SQL – *Structured Query Language*

Disciplina: Banco de Dados I

Prof: Aglaê Pereira Zaupa

Unoeste – Universidade do Oeste Paulista

FIPP – Faculdade do Oeste Paulista

Sumário

- Introdução (Histórico e definições)
- Padrões
- DDL – Data Definition Language
  - Create
  - Alter
  - Drop
- DML – Data Manipulation Language
  - Select
  - Update
  - Delete
- Visões em SQL

Introdução

- SQL é uma linguagem comercial de definição e manipulação de bancos de dados relacionais
- Inicialmente chamava-se SEQUEL (*Structured English QUery Language*) e surgiu no centro de pesquisas de San Jose da IBM, dentro do projeto System R
- SQL é padrão de direito (ISO) e (ANSI):
  - SQL1 ou SQL-89, aprovado em 1986, com modificações em 1989
  - SQL2 ou SQL-92, aprovado em 1992
  - SQL3 ou SQL-99, aprovado em 1999
  - SQL4 versão prévia em 2004
- ISO – Organização Internacional de Padrões
- ANSI – Instituto Nacional Americano de Padrões

Introdução

- SQL oferece as seguintes funcionalidades:
  - Uma DDL para definição do esquema da base de dados
  - Uma DML para programação de consultas e transações que inserem, removem e alteram linhas de tabelas
  - Uma versão de SQL embutida em linguagens de 3a. Geração (COBOL, C, ...), estendendo-as para manipulação de banco de dados
  - Instruções para definições de visões (tabelas virtuais vistas por um usuário ou uma classe de usuários)
  - Uma DCL para controle de autorização de acesso, ou seja, para exercer controle sobre os parâmetros do banco de dados
  - Instruções para controle de transações e concorrência
  - Instruções para especificação de restrições de integridade

Padrões

- A aderência a padrões de SQL é importante para aqueles que:
  - escrevem comandos de SQL em suas aplicações (SQL não fica oculto por algum gerador de telas, ...)
  - desejam portar aplicações a vários SGBD
- O padrão é irrelevante para aqueles que usam ferramentas como geradores de telas, relatórios, etc. proprietários – o usuário está preso ao fornecedor da ferramenta
- Praticamente todo fornecedor de SGBD afirma que o seu produto é compatível com o padrão

Padrões - Validação

- Compatibilidade com padrão somente pode ser verificada por um órgão independente
- Nos EUA, há um órgão do governo (NIST), que faz a validação de aderência ao padrão SQL
- Padrão SQL1 (1986/1989) é testado através do conjunto de testes FIPS 127-1
- Padrão SQL2 (1992) é testado através do conjunto de testes FIPS 127-2
- Grandes fornecedores (Oracle, Sybase, SQL Server DB2) normalmente respeitam pelo menos o padrão SQL2 (entry-level)

Padrões – Níveis do SQL2

- A norma SQL2 é formada por vários níveis, que oferecem cada vez mais funções:
  - entry-level, conjunto mínimo para considerar o produto como SQL2, implementado por vários fornecedores
  - transitional level (conjunto de testes somente apareceu em 1995)
  - intermediate level (conjunto de testes disponível a partir de 1996)
  - full level, norma completa, ainda sem testes nem produtos
- SGBD comerciais implementam variados níveis
- Não existe portabilidade real entre diferentes SGBD
- SQL3 começa a aparecer em alguns produtos (DB2, por exemplo)

DDL – criação do banco de dados

- SQL não oferece instruções para criação de banco de dados
- Alguns produtos (Ex.: SQL Server) têm instruções de DDL:
  - Create Database: cria uma base de dados vazia
  - Drop Database: elimina uma base de dados
- Outros têm abordagens variadas
  - Oracle cria o BD como parte da instalação
  - INGRES e Interbase tem um utilitário

DDL - Instruções

- SQL oferece três instruções para definição do esquema da base de dados:
  - Create Table - define a estrutura de uma tabela, suas restrições de integridade e cria a tabela vazia
  - Drop Table - elimina a tabela da base de dados
  - Alter Table - permite modificar a definição de uma tabela

DDL - Criação de tabelas

```
CREATE TABLE clientes (
        cli_codigo          integer NOT NULL,
        cli_nome            char(30),
        cli_endereco        char(50),
        cli_cpf             char(14),
        cli_sexo            char(1),
CONSTRAINT PK_clientes
                PRIMARY KEY (cli_codigo)
);
```

DDL - Criação de tabelas

- Observações
  - Em SQL2 o conjunto de domínios de valores é fixo
  - Desejável (SQL3) - domínio definível pelo usuário (exemplo: dias da semana, meses do ano, etc)
  - Nos SGBD comerciais são oferecidos domínios adicionais aos do padrão (CHAR, VARCHAR, INTEGER, REAL, etc) destinados a aplicações especiais como DATA, CURRENCY e domínios para armazenar campos longos (BLOB, até 2GB) destinados a conter imagens, sons, vídeos, etc. (maioria aparece no SQL2)
  - A cláusula NOT NULL especifica que uma coluna não admite o valor vazio (requerido para colunas que sejam chave primária)
  - Default é NULL permitido (exceto Sybase e SQL Server antigos)

DDL – Criação de Tabelas
Valor Padrão para Atributos

- É possível definir um valor default para um atributo por meio da adição da cláusula DEFAULT <valor> na definição de um atributo

```
CREATE TABLE Clientes (
    CLI_Codigo      INTEGER NOT NULL,
    CLI_Nome        CHAR(30),
    CLI_Endereco    CHAR(50),
    CLI_CPF         CHAR(14),
    CLI_sexo        char(1) default 'M',
    CONSTRAINT pk_clientes
        PRIMARY KEY (CLI_Codigo)
);
```

DDL – Criação de Tabelas
Restrição de Atributos

- Outro tipo de restrição pode limitar os valores do atributo ou de seu domínio pelo uso da cláusula CHECK

```
CREATE TABLE Clientes (
        CLI_Codigo      INTEGER NOT NULL,
        CLI_Nome        CHAR(30),
        CLI_Endereco    CHAR(50),
        CLI_CPF         CHAR(14),
        CLI_sexo        char(1) default 'M'
                CONSTRAINT c_sexo CHECK (cli_sexo in ('M','F')),
        CONSTRAINT pk_clientes
            PRIMARY KEY (CLI_Codigo)
);
```

DDL – Criação de Tabelas
Restrição de Atributos

```
CREATE TABLE Clientes (
      CLI_Codigo          INTEGER NOT NULL,
      CLI_Nome            CHAR(30),
      CLI_Endereco        CHAR(50),
      CLI_CPF             CHAR(14),
      CLI_sexo            char(1) default 'M',
      CLI_tipo            char(1) CHECK (cli_tipo in ('F','E')),
      CLI_saldo           decimal(8,2) CHECK (cli_saldo > 0),
      CONSTRAINT pk_clientes
           PRIMARY KEY (CLI_Codigo),
      CONSTRAINT c_sexo CHECK (cli_sexo in ('M','F'))
);
```

DDL – Criação de Tabelas
Restrição de Chave

- No SQL original (System R) e no SQL padrão original (1986) não haviam cláusulas para especificar chaves
- A única maneira de definir chave primária era através da criação de um índice sem duplicatas sobre a coluna
- Não havia forma declarativa de definir chaves estrangeiras
  - SGBD não dava suporte à integridade referencial
  - Usuário era obrigado a programar os testes de chaves em sua aplicação

DDL – Criação de Tabelas
Restrição de Chave

- Restrições de Chave Estrangeira nos SGBD comerciais
  - Padrão (1986/1989) foi estendido para especificar chaves:
    - primária
    - estrangeira
    - alternativa (unique key)
  - Praticamente todos os produtos comerciais incluem a definição de chaves

DDL – Criação de Tabelas
CONSTRAINT

- A tendência em SQL2 é tratar de maneira uniforme todas as restrições de integridade de chave (primária, alternativa e estrangeira)

DDL - Criação de tabelas
CONSTRAINT - EXEMPLOS

```
CREATE TABLE Clientes (
        CLI_Codigo      INTEGER NOT NULL,
        CLI_Nome        CHAR(30),
        CLI_Endereco    CHAR(50),
        CLI_CPF         CHAR(14),
        CONSTRAINT pk_clientes
            PRIMARY KEY (CLI_Codigo),
        CONSTRAINT uk_cpf UNIQUE (cli_cpf)
);
```

```
CREATE TABLE Pizza (
        PIZ_Numero      INTEGER NOT NULL,
        PIZ_Nome        CHAR(30),
        PIZ_Ingred      CHAR(200),
        CONSTRAINT pk_pizza
            PRIMARY KEY (PIZ_Numero)
);
```

```
CREATE TABLE Pizza_Tamanho (
        PIZ_Numero      INTEGER NOT NULL,
        PIT_Tamanho     CHAR(1) NOT NULL,
        PIT_Preco       DECIMAL(6,2),
        CONSTRAINT pk_pizza_tamanho
            PRIMARY KEY (PIZ_Numero, PIT_Tamanho),
        CONSTRAINT fk_pizzaTamanho_pizza
            FOREIGN KEY (PIZ_Numero)
                    REFERENCES Pizza
);
```

DDL - Criação de tabelas
CONSTRAINT - EXEMPLOS

```
CREATE TABLE Pedidos (
      PED_Numero        INTEGER NOT NULL,
      CLI_Codigo        INTEGER NOT NULL,
      PED_dtEmiss       DATE,
    CONSTRAINT pk_Pedidos
          PRIMARY KEY (PED_Numero),
    CONSTRAINT fk_pedidos_clientes
          FOREIGN KEY (CLI_Codigo)
                REFERENCES Clientes
);
```

```
CREATE TABLE Pedidos_Itens (
      PED_Numero        INTEGER NOT NULL,
      PIZ_Numero        INTEGER NOT NULL,
      PIT_Tamanho       CHAR(1) NOT NULL,
      ITE_Qtde          INTEGER,
      ITE_Preco         DECIMAL(6,2),
   CONSTRAINT pk_Pedidos_Itens
      PRIMARY KEY (PED_Numero, PIZ_Numero, PIT_Tamanho),
   CONSTRAINT fk_pedItens_pizzatamanho
      FOREIGN KEY (PIZ_Numero, PIT_Tamanho)
            REFERENCES Pizza_Tamanho,
   CONSTRAINT fk_pedItens_pedidos
      FOREIGN KEY (PED_Numero)
            REFERENCES Pedidos
);
```

DDL – Criação de Tabelas
Chave Estrangeira – cláusula on delete

- Caso nada seja especificado na definição de chave estrangeira, uma linha que contenha uma chave primária referenciada na chave estrangeira não pode ser excluída (regra RESTRICT)
- Também pode ser definida a propagação da exclusão da linha para as linhas que a referenciam

```
CREATE TABLE Pizza_Tamanho (
       PIZ_Numero   INTEGER NOT NULL,
       PIT_Tamanho  CHAR(1) NOT NULL,
       PIT_Preco    DECIMAL(6,2),
       CONSTRAINT pk_pizza_tamanho
              PRIMARY KEY (PIZ_Numero, PIT_Tamanho),
       CONSTRAINT fk_pizzaTamanho_pizza
              FOREIGN KEY (PIZ_Numero)
                 REFERENCES Pizza ON DELETE CASCADE
);
```

DDL – Criação de Tabelas
Chave Estrangeira – cláusula on delete

```
CREATE TABLE Pedidos_Itens (
      PED_Numero   INTEGER NOT NULL,
      PIZ_Numero    INTEGER NOT NULL,
      PIT_Tamanho  CHAR(1) NOT NULL,
      ITE_Qtde       INTEGER,
      ITE_Preco      DECIMAL(6,2),
   CONSTRAINT pk_Pedidos_Itens
      PRIMARY KEY (PED_Numero, PIZ_Numero, PIT_Tamanho),
   CONSTRAINT fk_pedItens_pizzatamanho
      FOREIGN KEY (PIZ_Numero, PIT_Tamanho)
                  REFERENCES Pizza_Tamanho,
   CONSTRAINT fk_pedItens_pedidos
      FOREIGN KEY (PED_Numero)
                  REFERENCES Pedidos ON DELETE CASCADE
);
```

DDL – Criação de Tabelas
Chave Estrangeira – cláusula on delete

- SET NULL
  - Especifica que as chaves estrangeiras que referenciam a linha excluída devem ser tornadas vazias
  - Válido somente para chaves estrangeiras opcionais
- RESTRICT (default)
  - Especifica que uma linha não pode ser excluída caso existam chaves estrangeiras que a referenciem

```
FOREIGN KEY ... REFERENCES ...
  ON DELETE {CASCADE | SET NULL | RESTRICT}
```

DDL – Criação de Tabelas
Chave Estrangeira – cláusula on update

- CASCADE
  - Alteração da chave primária é propagada para as chaves estrangeiras que a referenciam
- SET NULL
  - Chaves estrangeiras que referenciam a linha alterada devem ser tornadas vazias
  - Válido somente para chaves estrangeiras opcionais
- RESTRICT (default)
  - Uma chave primária não pode ser alterada, caso existam chaves estrangeiras que a referenciem

```
FOREIGN KEY ... REFERENCES ...
  ON UPDATE {CASCADE | SET NULL | RESTRICT}
```

DDL – Criação de Tabelas
Chave Estrangeira – cláusula on delete on update

```
CREATE TABLE Clientes (
       CLI_Codigo       INTEGER NOT NULL,
       CLI_Nome         CHAR(30),
       CLI_Endereco     CHAR(50),
       CLI_CPF          CHAR(14),
       BAI_CODIGO       INTEGER,
       CONSTRAINT pk_clientes
           PRIMARY KEY (CLI_Codigo),
       CONSTRAINT fk_clientes_bairro
           FOREIGN KEY (BAI_codigo) REFERENCES Bairro
             ON DELETE SET NULL
             ON UPDATE CASCADE
);
```

DDL – alteração na definição das tabelas

- A instrução ALTER TABLE serve para modificar a definição original da tabela
- Nem todas as modificações são permitidas
- Primeiros SGBD e SQL1 somente permitiam adicionar colunas

DDL – alteração na definição das tabelas

- Para modificar a estrutura de tabelas já existentes na base de dados, há uma instrução que permite adicionar colunas nas tabelas:
  - ALTER TABLE nomeTabela ADD atributo tipo
- Observe-se que:
  - A instrução adiciona uma nova coluna com o valor vazio para todas as linhas
  - Os valores para as diversas linhas devem ser adicionados através de instruções da DML
  - Não pode ser especificada a cláusula NOT NULL, já que a coluna é criada com o valor vazio (a menos que seja especificado um valor DEFAULT)

DDL – alteração na definição das tabelas

- ALTER TABLE também permite excluir colunas

```
ALTER TABLE clientes
   DROP CLI_Endereco;
```

DDL – alteração na definição das tabelas

- Se o SGBD não permite a alteração desejada:
  - Armazenar o conteúdo da tabela em uma tabela temporária ou arquivo do sistema operacional
  - Eliminar todas as referências à tabela antiga
  - Eliminar a tabela antiga (DROP TABLE)
  - Definir a nova tabela (CREATE TABLE)
  - Carregar a nova tabela a partir da tabela intermediária ou arquivo do sistema operacional criado no passo 1
  - Reincluir as referências à tabela

DDL – alteração na definição das tabelas

- ALTER TABLE também permite incluir ou excluir uma restrição (CONSTRAINT)

```
ALTER TABLE clientes
   DROP CONSTRAINT pk_clientes;

ALTER TABLE Pedidos
  ADD FOREIGN KEY (cli_codigo) REFERENCES clientes;
ALTER TABLE Pedidos_Itens
       ADD CONSTRAINT fk_pedItens_PizzaTamanho
           FOREIGN KEY (PIZ_Numero, PIT_Tamanho)
               REFERENCES Pizza_Tamanho;
```

DDL – alteração na definição das tabelas

- Para excluir uma restrição é necessário que ela tenha recebido um nome quando de sua definição

```
ALTER TABLE Pedidos_Itens
        ADD CONSTRAINT fk_pedItens_PizzaTamanho
            FOREIGN KEY (PIZ_Numero, PIT_Tamanho)
                REFERENCES Pizza_Tamanho;

ALTER TABLE Pedidos_Itens
       DROP CONSTRAINT fk_pedItens_PizzaTamanho;
```

DDL – alteração na definição das tabelas

- ALTER TABLE com definição de chaves permite separar a DDL em duas partes
  - A primeira parte contém apenas os CREATE TABLE com a estrutura da base de dados
  - A segunda parte contém as restrições de chave através de instruções ALTER TABLE
- Assim, não há a necessidade de ordenação na criação das tabelas

DDL – Eliminação de tabelas

- Para eliminar completamente uma tabela (vazia ou não) da base de dados é usada a instrução:
  - DROP TABLE nomeTabela
- SQL2 inclui cláusulas RESTRICT e CASCADE que informam se a exclusão deve ser propagada ou não para objetos definidos com base na tabela (chaves estrangeiras e visões)
- Exemplo:
  - DROP TABLE clientes RESTRICT
  - Exclui a tabela somente se não existirem visões definidas com base na tabela e ela não for referenciada como chave estrangeira (default para o Interbase)

DML – linguagem de manipulação

- SQL oferece quatro instruções para manipulação da base de dados, sendo uma delas para consultas e as outras três para atualização do conteúdo das tabelas existentes na base:
  - Select – consulta os dados armazenados nas tabelas
  - Insert - insere uma ou mais linhas em uma tabela
  - Update - altera os dados de uma ou mais linhas de uma tabela
  - Delete - exclui uma ou mais linhas de uma tabela

DML – Consulta

- A sintaxe básica de uma instrução de consulta é

```
SELECT <lista de colunas>
FROM <lista de tabelas>
[WHERE <critério>]
```

- O modelo básico de execução da instrução SQL é o seguinte:
  - É feito o produto cartesiano das tabelas envolvidas
  - São selecionadas as linhas da tabela que obedecem ao critério
  - É feita a projeção sobre as colunas que vão ao resultado

DML – Consulta

- Obter os dados de todos os clientes

```
SELECT cli_codigo, cli_nome
FROM clientes
```

ou

```
SELECT *
FROM clientes
```

DML – Consulta

- Obter os códigos e nomes de clientes do bairro 1 e que tem seu nome iniciado por ‘Jo’

```
SELECT cli_codigo, cli_nome
FROM clientes
WHERE bai_codigo = 1 AND
      cli_nome like “Jo%”
```

DML – Inserindo registros

- O usuário deve lembrar a ordem em que as colunas foram criadas quando do CREATE TABLE

```
INSERT INTO clientes VALUES
    (1, 'Epifânia', 'Rua SeiLá', '222.232.232-23',3)
```

- No exemplo abaixo, o usuário não necessita conhecer a ordem original de definição das colunas. Além disso, atributos de valor vazio podem ser omitidos

```
INSERT INTO clientes
(cli_codigo,cli_nome)
   VALUES(3,'Ciclano')
```

- A criação da linha somente será efetivada se as restrições de integridade de chave especificadas (valores não vazios, chaves primária e estrangeiras) forem obedecidas

DML – Inserindo registros

- Inserir o tamanho de Pizza P para todas as pizzas do cardápio que tenha palmito como ingrediente

```
INSERT INTO pizza_tamanho
   (piz_numero,pit_tamanho,pit_preco)
 SELECT piz_numero, 'P', 0
 FROM pizza
 WHERE piz_ingred like '%palmito%'
```

- Neste caso, é possível criar múltiplas linhas (definidas por uma instrução normal de consulta) em uma tabela

DML – Alteração de dados

- Modificar o cpf do cliente de código 4 para ‘221.342.232-44’

```
UPDATE clientes
  SET cli_cpf=‘221.342.232-44’
  WHERE cli_codigo=4
```

- Aumentar o preço das pizzas de tamanho P em 10%

```
UPDATE pizza_tamanho
  SET pit_preco= pit_preco*1.1
  WHERE pit_tamanho=‘P'
```

DML – Alteração de dados

- Incrementar de 10% o preço de todas as pizzas que foram vendidas no mês de julho de 2005

```
UPDATE pizza_tamanho
  SET pit_preco=pit_preco*1.1
  WHERE piz_numero IN
  (SELECT piz_numero
  FROM pedidos p, pedidos_itens pi
    WHERE p.ped_numero=pi.ped_numero and
          p.ped_dtemissao >= '01/07/2005' and
          p.ped_dtemissao <= '31/07/2005')
```

DML – Exclusão

- Excluir o clientes de código 3

```
DELETE FROM clientes
WHERE cli_codigo=3
```

- A sintaxe da cláusula WHERE é a mesma da instrução de consulta
- A exclusão somente é executada se nenhuma restrição de integridade (chave estrangeira) é violada - no caso, significa que o cliente somente será excluído se não possuir pedidos (cláusula RESTRICT), a menos que sejam utilizadas as cláusulas SET ou CASCADE.

DML – Exclusão

- Excluir todos os tamanhos de pizza ‘M’

```
DELETE FROM pizza_tamanho
WHERE pit_tamanho = ‘M’
```

- Excluir todos os tamanhos de pizza

```
DELETE FROM pizza_tamanho
```

Visões em SQL

- Uma visão em SQL é uma tabela virtual, isto é, uma tabela que não é armazenada fisicamente na base de dados
- O objetivo de visões é atender usuários que necessitam ver os dados de determinada forma, diferente da forma de armazenamento
- Visões são usadas em combinação com mecanismos de controle de acesso
- Quando a definição de tabelas básicas em uma base de dados é modificada, aplicações que a usam podem ser afetadas. Se a mudança nas tabelas básicas não afeta a visão, aplicações que usam a visão não são afetadas

Visões em SQL

- O conceito genérico de visão em BD é mais abrangente:
  - Qualquer forma de ver dados por um usuário
  - Visões de forma geral poderiam ser desnormalizadas (ex.: uma nota fiscal)
  - A instrução para criar uma visão em SQL é

    ```
    CREATE VIEW <nome> [(atributos)] AS <consulta SQL>
    ```
  - A instrução para eliminar uma visão é

    ```
    DROP VIEW <nome>
    ```

Visões em SQL - Exemplos

- Criar uma visão da tabela de pedidos, na qual apareça, os números de pizzas e de pedidos, bem como seus dados

```
CREATE VIEW vPedido AS
  SELECT p.ped_numero, p.ped_dtemis, pz.piz_nome,
     tp.pit_tamamho
  FROM Pedidos p,Pedidos_itens pi,Pizza pz,
       tamanho_pizza tp
  WHERE p.ped_numero = pi.ped_numero and
        pi.piz_numero = tp.piz_numero and
        pi.pit_tamanho = tp.pit_tamanho and
        pz.piz_numero = tp.piz_numero
```

Visões em SQL - Exemplos

- Também é possível especificar os nomes dos atributos da visão

```
CREATE VIEW vPedido2 (numero,data,pizza,tamanho) AS
  SELECT p.ped_numero, p.ped_dtemis, pz.piz_nome,
     tp.pit_tamamho
  FROM Pedidos p,Pedidos_itens pi,Pizza pz,
       tamanho_pizza tp
  WHERE p.ped_numero = pi.ped_numero and
        pi.piz_numero = tp.piz_numero and
        pi.pit_tamanho = tp.pit_tamanho and
        pz.piz_numero = tp.piz_numero
```

Visões em SQL - Exemplos

- Obter o número do pedido, o nome da pizza e o tamanho da pizza de cada pedido.

```
SELECT ped_numero,piz_nome, pit_tamanho
FROM vPedido
```

- A instrução acessa a visão vPedido
- O usuário não necessita especificar toda definição da junção das quatro tabelas
- A visão não existe fisicamente. A instrução SELECT de definição da visão é misturada ao SELECT da consulta em tempo de tradução (a instrução que cria a visão é inserida no **from** do SQL em questão)

Atualizações por meio de Visões

- SQL permite que uma tabela da base de dados seja atualizada (inserção, exclusão e alteração) através de uma visão
- A restrição é de que a atualização só pode ocorrer sobre uma visão definida sobre uma única tabela da base de dados e que as linhas da visão estejam em relação um-para-um com as linhas da tabela da base de dados
- Exemplo:
  - Atualizações sobre a visão vPedido definida anteriormente não são admissíveis, pois ela está definida sobre mais de uma tabela base

Referências

- ELMASRI, Ramez; NAVATHE, S. B. **Sistemas de Banco de Dados.** Addison Wesley, 4ª Ed., 2005.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados.** Makron Books, 3ª Ed., 1999