Consultas utilizando a linguagem SQL

Utilizarei o banco MySQL, atualize a sintaxe do SQL para a versão do seu banco, caso necessário

DQL - Data Query Language

Iremos abordar SELECT e JOIN que são possíveis através do DQL.

Tabela Profissoes

```
CREATE TABLE profissoes(
        profissao_id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        cargo VARCHAR(100)
);


INSERT INTO profissoes VALUES (1,'PROGRAMADOR');
INSERT INTO profissoes VALUES (2,'DBA');
INSERT INTO profissoes VALUES (3,'ANALISTA');
```

Tabela Clientes

```
CREATE TABLE clientes(
        cliente_id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        id_profissao BIGINT,
        nome VARCHAR(100),
        CONSTRAINT fk_clientes_profissoes FOREIGN KEY (id_profissao) REFERENCES
        profissoes(profissao_id)
);
INSERT INTO clientes VALUES (1,1,'Fabio');
INSERT INTO clientes VALUES (2,2,'Fabio');
INSERT INTO clientes VALUES (3,3,'Rafael');
INSERT INTO clientes VALUES (4,null,'Lucia');
```

SELECT em múltiplas tabelas

```
SELECT c.nome AS Cliente_Nome, p.cargo AS Cliente_Cargo
    FROM clientes AS c, profissoes AS p
    WHERE c.id_profissao = p.profissao_id;
```

Portugol:

```
SELECIONE c.nome COMO Cliente_Nome, p.cargo COMO Cliente_Profissão
    A PARTIR DE clientes COM APELIDO c, profissoes COM APELIDO p
    QUANDO c.id_da_profissao_do_cliente = p.id_da_profissao;
```

#	Cliente_Nome	Cliente_Profissão
1	Fabio	PROGRAMADOR
2	Fabio	DBA
3	Rafael	ANALISTA

Produto Cartesiano

Uma junção entre duas tabelas que origina uma terceira constituída por todos os elementos da primeira tabela combinada com todos os elementos da segunda.

SELECT * FROM profissoes, clientes;

#	profissao_id	cargo	cliente_id	id_profissa	nome
1	1	PROGRAMADOR	1	1	Fabio
2	2	DBA	1	1	Fabio
3	3	ANALISTA	1	1	Fabio
4	1	PROGRAMADOR	2	2	Fabio
5	2	DBA	2	2	Fabio
6	3	ANALISTA	2	2	Fabio
7	1	PROGRAMADOR	3	3	Rafael
8	2	DBA	3	3	Rafael
9	3	ANALISTA	3	3	Rafael
10	1	PROGRAMADOR	4	NULL	Lucia
11	2	DBA	4	NULL	Lucia
12	3	ANALISTA	4	NULL	Lucia

Junção Interna (Inner Join)

Uma junção interna é caracterizada por uma consulta que retorna apenas os dados que atendem às condições de junção, isto é, quais linhas de uma tabela se relacionam com as linhas de outras tabelas. Para isso utilizamos a cláusula ON, que é semelhante à cláusula WHERE.

```
SELECT cliente_id, c.nome, p.cargo

FROM clientes AS c INNER JOIN profissoes AS p

ON c.id_profissao = p.profissao_id;
```

#	Cliente_Nome	Cliente_Profissão
1	Fabio	PROGRAMADOR
2	Fabio	DBA
3	Rafael	ANALISTA

Junção Interna (Inner Join)

Portugol:

```
SELECIONE cliente_id, c.nome, p.cargo

A PARTIR DE clientes COM APELIDO c

JUNÇÃO INTERNA COM profissoes COM APELIDO p

QUANDO c.id_da_profissao_do_cliente = p.id_da_profissao;
```

#	Cliente_Nome	Cliente_Profissão
1	Fabio	PROGRAMADOR
2	Fabio	DBA
3	Rafael	ANALISTA

Junção Externa (Outer Join)

Uma junção externa é uma consulta que não requer que os registros de uma tabela possuam registros equivalentes em outra. Este tipo de junção se subdivide dependendo da tabela do qual admitiremos os registros que não possuem correspondência: a tabela da esquerda, a direita ou ambas.

Junção Externa à Esquerda (Left Outer Join)

O resultado desta consulta sempre contém todos os registros da tabela ESQUERDA (ou seja, a primeira tabela mencionada na consulta), mesmo quando não existam registros correspondentes na tabela direita. Desta forma, esta consulta retorna todos os valores da tabela ESQUERDA com os valores da tabela direita correspondente, e quando não há correspondência retorna um valor NULL.

```
SELECT * FROM clientes

LEFT OUTER JOIN profissoes ON clientes.id_profissao = profissoes.profissao_id;
```

#	cliente_id	id_profissao	nome	profissao_id	cargo
1	1	1	Fabio	1	PROGRAMADOR
2	2	2	Fabio	2	DBA
3	3	3	Rafael	3	ANALISTA
4	4	NULL	Lucia	NULL	NULL

Junção Externa à Esquerda (Left Outer Join)

Portugol:

SELECIONE A PARTIR DE clientes

FAZENDO JUNÇÃO EXTERNA À ESQUERDA COM profissoes

QUANDO clientes.id_da_profissao_do_cliente = profissoes.id_da_profissao;

#	cliente_id	id_profissao	nome	profissao_id	cargo
1	1	1	Fabio	1	PROGRAMADOR
2	2	2	Fabio	2	DBA
3	3	3	Rafael	3	ANALISTA
4	4	NULL	Lucia	NULL	NULL

Junção Externa à Direita (Right Outer Join)

O resultado desta consulta sempre contém todos os registros da tabela DIREITA (ou seja, a segunda tabela mencionada na consulta), mesmo quando não existam registros correspondentes na tabela esquerda. Desta forma, esta consulta retorna todos os valores da tabela DIREITA com os valores da tabela esquerda correspondente, e quando não há correspondência retorna um valor NULL.

```
SELECT * FROM clientes

RIGHT OUTER JOIN profissoes ON clientes.id_profissao = profissoes.profissao_id;
```

#	cliente_id	id_profissao	nome	profissao_id	cargo
1	1	1	Fabio	1	PROGRAMADOR
2	2	2	Fabio	2	DBA
3	3	3	Rafael	3	ANALISTA

Junção Externa à Direita (Right Outer Join)

Portugol:

SELECIONE A PARTIR DE clientes

FAZENDO JUNÇÃO EXTERNA À DIREITA COM profissoes

QUANDO clientes.id_da_profissao_do_cliente = profissoes.id_da_profissao;

#	cliente_id	id_profissao	nome	profissao_id	cargo
1	1	1	Fabio	1	PROGRAMADOR
2	2	2	Fabio	2	DBA
3	3	3	Rafael	3	ANALISTA

Full Outer Join

Esta consulta apresenta todos os dados das tabelas à esquerda e à direita, mesmo que não possuam correspondência em outra tabela. A tabela combinada possuirá assim todos os registros de ambas as tabelas e apresentará os valores nulos para os registros sem correspondência.

```
SELECT * FROM clientes

LEFT OUTER JOIN profissoes

ON clientes.id_profissao = profissoes.profissao_id UNION

SELECT * FROM clientes

RIGHT OUTER JOIN profissoes

ON clientes.id_profissao = profissoes.profissao_id;
```

Full Outer Join

Portugol:

SELECIONE A PARTIR DE clientes

FAZENDO JUNÇÃO EXTERNA À ESQUERDA COM profissoes

QUANDO clientes.id_da_profissao_do_cliente = profissoes.id_da_profissao;

UNINDO COM O RESULTADO DE

SELECIONE A PARTIR DE clientes

FAZENDO JUNÇÃO EXTERNA À DIREITA COM profissoes

QUANDO clientes.id_da_profissao_do_cliente = profissoes.id_da_profissao;

#	cliente_id	id_profissao	nome	profissao_id	cargo
1	1	1	Fabio	1	PROGRAMADOR
2	2	2	Fabio	2	DBA
3	3	3	Rafael	3	ANALISTA
4	4	NULL	Lucia	NULL	NULL

CROSS JOIN

Esta consulta é usada quando queremos juntar duas ou mais tabelas por cruzamento. Ou seja, para cada linha de uma tabela queremos todos os dados da outra tabela e vice-versa.

```
SELECT c.id_profissao, c.nome, p.cargo
FROM clientes AS c
CROSS JOIN profissoes AS p;
```

Portugol:

```
SELECIONE c.id_da_profissao_do_cliente, c.nome_do_cliente, p.cargo_da_profissao

A PARTIR DE cliente COM APELIDO c

FAZENDO JUNÇÃO CRUZADA EM profissoes COM APELIDO p;
```

OBS: o resultado desse SELECT é equivalente ao do produto cartesiano*

#	profissao_id	cargo	cliente_id	id_profissa	nome
1	1	PROGRAMADOR	1	1	Fabio
2	2	DBA	1	1	Fabio
3	3	ANALISTA	1	1	Fabio
4	1	PROGRAMADOR	2	2	Fabio
5	2	DBA	2	2	Fabio
6	3	ANALISTA	2	2	Fabio
7	1	PROGRAMADOR	3	3	Rafael
8	2	DBA	3	3	Rafael
9	3	ANALISTA	3	3	Rafael
10	1	PROGRAMADOR	4	NULL	Lucia
11	2	DBA	4	NULL	Lucia
12	3	ANALISTA	4	NULL	Lucia

SELF JOIN

Esta consulta é uma junção de uma tabela a si mesma. Para executar esse exemplo, vamos adicionar mais uma instância a tabela Clientes.

```
INSERT INTO clientes VALUES (5,1,'Raquel');

SELECT c1.nome AS Cliente1, c2.nome AS Cliente2, c1.id_profissao
FROM clientes AS c1
INNER JOIN clientes AS c2
ON c1.cliente_id  <> c2.cliente_id
AND c1.id_profissao = c2.id_profissao;

DELETE FROM clientes WHERE clientes.cliente_id=5;
```

#	Cliente1	Cliente2	id_profissao
1	Raquel	Fabio	1
2	Fabio	Raquel	1

SELF JOIN

Portugol:

SELECIONE c1.nome COMO Cliente1, c2.nome COMO Cliente2, c1.id_da_profissao

A PARTIR DE clientes COM APELIDO c1

FAZENDO JUNÇÃO INTERNA COM clientes COM APELIDO c2

QUANDO c1.id_do_cliente FOR DIFERENTE DE c2.id_do_cliente

E c1.id_da_profissao FOR IGUAL A c2.id_da_profissao

#	Cliente1	Cliente2	id_profissao
1	Raquel	Fabio	1
2	Fabio	Raquel	1

EQUI JOIN

Uma junção Equi-Join é um tipo específico de junção baseada em comparador, que usa apenas comparações de igualdade na junção.

```
SELECt * FROM profissoes JOIN clientes

ON clientes.id_profissao = profissoes.profissao_id;
```

OBS: o resultado desse SELECT é equivalente ao INNER JOIN*

#	Cliente_Nome	Cliente_Profissão
1	Fabio	PROGRAMADOR
2	Fabio	DBA
3	Rafael	ANALISTA

Natural Join

Uma junção Natual é um caso especial de Equi-Join. O resultado desta junção é o conjunto de todas as combinações que são iguais em seus nomes de atributos comuns.

OBS1: Esta junção só 'funciona' bem se os campos chaves (onde acontece os relacionamentos) tiverem o mesmo nome em ambas as tabelas.

Para testarmos, vamos criar duas tabelas.

Tabela Profissoes2

```
CREATE TABLE profissoes2(
    id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    cargo VARCHAR(100)
);

INSERT INTO profissoes2 VALUES (1,'PROGRAMADOR');
INSERT INTO profissoes2 VALUES (2,'DBA');
INSERT INTO profissoes2 VALUES (3,'ANALISTA');
```

Tabela Clientes2

```
CREATE TABLE clientes2(
    id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    id_profissao BIGINT,
    nome VARCHAR(100),
CONSTRAINT fk_clientes2_profissoes FOREIGN KEY (id_profissao) REFERENCES profissoes2(id)
);
INSERT INTO clientes2 VALUES (1,1,'Fabio');
INSERT INTO clientes2 VALUES (2,2,'Fabio');
INSERT INTO clientes2 VALUES (3,3,'Rafael');
INSERT INTO clientes2 VALUES (4,null,'Lucia');
```

Natural Join

SELECT * FROM clientes2 NATURAL JOIN profissoes2;

OBS1: O único campo comum entre as duas tabelas é o campo 'id'.

#	id	id_profissao	nome	cargo
1	1	1	Fabio	PROGRAMADOR
2	2	2	Fabio	DBA
3	3	3	Rafael	ANALISTA