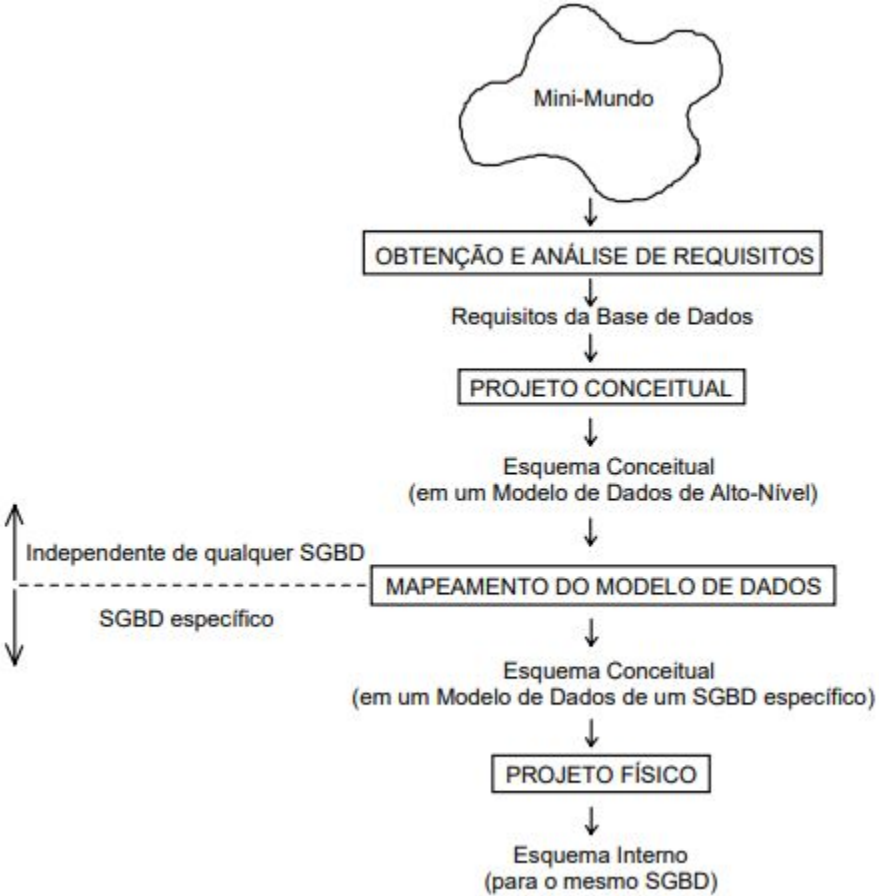
UVV

Design e Desenvolvimento de Banco de Dados I

SQL - Structured Query Language

Prof. Me Renato Sousa Botacim
Email: renato.botacim@uvv.br


Mini-Mundo
OBTENÇÃO E ANÁLISE DE REQUISITOS
Requisitos da Base de Dados
PROJETO CONCEITUAL
Esquema Conceitual
(em um Modelo de Dados de Alto-Nível)
Independente de qualquer SGBD
MAPEAMENTO DO MODELO DE DADOS
SGBD específico
Esquema Conceitual
(em um Modelo de Dados de um SGBD específico)
PROJETO FÍSICO
Esquema Interno
(para o mesmo SGBD)

Structured Query Language

- O SQL padrão sofreu várias alterações ao longo do tempo:
  - 1986 (SQL-86 e SQL-87) Publicado pela ANSI e ratificado pela ISO.
  - 1989 (SQL-89) 1992 (SQL-92) Também conhecido como SQL2.
  - 1999 (SQL:1999) Também conhecido como SQL 3. Inclui expressões regulares, queries recursivas, gatilhos, tipos não escalares e algumas funcionalidades orientadas a objetos.
  - 2003 (SQL:2003) Inclui suporte a XML e colunas com numeração automática.

Base álgebra relacional e cálculo relacional

Divisões da SQL

- DDL – Linguagem de Definição de Dados(Data Definition Language)
- DML – Linguagem de Manipulação de Dados
- DCL – Linguagem de Controle de Dados (autorização de dados e licença de usuários para controlar quem tem acesso aos dados).
- DTL - Linguagem de Transação de Dados.
- DQL – Linguagem de Consulta de Dados (Tem apenas um comando: SELECT).

Principais funcionalidades

definição (DDL) e manipulação (DML) de dados

definição de visões e autorizações de acesso

definição de restrições de integridade

definição de transações

comandos para embutir em Linguagens de Programação

SQL - DDL

- DDL é o conjunto de comandos SQL responsáveis pela definição dos dados, ou seja, pela criação de bancos, esquemas, tabelas, campos, tipos de dados, constraints, etc.
- A SQL/DDL (Data Definition Language) permite ao usuário definir tabelas e elementos associados.
- A SQL/DDL se caracteriza por poucos comandos básicos, embora implementações comerciais tenham várias extensões

DDL

Comando CREATE cria um objeto (uma Tabela, por exemplo) dentro da base de dados.

Comando ALTER altera um objeto do banco de dados.

Comando DROP apaga um objeto do banco de dados. Exemplos:

- ALTER TABLE
- CREATE INDEX
- ALTER INDEX
- DROP INDEX
- CREATE VIEW
- DROP VIEW

create table:

define a estrutura da tabela, suas restrições de integridade e cria uma tabela vazia

alter table:

- modifica a definição de uma tabela
- Inclusão, Exclusão e Alteração de atributos;
- Inclusão e Exclusão de Relações de Integridade.

Relações de Integridade Básicas:

- atributos chave não podem ser removidos
- atributos NOT NULL não podem ser inseridos

Os tipos de dados mais utilizados são:

- char(n) ou character(n) - Cadeia de caracteres de tamanho fixo n
- varchar(n) ou character varying(n) - Cadeia de caracteres com tamanho máximo n
- text - Cadeia de caracteres sem tamanho definido
- int ou integer - Números inteiros (4 bytes)
- numeric(precisão, escala) - Números reais sem limite de tamanho
- date e time - Data e hora
- datetime - Data + hora no mesmo campo
- boolean - Valores booleanos

Restrições

Em SQL podem ser definidas restrições de integridade de vários tipos:

CHECK

NOT-NULL

UNIQUE

CHAVE PRIMÁRIA

CHAVE ESTRANGEIRA

Check

Devemos sempre dar nomes às restrições para que seja mais fácil identificar a razão pela qual a inserção de dados falha:

Poderia ser assim:

salario numeric(9,2) CHECK (salario > 0)

Mas deve ser assim: salario numeric(9,2)

CONSTRAINT salario_positivo CHECK (salario > 0)

No caso da restrição abranger mais de uma coluna temos de usar uma restrição de tabela (após as definições das colunas):

```
CREATE TABLE empregado (
  matricula     INTEGER NOT NULL,
  ...
  salario       NUMERIC(9,2) NOT NULL,
  descontos     NUMERIC(9,2) NOT NULL,
  CONSTRAINT pk_empregado
    PRIMARY KEY (matricula),
  CONSTRAINT desconto_menor_salario
    CHECK (descontos < salario)
);
```

NOT-NULL

Para garantir que uma coluna não vai ter valores nulos podemos usar uma restrição do tipo NOT NULL

```
CREATE TABLE empregado (
    ...
    salario NUMERIC(9,2) NOT NULL
);
```

CHAVE PRIMÁRIA

Podemos definir uma, e só uma, chave primária para a tabela.

Uma chave primária não pode conter valores nulos nem pode ter valores repetidos

Exemplo:

```
CREATE TABLE empregado (
  matricula INTEGER NOT NULL,
  nome      VARCHAR(40) NULL,
  CONSTRAINT pk_empregado
    PRIMARY KEY (matricula)
);
```

CHAVE PRIMÁRIA

Uma chave primária pode ser composta por mais de uma coluna (campo).

Nesse caso temos de usar obrigatoriamente uma restrição de tabela.

```
CREATE TABLE empregado (
  setor      CHAR(3) NOT NULL,
  matricula INTEGER NOT NULL,
  nome       VARCHAR(40) NOT NULL,
  CONSTRAINT pk_empregado
    PRIMARY KEY (setor, matricula)
);
```

UNIQUE

Chaves candidatas alternativas podem ser definidas usando restrições do tipo UNIQUE.

Estas restrições são equivalentes às restrições de chave primária mas não obrigam os valores a ser não nulos

Unique

```
CREATE TABLE empregado2 (
  matricula    INTEGER NOT NULL,
  nome         VARCHAR(40) NOT NULL,
  cpf          CHAR(11) NULL,
  CONSTRAINT pk_empregado2
    PRIMARY KEY (matricula),
  CONSTRAINT un_empregado_cpf
    UNIQUE (cpf)
);
```

No caso da restrição UNIQUE for com mais de uma coluna (campo), obrigatório o uso da restrição de tabela.

Lembrar que a restrição UNIQUE cria um índice que terá um custo de manutenção para o SGBD.

Chaves Estrangeiras

Uma restrição do tipo FOREIGN KEY permite declarar chaves estrangeiras.

Uma chave estrangeira deve sempre referenciar uma chave primária ou única

Chaves Estrangeiras

```
CREATE TABLE empregado (
  matricula INTEGER     NOT NULL
    CONSTRAINT pk_empregado PRIMARY KEY,
  nome      VARCHAR(40) NOT NULL,
  cod_dep   INTEGER     NOT NULL,
  CONSTRAINT fk_empregado_departamento
    FOREIGN KEY (cod_dep)
    REFERENCES departamento(cod_depto)
 );
```

Chaves Estrangeiras

Podemos definir o que acontece quando uma chave estrangeira é violada durante uma operação de remoçã

```
CREATE TABLE empregado (
  matricula INTEGER     NOT NULL,
  nome      VARCHAR(40) NOT NULL,
  cod_depto INTEGER     NOT NULL,
  CONSTRAINT pk_empregado
    PRIMARY KEY (matricula),
  CONSTRAINT fk_empregado_departamento
    FOREIGN KEY (cod_depto)
    REFERENCES departamento(cod_depto)
    ON DELETE CASCADE
);
```

Chaves Estrangeiras

No exemplo, quando for apagado um departamento e caso exista(m) empregado(s) associados ao departamento que está sendo apagado, todos estes empregados também serão apagados.

CUIDADO

Chaves Estrangeiras

As alternativas para os valores de ON DELETE e ON UPDATE são:

- RESTRICT - Não deixa efetuar a operação
- CASCADE - Apaga os registros associados (delete) ou altera a chave estrangeira (update).
- SET NULL - A chave estrangeira passa a NULL
- SET DEFAULT - A chave estrangeira passa a ter o valor padrão

Se não for definido o funcionamento default é a restrição do tipo RESTRICT

Valor Default

Você pode definir um valor padrão que será colocado sempre quando houver uma inclusão na tabela e o valor da coluna não for especificado;

```
CREATE TABLE departamento2 (
   cod_depto  INTEGER     NOT NULL,
   nome_depto VARCHAR(40) NOT NULL
     DEFAULT 'Departamento Novo',
   CONSTRAINT pk_departamento2
     PRIMARY KEY (cod_depto)
);
```

CREATE TABLE

CREATE TABLE

Cria uma tabela vazia

```
CREATE TABLE nome_tabela (
    nome_atributo_1 tipo_1 [[NOT]NULL][UNIQUE]
    [{, nome_atributo_n tipo_n}]
    [, CONSTRAINT <nome_constraint> primary key
   (nome_atributo,...)]
    [{, CONSTRAINT <nome_constraint> foreign key
   (nome_atributo_local,...)  references
   nome_tabela_remota(nome_atributo_remoto,...)}]
);
```

```
CREATE TABLE ambulatorio (
   cod_ambulatorio INTEGER NOT NULL,
   andar           INTEGER NOT NULL,
   capacidade      INTEGER NULL,
   CONSTRAINT pk_ambulatorio
      PRIMARY KEY(cod_ambulatorio)
);
```

```
CREATE TABLE medico (
  cod_medico       INTEGER     NOT NULL,
  nome_medico      VARCHAR(40) NOT NULL,
  idade_medico     INTEGER     NOT NULL,
  especialidade    CHAR(20)    NULL,
  rg_medico        NUMERIC(10) NULL,
  cidade_medico    VARCHAR(30) NULL
    DEFAULT 'Cachoeiro de Itapemirim',
  cod_ambulatorio INTEGER      NOT NULL,
  CONSTRAINT pk_medico
    PRIMARY KEY(cod_medico),
  CONSTRAINT un_medico
    UNIQUE (rg_medico),
  CONSTRAINT fk_medico_ambulatorio
    FOREIGN KEY(cod_ambulatorio)
    REFERENCES ambulatorio(cod_ambulatorio)
);
```

DROP TABLE

DROP TABLE
remove uma tabela com todos os seus registros
não remove tabelas referenciadas por outras tabelas

sintaxe: DROP TABLE <nome_tabela>;

Exemplo: DROP TABLE empregado

ALTER TABLE

Depois de criada, uma tabela pode ser modificada.

```
ALTER TABLE medico ADD COLUMN telefone char(10) NULL;
ALTER TABLE medico ADD COLUMN cpf char(11) NULL;
ALTER TABLE medico DROP COLUMN telefone;
ALTER TABLE medico DROP COLUMN rg_medico;
ALTER TABLE medico ADD COLUMN rg_medico char(10) NULL;
ALTER TABLE medico ADD CONSTRAINT ck_medico_idade CHECK (idade_medico > 0);
```

```
ALTER TABLE medico ADD CONSTRAINT un_medico_cpf  UNIQUE (cpf);
ALTER TABLE medico ADD CONSTRAINT fk_medico_ambulatorio FOREIGN KEY (cod_ambulatorio) REFERENCES ambulatorio(cod_ambulatorio);
ALTER TABLE medico ALTER COLUMN especialidade SET NOT NULL;
ALTER TABLE medico DROP CONSTRAINT un_medico_cpf;
ALTER TABLE medico ALTER COLUMN especialidade DROP NOT NULL;
ALTER TABLE medico ALTER COLUMN rg_medico SET DEFAULT 'SEM RG';
ALTER TABLE medico ALTER COLUMN rg_medico DROP DEFAULT;
ALTER TABLE medico RENAME COLUMN cpf TO cpf_medico;
ALTER TABLE medico RENAME TO medicos;
```

```
ALTER TABLE ambulatorio
ADD COLUMN nome_ambulatorio VARCHAR(30) NULL;

ALTER TABLE medicos
DROP CONSTRAINT pk_medico;

ALTER TABLE ambulatorio
DROP COLUMN nome_ambulatorio;
```

Domains

Os domains são tipos de dados definidos pelo utilizador de forma a padronizar e evitar repetições.

Um DOMAIN pode conter um valor default, restrições do tipo NOT NULL e CHECK.

Criando o DOMAIN:

```
CREATE DOMAIN do_salario numeric(9,2) NOT NULL  DEFAULT 1
    CONSTRAINT ck_salario_maior_zero CHECK (VALUE > 0);
```

Usando o DOMAIN:

```
CREATE TABLE empregado (
  matricula      INTEGER      NOT NULL,
  salario        do_salario
);
```

Índices

Os índices são utilizados, principalmente, para melhorar o desempenho do banco de dados (embora a utilização não apropriada possa resultar em uma degradação de desempenho);

Definidos sobre atributos para acelerar o acesso físico a dados;

São definidos automaticamente índices para atributos chave pri(pk, unique);

Para criar um índice

- CREATE [UNIQUE] INDEX nome_índice ON nome_tabela (nome_atributo_1[{, nome_atributo_n }]);

Para apagar um índice

- D

```
CREATE INDEX ind_medico_1
  ON medico (nome_medico);
```