# – SQL –
# Linguagem de Manipulação de Dados

Banco de Dados

Profa. Dra. Cristina Dutra de Aguiar Ciferri

# 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 e funções>
FROM <lista de tabelas>
[ WHERE predicado ]
[ GROUP BY <atributos de agrupamento> ]
[ HAVING <condição para agrupamento> ]
[ ORDER BY <lista de atributos> ] ;

# SELECT

- Cláusula SELECT
  - lista os atributos e/ou as funções a serem exibidos no resultado da consulta
  - corresponde à operação de projeção da álgebra relacional
- Cláusula FROM
  - especifica as relações a serem examinadas na avaliação da consulta
  - corresponde à operação de produto cartesiano da álgebra relacional

# SELECT

- Cláusula WHERE
  - especifica as condições para a seleção das tuplas no resultado da consulta
    - as condições devem ser definidas sobre os atributos das relações que aparecem na cláusula FROM
  - inclui condições de junção
  - corresponde ao predicado de seleção da álgebra relacional
  - pode ser omitida

# SELECT

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

# Cláusula WHERE

```
SELECT
FROM
WHERE <atributo> <operador>
            <valor | atributo | lista de valores>
```

- Operador
  - conjunção de condições: AND
  - disjunção de condições: OR
  - negação de condições: NOT

# Cláusula WHERE

- Operadores de comparação

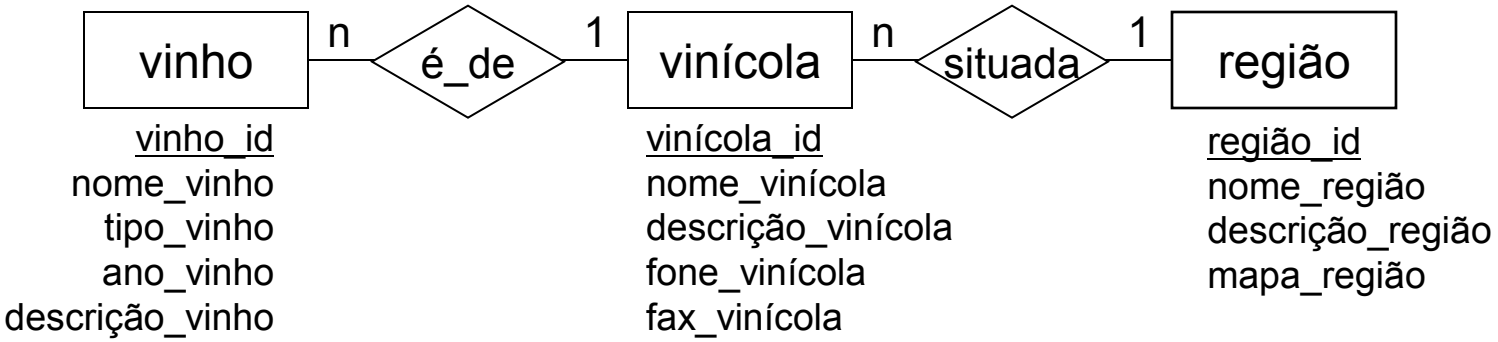
| | | | |
|---|---|---|---|
| igual a | = | diferente de | < > |
| maior que | > | maior ou igual a | >= |
| menor que | < | menor ou igual a | <= |
| entre *dois* valores | BETWEEN ... AND | de cadeias de caracteres | LIKE *ou* NOT LIKE |

- Precedência
  - NOT; operadores de comparação; AND; OR

# 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 (região_id, nome_região, mapa_região, descrição_região)
- vinícola (vinícola_id, nome_vinícola, descrição_vinícola, fone_vinícola, fax_vinícola, região_id)
- vinho (vinho_id, 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 de Conjuntos

| SQL | Álgebra Relacional |
|---|---|
| UNION | União (∪) |
| INTERSECT | Intersecção (∩) |
| MINUS | Diferença (–) |

- Observações
  - as relações participantes das operações precisam ser *compatíveis*
  - operações oferecidas dependem do SGBD

# Interbase

- UNION
  - une todas as linhas selecionadas por duas consultas, *eliminando* as linhas duplicadas
- UNION ALL
  - une todas as linhas selecionadas por duas consultas, *inclusive* as linhas duplicadas

# Exemplo

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

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

# Oracle 9*i* SQL

- UNION
  - une todas as linhas selecionadas por duas consultas, *eliminando* as linhas duplicadas
- INTERSECT
  - retorna as linhas selecionadas tanto pela 1ª consulta quanto pela 2ª consulta, *eliminando* linhas duplicadas que aparecem na resposta final
- UNION ALL
  - une todas as linhas selecionadas por duas consultas, *inclusive* as linhas duplicadas
- MINUS
  - retorna as linhas selecionadas pela 1ª consulta que não foram selecionadas pela 2ª consulta, *eliminando* linhas duplicadas que aparecem na resposta final

# Junção Natural

- SQL (primeiras versões)
  - não tem uma representação para a operação de junção
- Definida em termos de
  - um produto cartesiano
  - uma seleção
  - uma projeção

# Junção

- Não é representada explicitamente
- Cláusulas SELECT e WHERE
  - especificam atributos com mesmo nome usando o nome da tabela e o nome do atributo (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;

# Junção

- SQL-92
  - inclusão de operações adicionais na cláusula FROM
- Operações adicionais no Interbase
  - … [INNER] JOIN ... ON …
  - … LEFT [OUTER] JOIN ... ON …
  - … RIGHT [OUTER] JOIN ... ON …
  - … FULL [OUTER] JOIN ... ON …

# Exemplos

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

- SELECT nome_vinícola, nome_região,
  FROM vinícola LEFT OUTER JOIN região
  ON vinícola.região_id = região.região_id;

*Existe diferença entre os comandos?*

# Exemplos

- 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;

- SELECT nome_vinícola, nome_região,
  nome_vinho
  FROM vinícola JOIN região JOIN vinho
  ON vinho.vinícola = vinícola.vinícola_id
  ON vinícola.região_id = região.região_id;

*Existe diferença entre os comandos?*

# Cláusula AS

- Renomeia
  - atributos
    - deve aparecer na cláusula SELECT
    - útil para a visualização das respostas na tela
  - relações
    - deve aparecer na cláusula FROM
    - útil quando a mesma relação é utilizada mais do que uma vez na mesma consulta
- Sintaxe
  - nome_antigo AS nome_novo

# Exemplo

- SELECT

  nome_vinícola AS nome_da_vinícola ,

  nome_região AS localizada_na_região ,

  nome_vinho AS produz_o_vinho ,

  FROM vinícola AS V,

  região AS R,

  vinho AS Vi

  WHERE V.região_id = R.região_id

  AND Vi.vinícola_id = V.vinícola_id;

# Cláusula ORDER BY

- Ordena as tuplas que aparecem no resultado de uma consulta
  - asc (padrão): ordem ascendente
  - desc: ordem descendente
- Ordenação pode ser especificada em vários atributos
  - a 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 da relação vinícola. Ordene o resultado pelo nome da vinícola em ordem descendente e pela região da vinícola em ordem ascendente.

```
SELECT *
   FROM vinícola, região
   WHERE vinícola.região_id = região.região_id
   ORDER BY nome_vinícola desc,
                  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
- Entrada
  - sum( ) e avg( ): conjunto de números
  - demais funções: tipos de dados numéricos e não-numéricos

# 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

*1737,00*

- 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 conjunto de tuplas
- Grupo de conjunto de tuplas
  - conjunto de tuplas que possuem o mesmo valor para os atributos de agrupamento
- Semântica da respostas
  - atributos de agrupamento no GROUP BY também devem aparecer no SELECT

# 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 | f_1 |
|---|---|---|
| branco | 333 | 233,25 |
| rosê | 300 | 300 |
| tinto | 397 | 168 |

# Cláusula HAVING

- Funcionalidade
  - permite especificar uma condição de seleção para grupos, melhor do que para tuplas individuais
- 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 | f_1 |
|---|---|---|
| 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
- Predicado
  - pode ser complexo

# 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 deve 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 vinícola
    - tabela vinho (i.e., se a opção CASCADE foi especificada na cláusula ON DELETE do campo 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´
  - UPPER ´string´

# UPDATE ...

- Opera apenas em uma relação

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

# Exemplos

- Alterar os anos de produção de vinhos de 2007 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)