

BASE DE DADOS



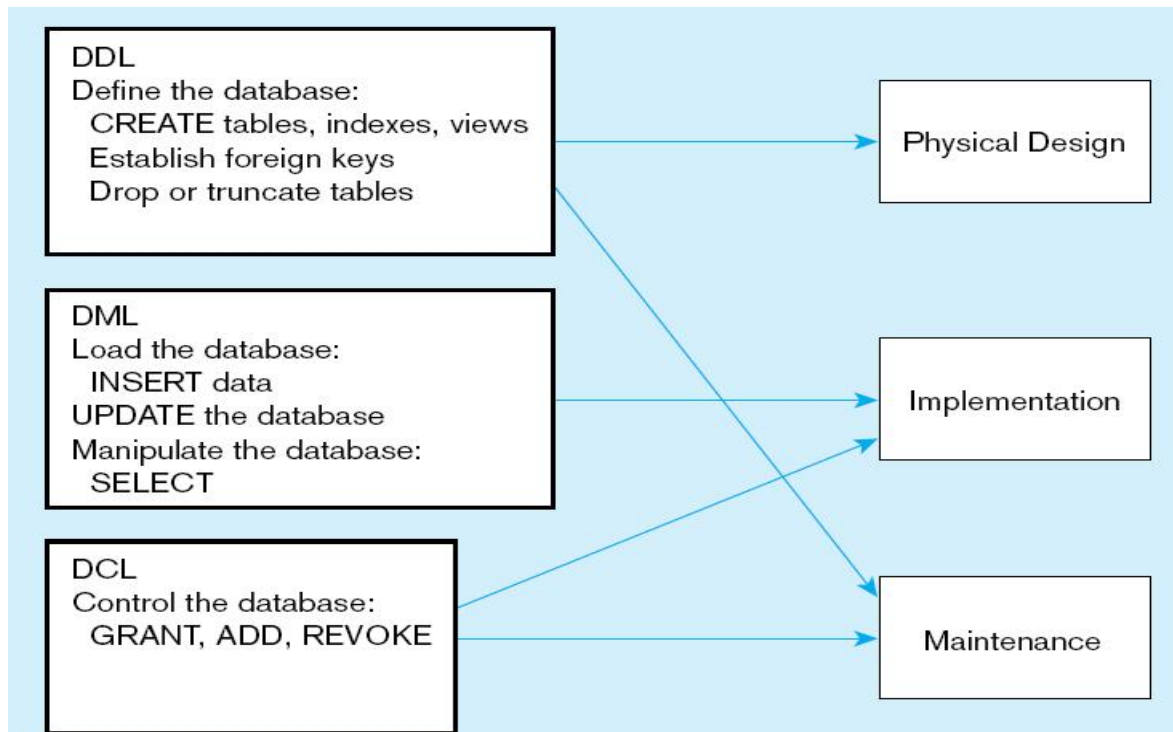
SQL
Structured Query
Language

Teóricas
Ano Lectivo 2018/2019

Linguagem SQL – DDL e DML

SQL- Structured Query Language

- ★ SQL é mais que uma linguagem de interrogação estruturada. Inclui características para a definição da estrutura de dados, para alterar os dados de uma base de dados, e para especificar esquemas de segurança. Estas características agrupam-se do seguinte modo:



Data Definition Language – Criar Tabelas

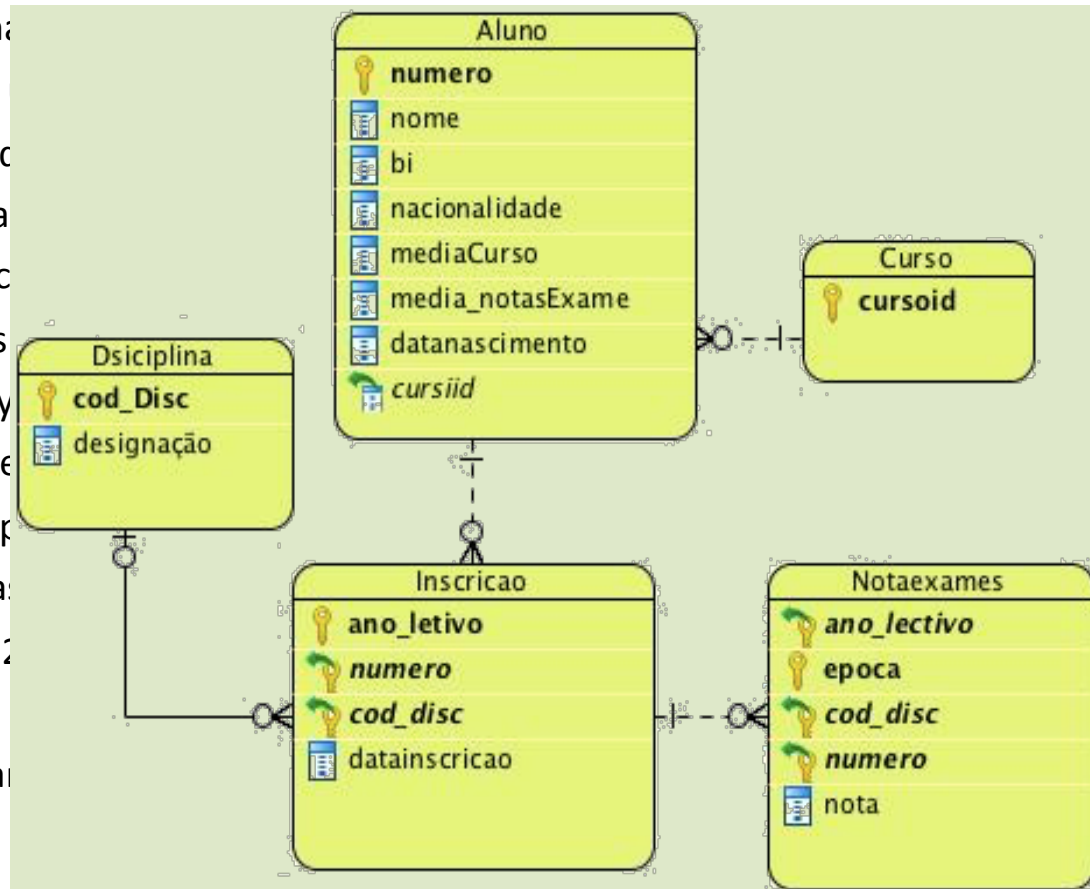
- ★ Na sua forma mais básica é preciso apenas indicar o nome da tabela, os nomes das várias colunas e o tipo de cada uma delas.

```
create table <nometabela> (  
    <nomecoluna>    <tipocoluna> >,  
    <nomecoluna>    <tipocoluna> );
```

Linguagem SQL – DDL e DML

Data Definition Language – Tipo de Dados

- **CHAR(n)** - Cadeia de caracteres de tamanho fixo n; por padrão o seu tamanho é um byte, ou seja, caso ele não receba valor no momento de sua utilização ele receberá seu tamanho padrão.
- **VARCHAR(n)**- Cadeia de caracteres com tamanho máximo n; e exige que um tamanho seja definido no momento de sua utilização. Seu tamanho máximo é variável todo espaço não utilizado é preenchido com zeros.
- **NCHAR e NVARCHAR2**- são tipos de dados para caracteres Unicode. O conjunto de caracteres de tipos de dados NCHAR e NVARCHAR2 é UTF8 e é especificado no momento da criação da tabela. AL16UTF16 e UTF8 são ambos tipos de dados para caracteres Unicode.
- **INTEGER** ou int - Números inteiros (4 bytes).
- **NUMBER(precisão, escala)** - Números reais. O valor de precisão varia de uma a trinta e oito (NUMBER(38)) porém independentemente do valor de precisão ocupará sempre o mesmo espaço na base de dados.
- **DATE** - data entre 4712-01-01 AC e 4712-12-31 AD, ano, mês, dia, hora, minuto e segundo.
- **TIMESTAMP**- Data + hora no mesmo campo.



Linguagem SQL – DDL e DML

Data Definition Language – Tipo de Dados

★ Exemplo

```
create table aluno (  
    numero integer ,  
    nome varchar (100),  
    bi integer,  
    nacionalidade varchar(30),  
    mediaCurso number (4 ,2),  
    media_notasExame number (4 ,2),  
    datanascimento date,  
    cursoid integer );
```



Data Definition Language – Valor por Omissão

- ★ Podem ainda ser definidos valores por omissão para cada coluna usando a palavra-chave **default**.

```
create table aluno (  
    numero integer ,  
    nome varchar (100),  
    bi integer,  
    nacionalidade varchar(30) default 'Portuguesa'  
    mediaCurso number (4 ,2) default 0,  
    media_notasExame number (4 ,2),  
    datanascimento date default Sysdate,  
    cursoid integer );
```



Data Definition Language - Restrições

- ✴ Em SQL podem ser definidas restrições de vários tipos.

Restrições de integridade: definem a chave primária e a chave estrangeira com a tabela e a chave primária referenciadas.

Restrições de valor: definem se os valores nulos não são permitidos, se são necessários valores únicos, e se apenas determinado conjunto de valores são permitidos numa coluna

- ✴ Uma restrição pode ser criada ao mesmo tempo em que a tabela é criada ou pode ser adicionada à tabela posteriormente.
- ✴ Existem dois níveis em que uma restrição é definida:
 - **Nível da coluna** - refere-se apenas a uma coluna e é descrita em frente à coluna em causa);
 - **Nível da tabela**- refere-se a mais do que a uma coluna e fica separada da definição das colunas).

Restrições – Chave Primária

- ★ Se a chave primaria é composta por um atributo, devemos usar uma restrição ao nível da coluna;

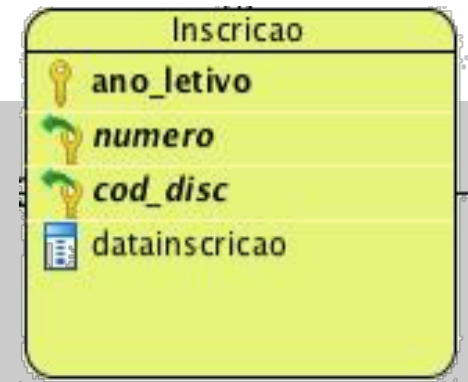
```
create table aluno (  
    numero integer Primary Key,  
    nome varchar (100),  
    bi integer  
    nacionalidade varchar(30) default 'Portuguesa'  
    mediaCurso number (4 ,2) default 0,  
    media_notasExame number (4 ,2),  
    datanascimento    date default Sysdate,  
    cursoid integer );
```



Restrições – Chave Primária

- ★ Se a chave primaria é composta por mais do que um atributo devemos usar uma restrição ao nível da tabela.

```
create table inscricao (  
    ano_letivo number(4),  
    numero integer,  
    cod_disc number(4),  
    datainscricao date,  
    constraint pk_cexamef Primary key (ano_letivo,numero,cod_disc));
```

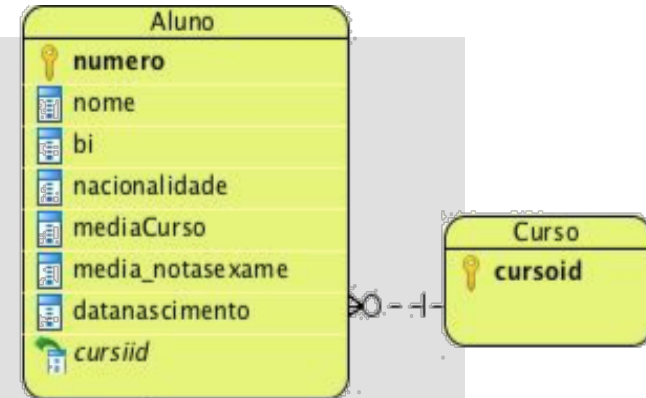


Podemos e devemos sempre dar nomes às restrições para que seja mais fácil identificar a razão pela qual a inserção de dados falha.

Restrições – Chaves Estrangeira

- ★ Uma restrição do tipo **foreign key** permite declarar chaves estrangeiras.
- ★ Uma chave estrangeira deve sempre referenciar uma chave primária ou única.

```
create table aluno (  
    numero integer Primary key,  
    nome varchar (100),  
    bi integer,  
    nacionalidade varchar(30) ) default 'Portuguesa',  
    mediaCurso number (4 ,2) default 0,  
    media_notasExame number (4 ,2),  
    datanascimento date default Sysdate,  
    cursoid integer references curso(id) );
```



atributo que é
chave primária
na tabela curso

Restrições – Chave Estrangeira

- ★ No caso da chave estrangeira ser **composta ou concatenada** (mais de uma coluna) usa-se uma restrição de tabela:

```
create table notaExames (
```

```
  ano_letivo number(4),
```

```
  epoca varchar(20),
```

```
  cod_disc number(4),
```

```
  numero integer,
```

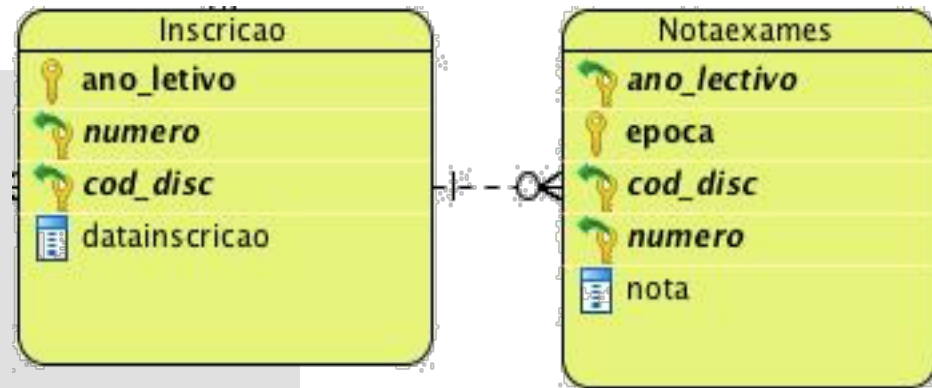
```
  nota number(4,2),
```

```
  .....
```

```
Constraint fk_notasexame Foreign Key
```

```
(ano_letivo, numero, cod_disc) references
```

```
inscricas (ano_letivo,numero,cod_disc));
```



Restrições – Chave Estrangeira

```
create table aluno (  
  numero integer primary key,  
  nome varchar (100),  
  bi integer,  
  nacionalidade varchar(30) ) default 'Portuguesa',  
  mediaCurso number (4 ,2) default 0,  
  media_notasExame number (4 ,2),  
  datanascimento date,  
  cursoid integer references curso(id) on delete set null on update  
cascade, ... );
```

Neste exemplo estamos a definir que no caso de um curso ser apagado os alunos que pertençam a esse curso devem ficar com o curso a null. No caso da chave primária do curso ser modificada, essa modificação deve propagar-se e modificar também o curso da tabela aluno.

Restrições – Check

- ★ As restrições do tipo **CHECK** permitem garantir que uma ou mais colunas seguem uma determinada regra que pode ser expressa como uma expressão matemática.

```
create table aluno (  
    numero integer primary key,  
    nome varchar (100),  
    bi integer,  
    nacionalidade varchar(30) ) default 'Portuguesa',  
    mediaCurso number (4 ,2) default 0,  
    Constraint ck_media check (mediaCurso >= 0),  
    media_notasExame number (4 ,2),  
    datanascimento date  
    ..... );
```

Restrições – Check

- ★ No caso da restrição abranger mais de uma coluna temos de usar uma restrição de tabela.

```
create table aluno (  
    numero Primary Key,  
    nome varchar (100) ,  
    bi integer ,  
    nacionalidade varchar(30) default 'Protuguesa',  
    mediaCurso number (4 ,2) default 0,  
    media_notasExame number (4 ,2),  
    datanascimento date,  
    constraint ck_mediaExames check (media_notasExame < médiaCurso)  
    ..... );
```

Restrições – Not Null

- ★ Para garantir que uma coluna não tenha valores nulos podemos usar uma restrição do tipo **not null**

```
create table aluno (  
    Nuero Primary Key,  
    nome varchar (100) not null,  
    bi integer ,  
    nacionalidade varchar(30) default 'Protuguesa',  
    mediaCurso number (4 ,2) default 0,  
    media_notasExame number (4 ,2),  
    constraint ck_mediaExames check (media_notasExame < médiaCurso)  
    datanascimento date,  
    ..... );
```

Restrições – Valores únicos

- ★ 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.

```
create table aluno (  
    numero Primary Key,  
    nome varchar (100) not null,  
    bi integer unique,  
    nacionalidade varchar(30) default 'Protuguesa',  
    mediaCurso number (4 ,2) default 0,  
    media_notasExame number (4 ,2),  
    constraint ck_mediaExames check (media_notasExame < médiaCurso),  
    datanascimento date default sysdate,  
    ..... );
```


Modificação de Tabelas

- ★ É possível alterar tabelas

- Acrescentar novas colunas, alterar o tipo de dados e eliminar colunas;
- Acrescentar ou retirar *constraints*.

- ★ **Acrescentar uma coluna**

- **ALTER TABLE** Aluno **ADD** (email varchar(100))

- ★ **Eliminar uma coluna**

- **ALTER TABLE** Aluno **DROP** column email

ATENÇÃO

- ★ **Alterar o tipo de dados de uma coluna**

- **ALTER TABLE** Aluno **MODIFY** (email varchar(75))

Para apagar uma Tabela
DROP TABLE
<nometabela>

Modificação de Tabelas

★ Eliminar uma constraint

- ALTER TABLE Aluno **DROP CONSTRAINT** FK_Nota_id_aluno_Aluno

★ Acrescentar uma constraint

- ALTER TABLE Aluno **ADD CONSTRAINT** FK_Nota_id_aluno_Aluno FOREIGN KEY (id_aluno) REFERENCES Aluno(id_aluno)

★ Considerações:

- A sintaxe dos comandos pode variar de SGBD para SGBD;
- Nem todos os SGBD permitem todas as versões do comando ALTER TABLE;
- Quando se adiciona uma *constraint*, terá sempre de ser no formato de uma table_constraint;
- **A alteração de tipo de dados de uma coluna não pode violar regras de integridade.**

Data Manipulation Language

- ★ Data Manipulation Language (DML) é utilizada para efetuar operações de seleção, ordenação, cálculo de informação guardada em tabelas, entre outras.

- ★ COMANDOS:
 - **INSERT** – inserir dados numa tabela;
 - **UPDATE** – atualiza os dados existentes numa tabela;
 - **DELETE** – elimina registos numa tabela;
 - **SELECT**- recuperar dados da base de dados.

Manipulação de Dados - **INSERT**

- ✳ O comando insert tem como função permitir inserir registo de dados em tabelas.


```
insert into <tabela> <lista de atributos>  
values <Conjunto de tuplos>
```

- ✳ Se forem indicadas o nome das colunas em que queremos inserir os dados, as restantes ficarão com o seu valor por omissão ou nulas.
- ✳ Podemos não indicar o nome das colunas. Neste caso somos obrigados a seguir a mesma ordem pela qual criamos a tabela.

```
INSERT INTO aluno (numero,nome,bi) VALUES (12,'Joao', 3333);
```

```
INSERT INTO aluno VALUES (1234 , ' Joao ' , 777777,'Espanhola', 12.50, 12,4,'12-12-1980',1));
```

```
INSERT INTO aluno VALUES (12 , ' Joao ' ,1234, default, default,12.6,default, 2) ;
```

Aluno	
	numero
	nome
	bi
	nacionalidade
	mediaCurso
	media_notasExame
	datanascimento
	cursiid

Manipulação de Dados- **UPDATE**

- ★ O comando update permite modificar os dados numa tabela.

determina sobre que
tabela se vão executar
as alterações.

```
update <tabela>  
set <Atributo> = <Expressão>, <Atributo> =  
    <Expressão>, ...  
where <Condição>
```

indica pares de colunas
e valores a atribuir a
colunas.

indica sobre que registo da tabela serão
executadas as alterações. A condição pode
ser uma subconsulta).

Manipulação de Dados- **UPDATE**

★ Exemplos

```
UPDATE Aluno  
SET numero=1234571  
WHERE nome='João'
```

```
UPDATE Aluno  
SET mediacurso=mediacurso * 1.2
```

```
UPDATE Aluno  
SET mediacurso=mediacurso * 1.2, numero = 123456  
WHERE nome='João'
```

Manipulação de Dados - **DELETE**

- ✳ Usa-se o comando delete para remover registos das tabelas.

delete from <tabela>
where <Condição>

- ✳ Permite apagar apenas registos inteiros. Não é possível apagar um campo com o Delete.

```
DELETE FROM aluno WHERE mediacurso > 12;
```

```
DELETE FROM aluno WHERE numero = 1234;
```

```
DELETE FROM aluno;
```