE6534','Ford','Fiesta',1999),
    ('DKL4598','Wolksvagen','Gol',2001),
    ('DKL7878','Ford','Fiesta',2001),
    ('JDM8776','Wolksvagen','Santana',2002),
    ('JJM3692','Chevrolet','Corsa',1999);
INSERT INTO corrida VALUES ('1755', 'DAE6534', '2003-02-15'), ('97', 'JDM8776', '2003-02-18');
```

Update

- Modifica os valores dos atributos das linhas da tabela em que o predicado especificado seja verdadeiro

UPDATE <tabela>

SET <campo$_1$> = <valor$_1$> [, …, <campo$_n$> <valor$_n$>]

WHERE <condição>

- Exemplo:

  UPDATE cliente SET nome = ‘Doris’ WHERE id = ‘93’

- Uma atualização no valor da chave primária pode propagar-se dependendo de como a restrição de chave estrangeira foi criada

Delete

- **DELETE FROM <tabela> WHERE <condição>**

- Exemplo:

  DELETE FROM corrida WHERE placa = ‘DAE0534’
- Exclui todos os registros da tabela em que o predicado especificado seja verdadeiro
- A exclusão não pode violar as restrições de integridade referencial (chave estrangeira)
- Alguns SGBDs permitem exclusões em cascata

Select

- Consulta os dados presentes no banco
- Estrtura básica:

```
SELECT <lista de atributos>]
FROM <lista de tabeas>]
WHERE <condição>
```

- Lista de atributos: nomes dos atributos a serem recuperados pela conulsta
  - Quando a lista de atributos envolver todos os atributos da relação, pode-se usar *
- Lista de tabelas: nomes das tabelas envolvidas no processamento da consulta
  - Mais de uma tabela -> produto cartesiano ou junção

Select

SELECT <lista de atributos>]
FROM <lista de tabeas>]
WHERE <condição>

- Condição: expressão booleana que identifica as linhas a serem recuperadas pela consulta
  - pode conter:
    - Conectivos logicos: AND, OR, NOT
    - Operadores de comparação: < ,<=, > ,>= , = , <>
    - Comparador de string: LIKE. Usado de duas maneiras:
      - LIKE ‘%<parte da string>%’
      - LIKE ‘_ _ _ <parte da string> _ _ _’

Select x Álgebra Relacional

Uma consulta típica:

SELECT $A1,A2,A3,\ldots,A_n$

FROM $R1,R2,R3,\ldots,R_n$

WHERE P

é equivalente em álgebra relacional a:

$$\Pi_{A1,A2,A3,\ldots,An} (\sigma_P (R1 \times R2 \times R3 \times \ldots \times R_n))$$

- Diferentemente da Álgebra, o SELECT não elimina repetições do resultado. É necessário forçar usando a palavra-chave DISTINCT. Exemplo:
  - SELECT DISTINCT <atributtos> FROM <ltabelas>

Select - Projeçao

Selecionar as marcas e modelos de táxi

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Select - Projeçao

SELECT marca, modelo FROM taxi

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Select - Projeçao

SELECT marca, modelo FROM taxi

Marca	Modelo
Ford	Fiesta
Wolksvagen	Gol
Ford	Fiesta
Wolksvagen	Santana
Chevrolet	Corsa

Select - Seleção

Selecionar os táxis fabricados após o ano 2000

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Select - Seleção

SELECT * FROM Taxi WHERE anofab > 2000

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Select - Seleção

SELECT * FROM taxi WHERE anofab > 2000

Placa	Marca	Modelo	AnoFab
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002

Select - Seleção

SELECT * FROM taxi WHERE anofab > 2000

Placa	Marca	Modelo	AnoFab
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002

Alias

- Uso de alias permite associar nomes alternativos tara tabelas e colunas
- Palavra-chave: **AS**
- Exemplo:
  - SELECT anofab AS fabri FROM taxi AS carro

Produto Cartesiano

```
SELECT *
FROM <tabela1> <tabela2>
```

- Não há associação de atributo da <tabela$_1$> com atributo da <tabela$_2$>
- Não há condição que ligue tabelas

Junção

SELECT ...
FROM <tabela$_1$> <tabela$_2$>
WHERE <tabela$_1$> <campo> = <tabela$_2$> <tabela$_2$>

- Condiçao de ligação entre as tabelas:
  <tabela$_1$> <campo> = <tabela$_2$> <tabela$_2$>

Ordenação

```
SELECT
....
 ORDER BY <lista de atributos> [ASC | DESC]
```

- Ordena a exibição dos registros
- ASC (crescente) | DESC (decrescente)
- Ordenação default: ASC

Funções Agregadas

```
SELECT <função de agregação>(<coluna>)
FROM ...
WHERE ...
```

Função de agregação pode ser:

- COUNT
- SUM
- AVG
- MAX
- MIN

Agrupamento: Group by

```
SELECT
....
GROUP BY <campo1>
```

- Agrupa linhas da tabela que compartilham os mesmo valores em todas as colunas da lista
- Exemplo:
  SELECT marca, count(*) FROM taxi group by marca;
- Resultado:

#	marca	count(*)
1	Chevrolet	1
2	Ford	2
3	Wolksvagen	2

Agrupamento: Having

```
SELECT …
GROUP BY <coluna_agrupar>
HAVING <condição_grupo>
```

- Restring os resultados do GROUP BY quando a condição é verdadeira
- Exemplo:
  SELECT marca FROM taxi GROUP BY marca HAVING count(*)>1;

- Resultado:

#	marca
1	Ford
2	Wolksvagen

RESUMO

```
SELECT <lista de colunas>
FROM <lista de tabelas>
[WHERE <condição>]
[GROUP BY <coluna_agrupar>]
[HAVING <condição_grupo>]
[ORDER BY <lista de atributos>]
```

- Apenas as cláusulas SELECT e FROM são obrigatórias
- Quando existentes, as cláusulas devem aparecer nessa ordem