SQL Tutorial

Introdução à Linguagem de Consulta Estruturada

Objetivos

Entender a Linguagem de Definição de Dados (DDL)

Entender a Linguagem de Manipulação de Dados (DML)

- SELECT, INSERT, UPDATE, DELETE

Utilizar a declaração **SELECT FROM WHERE** para construção de consultas

- Usar funçãoes de agregação (SUM, COUNT, AVG, ...)

Linguagem de Definição de Dados

Permite especificar as entidades e relacionamentos, bem como os atributos das entidades, incluindo:

- O banco de dados (schema) que contem as entidades e relacionamentos
- Os tipos de dados (domain) de cada atributo
- As restrições impostas às entidades (chave primaria), aos relacionamentos (chaves estrangeiras) e aos atributos (nulidade), que matêm a integridade do banco de dados

Tipos de Dados em SQL

Tipo	**Descrição**
CHAR(n)	Cadeia de caracteres de tamanho fixo. Tamanho especificado por **n**
VARCHAR(n)	Cadeia de caracteres de tamanho variável. Tamanho especificado por **n**
INTEGER	Inteiro. Ocupa 4 Bytes (32 bits) e permite mais de 4 bilhões de números distintos
SMALLINT	Inteiro. Ocupa 2 Bytes (16 bits) e permite 65536 números distintos
FLOAT(M,D)	Número de ponto flutuante (decimal). Total de M digitos com D digitos decimais.
DOUBLE(M,D)	Número de ponto flutuante com precisão dupla.

- Similar aos tipos de dados utilizados nas liguagens de programção.

CREATE DATABASE

Um banco de dados (schema) no MySQL é criado com o comando **CREATE DATABASE**:

- **create database** [*database name*]

Examplo

- **create database** *banco*

CREATE TABLE

Um tabela (entidade) no SQL é criada utilizando o comando **CREATE TABLE**:

- **create table** [*tablename*] *(A_1 T_1,A_2 T_2, … A_n T_n,*
  (integrity-constraint$_1$),
  …,
  (integrity-constraint$_k$));
- A_i é um campo (atributo) da tabela.
- T_i é o tipo de dado (domain) do campo A_i

Examplo

```
create table student (
   flashlineID    char(9) not null,
   name           varchar(30),
   age            integer,
   department     varchar(20),
   primary key (flashlineID));
```

Constraint

DROP e ALTER TABLE

- DROP TABLE exclui a tabela e todos os dados contidos nela.

```
drop table tabela <ação>
```

- **ALTER TABLE** é usado para adicionar, modificar ou excluir campos das tabelas:

```
alter table tabela <ação>
```

Ação pode ser:

```
add campo
drop campo
add primary key (campo,…)
drop primary key
```

Modificando o Banco de Dados

Comandos:

Incluir registro	**INSERT INTO** *tabela* **VALUES** *(Val$_1$, Val$_2$, … , Val$_n$)*
Alterar registro(s)	**UPDATE** *tabela* **SET** *A$_1$=val$_1$, A$_2$=val$_2$, …, A$_n$=val$_n$* **WHERE** *predicado*
Excluir registro(s)	**DELETE** **FROM** *tabela* **WHERE** *predicado*

Inclusão

Adicionar um novo registro na tabela ***student***

```
insert into student
values('999999999', 'Mike', 18, 'Math');

insert into student(flashlineID, name, age,
  department)
values('999999999', 'Mike', 18, 'computer science');
```

Adicionar uma novo registro na tabela ***student*** com campo ***age*** preenchido com **null**

```
insert into student
Values('999999999', 'Mike', null, 'Math');
```

Alteração

Atualizar todos os departamentos para 'computer science' da tabela **student**

```
update student
    set department = 'computer science';
```

Na tabela **conta** (numero_conta, *nome_agencia, saldo),* aumente o valor do saldo de todas as contas em 6%

```
update conta
set saldo = saldo * 1.06;
```

Exclusão

Excluir todos os registros da tabela **students**

```
delete from student;
```

Excluir todos os registros da tabela **students** em que o departamento seja computer science

```
delete from student
 where department = 'computer science';
```

Consultas (Query)

Uma típica consulta em SQL tem a forma:

select A_1, A_2, …, A_n
from $tabela_1$, $tabela_2$, …, $tabela_m$
where R

- A_i são os atributos (campos)
- $tabela_i$ são as tabelas
- R são as restrições (condições da consulta)

A consulta é equivalente a seguinte expressão em algebra relacional:

$$\prod_{A_1, A_2, \ldots, A_n} \left(\sigma_R \left(tabela_1, tabela_2, \ldots, tabela_m \right) \right)$$

Example

```
select flashlineID, name
  from student
 where department = 'computer science'
```

SELECT – Registros Duplicados

SQL não remove automaticamente os registros duplicados no resultado da consulta.

Para eliminar as duplicidades devemos utilizar a palavra reservada **distinct** após o **select**.

- Exemplo: Encontrar os nomes de todos os alunos da universidade e remover as duplicidades

```
select distinct name
   from student;
```

SELECT – Expressão *as*

Um asterisco em um comando **select** retorna todos os campos da tabela.

```
select * from student
```

Um campo pode receber um apelido utilizando a expressão ***as***

- Exemplo

```
select FlashlineID as ID
  from student
```

Nota: ***as*** também pode ser usado para apelidar as tabelas

```
select s.name as myname
  from student as s
```

WHERE

A cláusula **WHERE** define as condições (restrições) que satisfazem a consulta

- Podem utilizar os operadores de comparação e lógicos:
  - Operador de comparação: <, <=, >,>=, =, <>
  - Operador lógico: and, or, not

- Exemplo:Encontrar os nomes dos estudantes de computer science department com idade inferior a 18 anos

```
select names
  from student
 where department = 'computer science'
   and age < 18;
```

ORDER BY

A cláusula **ORDER BY** ordena o resultado de uma consulta

- Podem utilizar os operadores ASC (ascendente) ou DESC (descendente), ou ainda, uma combinação dos dois.

- Exemplo:Encontrar o nome e a idade dos estudantes do departamente de computer science com idade inferior a 18 anos. Ordenar o resultado por nome ascendente e idade descendente.

```
select names, age
  from student
 where department = 'computer science'
   and age < 18
 order by name asc, age desc;
```

Agregação

As funções de agregação operam nos campos de valor de um conjunto de registros e retornam o valor computado com os dados

avg(*campo*): valor médio
min(*campo*): campo com menos valor
max(*campo*): campo com o maior valor
sum(*campo*): soma dos valores dos campos
count(*campo*): quantidade de registros no conjunto de dados

Para remover as duplicidades de valor no campo do conjunto de dados, utilize a palavra **distinct** antes do campo:

avg(**distinct** *campo*)

Agregação – Valor NULL

Encontrar o estudante mais novo da universidade

```
select  *
  from student
 where age = min(age)
```

- A consulta acima ignora os valores do campos **age** iguais a **NULL**
- O resultado será **NULL** se todas as idades estiverem com valor **NULL**

As funções de agregação, com exceção de **count**, ignoram os campos com valor igual a **NULL**.

Agregação – GROUP BY

- **GROUP BY** opera da seguinte forma
  1. Agrupa os dados do conjunto em subconjuntos de campos do **select**
  2. Agrega os valores de cada subconjunto
  3. Retorna o valor agregado para os subconjuntos de campos

- Exemplo: Listar cada departamento e a quantidade de estudantes em cada um deles.

```
select department, count(name) as number
  from student
 group by department
```

Nota: Se um **select** contem uma função de agregação, então todos os campos que não participam da agregação têm que estar relacionados no **group by**.

No exemplo acima, como department não participa da função de agregação, então ele tem que estar no **group by**.

Agregação – HAVING

A cláusula **HAVING** filtra (impõe uma condição) o resultado de uma agragação.

Exemplo: Listar cada departamento e a quantidade de estudantes em cada um deles, somente para os departamentos com mais de dois estudantes.

```
select department, count(name) as number
  from student
 group by department
 order by department
having count(distinct name) > 2
```