Bancos (Bases) de Dados

Aula #8 – SQL

Prof. Eduardo R. Hruschka

* Slides baseados no material elaborado pelas professoras:
Cristina D. A. Ciferri
Elaine P. M. de Souza

SQL (*Structured Query Language*)

- Linguagem relacional;
- Exemplos de SGBD que utilizam SQL:
  - Oracle
  - Informix
  - Ingress
  - SQL Server
  - Interbase
  - SyBase
  - DB2
  - MySQL
  - PostgreSQL

Composição da SQL

- Linguagem de Definição dos Dados
  - *DDL*;
  - comandos para definir, modificar e remover relações (**tabelas**), além de criar e remover índices.
- Linguagem Interativa de Manipulação dos Dados
  - *DML*;
  - comandos para consultar, inserir, remover e modificar **tuplas**.

Composição da SQL

- DML embutida
  - pode ser utilizada a partir de linguagens de programação de propósito geral
- Definição de visões
  - SQL DDL inclui comandos para a criação e a remoção de visões
- Restrições de integridade
  - SQL DDL possui comandos para a especificação de restrições de integridade

Composição da SQL

- Autorização
  - SQL DDL inclui comandos para a especificação de direitos de acesso a relações e visões
- Gerenciamento de transações
  - introduz comandos para a especificação do início e do fim das transações
- Recuperação de falhas
  - introduz comandos para utilização do arquivo de *log*

SQL DDL

- CREATE DATABASE | SCHEMA
  - cria um esquema de BD relacional
- DROP DATABASE | SCHEMA
  - remove um esquema de BD relacional

CREATE DATABASE

```
CREATE {DATABASE | SCHEMA} nome
        [USER `username` [PASSWORD `password`] ]
         ... ;
```

- Cria um esquema de BD relacional
  - agrupa tabelas/comandos que pertencem à aplicação
  - identifica o proprietário do esquema
  - esquema inicial não possui tabelas/dados

DROP DATABASE

```
DROP {DATABASE | SCHEMA} nome
[CASCADE | RESTRICT] ;
```

- Remove um esquema de BD relacional
  - tabelas/dados
  - índices
  - arquivos de log

  } quaisquer elementos associados
- Usuários autorizados
  - proprietário do banco de dados
  - DBA

DROP DATABASE

- CASCADE
  - remove um esquema de BD, incluindo todas as suas tabelas e os seus outros elementos
- RESTRICT
  - remove um esquema de BD somente se não existirem elementos definidos para esse esquema

SQL DDL

- CREATE TABLE
  - cria uma nova tabela (relação) no BD
  - nova tabela não possui dados
- DROP TABLE
  - remove uma tabela e sua instância do BD
- ALTER TABLE
  - altera a estrutura de uma tabela já existente no BD

CREATE TABLE

```
CREATE TABLE nome_tabela ( A1  D1  R1,
                           A2  D2  R2,
                           ...
                           An  Dn  Rn ) ;
```

- Cria uma nova tabela (relação)
- Cria os atributos da nova tabela, com
  - nome do atributo: A_i ($1 \leq i \leq n$)
  - tipo de dado (domínio do atributo): D_i
  - restrições que atuam no atributo: R_i

Exemplos de Tipos de Dados

- Numéricos
  - Integer, float, ...
- Hora/Data
  - Date (YYYY-MM-DD), time (HH-MM-SS), ...
- Strings
- Etc.

Restrições de Integridade

- Valor nulo
  - representado por NULL
  - membro de todos os domínios
- Restrição NOT NULL
  - especificada quando NULL não é permitido
  - proíbe que o atributo receba valor nulo

Restrições de Integridade

- Cláusula PRIMARY KEY
  - identifica os atributos que formam a chave primária
    - NOT NULL
  - sintaxe

    PRIMARY KEY ($atributo_1$, $atributo_2$, ..., $atributo_X$)
- Cláusula UNIQUE
  - não permite valores duplicados para um atributo

Restrições de Integridade

- Cláusula DEFAULT
  - associa um valor *default* para um atributo, caso nenhum outro valor seja especificado
- Cláusula CHECK
  - especifica um predicado que precisa ser satisfeito por todas as tuplas de uma relação
  - exemplos
    - saldo int CHECK (saldo >= 0)
    - nível char(15) CHECK (nível IN `Bacharelado`,`Mestrado`, `Doutorado`))

Restrições de Integridade

- Integridade referencial
  - dependência existente entre a chave estrangeira de uma relação $\mathbf{R_1}$ (referência) e a chave primária da relação referenciada $\mathbf{R_2}$ (relação referida).
  - problemas
    - atualização ou exclusão de elementos da chave primária sem fazer um ajuste coordenado nas chaves estrangeiras:
      - Excluir tupla de **Empregado** que é gerente de **Departamento**.
    - inclusão ou alteração de valores não nulos na chave estrangeira de $\mathbf{R_1}$ que não existam na chave primária de $\mathbf{R_2}$

Restrições de Integridade

- Cláusula FOREIGN KEY
  - características
    - elimina a possibilidade de violação da integridade referencial
    - reflete nas chaves estrangeiras todas as alterações na chave primária
  - sintaxe

    FOREIGN KEY (atributos)

    REFERENCES nome_relação (atributos)

    [ON UPDATE [NO ACTION | CASCADE | SET NULL | SET DEFAULT]]

    [ON DELETE [NO ACTION | CASCADE | SET NULL | SET DEFAULT]]

DROP TABLE

```
DROP TABLE nome_tabela ;
```

- Remove uma tabela existente do BD
  - dados
  - índices, etc.
- Usuários autorizados
  - proprietário do banco de dados
  - DBA

ALTER TABLE

```
ALTER TABLE nome_tabela;
```

- Altera o esquema de uma tabela do BD
  - adiciona
  - remove
  - altera

} colunas ou restrições de integridade

Exemplos: ALTER TABLE

ALTER TABLE nome_tabela
ADD (A_1 D_1 R_1),
...
ADD (A_n D_n R_n)

- inclui novas colunas na tabela

ALTER TABLE nome_tabela DROP A_1

- elimina uma coluna já existente da tabela

Exemplos: ALTER TABLE

ALTER TABLE nome_tabela ALTER [COLUMN] A_1 TO A_2

- modifica o nome de uma coluna existente de A_1 para A_2

ALTER TABLE nome_tabela
ALTER [COLUMN] A_1 TYPE INT

– modifica o tipo de dado de uma coluna

SQL DDL

- CREATE DOMAIN
  - cria um domínio para um tipo de dados
- DROP DOMAIN
  - remove um domínio existente do BD
- ALTER DOMAIN
  - altera a definição de domínio

CREATE DOMAIN

```
CREATE DOMAIN nome_domínio [AS] tipo_dado
        [DEFAULT ... ]
        [NOT NULL]
        [CHECK ...]
         ... ;
```

- Cria um domínio para um tipo de dados
  - restrições de integridade
- Característica
  - a definição do domínio é global ao BD

DROP DOMAIN

```
DROP DOMAIN nome_domínio ;
```

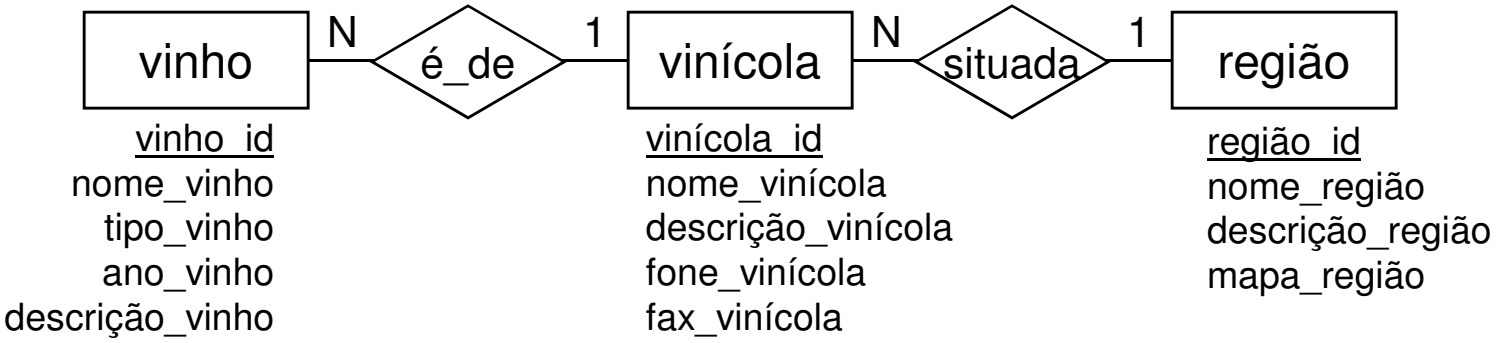
- Remove um domínio existente do BD
  - falha caso o domínio esteja definindo o tipo de dado de alguma coluna
- Usuários autorizados
  - proprietário do banco de dados
  - DBA

ALTER DOMAIN

```
ALTER DOMAIN nome_domínio... ;
```

- Altera um domínio existente do BD
  - remove ou define restrições de integridade

Exemplo



- região (<u>região_id</u>, nome_região, mapa_região, descrição_região)
- vinícola (<u>vinícola_id</u>, nome_vinícola, descrição_vinícola, fone_vinícola, fax_vinícola, região_id)
- vinho (<u>vinho_id</u>, nome_vinho, tipo_vinho, ano_vinho, descrição_vinho, vinícola_id)

Exemplo ...

```
CREATE DATABASE loja_vinhos;

CREATE TABLE região
(
   região_id int NOT NULL,
   nome_região varchar(100)  NOT NULL,
   mapa_região blob,
   descrição_região blob,
   PRIMARY KEY (região_id),
);
```

BLOB: *Binary Long Objects* (para armazenar áudio e vídeo)

Exemplo

```
CREATE TABLE vinícola
(
   vinícola_id int NOT NULL,
   nome_vinícola varchar(100) NOT NULL,
   descrição_vinícola blob,
   fone_vinícola varchar(15),
   fax_vinícola varchar(15),
   região_id int DEFAULT ′0′ NOT NULL,
   PRIMARY KEY (vinícola_id),
   FOREIGN KEY (região_id)
      REFERENCES região (região_id)
      ON UPDATE SET DEFAULT,
      ON DELETE SET DEFAULT,
);
```

Exemplo

```
CREATE TABLE vinho
(
   vinho_id int  NOT NULL,
   nome_vinho varchar(50)  DEFAULT ´´  NOT NULL,
   tipo_vinho varchar(10)  DEFAULT ´´  NOT NULL,
   ano_vinho int   DEFAULT ´0´  NOT NULL,
   descrição_vinho blob,
   vinícola_id int  DEFAULT ´0´  NOT NULL,
   PRIMARY KEY (vinho_id),
   FOREIGN KEY (vinícola_id)
      REFERENCES vinícola (vinícola_id),
      ON UPDATE CASCADE
      ON DELETE CASCADE,
);
```

SQL DML

- SELECT ... FROM ... WHERE ...
  - lista atributos de uma ou mais tabelas de acordo com alguma condição
- INSERT INTO ...
  - insere dados em uma tabela
- DELETE FROM ... WHERE ...
  - remove dados de tabelas já existentes
- UPDATE ... SET ... WHERE ...
  - altera dados específicos de uma tabela

SELECT

```
SELECT <lista de atributos>
FROM <lista de tabelas>
[ WHERE predicado/condição ]
[ GROUP BY <atributos de agrupamento> ]
[ HAVING <condição para agrupamento> ]
[ ORDER BY <lista de atributos> ] ;
```

SELECT

- Cláusula SELECT (lista de atributos)
  - lista os atributos cujos valores serão recuperados.
- Cláusula FROM (lista de tabelas)
  - especifica as relações necessárias para o processamento da consulta.
- Cláusula WHERE (condição)
  - especifica as condições para a seleção das tuplas a serem recuperadas.
  - pode ser omitida.

Exemplo:

- SELECT datanasc, endereco

  FROM Empregado

  WHERE Pnome=Eduardo AND Unome=Hruška

- Lembrar que, em princípio, SQL permite multiconjuntos.

SELECT

- Resultado de uma consulta
  - ordem de apresentação dos atributos
    - ordem dos atributos na cláusula SELECT
  - ordem de apresentação dos dados (parte das tuplas)
    - ordem ascendente ou descendente de acordo com a cláusula ORDER BY
    - sem ordenação
  - duas ou mais tuplas podem possuir valores idênticos de atributos
    - para eliminação de tuplas duplicadas
      - SELECT DISTINCT

Cláusula WHERE

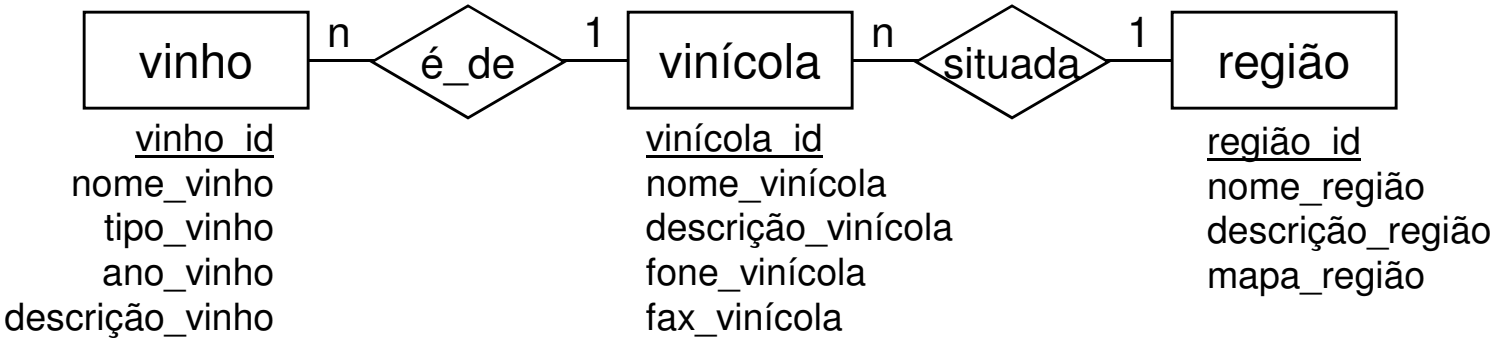
- Operadores de comparação

igual a	=	diferente de	< >
maior que	>	maior ou igual a	>=
menor que	<	menor ou igual a	<=
entre *dois* valores	BETWEEN ... AND	de cadeias de caracteres	LIKE *ou* NOT LIKE

Cláusula WHERE

- Operadores de comparação de cadeias de caracteres
  - % (porcentagem): substitui qualquer *string*
  - _ (*underscore*): substitui qualquer *caractere*
- Característica
  - operadores *sensíveis ao caso*
    - letras maiúsculas são consideradas diferentes de letras minúsculas

Relações Base



- região (<u>região_id</u>, nome_região, mapa_região, descrição_região)
- vinícola (<u>vinícola_id</u>, nome_vinícola, descrição_vinícola, fone_vinícola, fax_vinícola, região_id)
- vinho (<u>vinho_id</u>, nome_vinho, tipo_vinho, ano_vinho, descrição_vinho, vinícola_id)

Cláusula WHERE

- Exemplos
  - WHERE nome_região LIKE ‘Mar%’
    - qualquer string que se inicie com ‘Mar’
  - WHERE nome_região LIKE ‘Mar_’
    - qualquer string de 4 caracteres que se inicie com ‘Mar’

Exemplos

- SELECT *
  FROM região;

- SELECT região_id, nome_região
  FROM região
  WHERE nome_região LIKE ‘M%’ AND
  região_id >= 3 AND
  mapa_região IS NOT NULL;

Operações sobre conjuntos

SQL	Álgebra Relacional
UNION	União
INTERSECT	Intersecção
MINUS	Diferença

- Observações
  - as relações participantes das operações precisam ser *compatíveis.*

Exemplo

- Liste os anos de fabricação dos vinhos tintos e brancos:

```
SELECT ano_vinho
FROM vinho
WHERE tipo_vinho = ‘tinto’
UNION ALL
SELECT ano_vinho
FROM vinho
WHERE tipo_vinho = ‘branco’;
```

Junção

- Usar SELECT e WHERE
  - especificam atributos com mesmo nome usando nomes de tabelas e atributos (nome_tabela.nome_atributo)
- Cláusula FROM
  - possui mais do que uma tabela
- Cláusula WHERE
  - inclui as condições de junção

Exemplos

- SELECT nome_vinícola, nome_região
  FROM vinícola, região
  WHERE vinícola.região_id = região.região_id;

- SELECT nome_vinícola, nome_região, nome_vinho
  FROM vinícola, região, vinho
  WHERE vinícola.região_id = região.região_id
  AND vinho.vinícola_id = vinícola.vinícola_id;

Cláusula ORDER BY

- Ordena as tuplas resultantes de uma consulta
  - asc: ordem ascendente (padrão)
  - desc: ordem descendente
- Ordenação pode ser especificada em vários atributos
  - Ordenação referente ao primeiro atributo é prioritária.
  - Se houver valores repetidos, então é utilizada a ordenação referente ao segundo atributo, e assim por diante

Exemplo

- Liste os dados das vinícolas e suas regiões.
- Ordene o resultado pela região da vinícola em ordem ascendente.

```
SELECT *
  FROM vinícola, região
  WHERE vinícola.região_id = região.região_id
  ORDER BY nome_região asc
```

Funções de Agregação

- Funções
  - Média: AVG( )
  - Mínimo: MIN( )
  - Máximo: MAX( )
  - Total: SUM( )
  - Contagem: COUNT( )
- Observação
  - DISTINCT: não considera valores duplicados
  - ALL: inclui valores duplicados

Funções de Agregação

- Características
  - recebem uma coleção de valores como entrada;
  - retornam um único valor.

Funções de Agregação

vinho (vinho_id, nome_vinho, tipo_vinho, preço, vinícola_id)

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1
15	Daniela	branco	250,00	2
27	Eduarda	branco	150,00	2
48	Fernanda	tinto	7,00	2
13	Gabriela	tinto	397,00	3
12	Helena	branco	333,00	3

Exemplos

- Qual a *média* dos preços?

SELECT AVG (preço)

FROM vinho

217,125

- Qual a *soma* dos preços?

SELECT SUM (preço)

FROM vinho

1.737,00

- Qual o preço mais *baixo*?

SELECT MIN (preço)

FROM vinho

7,00

- Qual o preço mais *alto*?

SELECT MAX (preço)

FROM vinho

397,00

Exemplos

- *Quantos* vinhos existem na relação vinho?

  SELECT COUNT (vinho_id)

  FROM vinho *8*

- Quantos tipos de vinho *diferentes* existem na relação vinho?

  SELECT COUNT (DISTINCT tipo_vinho)

  FROM vinho *3*

Cláusula GROUP BY

- Funcionalidade:
  - permite aplicar uma função de agregação não somente a um conjunto de tuplas, mas também a um grupo de um conjunto de tuplas;
- Grupo de um conjunto de tuplas:
  - conjunto de tuplas que possuem o mesmo valor para os atributos de agrupamento;

Exemplo

- Qual o preço mais alto e a *média* dos preços *por tipo de vinho*?

  SELECT tipo_vinho, MAX (preço), AVG (preço)

  FROM vinho

  GROUP BY tipo_vinho

tipo_vinho	max	avg
branco	333	233,25
rosê	300	300
tinto	397	168

Cláusula HAVING

- Funcionalidade:
  - especificar uma condição de seleção para grupos;
- Resposta:
  - recupera os valores para as funções somente para aqueles grupos que satisfazem à condição imposta na cláusula HAVING;

Exemplo

- Qual o preço mais alto e a *média* dos preços *por tipo de vinho*, para médias de preços superiores a R$200,00

```
SELECT tipo_vinho, MAX (preço), AVG (preço)
FROM vinho
GROUP BY tipo_vinho
HAVING AVG (preço) > 200
```

tipo_vinho	max	avg
branco	333	233,25
rosê	300	300

Inserção

- Realizada através da especificação:
  - de uma tupla particular;
  - de uma consulta que resulta em um conjunto de tuplas a serem inseridas;
- Valores dos atributos das tuplas inseridas:
  - devem pertencer ao domínio do atributo;
- Atributos sem valores:
  - especificados por NULL ou valor DEFAULT;

INSERT

INSERT INTO nome_tabela
VALUES (V_1, V_2, ..., V_N) ;

- Ordem dos atributos deve ser mantida

INSERT

INSERT INTO nome_tabela (A_1, A_2, ..., A_n)
VALUES (V_1, V_2, ..., V_N) ;

- Ordem dos atributos não precisa ser mantida

INSERT

```
INSERT INTO nome_tabela
   SELECT ...
      FROM ...
      WHERE ...   ;
```

- Tuplas resultantes da cláusula SELECT serão inseridas na tabela nome_tabela

Exemplos

- INSERT INTO região

  VALUES (NULL, ‘nome região’, NULL,

  ‘descrição’);

- INSERT INTO região (nome_região, descrição_região)

  VALUES ‘nome região’, ‘descrição’;

DELETE

```
DELETE FROM nome_tabela
  WHERE predicado ;
```

- Cláusula WHERE
  - é opcional:
    - todas as tuplas da tabela são eliminadas
    - a tabela continua a existir

DELETE ...

- Remove tuplas inteiras
- Opera apenas em uma relação
- Tuplas de mais de uma relação a serem removidas:
  - um comando DELETE para cada relação

- A remoção de uma tupla de uma relação pode ser propagada para tuplas em outras relações devido às restrições de integridade referencial.

Exemplos

- DELETE FROM vinícola

  WHERE vinícola_id = 10;
  - remove a tupla referente a vinícola_id = 10;
  - tabela vinho (i.e., se CASCADE foi especificada na cláusula ON DELETE p/ vinícola_id desta tabela)
- DELETE FROM região
  - remove todos os dados da tabela região

UPDATE

```
UPDATE nome_tabela
   SET coluna = <valor>
   WHERE predicado ;
```

- Cláusula WHERE
  - é opcional
- Exemplos de <*valor*>
  - NULL
  - ´string´

UPDATE ...

- Opera apenas em uma relação

- A atualização da chave primária pode ser propagada para tuplas em outras relações devido às restrições de integridade referencial

Exemplos

- Alterar os anos de produção de vinhos de 2005 para 2003.

```
UPDATE vinho
    SET ano_vinho = 2003
    WHERE ano_vinho = 2005;
```

- Suponha o atributo adicional preço na tabela vinho. Aumentar os preços dos vinhos em 10%.

```
UPDATE vinho SET preço = preço * 1.10;
```

Exemplos

- UPDATE vinícola

  SET vinícola_id = 10

  WHERE vinícola_id = 2;

  - altera o valor de vinícola_id = 10 para vinícola_id = 2
    - tabela vinícola
    - tabela vinho (i.e., se a opção CASCADE foi especificada na cláusula ON UPDATE do campo vinícola_id desta tabela)