Universidade Federal
de Campina Grande

Banco de Dados I
Unidade 5: A linguagem SQL

Prof. Cláudio de Souza Baptista, Ph.D.
Laboratório de Sistemas de Informação – LSI
UFCG

Introdução

- O modelo relacional usa o *padrão SQL* (Structured Query Language)
- SQL foi desenvolvida pela IBM (70) - System R
- SQL = *interface*
  - para a definição de tabelas,
  - para as operações sobre as mesmas
  - para a definição de regras de integridade de bancos de dados

Introdução

- SQL: implementada em *todos* os sistemas de bancos de dados relacionais existentes;

- Por quê a indústria tem interesse em padronizar os sistemas de bancos de dados?
  - a portabilidade e
  - interoperabilidade

SQL too old?

- Fake news:
  - SQL está morto, com o surgimento do NOSQL – Not Only SQL (últimos cinco anos)?
- Realidade:
  - SQL cada vez mais sendo usado
  - Incluindo soluções para Big Data:
    - Hive, Impala, Spark-SQL
  - NewSQL DB: NuoDB, LeanXcale, SAP HANA, Google Spanner
  - Plataformas HTAP
  - Polystore databases
- Not Yet SQL?

SQL Intro

- SQL é baseado em bags (duplicatas) ao invés de sets (sem duplicatas)
- Muitos padrões (ISO/IEC 9075):
  - ANSI SQL
  - SQL 92 (a.k.a. SQL2)
  - SQL 99 (a.k.a. SQL3): Regex, Triggers, OO
  - SQL 2003: XML, windows, sequences
  - SQL 2008: TRUNCATE
  - SQL 2011: Temporal DB
  - SQL 2016: JSON, Polymorphic Table Functions
- A maioria dos SGDBR implementam SQL:92
  - Comparação das implemetações: http://troels.arvin.dk/db/rdbms/
  - Avanços do SQL em: https://modern-sql.com
- Relational SQL consiste em menos de 20% do Modern SQL (rich data types: arrays, JSON), nested tables (multisets), composite types (objects)

Padrão SQL: ISO/IEC 9075

- Dividido em 16 partes:
  - Part 1: *Framework* (SQL/Framework)
  - Part 2: *Foundation* (SQL/Foundation)
  - Part 3: *Call-Level Interface* (SQL/CLI)

Padrão SQL: ISO/IEC 9075

- Dividido em 16 partes (cont.):
  - Part 4: *Persistent stored modules* (SQL/PSM)
  - Part 6: *JavaScript Object Notation (JSON)*
  - Part 9: *Management of External Data* (SQL/MED)
  - Part 10: *Object language bindings* (SQL:OLB)
  - Part 11: *Information and definition schemas* (SQL/Schemata)

Padrão SQL: ISO/IEC 9075

- Dividido em 16 partes (cont):
  - Part 13: *SQL Routines and types using the Java TM programming language* (SQL/JRT)
  - Part 14: *XML-Related Specifications* (SQL/XML)
  - Part 15 - Multi dimensional arrays (SQL/MDA)
  - Part 16 - Property Graph Query (SQL/PGQ)

Padrão SQL: ISO/IEC 13249

- Complementa com a parte de Multimídia e Aplicações (Espacial/Temporal e DM)
  - ISO/IEC 13249-1:2016 Part 1: *Framework*
  - ISO/IEC 13249-2:2003 Part 2: *Full-Text*
  - ISO/IEC 13249-3:2016 Part 3: *Spatial*
  - ISO/IEC 13249-5:2003 Part 5: *Still image*
  - ISO/IEC 13249-6:2006 Part 6: *Data mining*
  - ISO/IEC 13249-7:2013 Part 7: *History*

Introdução

- Como vimos, um SGBD possui duas linguagens:
  - **DDL**: Linguagem de Definição de Dados. Usada para definir os esquemas, atributos, visões, regras de integridade, índices, usuários, segurança, etc;
  - **DML**: Linguagem de Manipulação de Dados. Usada para se ter acesso aos **dados** armazenados no BD.
    - CRUD: Insert, delete, update, select
  - **DCL**: Linguagem de Controle dos Dados: usada para controlar o acesso aos **dados** em um **banco de dados**

Introdução

A linguagem SQL tem diversas partes:

- Linguagem de Definição de Dados (DDL): fornece comandos para definições de esquemas de relação, criação/remoção de tabelas, criação de índices e modificação de esquemas.

- Linguagem de Manipulação de Dados (DML): Compreende comandos para inserir, consultar, remover e modificar tuplas num BD.
- Autorização (DCL): a SQL DCL (Data Control Language) inclui comandos para especificação de direitos de acesso às relações/ visões (grant e revoke).

Introdução

- **Tipos em SQL**
  - Numéricos:
    - INTEGER (INT), SMALLINT, BIGINT para representar inteiros
    - NUMERIC(p,s): tem uma precisão e uma escala(número de dígitos na parte fracionária). A escala não pode ser maior que a precisão. Muito usado para representar dinheiro
    - REAL: ponto flutuante de precisão simples
    - DOUBLE: ponto flutuante com precisão dupla (8 bytes)
    - FLOAT: ponto flutuante (4 bytes)
  - No Oracle, pode-se usar NUMBER para todos os tipos acima.

Introdução

- **Tipos em SQL**
  - Character
    - CHARACTER(x) (CHAR): representa um string de tamanho x. Se x for omitido então é equivalente a CHAR(1). Se um string a ser armazenado é menor do que x, então o restante é preenchido com brancos.
    - CHARACTER VARYING(x) (VARCHAR): representa um string de tamanho x. Armazena exatamente o tamanho do string (tam <= x) sem preencher o resto com brancos. Neste caso x é obrigatório.
    - CHARACTER LARGE OBJECT (CLOB): armazena strings longos. Usado para armazenar documentos.
    - **OBS.:** Existem os National character data types: NCHAR, NVARCHAR, NCLOB que permitem implementar internacionalização

Introdução

- **Tipos em SQL**
  - Bit string e Binary Strings (BLOB)
    - BIT(X): permite armazenar uma quantidade x de bits
    - BIT VARING(X) (VARBIT): permite armazenar uma quantidade variável de bits até o tamanho X
    - BINARY LARGE OBJECT (BLOB): para armazenar grande quantidades de bytes como fotos, vídeo, áudio, gráficos, mapas, etc.

Introdução

- **Tipos em SQL**
  - DATETIMES
    - DATE: armazena ano (4 dígitos), mês (2 dígitos) e dia(2 dígitos).
    - TIME: armazena hora(2 dígitos), minuto(2 dígitos) e segundo(2 dígitos)
    - TIMESTAMP: DATE + TIME
    - TIME WITH TIME ZONE: igual a time + UTC offset
    - TIMESTAMP WITH TIME ZONE: igual a TIMESTAMP + UTC offset

Introdução

- **Tipos em SQL**
  - Intervals
    - Um intervalo é a diferença entre duas datas e horas.
    - Year-month interval: contém apenas um valor de ano, ou de mês ou ambos. YEAR(p), MONTH(p), INTERVAL YEAR TO MONTH(p)
    - Day-Time interval: contém apenas um dia, uma hora, um minuto e/ou um segundo. INTERVAL DAY(p), INTERVAL DAY TO HOUR, INTERVAL DAY(6) TO MINUTE, INTERVAL SECOND(7), etc.
  - Booleans: lógica de três valores (TRUE, FALSE e UNKNOWN)
  - OBS: Oracle não possui o tipo Boolean!!!!

SQL-DDL

- Os comandos SQL para definição de dados são:
  - CREATE
  - DROP
  - ALTER
  - TRUNCATE

- **CREATE TABLE**: especifica uma nova tabela (relação), dando o seu nome, único no esquema, especificando as colunas(atributos) (cada uma com seu nome, tipo e restrições) e as restrições

Sintaxe:

```
CREATE TABLE tabela_base (colunas tipo_base + constraints)
```

SQL-DDL

- As definições das colunas têm o seguinte formato:

  coluna tipo[NOT NULL [UNIQUE]][DEFAULT valor]

- Onde:
  - **coluna:** nome do atributo que está sendo definido
  - **tipo:** domínio do atributo
  - **NOT NULL:** expressa que o atributo não pode receber valores nulos
  - **UNIQUE:** indica que o atributo tem valor único na tabela. Qualquer tentativa de se introduzir uma linha na tabela contendo um valor igual ao do atributo será rejeitada. Serve para indicar chaves secundárias
  - **DEFAULT:** indica um valor default para a coluna

Constraints
(Restrições de Integridade e de Domínio)

- **Integridade de Chave:**

  PRIMARY KEY(atributos_chave)

- **Integridade Referencial:**

  FOREIGN KEY (atributos) REFERENCES tabela_base(atributos)

- **Restrição de Integridade:**

  CHECK(condição)

SQL-DDL

```
CREATE TABLE Empregado (
            matricula char(9),
             nome VARCHAR(15) NOT NULL,
             dataNasc            DATE,
             endereco            VARCHAR(30),
             sexo CHAR,
             salario NUMERIC(10,2),
             supervisor CHAR(9),
             depto     INT NOT NULL,
             PRIMARY KEY (matricula),
             CHECK (salario >= 0),
            FOREIGN KEY (supervisor) REFERENCES empregado(matricula),
            FOREIGN KEY (depto) REFERENCES departamento(codDep)
)
```

SQL-DDL

```
CREATE TABLE Departamento (
      codDep                INT,
       nomeDep   VARCHAR(15)  NOT NULL UNIQUE,
       gerente               CHAR(9)         NOT NULL,
       dataInicioGer         DATE,
       PRIMARY KEY(codDep),
       FOREIGN KEY (gerente) REFERENCES
                     empregado(matricula)
)
```

SQL-DDL

- Problema no exemplo anterior: **Como criar as tabelas que dependem uma das outras.**

- **Ex.:** Ovo ou galinha

```
CREATE TABLE chicken
(cID INT PRIMARY KEY,
 eID INT REFERENCES egg(eID));

CREATE TABLE egg
(eID INT PRIMARY KEY,
     cID INT REFERENCES chicken(cID));
```

SQL-DDL

- Solução no Oracle:

```
CREATE TABLE chicken
(cID INT PRIMARY KEY,   eID INT)

CREATE TABLE egg
(eID INT PRIMARY KEY,  cID INT)

ALTER TABLE chicken ADD CONSTRAINT chickenREFegg
        FOREIGN KEY (eID) REFERENCES egg(eID)
INITIALLY DEFERRED DEFERRABLE

ALTER TABLE egg ADD CONSTRAINT eggREFchicken
        FOREIGN KEY (cID) REFERENCES chicken(cID)
INITIALLY DEFERRED DEFERRABLE
```

SQL-DDL

- Solução:

  E se quisermos interir Chicken(1,2) e Egg(2,1)?

```
INSERT INTO chicken VALUES(1, 2);
INSERT INTO egg VALUES(2, 1);
COMMIT;
```

SQL-DDL

- **Chave estrangeira**
  - Como vimos é definida com a cláusula **FOREIGN KEY**. Alguns SGBDs permitem que se use uma notação abreviada para chave estrangeira quando esta é formada por um único atributo

```
CREATE TABLE Empregado
        (
            matricula CHAR(9) NOT NULL,
            nome VARCHAR(15) NOT NULL,
            …
            supervisor CHAR(9) REFERENCES Empregado(matricula),
            codDep      INT  NOT NULL REFERENCES  Departamento(codigo),
            …
        );
```

SQL-DDL

- Uma cláusula **FOREIGN KEY** inclui regras de remoção/ atualização:

```
FOREIGN KEY (coluna)
REFERENCES tabela [ON DELETE
{RESTRICT|CASCADE|SET NULL| SET DEFAULT}]
[ON UPDATE
        {RESTRICT|CASCADE|SET NULL| SET DEFAULT}]
```

- Supondo que T2 tem uma chave estrangeira para T1, vejamos as cláusulas **ON DELETE** e **ON UPDATE**

SQL-DDL

- **ON DELETE:**
  - RESTRICT: (default) significa que uma tentativa de se remover uma linha de T1 falhará se alguma linha em T2 combina com a chave de T1
  - CASCADE: remoção de uma linha de T1 implica em remoção de todas as linhas de T2 que combina com a chave de T1
  - SET NULL: remoção de T1 implica em colocar NULL em todos os atributos da chave estrangeira de cada linha de T2 que combina com a chave de Tq.
  - SET DEFAULT: remoção de linha em T1 implica em colocar valores DEFAULT nos atributos da chave estrangeira de cada linha de T2 que combina com a chave de T1.

SQL-DDL

- **ON UPDATE:**
  - RESTRICT: (default) update de um atributo de T1 falha se existem linhas em T2 combinando
  - CASCADE: update de atributo em T1 implica que linhas que combinam em T2 também serão atualizadas
  - SET NULL: update de T1 implica que valores da chave estrangeira em T2 nas linhas que combinam são postos par NULL.
  - SET DEFAULT: update de T1 implica que valores da chave estrangeira de T2 nas linhas que combinam terão valores default aplicados.

SQL-DDL

- As restrições de integridade podem ter um nome e serem especificadas com a cláusula CONSTRAINT. Isto permite que possamos no futuro eliminar (DROP) ou alterar (ALTER) o constraint.

- O exemplo a seguir mostra o uso de CONSTRAINT, DEFAULT, ON DELETE e ON UPDATE

SQL-DDL

```
CREATE TABLE empregado
        (
                …
                depto     INT       NOT NULL        DEFAULT 1,
                CONSTRAINT empCP PRIMARY KEY(matricula),
                CONSTRAINT empSuperCE FOREIGN KEY(supervisor)
                REFERENCES empregado(matricula) ON DELETE
    SET NULL ON UPDATE CASCADE,
                CONSTRAINT deptoCE FOREIGN KEY (depto)
                REFERENCES departamento(codigo) ON DELETE
    SET DEFAULT ON UPDATE CASCADE
      );
```

SQL-DDL

- **ALTER TABLE**
  - Para adicionar / remover / alterar os atributos de uma tabela
  - Pode-se também alterar as restrições da tabela.
- Ao incluirmos uma coluna devemos especificar o seu tipo de dado, não podendo esta coluna ser NOT NULL.

SQL-DDL

- **ALTER TABLE**
  - Sintaxe: Para adicionar uma nova coluna a uma tabela

```
ALTER TABLE tabela_base
ADD [COLUMN] atributo tipo_dado
```

Para modificar uma coluna de uma tabela

```
ALTER TABLE tabela_base
ALTER [COLUMN] atributo
                    SET valor-default
                    ou  DROP DEFAULT
```

Obs.: no Oracle a cláusula opcional COLUMN não existe!

SQL-DDL

- **ALTER TABLE**
  - Para remover uma coluna de uma tabela:

```
ALTER TABLE tabela_base
DROP [COLUMN] atributo
```

  - Para adicionar uma restrição a uma tabela

```
ALTER TABLE tabela_base
ADD  CONSTRAINT …
```

  - Para remover uma restrição de um tabela

```
ALTER TABLE tabela_base
DROP CONSTRAINT nome-contraint
```

SQL-DDL

Ex.:

```
ALTER TABLE Peca
ADD espessura INT
```

- Podemos remover um atributo usando a sintaxe

```
ALTER TABLE tabela_base
DROP atributo [CASCADE|RESTRICT]
```

-

SQL-DDL

Ex.:

```
ALTER TABLE empregado DROP endereco CASCADE;
```

```
ALTER TABLE departamento ALTER gerente DROP DEFAULT
```

```
ALTER TABLE departamento ALTER gerente
SET DEFAULT “333444555”
```

```
ALTER TABLE empregado
        DROP CONSTRAINT empsuperCE CASCADE;
```

```
ALTER TABLE empregado
        ADD CONSTRAINT empsuperCE FOREIGN KEY
        (supervisor) REFERENCES empregado(matricula)
```

SQL-DDL

- **DROP TABLE**: remove uma tabela-base do BD. Remove tanto os dados quanto a definição da tabela

- Sintaxe:

```
DROP TABLE <nomeTabela>
```

Ex.:

```
DROP TABLE Peca
```

SQL-DDL

- **TRUNCATE TABLE**: remove apenas os dados da tabela, deixando sua definição (metadados) intacta.

- Sintaxe:

```
TRUNCATE TABLE <nomeTabela>
```

Ex.:

```
TRUNCATE TABLE Peca
```

SQL-DDL

- **Especificando índices em SQL**
  - SQL possui comandos para criar e remover índices
  - O Oracle cria automaticamente índices em chaves primárias e colunas com UNIQUE

Ex.: Criar um índice no atributo nome da relação Empregado.

-

```
CREATE INDEX nome-índice
ON Empregado(nome)
```

Sequências

Sequências

- Sintaxe do Padrão SQL:

CREATE SEQUENCE < nome-sequencia >
        START WITH <valor inteiro>
        INCREMENT BY <valor do incremento inteiro>
[CYCLE | NOCYCLE]

Sequências

- Ex 1:

- CREATE SEQUENCE SEQ_1;

Sequências

- Ex 2:

```
CREATE SEQUENCE  SEQ_2
START WITH 500
INCREMENT BY -10
MAXVALUE 500
MINVALUE 0
```

Sequências

- Ex 3:

```
CREATE SEQUENCE  SEQ_3
START WITH 500
INCREMENT BY -10
MAXVALUE 500
MINVALUE 0
CYCLE
```

ACESSANDO SEQUÊNCIAS - Oracle

SELECT SEQ1.CURRVAL FROM DUAL

-- MOSTRA O VALOR CORRENTE DA

-- SEQUENCIA SEQ1- DUAL é uma

-- DUMMY TABLE no ORACLE

SELECT SEQ1.NEXTVAL FROM DUAL

-- AVANÇA A SEQ1 E MOSTRA SEU VALOR

INSERINDO DADOS NUMA TABLE

```
INSERT INTO EMPREGADO VALUES (
SEQ2.NEXTVAL,
‘MARIA DA PAZ’,
9850,00,
‘D1’)
```

Auto-incremento

- Implementação da ideia de gerar números automaticamente, sintaxe usada no MYSQL:

```
CREATE TABLE Pessoas (
        ID int  AUTO_INCREMENT,
        Nome varchar(255) NOT NULL,
        Idade int,
        PRIMARY KEY (ID)
);
INSERT INTO Pessoas (Nome, Idade)
VALUES ('Larissa Silva',42);
```

- OBS.: Perceba que na inserção não se coloca o campo ID (do auto-incremento)

Identity

- Implementação da ideia de gerar números automaticamente, sintaxe usada no MS SQL Server:

```
CREATE TABLE Pessoas (
        ID int IDENTITY(1,1) PRIMARY KEY,
        Nome varchar(255) NOT NULL,
        Idade int
);
INSERT INTO Pessoas (Nome, Idade)
VALUES ('Larissa Silva',42);
```

- OBS.: Perceba que na inserção não se coloca o campo ID (do Identity)