

4. SQL – Parte II

4.1. Operadores *and*, *or* e *not* com a cláusula *where*

Usados principalmente em consultas nas quais são necessárias mais de uma única condição.

Exemplo:

```
select  nome_cliente
from    clientes
where   idade > 30 and idade <= 35
```

Operadores de comparação:

<
>
<=
>=
=
<>

Selecionando valores entre intervalos de valores:

```
select  nome_produto
from    produto
where   valor between 30,00 and 300,00
```

ou

```
select  nome_funcionário
from    funcionários
where   salário <= 1.000 and salário >= 3.000
```

4.2. Rename

Para renomear relações ou atributos:

nome_antigo as nome_novo

Exemplo:

```
select distinct nome_cliente, devedor.nro_empréstimo
from    devedor, empréstimo
where   devedor.nro_empréstimo = empréstimo.nro_empréstimo
and     nome_agência = "Lages"
```

nome_cliente	nro_empréstimo
João da Silva	150
Carlos Honório	212

Para melhorar a apresentação da consulta:

```
select distinct nome_cliente as Nome,  
                devedor.nro_empréstimo as Identificador  
from    devedor, empréstimo  
where   devedor nro_empréstimo = empréstimo.nro_empréstimo  
and     nome_agência = "Lages"
```

Nome	Identificador
João da Silva	150
Carlos Honório	212

4.3. Variáveis Tuplas (ou *alias*)

“Apelidos” para relações ou atributos, para facilitar a criação de consultas.
Exemplo:

```
select distinct nome_cliente, T.nro_empréstimo  
from    devedor as T, empréstimo as S  
where   T.nro_empréstimo = S.nro_empréstimo
```

4.4. Chaves

Usa-se o comando *primary_key* (*chave*) para criar chaves identificadoras em SQL. Exemplo:

```
create table cliente (nome_cliente char(30) not null,  
                      cpf_cliente char(11) not null,  
                      endereço_cliente char(100) not null,  
                      primary_key (cpf_cliente))
```

Para certificar-se de que um valor, durante a inserção, estará dentro de um determinado domínio, pode-se usar o comando *check*, como nos exemplos abaixo:

```
create table conta (nro_conta char(10) not null,  
                   nome_conta char(30) not null,  
                   saldo decimal(10,2) not null,  
                   primary_key (nro_conta),  
                   check (saldo >= 0))  
  
create table estudante (nome_estudante char(30) not null,  
                        id_estudante char(10) not null,  
                        nível char(15) not null,  
                        primary_key (id_estudante),  
                        check (nível in ("Bacharelado", "Mestrado",  
                        "Doutorado")))
```

- Remover um atributo (coluna completa):

```
alter table T1 drop nome
```

4.5. Operações em *Strings*

- Operações com *strings* podem ser úteis para:
 - Verificar coincidências entre pares
- Caracter *%* (porcentagem) compara qualquer *substring*
- Caracter *_* (sublinhado) compara qualquer caracter
- Comparações em *strings* são *case-sensitive*.

"La%" – qualquer *substring* que comece com "La". Ex.: Lages, Lagoa Vermelha, etc.

"%na%" – qualquer *string* que possua uma *substring* "na". Ex.: Canasvieiras, Parnaíba, etc.

"___" - qualquer *string* com exatamente 3 caracteres. Ex.: uma, duo, ema, etc.

"___%" – qualquer *string* com pelo menos 3 caracteres. Ex.: Adão, Eva, etc.

Usa-se o operador *like* para comparações, como no exemplo:

```
select  nome_ rua
from    ruas
where   nome_ rua like "Jo%"
```

Ex.: João Machado, Jonas Campos, etc.

Para usar diferenças em comparações, pode ser usado o operador *not like*.

4.6. Ordenação e apresentação de tuplas

- cláusula *order by*
- ordenação é feita, por *default*, por ordem ascendente. Para especificar a forma de ordenação desejada, usamos *desc* para descendente e *asc* para ascendente.

Exemplo:

```
select  nome_cliente
from    devedor, empréstimo
where   devedor.nro_empréstimo = empréstimo.nro_empréstimo
        and nome_agência = "Lages"
order by nome_cliente
```

Exemplo de ordenações especificadas:

```
select  *                (seleciona tudo)
from    empréstimo
order by total desc, nro_empréstimo asc
```

4.7. Operações de conjuntos

- Operações *union*, *intersect* e *except* (união, interseção e diferença).
- condição: as relações precisam ter o mesmo conjunto de atributos

- certos SGBDs não oferecem suporte para essas operações
Conjuntos de exemplo:

- *todos os clientes com conta no banco:*

```
select  nome_cliente  
from    depositante
```

- *todos os clientes com empréstimo:*

```
select  nome_cliente  
from    devedor
```

4.7.1. Operação de união (*union*)

- *encontrar todos os clientes com conta, empréstimo ou ambos:*

```
(select  nome_cliente  
  from    depositante)  
union  
(select  nome_cliente  
  from    devedor)
```

- *union* automaticamente elimina as repetições, não sendo necessário usar *distinct*. Para incluir as repetições:

```
(select  nome_cliente  
  from    depositante)  
union all  
(select  nome_cliente  
  from    devedor)
```

4.7.2. Operação de interseção (*intersect*)

- *encontrar todos os clientes que possuam tanto empréstimos quanto contas no banco:*

```
(select  nome_cliente  
  from    depositante)  
intersect  
(select  nome_cliente  
  from    devedor)
```

- para incluir repetições:

```
(select  nome_cliente  
  from    depositante)  
intersect all  
(select  nome_cliente  
  from    devedor)
```

4.7.3. Operação de diferença (*except*)

- encontrar todos os clientes que possuam conta e nenhum empréstimo:

```
(select nome_cliente
 from depositante)
except
(select nome_cliente
 from devedor)
```

- para incluir repetições:

```
(select nome_cliente
 from depositante)
except all
(select nome_cliente
 from devedor)
```

4.8. Funções agregadas

Funções que tomam uma coleção de valores como entrada, e retornam um valor simples. Cinco funções pré-programadas em SQL:

- Média (*average*): *avg*
- Mínimo (*minimum*): *min*
- Máximo (*maximum*): *max*
- Total (*total*): *sum*
- Contagem (*count*): *count*

- *sum* e *avg* aplicam-se somente a conjuntos de números.

- outras funções: dados não-numéricos, como *strings* e semelhantes.

Exemplo de média:

- *média dos saldos em contas na agência "Lages"*:

```
select avg (saldo)
from conta
where nome_agência = "Lages"
```

- melhorando a apresentação da consulta:

```
select avg (saldo) as Média de saldos
from conta
where nome_agência = "Lages"
```

Exemplo de contagem:

- *todos os números de depositantes de cada agência (um depositante contado somente uma vez, não importando o número de contas que possua):*

```
select nome_agência, count (distinct nome_cliente)
from depositante, conta
where depositante.nro_conta = conta.nro_conta
group by nome_agência
```

- quais agências possuam médias dos saldos em contas maior de \$ 1.200:

```
select  nome_agência, avg (saldo)
from    conta
group by nome_agência
having  avg (saldo) > 1200
```

- média de todas as contas:

```
select  avg (saldo)
from    conta
```

- número de tuplas de uma relação (tabela):

```
select  count (*)
from    cliente
```

- encontre o saldo médio para cada cliente que mora em Lages e tenha ao menos três contas:

```
select  depositante.nome_cliente, avg (saldo)
from    depositante, conta, cliente
where   depositante.nro_conta = conta.nro_conta and
        depositante.nome_cliente = cliente.nome_cliente and
        cidade_agência = "Lages"
group by depositante.nome_cliente
having  count (distinct depositante.nro_conta) >= 3
```