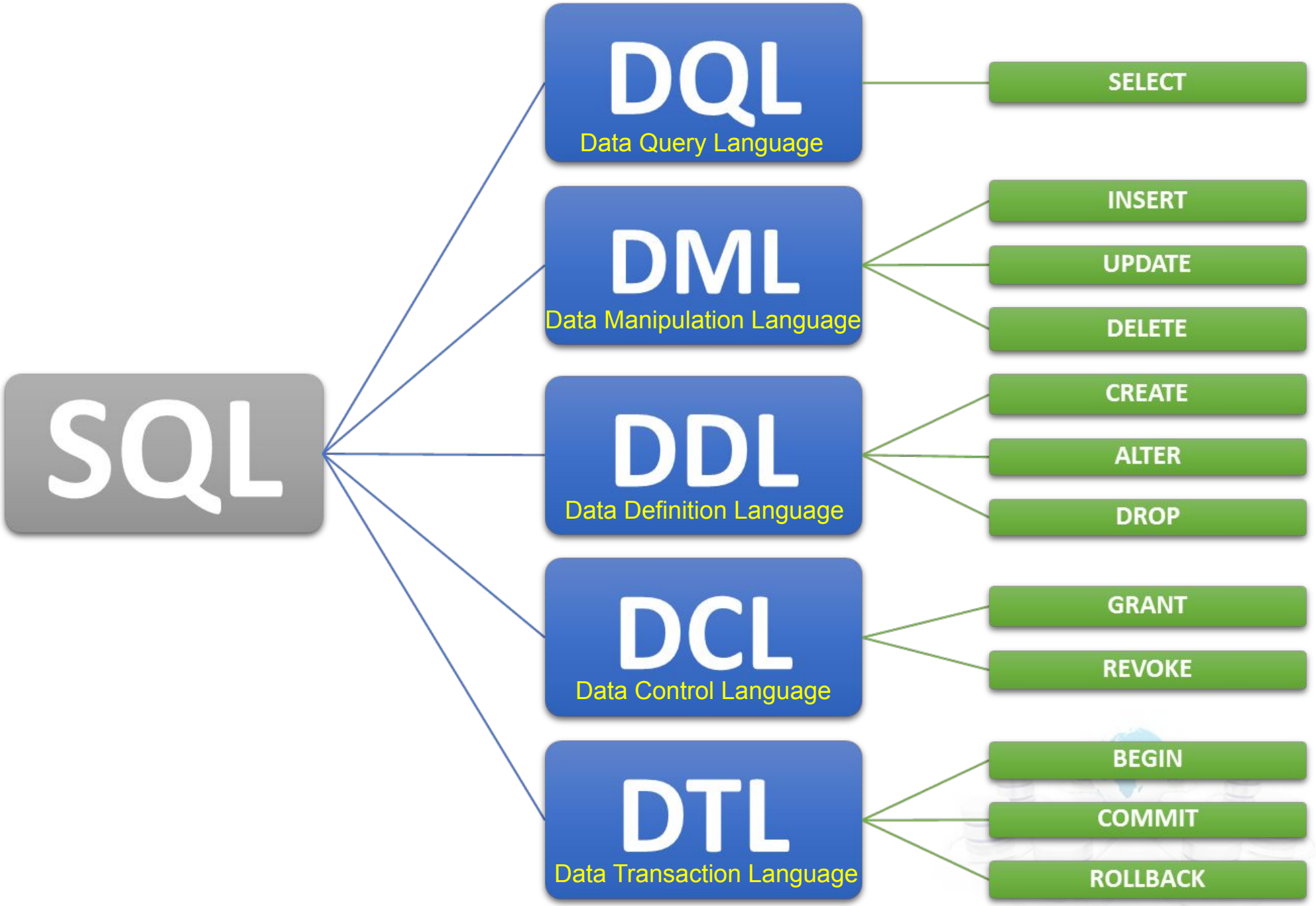
Banco de Dados I

SQL

Structured Query Language

Professor
Marcus Aurelius

SQL



SQL

Entradas SQL

- Correspondem a sequências de comandos SQL;

- Um comando é composto por uma sequências de símbolos:
  - palavra-chave;
  - identificador
    - começam com letras (a-z) ou sublinha (_);
    - tem o máximo de 63 caracteres;
  - identificador entre aspas;
  - literal (constante);
  - símbolo especial;
  - A linha de comando é encerrada com ponto-e-vírgula (;);

- Os símbolos são separados por espaços;
  - espaço;
  - tabulação
  - nova linha;

Exemplo:

- SELECT * FROM TABELA_ALUNO;

- UPDATE TABELA_ALUNO SET IDADE = 25;

- INSERT INTO TABELA_ALUNO VALUES (3, 'Joaquim Lima Borges');

SQL

Entradas SQL

- Comentários
  - --;
    - converte uma linha em comentário;
  - /* */
    - define um bloco de comentário

Exemplo:

```
-- select * from aluno;

/*
update aluno set situacao = “aprovado”;
*/
```

SQL

Tabelas

- são composta por linhas e colunas;
- a quantidade de colunas (campos) é fixa, definida na construção da tabela;
- a quantidade de linhas (registro) varia em função das operações de inclusão e exclusão de dados;
- cada coluna tem um tipo, correspondente ao tipo do dado que nela será armazenado;

Criando tabelas

Sintaxe:

```
create table <nome da tabela> (
        <campo> <tipo>,
        <campo> <tipo>,
        ...
);
```

Exemplo:

```
create table curso (
        cur_codigo integer not null,
        cur_nome varchar(50) not null,
        cur_cargaHoraria integer default 60
);
```

SQL

Tabelas

- **not null**
  - define um campo que não pode ter valor;
  - trata-se de uma restrição de coluna;

Sintaxe:

```
create table <nome da tabela> (
    ...
    <campo> <tipo> not null,
    <campo> <tipo>,
    <campo> <tipo>,
    ...
);
```

Exemplo:

```
create table aluno (
    ...,
    alu_matricula integer not null,
    alu_notaParcial float;
    ...
);
```

SQL

Tabelas

- **default**
  - define um valor padrão para um campo não nulo;

Sintaxe:

```
create table <nome da tabela> (
    ...
    <campo> <tipo> default <valor padrão>,
    <campo> <tipo>,
    ...
);
```

Exemplo:

```
create table aluno (
    ...,
    alu_notaParcial float default 0.0;
    ...
);
```

Tabelas

- **unique**
  - garante que o valor armazenado em um campo será único dentre todos os demais registros inseridos na tabela;
  - pode ser definido também em forma de “constraint”. Neste caso pode-se listar vários campos simultaneamente, separando os nomes com vírgula. Assim a combinação dos valores desses campos tem que ser única;

Sintaxe:

```
create table <nome da tabela> (
        ...
        <campo> <tipo> unique,
        <campo> <tipo>,
        ...
);
```

Exemplo:

```
create table aluno (
        ...,
        alu_eMail varchar(60) unique;
        ...
);

create table aluno (
        ...,
        alu_eMail varchar(60);
        ...,
        unique(alu_eMail)
);
```

SQL

Tabelas

- **primary key**
  - indica que uma coluna, ou grupo de colunas, são usados como identificador único de uma linha de dados na tabela. Esses dados devem ser únicos e não nulos;

    Sintaxe:

```
create table <nome da tabela> (
        ...
        <campo> <tipo> primary key,
        <campo> <tipo>,
        ...
);

create table <nome da tabela> (
        ...
        <campo> <tipo>,
        ...
        primary key (<campo chave>)
);

create table <nome da tabela> (
        ...
        <campo> <tipo>,
        ...
        constraint <nome da restrição> primary key (<campo chave>)
);
```

SQL

Tabelas

- **primary key**
  - indica que uma coluna, ou grupo de colunas, são usados como identificador único de uma linha de dados na tabela. Esses dados devem ser únicos e não nulos;

    Exemplo:

```
create table aluno (
        alu_matricula integer primary key,
        alu_eMail varchar(60) unique;
        ...
);

create table aluno (
        alu_matricula integer not null,
        alu_eMail varchar(60);
        ...
        primary key (alu_matricula)
);

create table aluno (
        alu_matricula integer not null,
        alu_eMail varchar(60);
        ...
        constraint pk_aluno primary key (alu_matricula)
);
```

SQL

Tabelas

- **foreign key**
  - especifica que o valor de uma coluna, ou de um grupo de colunas, deve combinar com o valor existente em algum registro de outra tabela, mantendo o referencial de integridade entre as duas tabelas;
  - caso a chave referenciada seja composta, é necessário listar os campos na mesma ordem da tabela referenciada, separando-os com uma vírgula;

    Exemplo:

```
create table curso (
        cur_codigo integer not null,
        cur_nome varchar(50) not null unique;
        constraint pk_curso primary key (cur_codigo)
);

create table aluno (
        ...
        alu_curso integer references curso(cur_codigo),
        ...
);

create table aluno (
        ...
        alu_curso integer not null,
        ...
        constraint pk_aluno primary key (alu_matricula),
        constraint fk_aluno_curso (alu_curso) references curso(cur_codigo)
);
```

SQL

Tabelas

- **on delete**
  - define o que acontecerá com os registros que referenciam registros de outras tabelas quando estes registros referenciados forem excluídos.
    - no action
      - não conclui a operação, e gera uma mensagem de erro. É a opção padrão do banco de dados;
    - cascade
      - quando o registro referenciado é excluído, as linhas de outras tabelas que o referenciam são também excluídos de suas respectivas tabelas;
    - set null
      - atribui o valor “null” aos campos referenciados quando o campo referência é excluído;
    - set default
      - atribui o valor default definido para o campo referenciado quando o campo referência é excluído. É necessário que um valor padrão seja especificado para este campo, caso contrário um erro será gerado;

SQL

Tabelas

- **on delete**

Exemplo:

```
create table curso (
        cur_codigo integer not null,
        cur_nome varchar(50) not null unique;
        constraint pk_curso primary key (cur_codigo)
);

create table aluno (
        ...
        alu_curso integer references curso(cur_codigo),
        ...
);

create table aluno (
        ...
        alu_curso integer not null,
        ...
        constraint pk_aluno primary key (alu_matricula),
        constraint fk_aluno_curso (alu_curso) references curso(cur_codigo) on delete
cascade
);
```

Tabelas

Removendo tabelas

Sintaxe:

drop table <nome da tabela>;

Exemplo:

drop table curso;

Tabelas

Inserindo dados em uma tabela

Sintaxe:

insert into <nome da tabela> values (<valorCampo1>,<valorCampo2>,...,<valorCampoN>);

Obs: É necessário conhecer a ordem dos campos existente na tabela.

Exemplo:

```
insert into curso (cur_codigo,cur_nome) values
        (1,'Informática'),
        (2,'Análise e Desenvolvimento de Sistemas');

insert into curso values
        (3,'Subsequente de Informática'),
        (4,'Integrado de Informática');
```

Obs: É possível listar os campos fora da ordem que que se encontram na tabela, desde que os respectivos valores estejam na mesma ordem dos campos relacionados na linha de comando;

Obs: É possível inserir dados apenas em campos específicos desde que os campos omitidos não sejam de preenchimento obrigatório.

Tabelas

Consultando dados existentes nas tabelas

Sintaxe:

select * from <nome da tabela>;

select (<campo1>,<campo2>,<campo3>,...,<campoN>) from <nome da tabela>;

Exemplo:

select * from curso

select cur_nome from curso order by cur_nome;

Tabelas

Atualizando dados existentes nas tabelas

- Atualiza os valores dos campos de uma tabela;
- Permite atualizar o valor de um ou mais campos na mesma linha de comando;
- Permite o uso de expressão matemática para o cálculo do novo valor;

Sintaxe:

update <nome da tabela> set <nome do campo> = <novo valor> where <campoChave> = <valor desejado>;

Obs: Caso o valor da chave de pesquisa não seja informado, todos os campos listados na linha de comando receberão o novo valor.

Exemplo:

```
update aluno set alu_mediaSemestre1=10.0,
        alu_mediaSemestre2 = 8.9,
        alu_mediaSemestre3 = 9.4,
        alu_mediaSemestre4 = 8.1,
        alu_mediaSemestre5 = 8.6,
        alu_mediaSemestre6 =9.8
    where alu_matricula = 1;
```

Tabelas

Excluindo dados existentes nas tabelas

- Remove uma linha inteira de dados da tabela;
- Permite excluir linhas de dados a partir de uma condição definida;
- Caso não seja definida uma condição, todos as linhas de dados serão apagadas da tabela;;

Sintaxe:

delete from <nome da tabela>;

delete from <nome da tabela> where <nome do campo> = <valor pesquisado>;

Obs: Caso o valor da chave de pesquisa não seja informado, todos os campos listados na linha de comando receberão o novo valor.

Exemplo:

delete from aluno where alu_matricula = 4;

Tabelas

Modificando tabelas

É possível, em um banco de dados alterar a estrutura de uma tabela, através das funcionalidades::

- adicionar coluna;
- remover coluna;
- adicionar restrição (constraint);
- remover restrição (constraint);
- mudar o valor default de uma coluna;
- mudar o tipo de uma coluna;
- renomear uma coluna;
- renomear uma tabela;

Tabelas

Modificando tabelas - adicionando coluna

- o campo é inicialmente preenchido com o valor default definido para ela, ou null caso caso o valor default não tenha sido definido;
- é possível também adicionar constraints a esse novo campo, na linha de comando da sua criação;

Sintaxe:

alter table <tabela> add column <campo> <tipo>

Exemplo:

...
alter table aluno add column alu_sexo varchar(1),
...

Tabelas

Modificando tabelas - removendo coluna

- a coluna é removida da tabela e os respectivos dados e restrições são excluídos;
- caso a coluna seja referenciada por outras tabelas, “cascade” adicionada à linha de comando autoriza a exclusão e exclui também todos os campos dependentes, em suas respectivas tabelas;

Sintaxe:

alter table <tabela> drop column <campo> [cascade];

Tabelas

Modificando tabelas - adicionar restrições (constraints)

- adiciona restrições a uma coluna já existente na tabela;

Sintaxe:

- adicionando restrições à tabela

  alter table <tabela> add constraint <nome da restrição> unique (<coluna>);

  alter table <tabela> add foreign key (<coluna>) references <tabela>(<coluna referenciada>);

- adicionando restrições “not null” a uma coluna

  alter table <campo> alter column <coluna> set not null;

Tabelas

Modificando tabelas - remover restrições (constraints)

- remove restrições já existentes na tabela;
- é necessário conhecer o nome da restrição;
- usa-se “cascade” no caso de apagar restrições que são referenciadas por outras restrições, As restrições que fazem referência à que vai ser excluída serão excluídas também;

Sintaxe:

- removendo restrições da tabela

```
alter table <tabela> drop constraint  <nome da restrição>;
```

Tabelas

Modificando tabelas - modificando o valor default da coluna

- define um novo valor default para a coluna;

Sintaxe:

- definindo novo valor default

  alter table <tabela> alter column <coluna> set default <valor default>;

- removendo o valor default de uma coluna

  alter table <tabela> alter column <coluna> drop default;

Tabelas

Modificando tabelas - modificando o tipo da coluna

- converte a coluna para receber um novo tipo de dado;
- a operação será realizada se os dados da coluna forem compatíveis com o novo tipo;

Sintaxe:

- definindo novo tipo

```
alter table <tabela> alter column <coluna> type <novo tipo>;
```

Tabelas

Modificando tabelas - renomeando uma coluna

- modifica o nome de uma coluna;

Sintaxe:

- Atribuindo o novo nome à coluna

    alter table <tabela> rename column <nome da coluna> to <novo nome da coluna>;

Tabelas

Modificando tabelas - renomeando uma tabela

- modifica o nome da tabela;

Sintaxe:

- Atribuindo o novo nome da tabela

    alter table <tabela> rename to <novo nome>;

Referências

DEVMEDIA. **Guia completo de SQL**. Disponível em: <https://www.devmedia.com.br/guia/guia-completo-de-sql/38314#plsql>. Acesso em: jan 2021.

ELMASRI, Ramez. Sistemas de banco de dados. 6. ed. São Paulo : Pearson Addison Wesley, 2011.

GOMES, Eduardo Henrique. **Introdução:** Uma visão geral de Banco de Dados, seus fundamentos, características e sua terminologia. Disponível em: <http://ehgomes.com.br/disciplinas/bdd/introducao.php>. Acesso em: jan 2021.

GOOGLE. Pesquisa. Disponível em: <www.google.com.br>. Acesso em: jan

POSTGREESQL. **PostgreSQL 13.1 Documentation**. Disponível em: <https://www.postgresql.org/docs/current/index.html>. Acesso em: jan 2021.

ROSSITO NETO, Thomaz Antônio. **Comandos DML, DDL, DCL, TCL, SQL SERVER**. Disponível em: <https://www.thomazrossito.com.br/comandos-dml-ddl-dcl-tcl-sql-server/>. Acesso em: jan 2021.

Perguntas?