Guia Rápido – SQL

SQL, sigla de Structured Query Language (linguagem de consulta estruturada), é uma linguagem declarativa padrão para manipulação de bancos de dados relacionais.

1. Comandos de definição de dados (DDL – *Data Definition Language*)

1.1. CREATE

Comando empregado para criar objetos em um SGBD.

Exemplos:

- Criando um database:

```
create database embarcacoes;
```

- Criando tabelas:

```
create table classe (
   cod integer not null,
   descr varchar(20) not null,
   primary key (cod));


create table velejador (
   codv integer not null,
   nome varchar(30) not null,
   dtnasc date not null,
   classe integer not null,
   primary key(codv).
   foreign key (classe) references classe(cod));


create table barco (
   codb integer not null,
   nome varchar(20) not null,
   cor  varchar(10) not null check (cor='Azul' or cor='Branco' or
cor='Vermelho' or cor='Cinza'),
   tam  integer not null,
   pot  integer not null,
```

```
    primary key (codb));

create table reserva (
    codb integer not null,
    codv integer not null,
    data date not null,
    dhoras float,
    primary key (codb,codv, data),
    foreign key (codb) references barco(codb),
    foreign key (codv) references velejador(codv));
```

1.2. ALTER

Comando para alterar objetos em um bd.

Exemplos:

- Alterando um database (trocando o nome):

```
alter database nome_atual rename to novo_nome;
```

- Alterando uma tabela (adicionando uma coluna):

```
alter table velejador add column age integer;
```

- Alterando uma tabela (renomeando uma coluna):

```
alter table velejador rename column age to idade;
```

- Alterando uma tabela (adicionando uma restrição not null):

```
alter table velejador alter column idade set not null;
```

- Alterando uma tabela (retirando uma restrição not null):

```
alter table velejador alter column idade drop not null;
```

- Alterando uma tabela (alterando o tipo de uma coluna):

```
alter table velejador alter column idade type float;
```

- Alterando uma tabela (removendo uma coluna):

```
alter table velejador drop column idade;
```

1.3. DROP

Remove objetos do banco de dados.

Exemplos:

- Apagando uma tabela (apaga também todas as tuplas):

```
drop table velejador;
```

- Apagando um database (apaga todos os objetos contidos no database):

```
drop database embarcações;
```

2. Comandos de manipulação de dados (DML – *Data Manipulation Language)*

2.1. Insert

Comando utilizado para inserir tuplas em uma relação. Possui a seguinte sintaxe:

```
insert into <tabela> [(<atributos>)] values (<valores>);
```

Quando são fornecidos valores para todos os atributos, na mesma ordem em que aparecem no esquema da relação, a lista de atributos pode ser suprimida do comando insert. Por este motivo, aparece como opcional (entre colchetes) na sintaxe acima.

Exemplos:

- Inserindo um velejador com todos os atributos preenchidos:

```
insert into velejador (cod, nome, dtnasc, classe) values (1,'Almir
Klink','10/03/1963',4);
```

ou

```
insert into velejador values (1,'Almir Klink','10/03/1963',4);
```

- Inserindo uma reserva com o atributo dhoras vazio:

```
insert into reserva (codb, codv, data) values (4, 1, '2014/10/24');
```

ou

```
insert into reserva (codb, codv, data, dhoras) values
(4,1,'10/03/2013',NULL);
```

ou

```
insert into reserva values (4,1,'10/03/2013',NULL);
```

A data pode ser escrita nos formatos 'AAAA-MM-DD' ou 'DD-MM-AAAA', utilizando hífen (-) ou barra (/). Valores do tipo datahora ou caractere devem vir sempre entre aspas simples.

2.2. Update

Comando utilizado para atualizar valores de atributos nas tuplas. Possui a seguinte sintaxe:

```
update <tabela> set <atributo>=<novo valor> [where <predicado>];
```

Pode-se alterar mais de um atributo por vez, separando os pares <atributo> = <novo valor> por vírgula. O predicado é uma condição lógica que deve ser atendida pelas tuplas para que os valores sejam atualizados. Se não for especificado nenhum predicado, a alteração afetará todas as tuplas da relação.

Exemplos:

- Atualizando o nome do velejador:

```
update velejador set nome = 'Martin Fowler' where cod = 1;
```

- Atualizando o nome e a data de nascimento do velejador:

```
update velejador set nome = 'Martin Fowler', dtnasc='01-01-1963'
where cod = 1;
```

- Atualizando dhoras de todas as tuplas:

```
update reserva set dhoras = null;
```

2.3. Delete

Comando destinado a remover tuplas de uma relação. Sua sintaxe é:

```
delete from <tabela> [where <predicado>];
```

O predicado é uma condição lógica que deve ser atendida pelas tuplas que serão removidas. Se não for especificado nenhum predicado, todas as tuplas da relação serão apagadas.

Exemplos:

- Apagando o velejador Martim Fowler:

```
delete from velejador where nome = 'Martim Fowler';
```

- Apagando todos os velejadores:

```
delete from velejador;
```

3. Comandos de consulta (select e suas variações)

Para acessar dados armazenados nas tabelas, empregamos o comando select. Sua sintaxe completa pode ser vista abaixo:

```
select [distinct] <atributos ou *>
from <tabela(s)> [<joins>]
[where <predicados>]
[having <predicados com funções de grupo>]
[group by <atributos de agrupamento>]
[order by <atributos>]
```

Corresponde à projeção (pi)

Corresponde à seleção (sigma)

Utilizado apenas se existir uma função de grupo

Ordena o resultado por um ou mais atributos

3.1. Sintaxe básica do SELECT

A sintaxe mínima da instrução SELECT é:

```
select <campos> from <tabela>
```

Exemplos:

- Selecione somente o nome dos barcos.

```
select nome from barco;
```

- Selecione o nome, tamanho e potência dos barcos.

```
select nome, tam, pot from barco;
```

- Selecione todos os campos da tabela barco.

```
select * from barco;
```

3.1.1. Distinct

Por padrão, quando selecionamos um atributo, estamos selecionando todos os seus valores, ainda que repetidos. Isso ocorre porque, implicitamente, um comando **select** tem o mesmo efeito de um comando **select all** (selecionar todos).

Para selecionar apenas valores distintos, podemos introduzir a expressão **distinct** antes do nome do atributo.

Exemplos:

- Selecione todas as cores de barco, sem repetição

```
select distinct cor from barco;
```

3.1.2. Operações aritméticas na consulta

Além de exibir os dados tal qual aparecem no banco de dados, uma consulta também pode gerar valores calculados, utilizando as operações aritméticas de soma, subtração, multiplicação e divisão.

Exemplos:

- Selecione os nomes dos barcos e sua potência convertida em kw:

```
select nome, pot, pot/1.36 from barco;
```

3.1.3. Renomeando colunas (AS)

Os nomes das colunas, tanto as pertencentes ao esquema da relação quanto aquelas derivadas, podem ter seus nomes alterados no resultado da consulta através da inclusão da palavra as seguida do novo nome.

Exemplos:

- Selecione os nomes dos velejadores, renomeando a coluna para 'cliente'.

```
select nome as cliente from velejador;
```

- Selecione os nomes dos barcos e sua potência convertida em kw:

```
select nome, pot, pot/1.36 as kw from barco;
```

3.1.4. Funções agregadoras

São funções que servem para calcular e resumir valores. Podemos utilizar as seguintes funções agregadoras no comando SELECT:

Função	Resultado
`count(coluna)`	Conta número de linhas com valores não nulos da coluna. Se for utilizado count(*), conta o número de linhas da tabela, incluindo valores nulos.
`max(coluna)`	Retorna maior valor da coluna.
`min(coluna)`	Retorna menor valor da coluna.
`sum(coluna)`	Soma todos os valores da coluna.
`avg(coluna)`	Media aritmética dos valores da coluna.

Exemplos:

- Contando o número de linhas da tabela reserva:

```
select count(*) from reserva;
```

- Contando o número de valores não nulos de dhoras:

```
select count(dhoras) from reserva;
```

- Barco com maior potência:

```
select max(pot) from barco;
```

- Barco com menor tamanho:

```
select min(pot) from barco;
```

- Potência média:

```
select avg(pot) from barco;
```

- Soma de dhoras:

```
select sum(dhoras) from reserva;
```

3.2. Cláusula WHERE

A cláusula WHERE é opcional e indica um critério que os registros devem obedecer para aparecerem no resultado da consulta (relembre a operação seleção (σ) da álgebra relacional). Pode-se utilizar os seguintes operadores na cláusula WHERE:

Operador	Significado
`=`	Igual a
`<>`	Diferente de
`>`	Maior que
`>=`	Maior ou igual a
`<`	Menor que
`<=`	Menor ou igual a
`between <x> and <y>`	Entre dois valores (inclusive)
`in (<x>, <y>, <z>)`	Dentro da lista de valores
`like 'x%'`	Que seja igual aos caracteres da amostra (% como caractere curinga)
`is null`	É um valor nulo

Qualquer das expressões acima pode ser precedida do operador de negação **not**. Duas condições podem ser combinadas através dos operadores lógicos **and** ou **or**.

Exemplos:

- Listar dados de todas as reservas que ainda não encerraram:

```
select * from reserva where dhoras is null
```

- Listar dados de todas as reservas já encerradas:

```
select * from reserva where dhoras is not null
```

- Listar dados dos velejadores cujo nome começa com 'A':

```
select * from velejador where nome like 'A%'
```

- Listar dados dos velejadores cujo nome possui a string 'ar' em qualquer posição:

```
select * from velejador where nome like '%ar%'
```

- Listar dados dos velejadores cujo nome termina com 'A':

```
select * from velejador where nome not like '%A%'
```

- Listar todos os barcos com tamanho entre 10 e 20 (inclusive):

```
select * from barco where tam >= 10 and tam =<20;
```

ou

```
select * from barco where tam between 10 and 20;
```

- Listar todos os barcos com tamanho igual a 10 ou 15 ou 20:

```
select * from barco where tam = 10 or tam = 15 or tam = 20;
```

ou

```
select * from barco where tam in(10, 15, 20);
```

3.3. Cláusula ORDER BY

A cláusula ORDER BY é opcional, sendo usada para exibir os dados em ordem crescente ou decrescente pelo valor de um ou mais campos. O padrão da ordem de classificação é ascendente (A a Z, 0 a 9), denominado ASC. Para utilizar ordem decrescente, devemos informar através do comando DESC. Como a ordem crescente é padrão, não é necessário colocar a palavra ASC ao final da declaração SQL.

Podemos incluir campos adicionais na cláusula ORDER BY. Os registros são classificados primeiramente pelo primeiro campo listado em ORDER BY. Os registros que tiverem valores iguais naquele campo são classificados pelo valor no segundo campo listado e assim por diante.

Exemplos:

- Listando todos os barcos ordenados por cor em ordem crescente:

```
select * from barco order by cor;
```

- Listando todos os barcos ordenados por cor em ordem decrescente:

```
select * from barco order by cor desc;
```

- Listando todos os barcos ordenados por cor, e para os que têm a mesma cor, ordenar por nome:

```
select * from barco order by cor, nome;
```

3.4. Cláusula GROUP BY

Agrupa os registros por um determinado campo, quando se utiliza alguma função agregadora como count, sum, etc.

Exemplos:

- Contando a quantidade de barcos por cor:

```
select cor, count(*) from barco group by cor;
```

- Calculando a média de potência por cor:

```
select cor, avg(pot) from barco group by cor;
```

3.5. Cláusula HAVING

Quando agrupamos dados com group by, a cláusula HAVING pode ser utilizada para especificar quais grupos deverão ser exibidos, restringindo-os. Isso porque as funções agregadoras não podem ser utilizadas na cláusula where.

Exemplos:

- Contando a quantidade de barcos por cor, e exibindo somente as cores que possuem 2 ou mais barcos:

```
select cor, count(*) from barco group by cor having count(*) >=2;
```

- Contando a quantidade de velejadores por categoria (usando junção), mostrando somente classes que têm mais de um velejador:

```
select classe.descr, count(*)
from velejador join classe on velejador.classe=classe.cod
group by classe.descr
having count(*) > 1
```

3.6. Junções internas

Com frequência necessitamos obter dados provenientes de diversas tabelas, pois trabalhando com o modelo relacional de banco de dados, durante a normalização os dados serão colocados em tabelas diferentes para evitar a repetição de informações. Sempre que existe um relacionamento entre duas tabelas, pode-se uni-las através de uma operação de junção.

As junções internas usam um operador de comparação para relacionar duas tabelas com base nos valores em colunas comuns de cada tabela. O resultado conterá SOMENTE os registros das duas tabelas que possuem relacionamento. Se houver algum registro, em qualquer das duas tabelas, que não esteja relacionado, ele não aparecerá no resultado de uma junção interna.

3.6.1. Equi-junção

A junção interna do tipo equi-junção é especificada através da palavra chave **JOIN** (ou da expressão INNER JOIN) na cláusula **FROM** de uma instrução **SELECT**. É necessário também especificar a condição de junção, isto é, quais campos estão ligando as duas tabelas. Em uma equi-junção interna, estes campos devem ser comparados através de uma igualdade, conforme sintaxe abaixo:

```
SELECT <campo1, ..., campoN>
FROM tabela1 JOIN tabela2 ON tabela1.PK = tabela2.FK
```

onde PK é o nome do campo chave primária, e FK é o nome do campo chave estrangeira das respectivas tabelas.

Quando há quaisquer atributos com nome idêntico nas tabelas em questão, é imprescindível prefixar o nome do atributo com o nome da tabela, sob pena de termos um erro de referência ambígua. Este mecanismo também pode ser utilizado opcionalmente nos casos em que os nomes dos atributos não são ambíguos, mas a adição do nome da tabela como prefixo proporciona maior clareza. Também podemos atribuir um apelido para cada tabela e utilizá-lo no interior da consulta.

Exemplos:

- Listando nomes dos barcos e data da reserva:

```
select nome, data
from barco join reserva on barco.codb = reserva.codb;
```

ou

```
select barco.nome, reserva.data
from barco join reserva on barco.codb = reserva.codb;
```

ou

```
select b.nome, r.data
from barco b join reserva r on b.codb=r.codb;
```

- Listando nomes dos velejadores e respectivas classes:

```
select nome, descr
from velejador v join classe c on v.classe=c.cod;
```

- Listando o nome do velejador e do barco cuja reserva ainda não foi finalizada:

```
select v.nome, b.nome
from   velejador v join reserva r on v.cod=r.codv
join barco b on r.codb=b.codb
where  r.dhoras is null;
```

- Listando nome do velejador, nome do barco, data e duração das reservas, somente para velejadores da classe iniciante:

```
select v.nome, b.nome, r.data, r.dhoras
from velejador v join classe c on v.classe=c.cod
join reserva r on r.codv = v.cod
join barco b on r.codb=b.codb
where  c.descr='Iniciante';
```

3.6.2. Junção natural

A junção natural pode ser utilizada quando os atributos de junção possuem o mesmo nome. Neste caso, é desnecessário declarar a condição de junção. A única diferença em relação aos exemplos

anteriores é que o atributo de junção aparecerá apenas uma vez no resultado da consulta.

- Listando nomes dos barcos e data da reserva:

```
select nome, data
from barco natural join reserva;
```

3.7. Junções externas

3.7.1. Left join

O conjunto de resultados de uma junção externa inclui todas as linhas de uma tabela esquerda especificadas na cláusula LEFT JOIN, não somente aquelas nas quais as colunas unidas se correspondem. Quando uma linha na tabela esquerda não possui linhas correspondentes na tabela direita, a linha associada ao conjunto de resultados conterá valores nulos para todas as colunas da lista de seleção vindos da tabela direita.

O formato de uma consulta em duas tabelas utilizando o LEFT JOIN (ou LEFT OUTER JOIN) para especificar a condição de junção é:

```
SELECT <campo1, ..., campoN>
FROM tabela1 LEFT JOIN tabela2
ON tabela1.PK = tabela2.FK
```

onde PK é o nome do campo chave primária, e FK é o nome do campo chave estrangeira das respectivas tabelas.

Exemplos:

- Listando todos os barcos e suas reservas (se houver):

```
select * from barco left join reserva on barco.codb = reserva.codb;
```

3.7.2. Right Join

Uma junção externa direita é o oposto de uma junção externa esquerda. São retornadas todas as linhas da tabela direita. São retornados valores nulos para a tabela esquerda sempre que uma tabela direita não possuir alguma linha de correspondência na tabela esquerda.

Obviamente, se trocarmos os nomes das tabelas de posição, uma junção a direita torna-se esquerda, e vice-versa.

O formato de uma consulta em duas tabelas utilizando o RIGHT JOIN (ou RIGHT OUTER JOIN) para especificar a condição de junção é:

```
SELECT <campo1, ..., campoN>
FROM tabela1 RIGHT JOIN tabela2
ON tabela1.PK = tabela2.FK
```

onde PK é o nome do campo chave primária, e FK é o nome do campo chave estrangeira das respectivas tabelas.

Exemplos:

- Listando todos os barcos e suas reservas (se houver):

```
select * from reserva right join barco on barco.codb = reserva.codb;
```

3.7.3. Full join

Uma junção externa completa retorna todas as linhas em ambas as tabelas, direita e esquerda. Sempre que uma linha não tiver correspondência com a outra tabela, as colunas da lista de seleção da outra tabela conterão valores nulos. Quando houver correspondência entre as tabelas, todas as linhas do conjunto de resultados conterão valores de dados de tabelas base. Basta utilizar FULL JOIN.

3.8. Operações de conjunto

Pode-se combinar os resultados de duas consultas utilizando as operações de conjunto união, interseção e diferença. Em qualquer caso, as duas consultas devem ser "compatíveis", isto é, ambas devem retornar o mesmo número de colunas e as colunas correspondentes devem possuir tipos de dado compatíveis.

3.8.1. União

A operação UNION anexa o resultado da consulta2 ao resultado da consulta1 (embora não haja garantia que esta seja a ordem que as linhas realmente retornam). Além disso, são eliminadas do resultado as linhas duplicadas, a não ser que seja utilizado UNION ALL.

Sintaxe:

```
consulta1 UNION [ALL] consulta2
```

onde consulta1 e consulta2 são consultas que podem utilizar qualquer uma das funcionalidades mostradas até aqui.

Exemplos:

- União dos velejadores que reservaram o barco 1 e dos que reservaram o barco 3:

```
select v.nome from velejador v join reserva r on v.cod=r.codv and r.codb=1
union
select v.nome from velejador v join reserva r on v.cod=r.codv and r.codb=3
```

3.8.2. Intersecção

A operação INTERSECT retorna todas as linhas presentes tanto no resultado da consulta1 quanto no resultado da consulta2. As linhas duplicadas são eliminadas, a não ser que seja utilizado

INTERSECT ALL.

Sintaxe:

```
consulta1 INTERSECT [ALL] consulta2
```

onde consulta1 e consulta2 são consultas que podem utilizar qualquer uma das funcionalidades mostradas até aqui.

Exemplos:

- Velejadores que reservaram o barco 1 e também o barco 3:

```
select v.nome from velejador v join reserva r on v.cod=r.codv and r.codb=1
intersect
select v.nome from velejador v join reserva r on v.cod=r.codv and r.codb=3
```

3.8.3. Diferença

A operação EXCEPT retorna todas as linhas presentes no resultado da consulta1, mas que não estão presentes no resultado da consulta2 (diferença entre duas consultas). Novamente, as linhas duplicadas são eliminadas, a não ser que seja utilizado EXCEPT ALL.

Sintaxe:

```
consulta1 EXCEPT [ALL] consulta2
```

onde consulta1 e consulta2 são consultas que podem utilizar qualquer uma das funcionalidades mostradas até aqui.

Exemplos:

- Velejadores que reservaram o barco 1, mas não o barco 3:

```
select v.nome from velejador v join reserva r on v.cod=r.codv and r.codb=1
except
select v.nome from velejador v join reserva r on v.cod=r.codv and r.codb=3
```

3.9. Subconsultas

Subconsultas ou consultas aninhadas são “consultas dentro de consultas”. Uma subconsulta é sempre especificada entre parênteses e retorna resultados que podem ser referenciados pela consulta mais externa.

Exemplos:

- Nome do barco e total de reservas:

```
select b.nome, (select count(*) from reserva r where r.codb =
b.codb) as num from barco b
```

- Nome dos barcos nunca reservados:

```
select b.nome
from barco b where not exists (select * from reserva r where
r.codb=b.codb)
```

- Dados do maior barco:

```
select *
from barco
where tam = (select max(tam) from barco);
```

- Dados dos barcos com tamanho acima da média:

```
select *
from barco
where tam > (select avg(tam) from barco);
```

3.10. Operação de divisão

A operação de divisão, vista na álgebra relacional, não possui comando equivalente na SQL. No entanto, podemos obter este resultado através de 2 “not exists”.

Exemplo:

- Velejadores que reservaram todos os barcos:

```
select distinct codv from reserva r1 where not exists
(select * from barco where not exists
(select * from reserva r2 where r1.codv=r2.codv and
barco.codb=r2.codb));
```

A consulta acima pode ser interpretada como: “selecione os códigos dos velejadores cujo conjunto de barcos não reservados por ele seja vazio”. Ou seja, os velejadores que reservaram todos os barcos.