4.6. SQL - Structured Query Language

SQL é um conjunto de declarações que é utilizado para acessar os dados utilizando gerenciadores de banco de dados. Nem todos os gerenciadores utilizam SQL. SQL não é uma linguagem procedural pois processa conjuntos de registros, ao invés de um por vez, provendo navegação automática através dos dados, permitindo ao usuário manipular tipos complexos de dados. SQL pode ser utilizada para todas as atividades relativas a um banco de dados podendo ser utilizada pelo administrador de sistemas, pelo DBA, por programadores, sistemas de suporte à tomada de decisões e outros usuários finais.

4.6.1. Definição de Dados Utilizando SQL

4.6.1.1. Comando CREATE TABLE

O comando **create table** permite ao usuário criar uma nova tabela (ou relação). Para cada atributo da relação é definido um nome, um tipo, máscara e algumas restrições. Os tipos de uma coluna são:

- **char(*n*)**: caracteres e strings onde ***n*** é o número de caracteres;
- **integer**: inteiros
- **float**: ponto flutuante;
- **decimal(*m*,*n*)**: onde ***m*** é o número de casas inteiras e ***n*** o número de casas decimais.

A restrição **not null** indica que o atributo deve ser obrigatoriamente preenchido; se não for especificado, então o "default" é que o atributo possa assumir o valor nulo.

A forma geral do comando **create table** então é:

create table <nome_tabela> (<nome_coluna$_1$> <tipo_coluna$_1$> <NOT NULL>,
<nome_coluna$_2$> <tipo_coluna$_2$> <NOT NULL>,
:
<nome_coluna$_n$> <tipo_coluna$_n$> <NOT NULL>);

Por exemplo, para criar a tabela EMPREGADOS do apêndice A, teríamos o seguinte comando:

```
create table EMPREGADOS  (  nome          char (30)        NOT
NULL,
                            rg            integer          NOT NULL,
                            cic           integer,
                            depto         integer          NOT NULL,
                            rg_supervisor integer,
                            salario,      decimal (7,2)    NOT NULL );
```

4.6.1.2. Comando DROP TABLE

O comando **drop table** permite a exclusão de uma tabela (relação) em um banco de dados.

A forma geral para o comando **drop table** é:

drop table <nome_tabela>;

Por exemplo, para eliminar a tabela EMPREGADOS do apêndice A teríamos o seguinte comando:

drop table EMPREGADOS;

Observe que neste caso, a chave da tabela EMPREGADOS, (rg) é utilizada como chave estrangeira ou como chave primária composta em diversos tabelas que devem ser devidamente corrigidas.

Este processo não é assim tão simples pois, como vemos neste caso, a exclusão da tabela EMPREGADOS implica na alteração do projeto físico de diversas tabelas. Isto acaba implicando na construção de uma nova base de dados.

4.6.1.3. Comando ALTER TABLE

O comando **alter table** permite que o usuário faça a inclusão de novos atributos em uma tabela. A forma geral para o comando **alter table** é a seguinte:

alter table <nome_tabela> add <nome_coluna> <tipo_coluna>;

No caso do comando **alter table**, a restrição NOT NULL não é permitida pois assim que se insere um novo atributo na tabela, o valor para o mesmo em todas as tuplas da tabela receberão o valor NULL.

4.6.2. Consultas em SQL

4.6.2.1. O comando SELECT

O comando **select** permite a seleção de tuplas e atributos em uma ou mais tabelas. A forma básica para o uso do comando **select** é:

select *<lista de atributos>*
from *<lista de tabelas>*
where *<condições>;*

Por exemplo, para selecionar o nome e o rg dos funcionários que trabalham no departamento número 2 na tabela EMPREGADOS utilizamos o seguinte comando:

select nome, rg
from EMPREGADOS
where depto = 2;

obteremos então o seguinte resultado:

Nome	***RG***
Fernando	*20202020*
Ricardo	*30303030*
Jorge	*40404040*

A consulta acima é originária da seguinte função em álgebra relacional:

$$\pi_{\text{nome, rg}} (\sigma_{\text{depto} = 2}^{(\text{EMPREGADOS})});$$

Em SQL também é permitido o uso de condições múltiplas. Veja o exemplo a seguir:

select nome, rg, salario
from EMPREGADOS
where depto = 2 AND salario > 2500.00;

que fornece o seguinte resultado:

Nome	***RG***	***Salário***
Jorge	*40404040*	*4.200,00*

e que é originária da seguinte função em álgebra relacional:

$$\pi_{\text{nome, rg, salario}} (\sigma_{\text{depto = 2 .and. salario > 3500.00}}^{\text{(EMPREGADOS)}});$$

A operação *select-from-where* em SQL pode envolver quantas tabelas forem necessárias. Leve em consideração a seguinte consulta:

selecione o número do departamento que controla projetos localizados em Rio Claro;

select t1.numero_depto
from departamento_projeto t1, projeto t2
where t1.numero_projeto = t2.numero;

Na expressão SQL acima, *t1* e *t2* são chamados *"alias"* (apelidos) e representam a mesma tabela a qual estão referenciando. Um "alias" é muito importante quando há redundância nos nomes das colunas de duas ou mais tabelas que estão envolvidas em uma expressão. Ao invés de utilizar o "alias", é possível utilizar o nome da tabela, mas isto pode ficar cansativo em consultas muito complexas além do que, impossibilitaria a utilização da mesma tabela mais que uma vez em uma expressão SQL. Considere a seguinte consulta:

selecione o nome e o rg de todos os funcionários que são supervisores;

select e1.nome, e1.rg
from empregado e1, empregado e2
where e1.rg = e2.rg_supervisor;

que gera o seguinte resultado:

Nome	***RG***
João Luiz	*10101010*
Fernando	*20202020*

A consulta acima é decorrente da seguinte expressão em álgebra relacional:

$$\pi_{\text{nome, rg}} (\bowtie \text{EMPREGADOS}_{\text{ tg\_t1 = rg\_supervisor\_t2}} \text{EMPREGADOS});$$

O operador ***** dentro do especificador *select* seleciona todos os atributos de uma tabela, enquanto que a exclusão do especificador *where* faz com que todas as tuplas de uma tabela sejam selecionadas. Desta forma, a expressão:

*select **
from empregados;

gera o seguinte resultado:

Nome	***RG***	***CIC***	***Depto.***	***RG Supervisor***	Salário
João Luiz	*10101010*	*11111111*	*1*	***NULO***	*3.000,00*
Fernando	*20202020*	*22222222*	*2*	*10101010*	*2.500,00*
Ricardo	*30303030*	*33333333*	*2*	*10101010*	*2.300,00*
Jorge	*40404040*	*44444444*	*2*	*20202020*	*4.200,00*
Renato	*50505050*	*55555555*	*3*	*20202020*	*1.300,00*

Diferente de álgebra relacional, a operação *select* em SQL permite a geração de tuplas duplicadas como resultado de uma expressão. Para evitar isto, devemos utilizar o especificador **distinct**. Veja a seguir os exemplos com e sem o especificador **distinct**.

```
select depto                    select distinct depto
from empregado;                 from empregado;
```

que gera os seguintes resultados:

Depto.
1
2
2
2
3

Depto.
1
2
3

Podemos gerar consultas aninhadas em SQL utilizando o espeficicador **in**, que faz uma comparação do especificador **where** da consulta mais externa com o resultado da consulta mais interna. Considere a consulta a seguir:

selecione o nome de todos os funcionários que trabalham em projetos localizados em
Rio Claro;

```
select e1.nome, e1.rg, e1.depto
from empregado e1, empregado_projeto e2
where e1.rg = e2.rg_empregado
      and e2.numero_projeto in (   select numero
                                   from projeto
                                   where localizacao = 'Rio Claro');
```

Para selecionar um conjunto de tuplas de forma ordenada devemos utilizar o comando **order by**. Leve em consideração a seguinte consulta:

selecione todos os empregados por ordem alfabética:

```
select nome, rg, depto
from empregado
order by nome;
```

4.6.3. Inserções e Atualizações

Para elaborar inserções em SQL, utiliza-se o comando **insert into**. A forma geral para o comando **insert into** é:

insert into *<nome da tabela> <(lista de colunas)>*
values *<(lista de valores)>;*

Considere a seguinte declaração:

insira na tabela empregados, os seguintes dados:

nome: *Jorge Goncalves*

rg: *60606060*

cic: *66666666*

departamento: *3*

rg_supervisor: *20202020*

salário: *R$ 4.000,00*

insert into empregados
values ('Jorge Goncalves', '60606060', '66666666', 3, '20202020', 4000,00);

ou ainda:

insira na tabela empregados os seguintes dados:

nome: *Joao de Campos*

rg: *70707070*

cic: *77777777*

departamento: *3*

salário: *R$2.500,00*

insert into empregados (nome, rg, cic, depto, salario)
values ('*Joao de Campos*, '*70707070*', '*77777777*', 3, 2500,00);

Como na primeira inserção todos os campos foram inseridos, então não foi necessário especificar o nome das colunas. Porém, na segunda inserção, o campo *rg_supervisor* não foi inserido, então especificou-se as colunas. Outra forma de se elaborar esta inserção seria:

insert into empregados
values ('*Joao de Campos*, '*70707070*', '*77777777*', 3, '', 2500,00);

Neste caso, utilizou-se os caracteres **''** para se declarar que um valor nulo seria inserido nesta coluna.

Para se efetuar uma alteração em uma tabela, é utilizado o comando **update**. A forma geral do comando **update** é:

update *<tabela>*
set *<coluna> = <expressão>*
where *<condição>*

Considere a seguinte declaração:

atualize o salário de todos os empregados que trabalham no departamento 2 para R$ 3.000,00;

```
update   empregado
set      salario = 3.000,00
where      depto = 2;
```

Para se eliminar uma tupla de uma tabela, utiliza-se o comando **delete**. A forma geral do comando **update** é:

```
delete from <tabela>
where          <condição>;
```

Leve em consideração a seguinte expressão:

elimine os registros nos quais o empregado trabalhe no departamento 2 e possua salário maior que R$ 3.500,00;

```
delete from empregado
where      salario > 3.500,00 and depto = 2;
```

Nos casos de atualização que foram vistos, todas as <*condições*> podem ser uma consulta utilizando o comando **select**, onde o comando será aplicado sobre todos os registros que satisfizerem as condições determinadas pelo comando de seleção.

5. Bibliografia

Fundamentals of Database Systems; Ramez Elmasri, Shamkant Navathe; The Benjamin CummingsPublishing Company; 1989;

Sistema de Banco de Dados; Henry F. Korth, Abraham Silberschatz; Makro Books; 1995;

SQL Language - Oracle Reference Manual; Version 7.2;

Apêndice A - Exemplo de um Banco de Dados

Tabela EMPREGADO					
Nome	RG	CIC	Depto.	RG Supervisor	Salário
João Luiz	*10101010*	*11111111*	*1*	***NULO***	*3.000,00*
Fernando	*20202020*	*22222222*	*2*	*10101010*	*2.500,00*
Ricardo	*30303030*	*33333333*	*2*	*10101010*	*2.300,00*
Jorge	*40404040*	*44444444*	*2*	*20202020*	*4.200,00*
Renato	*50505050*	*55555555*	*3*	*20202020*	*1.300,00*

Tabela DEPARTAMENTO		
Nome	Número	RG Gerente
Contabilidade	*1*	*10101010*
Engenharia Civil	*2*	*30303030*
Engenharia Mecânica	*3*	*20202020*

Tabela PROJETO		
Nome	Número	Localização
Financeiro 1	*5*	*São Paulo*
Motor 3	*10*	*Rio Claro*
Prédio Central	*20*	*Campinas*

Tabela DEPENDENTES				
RG Responsável	Nome Dependente	Dt. Nascimento	Relação	Sexo
10101010	Jorge	27/12/86	Filho	Masculino
10101010	Luiz	18/11/79	Filho	Masculino
20202020	Fernanda	14/02/69	Conjuge	Feminino
20202020	Angelo	10/02/95	Filho	Masculino
30303030	Adreia	01/05/90	Filho	Feminino

Tabela DEPARTAMENTO_PROJETO	
Número Depto.	Número Projeto
2	*5*
3	*10*
2	*20*

Tabela EMPREGADO_PROJETO		
RG Empregado	Número Projeto	Horas
20202020	*5*	*10*
20202020	*10*	*25*
30303030	*5*	*35*
40404040	*20*	*50*
50505050	*20*	*35*