Apostila de SQL

1. Introdução

A primeira versão da linguagem SQL, chamada SEQUEL (Structured Query English Language), surgiu em 1974 nos laboratórios da IBM (Califórnia). Entre 1976 e 1977 ela foi revisada e ampliada, tendo então o seu nome alterado para SQL.

Devido ao sucesso da nova forma de consulta e manipulação de dados dentro de um ambiente de banco de dados, sua utilização tornou-se cada vez maior. Vários SGBD's atuais utilizam o SQL como a linguagem padrão para o acesso às bases de dados. Entre eles podemos citar:

- ✓ DB2 da IBM
- ✓ ORACLE da Oracle Corporation;
- ✓ RDB da Digital
- ✓ SYBASE da Sybase INC
- ✓ SQL Server da Microsoft
- ✓ Ingres da Computer Associates

Em 1982 o American National Standard Institute (ANSI) tornou a SQL a linguagem padrão para a manipulação de dados em ambiente relacional.

2. A Linguagem SQL

A linguagem SQL pode ter vários enfoques:

Linguagem interativa de consulta (query AdHoc)

Através de comandos SQL os usuários podem montar consultas poderosas, sem a necessidade da criação de um programa, podendo utilizar ferramentas front-end para a montagem de relatórios.

Linguagem de programação para acesso às bases de dados

Comandos SQL embutidos em programas de aplicação (escritos em C, C++, Java, Visual Basic e etc) acessam os dados armazenados em uma base de dados relacional.

Linguagem de administração de banco de dados

O responsável pela administração do banco de dados (DBA) pode utilizar comandos SQL para realizar tarefas relacionadas com a manutenção dos schemas do banco de dados.

Linguagem de consulta em ambiente cliente/servidor

Os programas sendo processados nos computadores dos clientes (front ends) usam comandos SQL para se comunicarem, através de uma rede, com um SGBD sendo processado em uma máquina servidora (back end);

Linguagem para bancos de dados distribuídos

A linguagem SQL é também a linguagem padrão para a manipulação de dados em uma base de dados distribuída.

Linguagem de definição de dados (DDL)

Permite ao usuário a definição da estrutura e organização dos dados armazenados, e das relações existentes entre eles.

Linguagem de manipulação de dados (DML)

Permite a um usuário, ou a um programa de aplicação, a execução de operações de inclusão, remoção, seleção ou atualização de dados previamente armazenados na base de dados.

Controle de acesso

Protege os dados de manipulações não autorizadas.

Integridade dos dados

Auxilia no processo de definição da integridade dos dados, protegendo contra corrupções e inconsistências geradas por falhas do sistema de computação, ou por erros nos programas de aplicação.

3. Vantagens e Desvantagens da Linguagem SQL

Podemos apontar as seguintes vantagens no uso da linguagem SQL:

Independência de fabricante

A linguagem SQL é adotada por praticamente todos os SGBD's relacionais existentes no mercado, além de ser uma linguagem padronizada (ANSI). Com isso, pelo menos em tese, posso mudar de SGBD sem me preocupar em alterar os programas de aplicação.

Portabilidade entre plataformas de hardware e software

Pode ser utilizada tanto em máquinas Intel rodando Windows, passando por workstations RISC rodando UNIX, até mainframes rodando sistemas operacionais proprietários.

Redução dos custos com treinamento

Com base no item anterior, as aplicações podem se movimentar de um ambiente para o outro sem que seja necessária uma reciclagem da equipe de desenvolvimento.

Usa inglês estruturado de alto nível

O SQL é formado por um conjunto bem simples de sentenças em inglês, oferecendo um rápido e fácil entendimento.

Permite consultas interativas

Permite aos usuários acesso fácil e rápido aos dados a partir de um front end que permita a edição e a submissão de comandos SQL.

Múltiplas visões dos dados

Permite ao criador do banco de dados levar diferentes visões dos dados aos diferentes usuários.

Definição dinâmica dos dados

Através da linguagem SQL pode-se alterar, expandir ou incluir, dinamicamente, as estruturas dos dados armazenados, com máxima flexibilidade.

Porém, existem também algumas desvantagens no uso da linguagem SQL:

Críticas (segundo C.J. Date)

- ✓ Falta de ortogonalidade nas expressões, funções embutidas, variáveis indicadoras, referência a dados correntes, constante NULL, conjuntos vazios, e etc;
- ✓ Definição formal da linguagem após sua criação;
- ✓ Discordância com as linguagens hospedeiras (geralmente procedurais e orientadas para registros e não para conjuntos);
- ✓ Falta de algumas funções;
- ✓ Não dá suporte a alguns aspectos do modelo relacional (join explícito, domínios, e etc.)

4. Tabelas dos Exemplos

As tabelas a seguir serão usadas nos exemplos que se seguem:

CLIENTE (cod_cli, nome_cli, endereco, cidade, cep, uf)

VENDEDOR (cod_vend, nome_vend, sal_fixo, faixa_comiss)

PEDIDO (num_ped, prazo_entr, cd_cli, cd_vend)

ITEM_PEDIDO (no_ped, cd_prod, qtd_ped)

PRODUTO (cod_prod, unid_prod, desc_prod, val_unit)

5. Criação e Destruição de Tabelas

O comando CREATE TABLE é usado para criar uma tabela. A sua forma geral é:

CREATE TABLE <nome_tabela>
(<descrição das colunas>,
<descrição das chaves>);

onde:

<nome_tabela> dever ser substituído pelo nome da tabela a ser criada.

<descrição das colunas> deve ser substituída pela relação das colunas da tabela e seus respectivos tipos de dados (por exemplo, smallint, char, varchar, integer, number, float e etc).

<descrição das chaves> deve ser substituída pela lista das colunas que são tratadas como chaves estrangeiras.

Algumas colunas podem receber o valor NULL (nulo), e a coluna definida como chave primária, além de não poder receber NULL, deve ser uma coluna UNIQUE (sem repetições; isto é, chave primária)

Script de Criação das Tabelas dos Exemplos

```
create table cliente
      (
         cod_cli       smallint    not null,
         nome_cli      varchar(40) not null,
         endereco      varchar(40)     null,
         cidade        varchar(20)     null,
         cep           char(08)        null,
         uf            char(02)        null,
         primary key (cod_cli));

create table vendedor
      (
         cod_vend      smallint    not null,
         nome_vend     varchar(40) not null,
         sal_fixo      number(9,2) not null,
```

```
         faixa_comiss  char(01)    not null,
         primary key (cod_vend));

create table produto
      (
         cod_prod      smallint    not null,
         unid_prod     char(03)    not null,
         desc_prod     varchar(20) not null,
         val_unit      number(9,2) not null,
         primary key (cod_prod));

create table pedido
      (
         num_ped       smallint    not null,
         prazo_entr    smallint    not null,
         cd_cli        smallint    not null
REFERENCES CLIENTE (cod_cli),
         cd_vend       smallint    not null
REFERENCES VENDEDOR (cod_vend),
         primary key (num_ped));

create table item_pedido
      (
         no_ped        smallint    not null
REFERENCES PEDIDO (num_ped),
         cd_prod       smallint    not null
REFERENCES PRODUTO (cod_prod),
         qtd_ped       float       not null);
```

Para excluirmos uma tabela existente devemos usar o comando DROP TABLE. A sua forma geral é:

DROP TABLE <nome_tabela>;

onde:

<nome_tabela> dever ser substituído pelo nome da tabela a ser excluída.

Exemplos

```
drop   table item_pedido;
drop   table pedido;
drop   table vendedor;
drop   table produto;
drop   table cliente;
```

6. Executando Consultas sobre as Tabelas

6.1 Selecionando Colunas Específicas de uma Tabela

SELECT <lista_de_colunas>
FROM <nome_tabela>;

Problema:

Listar todos os produtos com as respectivas descrições, unidades e valores unitários.

```
select desc_prod,unid_prod,val_unit
  from produto;

DESC_PROD            UNI VAL_UNIT
-------------------- --- ----------
Chapa de Aco         kg         2,5
Cimento              kg         4,5
parafuso 3.0X10.5 mm kg           2
Fio plastico         m            ,2
Solvente PRW         l             5
5 linhas selecionadas.
```

Problema:

Listar os nomes dos clientes, as cidade e os estados onde eles estão localizados.

```
select nome_cli,cidade,uf
  from cliente;

NOME_CLI                               CIDADE               UF
-------------------------------------- -------------------- --
Supermercado Carrefour                 rio de janeiro       rj
Supermercado Baratao                   rio de janeiro       rj
Supermercado Arariboia                 niteroi              rj
UFF                                    niteroi              rj
CSN                                    volta redonda        rj
Pegeout                                resende              rj
Ind. Quimicas Paulistas                sao paulo            sp
Ford Caminhoes                         sao paulo            sp
Riocel Celulose                        guaiba               rs
Elevadores Sur                         guaiba               rs
```

6.2 Selecionando todas as Colunas de uma Tabela

SELECT *
FROM <nome_tabela>;

Problema:

Listar o conteúdo de todas as tabelas da base de dados dos exemplos.

select * from cliente;

```
COD_CLI    NOME_CLI                                 ENDERECO
CIDADE               CEP      UF
---------- ---------------------------------------- --------------------
------------------ ------------------- -------- --
      1000 Supermercado Carrefour                   Av. das Americas
rio de janeiro       20000001 rj
      2000 Supermercado Baratao                     Rua Rolando Lero
rio de janeiro       20000002 rj
      3000 Supermercado Arariboia                   Rua Itaoca
niteroi              20000003 rj
      4000 UFF                                      Cidade Univers.
niteroi              20000004 rj
      5000 CSN                                      Rua do Aco
volta redonda        20000005 rj
      6000 Pegeout                                  Rodovia Pres. Dutra
resende              20000006 rj
      7000 Ind. Quimicas Paulistas                  Rua Tuiuti
sao paulo            11000001 sp
      8000 Ford Caminhoes                           Rua Henry Ford
sao paulo            11000002 sp
      9000 Riocel Celulose                          Rua Gen. Arouca
guaiba               30000001 rs
     10000 Elevadores Sur                           Rua Planejada
guaiba               30000001 rs
```

select * from produto;

```
COD_PROD   UNI DESC_PROD            VAL_UNIT
---------- --- -------------------- ----------
       100 kg  Chapa de Aco                2,5
       200 kg  Cimento                     4,5
       300 kg  parafuso 3.0X10.5 mm          2
       400 m   Fio plastico                 ,2
       500 l   Solvente PRW                  5
```

select * from vendedor;

```
COD_VEND   NOME_VEND                                  SAL_FIXO   F
---------- ---------------------------------------- ---------- -
        11 Paulo Alberto                                  1500 b
        12 Ana Cristina                                   2100 a
        13 Cassia Andrade                                  900 c
        14 Armando Pinto                                  2500 a
        15 Maria Paula                                     900 c
```

select * from pedido;

```
NUM_PED    PRAZO_ENTR CD_CLI     CD_VEND
---------- ---------- ---------- ----------
      1111         10       1000         11
      1112          5       1000         11
      1113         30       1000         15
      2111         15       3000         14
      2112         18       3000         15
      2113          3       3000         12
      3111         13       4000         12
      3112          7       4000         11
      4111          7       6000         11
      4112          7       6000         14
      5111         10       8000         14
      6111         30       9000         14
      6112         60       9000         12
      7111         20      10000         15
```

select * from item_pedido;

```
NO_PED     CD_PROD    QTD_PED
---------- ---------- ----------
      1111        100        100
      1111        200        100
      1111        300        200
      1112        400        100
      1112        500       1000
      1113        100        300
      2111        100        500
      2111        500        400
      2112        200        100
      2112        300        200
      2113        500        500
      3111        400        300
      3112        100        400
      3112        200        600
      4111        300        700
      4112        500       1000
      4112        500        500
      5111        200        100
      5111        300        500
      6111        400        100
      6112        300        400
      6112        400        200
      7111        100        500
```

6.3 Selecionando Apenas Alguns Registros da Tabela

SELECT <lista_de_colunas>
FROM <nome_tabela>
WHERE <condição_de_seleção>;

Onde a cláusula WHERE tem a seguinte forma:

WHERE <nome_da_coluna> <operador> <valor>

6.3.1 Operadores Relacionais:

=	igual
<> ou !=	diferente
<	menor que
>	maior que
>=	maior ou igual a
<=	menor ou igual a

Quando a coluna é do tipo **caracter**, o <valor> deve estar entre aspas simples (').

Exemplo: 'parafuso'

Observação:

Na linguagem SQL existe diferença entre caracteres maiúsculos e minúsculos; logo, 'PARAFUSO' é diferente de 'parafuso'.

Problema:

Listar o número do pedido, o código do produto e a quantidade pedida dos itens de um pedido, onde a quantidade pedida seja igual a 500.

```
select no_ped,cd_prod,qtd_ped
  from item_pedido
  where qtd_ped = 500;
```

```
NO_PED     CD_PROD    QTD_PED
---------- ---------- ----------
      2111        100        500
      2113        500        500
      4112        500        500
      5111        300        500
      7111        100        500
5 linhas selecionadas.
```

Problema:

Quais são os clientes localizados na cidade de Niterói?

```
select nome_cli
  from cliente
  where cidade = 'niteroi';

NOME_CLI
----------------------------------------
Supermercado Arariboia
UFF
2 linhas selecionadas.
```

6.3.2 Operadores Lógicos

AND	conjunção
OR	disjunção
NOT	negação

Problema:

Quais são os produtos que têm unidade igual a 'kg' e valor unitário maior do que R$ 2,00?

```
select desc_prod
  from produto
  where unid_prod = 'kg' and val_unit > 2.00;

DESC_PROD
--------------------
Chapa de Aco
Cimento
2 linhas selecionadas.
```

Problema:

Liste todos os clientes localizados na cidade de São Paulo ou que tenham CEP entre 20000005 e 20000010.

```
select nome_cli,cidade,cep
  from cliente
  where cidade = 'sao paulo' or (cep>='20000005' and cep<='20000010');

NOME_CLI                                 CIDADE               CEP
---------------------------------------- -------------------- --------
CSN                                      volta redonda        20000005
Pegeout                                  resende              20000006
Ind. Quimicas Paulistas                  sao paulo            11000001
Ford Caminhoes                           sao paulo            11000002
4 linhas selecionadas.
```

Observação:

A prioridade do operador AND é maior do que a prioridade do operador OR; logo, neste exemplo, a utilização dos parênteses é opcional.

Problema:

Mostrar todos os pedidos que não tenham prazo de entrega superior a 15 dias.

```
select num_ped
  from pedido
  where not (prazo_entr > 15);

NUM_PED
----------
      1111
      1112
      2111
      2113
      3111
      3112
      4111
      4112
      5111
9 linhas selecionadas.
```

6.3.3 Operadores BETWEEN e NOT BETWEEN

WHERE <nome_coluna> BETWEEN <valor1> AND <valor2>

WHERE <nome_coluna> NOT BETWEEN <valor1> AND <valor2>

Este operador possibilita a seleção de uma faixa de valores sem a necessidade do uso dos operadores >=, <= e AND.

<valor1> e <valor2> têm que ter o mesmo tipo de dado que <nome_coluna>.

Problema:

Liste o código e a descrição dos produtos que tenham o valor unitário na faixa de R$ 0,10 a R$ 3,00.

```
select cod_prod,desc_prod
  from produto
  where val_unit between 0.10 and 3.00;

COD_PROD   DESC_PROD
---------- --------------------
       100 Chapa de Aco
       300 parafuso 3.0X10.5 mm
       400 Fio plastico
3 linhas selecionadas.
```

6.3.4 Operadores LIKE e NOT LIKE

WHERE <nome_coluna> LIKE <valor>

WHERE <nome_coluna> NOT LIKE <valor>

Aplicáveis apenas a colunas dos tipos CHAR e VARCHAR.

Funcionam de modo análogo aos operadores = e <>, porém o poder dos operadores LIKE e NOT LIKE está na utilização dos símbolos **%** e **_**, que podem fazer o papel de “coringa”:

%	substitui uma palavra
_	substitui um caracter qualquer

Exemplos:

'apis%' se aplicaria às seguintes cadeias de caracteres:

'lapis preto'
'lapis cera'
'lapis borracha'

'broca n_' se aplicaria às seguintes cadeias de caractere:

'broca n1'
'broca n9'
'broca n3'

Problema:

Listar todos os produtos que tenham a sua unidade começando por k (lembre-se de que a coluna unid_prod foi definida como char(03)).

```
select cod_prod,desc_prod
  from produto
  where unid_prod like 'k__';

COD_PROD   DESC_PROD
---------- --------------------
       100 Chapa de Aco
       200 Cimento
       300 parafuso 3.0X10.5 mm
3 linhas selecionadas.
```

Problema:

Listar todos os vendedores cujos os nome não comecem por 'A'.

```
select cod_vend,nome_vend
  from vendedor
  where nome_vend not like 'A%';

COD_VEND   NOME_VEND
---------- ----------------------------------------
        11 Paulo Alberto
        13 Cassia Andrade
        15 Maria Paula
3 linhas selecionadas.
```

6.3.5 Operadores IN e NOT IN

WHERE <nome_coluna> IN <lista_de_valores>

WHERE <nome_coluna> NOT IN (<lista_de_valores>)

Seleciona as linhas cujo o valor da coluna <nome_coluna> pertença ao conjunto <lista_de_valores>.

Problema:

Listar todos os vendedores cujas as faixas de comissão sejam 'a' ou 'b'.

```
select cod_vend,nome_vend
  from vendedor
  where faixa_comiss in ('a','b');

COD_VEND   NOME_VEND
---------- ----------------------------------------
        11 Paulo Alberto
        12 Ana Cristina
        14 Armando Pinto
3 linhas selecionadas.
```

6.3.6 Operadores IS NULL e IS NOT NULL

WHERE <nome_coluna> IS NULL

WHERE <nome_coluna> IS NOT NULL

A utilização do valor nulo (NULL) é problemática, pois as diversas implementações da linguagem SQL podem adotar qualquer representação para o valor nulo.

Problema:

Mostrar os clientes que não tenham endereço cadastrado.

```
select nome_cli
  from cliente
  where endereco is null;

NOME_CLI
----------------------------------------
0 linhas selecionadas.
```

6.4 Ordenando os Dados Selecionandos

SELECT <lista_de_colunas>
FROM <nome_tabela>
WHERE <condição_de_seleção>
ORDER BY {<nome_coluna>|<num_col> [ASC|DESC]}

Onde <nome_coluna> se refere à coluna segundo a qual as linhas serão ordenadas, e <num_col> se refere à posição relativa da coluna na <lista_de_colunas> projetadas, contada da esquerda para a direita, e não à posição na tabela original.
As cláusulas ASC e DESC denotam ordenação ascendente e descendente respectivamente. A forma ascendente de ordenação é assumida caso nenhuma opção seja informada explicitamente.

Problema:

Mostrar em ordem alfabética a lista de vendedores e seus respectivos salários fixos.

```
select nome_vend,sal_fixo
  from vendedor
  order by nome_vend;
```

```
NOME_VEND                                SAL_FIXO
---------------------------------------- ----------
Ana Cristina                                   2100
Armando Pinto                                  2500
Cassia Andrade                                  900
Maria Paula                                     900
Paulo Alberto                                  1500
5 linhas selecionadas.
```

Problema:

Listar os nomes, as cidades e os estados de todos os clientes, ordenados por estado e cidade de forma descendente.

```
select nome_cli,cidade,uf
  from cliente
  order by uf desc,cidade desc;
```

```
NOME_CLI                                 CIDADE               UF
---------------------------------------- -------------------- --
Ind. Quimicas Paulistas                  sao paulo            sp
Ford Caminhoes                           sao paulo            sp
Riocel Celulose                          guaiba               rs
Elevadores Sur                           guaiba               rs
CSN                                      volta redonda        rj
Supermercado Carrefour                   rio de janeiro       rj
Supermercado Baratao                     rio de janeiro       rj
Pegeout                                  resende              rj
Supermercado Arariboia                   niteroi              rj
UFF                                      niteroi              rj
10 linhas selecionadas.
```

Problema:

Mostrar a descrição e o valor unitário de todos os produtos que tenham unidade 'kg' em ordem ascendente de valor unitário.

```
select desc_prod,val_unit
  from produto
  where unid_prod = 'kg'
  order by 2;
```

```
DESC_PROD            VAL_UNIT
-------------------- ----------
parafuso 3.0X10.5 mm          2
Chapa de Aco                2,5
Cimento                     4,5
3 linhas selecionadas.
```

6.5 Realizando Cálculos sobre a Informação Selecionada

Podemos criar dinamicamente um campo que não pertença à tabela original através de operações executadas sobre os campos projetados.

Problema:

Exibir o novo salário fixo dos vendedores da faixa de comissão ‘C’, calculado com base no reajuste de 75% sobre o salário atual acrescido de R$ 120,00 de bonificação. Ordene a relação resultante pelo nome do vendedor.

```
select nome_vend,((sal_fixo*1.75)+120) as
  from vendedor
  where faixa_comiss='c'
  order by nome_vend;

NOME_VEND                                NOVO_SAL
---------------------------------------- ----------
Cassia Andrade                                 1695
Maria Paula                                    1695
2 linhas selecionadas.
```

6.5.1 Máximos (MAX) e Mínimos (MIN)

Problema:

Mostrar o menor e o maior salário entre os vendedores.

```
select  MIN(sal_fixo),MAX (sal_fixo)
  from vendedor;

MIN(SAL_FI MAX(SAL_FI
---------- ----------
       900       2500
1 linha selecionada.
```

6.5.2 Totalizando Colunas (SUM)

Problema:

Mostrar a quantidade total pedida para o produto cimento, de código 200.

```
select SUM(qtd_ped)
  from item_pedido
  where cd_prod=200;

SUM(QTD_PE
----------
       900
1 linha selecionada.
```

6.5.3 Calculando Médias (AVG)

Problema:

Qual é a média dos salários fixos dos vendedores?

```
select AVG(sal_fixo)
  from vendedor;

AVG(SAL_FI
----------
      1580
1 linha selecionada.
```

6.5.4 Contando as Linhas (COUNT)

Problema:

Quantos vendedores ganham acima de R$ 2.000,00 de salário fixo?

```
select COUNT(*)
  from vendedor
  where sal_fixo>2000;

COUNT(*)
----------
         2
1 linha selecionada.
```

6.5.5 A Palavra-Chave DISTINCT

Várias linhas de uma tabela podem conter os mesmos valores para as suas colunas (duplicidade), com exceção da chave primária. Quando desejarmos eliminar a duplicidade, podemos inserir a palavra-chave DISTINCT após a palavra-chave SELECT.

Problema:

Em que cidades as nossa empresa possui clientes?

```
select DISTINCT cidade
  from cliente;

CIDADE
--------------------
guaiba
niteroi
resende
rio de janeiro
sao paulo
volta redonda
6 linhas selecionadas.
```

6.5.6 Agrupando a Informação Selecionada (GROUP BY)

Existem ocasiões em que desejamos aplicar uma função de agregação não somente a um conjunto de tuplas, mas também organizar a informação em determinadas categorias. Isto é possível através do uso da cláusula GROUP BY.

Problema:

Listar o número de itens existente em cada pedido.

```
select no_ped, count(*) as total_itens
  from item_pedido
  group by no_ped;

NO_PED     TOTAL_ITEN
---------- ----------
      1111          3
      1112          2
      1113          1
      2111          2
      2112          2
      2113          1
      3111          1
      3112          2
      4111          1
      4112          2
      5111          2
```

```
      6111          1
      6112          2
      7111          1
14 linhas selecionadas.
```

Inicialmente as linha são agrupadas através do(s) atributo(s) fornecido(s) na cláusula GROUP BY; neste caso, no_ped. Em um segundo passo, é aplicada a operação COUNT(*) para cada grupo de linhas que tenha o mesmo número de pedido. Após a operação de contagem de cada grupo, o resultado da consulta é apresentado.
Normalmente, a cláusula GROUP BY é utilizada em conjunto com as operações COUNT e AVG.

6.5.7 A Cláusula HAVING

Às vezes temos que definir condições e aplicá-las aos grupos ao invés de fazê-lo a cada linha separadamente. Por exemplo, suponha que desejemos listar todos os pedidos que possuam mais de um item. Esta condição não se aplica a uma única linha separadamente, mas a cada grupo definido pela cláusula GROUP BY. Para exprimir tal consulta, usamos a cláusula HAVING. A condição da cláusula HAVING é aplicada após a formação dos grupos; logo, podemos usar funções de agregação na construção das condições da cláusula HAVING.

Problema:

Listar os pedidios que possuam mais de um item.

```
select no_ped,count(*) as total_itens
  from item_pedido
  group by no_ped
  having count(*)>1;

NO_PED     TOTAL_ITEN
---------- ----------
      1111          3
      1112          2
      2111          2
      2112          2
      3112          2
      4112          2
      5111          2
      6112          2
8 linhas selecionadas.
```

6.6 Recuperando Dados de Várias Tabelas (JOINS)

Algumas consultas necessitam acessar simultaneamente várias tabelas, o que leva à realização de junções (JOINS) entre as tabelas para poder extrair as informações necessárias para a consulta formulada.

6.6.1 Qualificadores de Nomes

Um qualificador de nome consiste do nome da tabela, seguido de um ponto, seguido por um nome de uma coluna da tabela. Por exemplo, o qualificador da coluna DESC_PROD da tabela PRODUTO será PRODUTO.DESC_PROD.
Os qualificadores de nome são utilizados em uma consulta para efetivar a junção (JOIN) entre as tabelas.

Problema:

Faça uma junção da tabela de clientes com a de pedidos, exibindo o nome do cliente, o código do cliente e o número do pedido.

```
select cod_cli,nome_cli,pedido.num_ped
  from cliente,pedido;

COD_CLI    NOME_CLI                                 NUM_PED
---------- ---------------------------------------- ----------
      1000 Supermercado Carrefour                         1111
      2000 Supermercado Baratao                           1111
      3000 Supermercado Arariboia                         1111
      4000 UFF                                            1111
      5000 CSN                                            1111
      6000 Pegeout                                        1111
      7000 Ind. Quimicas Paulistas                        1111
      8000 Ford Caminhoes                                 1111
      9000 Riocel Celulose                                1111
     10000 Elevadores Sur                                 1111
      1000 Supermercado Carrefour                         1112
      2000 Supermercado Baratao                           1112
      3000 Supermercado Arariboia                         1112
      4000 UFF                                            1112
      5000 CSN                                            1112
      6000 Pegeout                                        1112
      7000 Ind. Quimicas Paulistas                        1112
      8000 Ford Caminhoes                                 1112
      9000 Riocel Celulose                                1112
     10000 Elevadores Sur                                 1112
      1000 Supermercado Carrefour                         1113
      2000 Supermercado Baratao                           1113
```

```
      3000 Supermercado Arariboia                          1113
      4000 UFF                                             1113
      5000 CSN                                             1113
      6000 Pegeout                                         1113
      7000 Ind. Quimicas Paulistas                         1113
      8000 Ford Caminhoes                                  1113
      9000 Riocel Celulose                                 1113

.............................................................

      1000 Supermercado Carrefour                          6112
      2000 Supermercado Baratao                            6112
      3000 Supermercado Arariboia                          6112
      4000 UFF                                             6112
      5000 CSN                                             6112
      6000 Pegeout                                         6112
      7000 Ind. Quimicas Paulistas                         6112
      8000 Ford Caminhoes                                  6112
      9000 Riocel Celulose                                 6112
     10000 Elevadores Sur                                  6112
      1000 Supermercado Carrefour                          7111
      2000 Supermercado Baratao                            7111
      3000 Supermercado Arariboia                          7111
      4000 UFF                                             7111
      5000 CSN                                             7111
      6000 Pegeout                                         7111
      7000 Ind. Quimicas Paulistas                         7111
      8000 Ford Caminhoes                                  7111
      9000 Riocel Celulose                                 7111
     10000 Elevadores Sur                                  7111
140 linhas selecionadas.
```

Neste exemplo foi executado um produto cartesiano das tabelas CLIENTE e PEDIDO, seguido de uma projeção das colunas exibidas. Neste caso, poucas informações úteis podem ser extraídas da relação resultante. Devemos então aplicar critérios de seleção à junção para podermos obter algum resultado concreto.

Problema:

A que clientes estão associados os pedidos existentes? Listar pelos nomes dos clientes.

```
select nome_cli,pedido.cd_cli,pedido.num_ped
  from cliente,pedido
  where cliente.cod_cli=pedido.cd_cli;

NOME_CLI                                CD_CLI     NUM_PED
--------------------------------------- ---------- ----------
Supermercado Carrefour                        1000       1111
Supermercado Carrefour                        1000       1112
```

```
Supermercado Carrefour                        1000       1113
Supermercado Arariboia                        3000       2111
Supermercado Arariboia                        3000       2112
Supermercado Arariboia                        3000       2113
UFF                                           4000       3111
UFF                                           4000       3112
Pegeout                                       6000       4111
Pegeout                                       6000       4112
Ford Caminhoes                                8000       5111
Riocel Celulose                               9000       6111
Riocel Celulose                               9000       6112
Elevadores Sur                               10000       7111
14 linhas selecionadas.
```

A equação apresentada na cláusula WHERE é chamada de EQUAÇÃO DE JUNÇÃO.
Podemos utilizar os operadores LIKE, NOT LIKE, IN, NOT IN, NULL, NOT NULL, os operadores relacionais e operadores AND, OR e NOT, na cláusula WHERE de uma junção de tabelas.

Problema:

Quais são os clientes que têm pedidos com prazos de entrega superiores a 15 dias e que estão localizados nos estados de São Paulo ou do Rio de Janeiro?

```
select nome_cli,uf,pedido.num_ped,pedido.prazo_entr
  from cliente,pedido
  where cliente.cod_cli=pedido.cd_cli and
    uf in ('rj','sp') and prazo_entr>15;

NOME_CLI                                 UF NUM_PED    PRAZO_ENTR
---------------------------------------- -- ---------- ----------
Supermercado Carrefour                   rj       1113         30
Supermercado Arariboia                   rj       2112         18
2 linhas selecionadas.
```

Problema:

Mostrar os pedidos dos clientes e seus respectivos prazos de entrega, ordenados do maior para o menor.

```
select nome_cli,pedido.num_ped,pedido.prazo_entr
  from cliente,pedido
  where cliente.cod_cli=pedido.cd_cli
  order by prazo_entr DESC;
```

```
NOME_CLI                                    NUM_PED    PRAZO_ENTR
---------------------------------------- ---------- ----------
Riocel Celulose                               6112         60
Supermercado Carrefour                        1113         30
Riocel Celulose                               6111         30
Elevadores Sur                                7111         20
Supermercado Arariboia                        2112         18
Supermercado Arariboia                        2111         15
UFF                                           3111         13
Supermercado Carrefour                        1111         10
Ford Caminhoes                                5111         10
UFF                                           3112          7
Pegeout                                       4111          7
Pegeout                                       4112          7
Supermercado Carrefour                        1112          5
Supermercado Arariboia                        2113          3
14 linhas selecionadas.
```

6.6.2 Sinônimos

Para que não seja necessário escrever o nome da tabela nas qualificações de nomes, podemos utilizar ALIASES definidos na própria consulta.
A definição dos ALIASES é feita na cláusula FROM, sendo então utilizada nas outras cláusulas (WHERE, ORDER BY, GROUP BY, HAVING e SELECT) de uma consulta.

Problema:

Exibir os vendedores (ordenados por nome) que emitiram pedidos com prazos de entrega superiores a 15 dias e que tenham salários fixos iguais ou superiores a R$ 1.000,00.

```
select distinct nome_vend,prazo_entr
  from vendedor V, pedido P
  where V.cod_vend=P.cd_vend and
    V.sal_fixo>1000 and prazo_entr>15
  order by nome_vend;

NOME_VEND                                   PRAZO_ENTR
---------------------------------------- ----------
Ana Cristina                                     60
Armando Pinto                                    30
2 linhas selecionadas.
```

6.6 Junções Envolvendo Três ou mais Tabelas

Problema:

Exiba a relação dos clientes localizados no Rio de Janeiro (ordenados alfabeticamente) que têm pedidos do produto **Chapa de Aco** com prazos de entrega superirores a 15 dias.

```
select nome_cli,desc_prod,no_ped,prazo_entr
  from cliente C,pedido P,item_pedido IP,produto PR
  where C.cod_cli=P.cd_cli and
    P.num_ped=IP.no_ped and
    IP.cd_prod=PR.cod_prod and
    PR.desc_prod='Chapa de Aco' and
    P.prazo_entr>15 and
    C.uf='rj'
  order by C.nome_cli;

NOME_CLI                                      DESC_PROD             NO_PED
PRAZO_ENTR
----------------------------------------  -------------------  ----------
----------
Supermercado Carrefour                    Chapa de Aco              1113
30
1 linha selecionada.
```

Problema:

Mostre os nome de todos os vendedores que venderam **Chapa de Aco** em quantidade superior a 300 Kg.

```
select distinct nome_vend
  from vendedor V, pedido P, item_pedido IP, produto PR
  where V.cod_vend=P.cd_vend and
    p.num_ped=IP.no_ped and
    IP.cd_prod=PR.cod_prod and
    IP.qtd_ped>300 and
    PR.desc_prod='Chapa de Aco';

NOME_VEND
----------------------------------------
Armando Pinto
Maria Paula
Paulo Alberto
3 linhas selecionadas.
```

Problema:

Quantos clientes fizeram pedidos com a vendedora **Ana Cristina**?

```
select count(distinct cod_cli)
  from cliente C, pedido P, vendedor V
  where C.cod_cli=P.cd_cli and
    p.cd_vend=V.cod_vend and
    V.nome_vend='Ana Cristina';

COUNT(COD_
----------
         3
1 linha selecionada.
```

Problema:

Quantos clientes das cidades do Rio de Janeiro e Niterói tiveram seus pedidos tirados com a vendedora **Ana Cristina**?

```
select C.cidade,count(distinct cod_cli) as num_clientes
  from cliente C, pedido P, vendedor V
  where C.cod_cli=P.cd_cli and
    C.cidade in ('rio de janeiro','niteroi') and
    p.cd_vend=V.cod_vend and
    V.nome_vend='Ana Cristina'
  group by C.cidade;

CIDADE               NUM_CLIENT
-------------------- ----------
niteroi                       2
1 linha selecionada.
```

6.7 Utilizando Consultas Aninhadas (Subqueries)

Chamamos de consulta aninhada à consulta cujo o resultado é utilizado por outra consulta, de forma encadeada e contida no mesmo comando SQL.

Problema:

Que produtos estão incluídos em um pedido qualquer com a quantidade pedida igual a 100?

```
select desc_prod
  from produto
  where cod_prod IN
    (select cd_prod
       from item_pedido
       where qtd_ped=100);

DESC_PROD
--------------------
Chapa de Aco
Cimento
Fio plastico
3 linhas selecionadas.
```

Problema:

Quais vendedores ganham um salário fixo abaixo da média?

```
select nome_vend
  from vendedor
  where sal_fixo <
    (select avg(sal_fixo)
       from vendedor);

NOME_VEND
----------------------------------------
Paulo Alberto
Cassia Andrade
Maria Paula
3 linhas selecionadas.
```

Problema:

Quais os vendedores que só venderam produtos comercializados em quilogramas (Kg)?

```
select distinct cod_vend,nome_vend
  from vendedor V
  where 'kg'=ALL
    (select unid_prod
       from pedido P,item_pedido IP,produto PR
       where P.num_ped=IP.no_ped and
         IP.cd_prod=PR.cod_prod and
         P.cd_vend=V.cod_vend);
```

```
COD_VEND   NOME_VEND
---------- ----------------------------------------
15 Maria Paula
1 linha selecionada.
```

Problema:

Quais clientes realizaram mais de dois pedidos?

```
select nome_cli
  from cliente C
  where exists
    (select count(*)
       from pedido
       where cd_cli=C.cod_cli
       having count(*)>2);

NOME_CLI
----------------------------------------
Supermercado Carrefour
Supermercado Arariboia
2 linhas selecionadas.
```

Problema:

Quais os produtos que não estão presentes em nenhum pedido?

```
select cod_prod,desc_prod
  from produto P
  where not exists
    (select *
       from item_pedido
       where cd_prod=P.cod_prod);

COD_PROD   DESC_PROD
---------- --------------------
0 linhas selecionadas.
```

7. Inserindo, Modificando e Excluindo Registros

7.1 Inserindo Registros em uma Tabela

INSERT INTO <nome_tabela> (<lista_de_colunas>)
VALUES (<lista_de_valores);

Problema:

Inserir o produto **Tinta de PVC** na tabela de produtos.

```
insert into produto
  values (600,'l','Tinta PVC',15.80);

1 linha processada.
```

7.2 Inserindo Registros Usando um SELECT

INSERT INTO <nome_tabela> (<lista_de_colunas>)
SELECT <lista_de_colunas>
FROM <nome_tabela>
WHERE <condição_de_seleção>;

Problema:

Cadastrar como clientes os vendedores que emitiram mais de 3 pedidos. Usar para o código de cliente o mesmo código do vendedor. O restante das colunas devem ser preenchidas com um espaço em branco.

```
insert into cliente (cod_cli, nome_cli,endereco,cidade,cep,uf)
  select cod_vend,nome_vend,' ', ' ',' ',' '
  from vendedor V
  where 3<(select count(*)
           from pedido P
           where V.cod_vend=P.cd_vend);

2 linhas processadas.
```

7.3 Modificando um Registro

UPDATE <nome_tabela>
SET {<nome_coluna> = <expressão>}
WHERE <condição_de_seleção>;

Problema:

Alterar o preço unitário do **Cimento** para R$ 5,00.

```
update produto
  set val_unit=5.00
  where desc_prod='Cimento';

1 linha processada.
```

Problema:

Atualizar o salário fixo de todos os vendedores em 27% mais uma bonificação de R$ 100,00.

```
update vendedor
  set sal_fixo=(sal_fixo*1.27+100.00);

5 linhas processadas.
```

Problema:

Acrescentar 2,5% ao preço unitário dos produtos que estejam abaixo da média dos preços dos produtos comprados a quilo.

```
update produto
  set val_unit=val_unit*1.025
  where val_unit <
  (select avg(val_unit)
   from produto
   where unid_prod='kg');

3 linhas processadas.
```

7.4 Excluindo Registros

DELETE FROM <nome_tabela>
WHERE <condição_de_seleção>;

Problema:

Excluir todos o itens de pedido que tenham quantidade pedida inferior a 200.

```
delete from item_pedido
  where qtd_ped < 200;

6 linhas processadas.
```

Problema

Excluir todos os vendedores que não realizaram nenhum pedido.

```
delete from vendedor V
  where 0=(select count(*)
          from vendedor,pedido
          where V.cod_vend=cd_vend);

1 linha processada.
```

8. Transações

A execução de um programa que inclui operações de acesso a um banco de dados é chamada de **transação**. Se as operações em questão não alteram os dados do banco de dados, a transação é chamada de **read-only transaction**. Neste capítulo estamos interessados apenas nas transações que executam operações de atualização. Logo, a palavra **transação** se refere a um programa que realiza operações que alteram registros de um banco de dados.

8.1 Propriedades das Transações

Uma transação deve possuir as seguintes propriedades (chamadas de propriedades **ACID**):

1. **Atomicidade**: uma transação é uma unidade atômica de processamento; ou ela é executada na sua totalidade, ou então nada é executado.
2. **Consistência**: a execução de uma transação deve manter a consistência de um banco de dados.
3. **Isolamento**: uma transação não deve tornar visível para outras transações as modificações feitas em um banco de dados até que ela seja encerrada com sucesso.
4. **Durabilidade**: uma vez executada com sucesso, as alterações feitas por uma transação devem persistir, mesmo se houver falhas subseqüentes no sistema.

8.2 Definição de Transações em SQL

O padrão SQL especifica que uma transação deve começar de modo subentendido. As transações são terminadas por um dos seguintes comandos SQL:

1. **commit**: efetiva a transação corrente e inicia uma nova.
2. **rollback**: aborta a transação corrente.

Exemplo

```
update vendedor
  set sal_fixo=1000.00;
rollback;
```

No exemplo acima, as alterações feitas nos salários fixos dos vendedores não serão efetivadas, pois o comando **rollback** é executado antes que a transação seja terminada com sucesso. Logo, os salários fixos dos vendedores permanecerão com os mesmos valores que possuíam imediatamente antes do início da transação.

9. Visões

Considerações sobre segurança podem exigir que determinados dados não estejam disponíveis para alguns usuários. Logo, não é desejável que o modelo lógico possa ser acessado indiscriminadamente. Com base nas questões de segurança, podemos criar uma coleção de relações personalizadas que se ajustem melhor às necessidades dos usuários e que levem em conta as questões de segurança. Tais relações são chamadas de **visões**.
Uma **visão**, na terminologia SQL, é uma tabela que é derivada de outras tabelas. Estas outras tabelas podem ser tabelas-base (criadas através do comando **CREATE TABLE**) ou outras visões previamente definidas. Uma **visão** não está, necessariamente, fisicamente armazenada no banco de dados; ela pode existir apenas virtualmente, em contraste com as tabelas-base, cujas as tuplas se encontram fisicamente armazenadas no banco de dados.

9.1 Definindo Visões

Em SQL, uma visão é definida através do comando **create view**, cuja a forma geral é:

CREATE VIEW <nome_visão> (<nomes das colunas>)
AS <expressão_de_consulta>;

Problema:

Criar uma visão que contenha apenas os produtos vendidos em quilograma.

```
create view prod_kg
  (codigo,descricao,unidade)
  as select cod_prod,desc_prod,unid_prod
       from produto
       where unid_prod='kg';

Instrução processada.
```

Problema

Criar uma visão contendo o código do vendedor, o seu nome e o salário fixo anual.

```
create view salario_anual
  (codigo,nome,sal_anual)
  as select cod_vend,nome_vend,sal_fixo*12
       from vendedor;

Instrução processada.
```

Problema

Criar uma visão contendo os vendedores, seus pedidos efetuados e os respectivos produtos.

```
create view lista_pedidos
  as select nome_vend,num_ped,desc_prod
       from vendedor V,pedido P,item_pedido I, produto PR
       where V.cod_vend=P.cd_vend and
             P.num_ped=I.no_ped and
             I.cd_prod=PR.cod_prod;

Instrução processada.
```

9.2 Utilizando Visões

Problema

Com base na visão **salario_anual**, mostrar os vendedores que têm salário fixo anual superior a R$ 20.000,00.

```
select nome,sal_anual
  from salario_anual
  where sal_anual > 20000;

NOME                                     SAL_ANUAL
---------------------------------------- ----------
Ana Cristina                                  25200
Armando Pinto                                 30000
2 linhas selecionadas.
```

Problema

Inserir uma linha na visão **prod_kg** com código **800**, descrição **Cimento Branco**, e unidade **kg**.

```
insert into prod_kg
  values (800,'Cimento Branco','kg');
ORA-01400: não é possível inserir NULL em ("xxx"."PRODUTO"."VAL_UNIT")
```

Problema

Excluir da visão salario_anual todos os vendedores que têm salário anual abaixo de R$ 15.000,00.

```
delete from salario_anual
  where sal_anual<15000;
ORA-02292: restrição de integridade (xxx.SYS_C001449) violada - registro
filho localizado
```

9.3 Eliminando Visões

Podemos eliminar uma visão através do comando **drop view**, cuja a forma geral é:

DROP VIEW <nome_visão>;

Problema:

Eliminar a visão **salario_anual**;

```
drop view salario_anual;
Instrução processada.
```

10. Índices

Os SGBD's utilizam índices para tornar mais eficiente a recuperação dos dados de um banco de dados.
Um índice é uma estrutura de dados, onde são armazenados valores e ponteiros organizados de forma ascendente ou descendente, que torna possível localizar rapidamente as linhas de uma tabela nas quais o valor desejado está armazenado.
Os índices são utilizados internamente pelo SGBD, ficando totalmente transparente ao usuário a sua utilização.

10.1 Criando Índices

Um índice é construído sobre uma ou mais colunas de uma determinada tabela. Em SQL, um índice é criado através do comando **create index**, cuja a forma geral é:

CREATE [UNIQUE] INDEX <nome_índice>
ON <nome_tabela> (<lista_de_colunas>);

A cláusula UNIQUE é opcional, e a sua inclusão assegura a não existência de valores duplicados no índice a ser criado.

Problema:

Criar um índice chamado **produto_desc_prod_idx** sobre a coluna **desc_prod** da tabela **produto**.

```
create index produto_desc_prod_idx
on produto(desc_prod);
Instrução processada.
```

Problema:

Criar um índice único chamado **cliente_nome_cli_idx** sobre a coluna **nome_cli** da tabela **cliente**.

```
create unique index cliente_nome_cli_idx
on cliente(nome_cli);
Instrução processada.
```

Problema:

Criar um índice único chamado **pedido_num_ped_cd_vend_idx** sobre as colunas **num_ped** e **cd_vend** da tabela **pedido**.

```
create unique index pedido_num_ped_cd_vend_idx
on pedido(num_ped,cd_vend);
Instrução processada.
```

Problema:

Criar um índice único chamado **cliente_cod_cli_idx** sobre a coluna **cod_cli** da tabela **cliente**.

```
create unique index cliente_cod_cli_idx
on cliente(cod_cli);
ORA-01408: esta lista de colunas já está indexada
```

Obs: um índice único é automaticamente construído quando uma coluna é definida como **chave primária** no comando CREATE TABLE.

10.2 Eliminado Índices

Podemos eliminar uma índice através do comando **drop index**, cuja a forma geral é:

DROP INDEX <nome_índice>;

Problema:

Eliminar os índices criados nos problemas acima.

```
drop index produto_desc_prod_idx;
Instrução processada.
drop index cliente_nome_cli_idx;
Instrução processada.
drop index pedido_num_ped_cd_vend_idx;
Instrução processada.
```

11. Segurança

A informação é vital para o sucesso de um negócio, mas quando ela é tratada de forma incorreta ou quando cai em mão erradas, pode tornar-se um sério obstáculo para se atingir o sucesso.
Visando garantir a segurança dos dados, os SGBD's disponíveis no mercado fornecem um série de facilidades para salvaguardar as informações por eles mantidas. Tais facilidades são implementadas concedendo-se e retirando-se privilégios dos usuários sobre os dados mantidos por um SGBD.

11.1 O Comando GRANT

Quando uma tabela ou uma visão é criada, o nome do usuário que a criou é anexado implicitamente ao nome da tabela ou da visão.

Exemplo:

Se a tabela **produto** foi criada pelo usuário **pedro**; então, internamente, ela será conhecida como **pedro.produto**.
O criador de uma tabela ou de uma visão tem todos os privilégio sobre o objeto criado, podendo inclusive conceder tais privilégios para outros usuários.
Em SQL, os privilégios são concedidos através do comando **grant**, cuja a forma geral é:

GRANT <lista_de_privilégios> ON <nome_objeto>
TO <lista_de_usuários>;

Os privilégios concedidos são os seguintes:

select → permite executar consultas

insert → permite inserir registros

delete → permite excluir registros

update → permite modificar registros

all → permite executar qualquer operação

Observações:

a) Os usuário que irão receber os privilégios têm que estar previamente cadastrados no banco de dados.
b) Podemos conceder privilégios para todos os usuários do banco de dados colocando a palavra **PUBLIC** no lugar reservado para a <**lista_de_usuários**>.

Exemplos:

GRANT select on produto to pedro;

Permite apenas consultas ao usuário Pedro sobre a tabela **produto**.

GRANT select, insert, update on pedido to ana;

Concede ao usuário Ana os privilégios de consulta, inclusão e alteração sobre a tabela **pedido**.

GRANT select on cliente to ana,pedro;

Concede aos usuários Ana e Pedro o privilégio de consulta sobre a tabela **cliente**.

GRANT all on vendedor to PUBLIC;

Permite todos os privilégios a todos os usuários cadastrados sobre a tabela **vendedor**.

Problema:

Disponibilizar para seleção, a todos os usuários, a visão **salario_anual**.

```
grant select on salario_anual to public;
Instrução processada.
```

Problema:

Disponibilizar para alteração as colunas endereco, cidade, cep e uf, da tabela **cliente**, para todos os usuários.

```
grant update (endereco,cidade,cep,uf) on cliente to public;
Instrução processada.
```

Podemos passar nossa concessão de privilégio a outros usuários através da cláusula WITH GRANT OPTION.

Exemplo:

Conceder ao usuário Ana o poder de dar a concessão de todos os privilégios sobre a tabela **pedido** a outros usuários.

```
grant all on pedido to ana with grant option;
Instrução processada.
```

11.2 O Comando REVOKE

Da mesma forma que podemos conceder privilégios de acesso a outros usuários, podemos também retirá-los através do comando REVOKE, cuja a forma geral é:

REVOKE <lista_de_privilégios> ON <nome_objeto>
FROM <lista_de_usuários>;

Exemplo:

Retirar o privilégio de seleção sobre a tabela **pedido** do usuário Ana.

REVOKE select on pedido from ana;

Problema:

Retirar todos os privilégios concedidos a todos os usuários sobre a visão **salario_anual**.

```
revoke all on salario_anual from public;
Instrução processada.
```

Problema:

Retirar os privilégios de atualização e inserção concedidos ao usuário Ana sobre a tabela **pedido**.

```
revoke insert, update on pedido from ana;
Instrução processada.
```