

SQL – PARTE 1

Sérgio Mergen

História

- IBM Sequel language desenvolvido como parte do Sistema R
- Renomeado para Structured Query Language (SQL)
- Padrões ANSI e ISO:
 - SQL-86
 - SQL-89
 - ...
 - SQL:2011
 - SQL:2016
- Sistemas comerciais oferecem grande parte (se não todos) recursos do SQL-92, além de recursos proprietários e oriundos de padrões mais recentes.

SQL

- Subconjuntos da SQL
 - DDL – Linguagem de Definição de dados
 - DML – Linguagem de Manipulação de dados
 - DQL – Linguagem de Consulta de dados
 - DCL – Linguagem de Controle de dados
 - DTL – Linguagem de Transação de dados

DDL - Data Definition Language

- Permite a especificação de relações, e todas informações referentes a elas, como:
 - O esquema da relação.
 - Domínio de valores dos atributos.
 - Restrições de Integridade
 - Índices.

Manipulação dos Metadados do Banco

- Comandos DDL – Data Definition Language
 - Criação de tabela / visão / índice
 - Alteração de tabela / índice
 - Remoção de tabela / visão / índice

Criação de Tabela

- Uma tabela é criada com o comando **create table**:

```
create table t (A1 D1, A2 D2, ..., An Dn,  
                                     (restrição-de-integridade1), ...,  
                                     (restrição-de-integridadek))
```

- *t* é o nome da relação
 - cada *A_i* é o nome de um atributo da tabela *t*
 - *D_i* é o tipo de dados aceito pelo atributo *A_i*
- Exemplo: criação da tabela para funcionários

```
create table func (  
    idFunc    int    not null,  
    nome      varchar(50) not null,  
    salario    decimal(8,2),  
    idDepto   int    )
```

Restrições de Integridade

- **not null**
- **primary key** (A_1, \dots, A_n)
- **foreign key** ...

Exemplo: Declarar *idFunc* como a chave primária da tabela *func*

```
create table func (  
    idFunc          int,  
    nome            varchar(50) not null,  
    salario         decimal(8,2),  
    idDepto         int,  
    primary key (idFunc)  
    foreign key(idDepto) references depto(idDepto) )
```

Conforme SQL-92, atributos declarados como **primary key** são automaticamente **not null**

Tipos de dados em SQL

- “Alguns” tipos disponíveis
 - INT, INTEGER, SHORT, LONG, NUMBER, NUMERIC, SMALLINT, INTEGER2, INTEGER4
 - CHAR, VARCHAR, ALPHANUMERIC, CHARACTER, STRING
 - DATE, TIME, DATETIME , TIMESTAMP
 - DECIMAL, REAL, DOUBLE , FLOAT, FLOAT4, FLOAT8
 - BINARY, BIT, BYTE, YESNO, BOOLEAN
 - CURRENCY, MONEY
 - TEXT, OLEOBJECT, LONGTEXT, MEMO, NOTE, GENERAL

Remoção de Tabela

- **drop table:** remove uma tabela.
- Ex. remoção da tabela de funcionários
drop table *func*

Alteração de Tabela

- **alter table:** altera as propriedades de uma tabela:
- Para adição de uma coluna

alter table *r* add *A D*

- - *A* é o nome do atributo a ser adicionado na relação *r*
 - *D* é o tipo de dados de *A*

- Ex. criação do atributo email

alter table *func* add *email varchar(50)*

Obs. Todas tuplas recebem *null* como valor do novo atributo.

Alteração de Tabela

- **alter table:** altera as propriedades de uma tabela:
- Para remoção de uma coluna

alter table *r* drop *A*

- - *A* é o nome do atributo a ser removido da relação *r*

- Ex. remoção do atributo email

alter table *func* drop *email*

obs. Chaves primárias não podem ser excluídas

Alteração de Tabela

- **alter table:** altera as propriedades de uma tabela:
- Para adicionar novas restrições:

alter table *r* add constraint *N R*

- *N* é um nome dado à nova restrição
- *R* define a restrição.

Alteração de Tabela

- **alter table:** altera as propriedades de uma tabela:

Ex. Criação de uma restrição de domínio:

```
alter table func add constraint salario_minimo  
check (salario > 0)
```

Ex. Criação de uma restrição de integridade (chave estrangeira):

```
alter table func add constraint fk_depto  
foreign key(idDepto) references  
depto(idDepto)
```

obs. Para funcionar, Depto.idDepto já deve existir

Alteração de Tabela

- **alter table:** altera as propriedades de uma tabela:
- Para remover restrições (definidas previamente com um nome)

alter table *r* drop constraint *N*

Ex.:

alter table *func* drop constraint *salario_minimo*

DML - Data Manipulation Language

- Permite modificar as instâncias armazenadas no esquema de um banco de dados, através de:
 - Inserção de registros
 - Remoção de registros
 - Atualização de registros

Modificação da base de dados – Inserção

- A inserção de tuplas numa tabela (ou visão) é feita em SQL com a instrução

insert into *<tabela ou visão>*
values *<Conjunto de tuplas>*

- Exemplos

insert into *func (idFunc, nome, salario)* **values** (1, 'joão', 1200)

insert into *func* **values** (1, 'joão', 1200, 3)

insert into *func* **values** (1, 'joão', 1200, *null*)

Modificação da base de dados – Inserção

- Mais exemplos de inserção

- Inserção de mais de um registro

```
insert into func values (1, 'joão',1200, 3), (2, 'zé',2200, 5)
```

- Inserção a partir de outra tabela

```
insert into func (idFunc,nome, salario)
```

```
select cod,nomeFunc,salario from func2
```

Modificação da base de Dados – Remoção

- A remoção de tuplas de uma tabela (ou visão) é feita em SQL com a instrução

delete from *<tabela ou visão>*
where *<Condição>*

- Exemplo: Apagar todos funcionários do departamento 2

delete from *func*
where *idDepto = 2*

obs. O SGBD pode impedir a remoção se o registro for referenciado por chave estrangeira

Modificação da base de dados – Atualização

- A atualização de tuplas de uma tabela (ou visão) é feita em SQL com a instrução

update *<tabela ou visão>*

set *<Atributo> = <Expressão>, <Atributo> = <Expressão>, ...*

where *<Condição>*

- Exemplo: Aumentar salário em 6% a todos funcionários do departamento de código 2

update *func*

set *salario = salario * 1.06*

where *idDepto = 2*

Modificação da base de dados – Atualização

- Aumentar salário em 6% a todos funcionários que ganhem mais do que 10.000, e aumentar 5% no salário dos demais.
 - Escrever duas instruções de **update**:

```
update func  
set salario= salario * 1.06  
where salario > 10000
```



```
update func  
set salario= salario * 1.05  
where salario ≤ 10000
```
- A ordem é importante
- Pode ser efetuado de maneira mais “limpa” recorrendo à instrução **case**

Instrução Case para Atualizações condicionais

- Aumentar salário em 6% a todos funcionários que ganhem mais do que 10.000, e aumentar 5% no salário dos demais.

update *func*

set *salario* = **case**

when *salario* <= 10000 **then** *salario* * 1.05

else *salario* * 1.06

end

Atividade Individual

- Crie um banco de dados em MySQL com base no modelo relacional simplificado abaixo.

juiz (idJuiz, nome, pais)

jogador (idJogador, nome, time)

Tipo_cartao (data, tempo, idJogador, cartão, idJuiz)

idJuiz referencia juiz

idJogador referencia jogador

Atividade Individual

- Em seguida, alimente o banco com os dados presentes na tabela abaixo.
- Crie ids artificiais para representar unicamente os registros

Data	Jogador	Cartão	Tempo	Time	Juiz	País
15/06/2014	Higuaita	amarelo	23:00	Colômbia	Meira Ricci	Brasil
15/06/2014	Maradona	vermelho	11:00	Argentina	Óscar Ruiz	Colômbia
16/06/2014	Messi	amarelo	60:00	Argentina	Amarilla	Paraguai
16/06/2014	F. Melo	amarelo	14:00	Brasil	Óscar Ruiz	Colômbia
16/06/2014	F. Melo	vermelho	24:00	Brasil	Óscar Ruiz	Colômbia
16/06/2014	Messi	amarelo	23:00	Argentina	Amarilla	Paraguai

Atividade Individual

- Para essa atividade, será necessário instalar o servidor de banco de dados MySQL.
- Também será necessário um cliente para submeter comandos ao servidor.
 - Recomendo o MySQL WorkBench, que é um editor gráfico bem fácil de usar.
- A atividade envolve a criação de comandos DDL e DML.
 - Para a entrega, coloque todos os comandos em um mesmo arquivo de texto.

SQL – PARTE 1
