Resumo SQL:

Inserção de dados,

**INSERT** **INTO** nome\_tabela [(lista\_dos\_atributos) **VALUES** (lista\_de\_atributos)

**INSERT INTO** Medicos (codm, nome, idade, especialidade, CPF, cidade) **VALUES** (4, ‘Carlos’, 28,’ortopedia’, 11000110000, ‘Joinville’);

Alteração de dados,

**UPDATE** nome\_tabela SET nome\_atributo\_1 = Valor [{, nome\_atributo\_n = Valor}] [**WHERE** condição]

**UPDATE** Medicos **SET** cidade = ‘Florianopolis’ UPDATE Ambulatorios SET capacidade = capacidade + 5, andar = 3 **WHERE** nroa = 2

Exclusão de dados,

**DELETE** **FROM** nome\_tabela [WHERE condição]

**DELETE** **FROM** Medicos **WHERE** especialidade = ‘cardiologia’ or cidade < > ‘Florianopolis’

Consulta de dados,

**SELECT** lista\_de\_atributos **FROM** tabela **WHERE** condição

Posso utilizar funções de agregação, que são:

* COUNT (contador de ocorrências [de um atributo])
* MAX / MIN (valores máximos / mínimo de um atributo)
* SUM (somador de valores de um atributo)
* AVG (média de valores de um atributo)

**Eliminação de duplicatas:** SELECT **DISTINCT** LISTA\_DE\_ATRIBUTOS etc...

**SELECT** distinct especialidade **FROM** Médicos

Vai retornar as especialidades dos médicos, mas sem repetir resultados iguais.

A **importância** do comando **SELECT**, lembrando que **as funções de agregação só podem ser utilizadas no comando SELECT**!

Exemplo: quantos grupos de 5 leitos podem ser formados em cada ambulatório?

**select** nroa, capacidade/5 as grupos5 **from** ambulatórios, o SQL me permite realizar essas operações!

Exemplificando a função **COUNT**:

– informar o total de médicos ortopedistas **select** count(\*) as TotalOrtopedistas **from** Médicos **where** especialidade = ´ortopedia´

– Total de médicos que atendem em ambulatórios **select** count(nroa) as Total **from** médicos

Função **SUM**:

**SELECT** sum(capacidade) as TotalAndar1 **FROM** Ambulatórios **WHERE** andar = 1

Função **AVG**:

**SELECT** avg(idade) as MediaPacFpolis **FROM** Pacientes **WHERE** cidade = ´Florianópolis´

Função **MAX/MIN:**

**SELECT** **MIN**(salário) as mínimos, **MAX**(salário) as máximo **FROM** Funcionários **WHERE** depto = ´Pessoal´ **AND** idade > 50

• Funções de Agregação com distinct

– Valores duplicados não são computados

– Exemplos:

**SELECT** **COUNT** (distinct especialidade) **FROM** Médicos

**SELECT** **AVG** (distinct salário) **FROM** Funcionários

Função **WHERE:**

Cláusulas especiais:

* **[NOT] LIKE**
* **IS [NOT] NULL**
* **[NOT] BETWEEN valor1 AND valor2**
* **NOT IN**

Busca por padrões:

**– Buscar CPF e nome dos médicos com inicial M:**

**SELECT** CPF, nome **FROM** Médicos **WHERE** nome like ´M%´

**– Buscar nomes de pacientes cujo CPF termina com 20000 ou 30000**

**SELECT** nome **FROM** Pacientes **WHERE** CPF **LIKE** '%20000‘ OR CPF **LIKE** '%30000‘

**Teste de valores nulos**

**-Exemplo:**

**-Buscar o CPF e o nome dos médicos que não dão atendimento em ambulatórios**

**SELECT** CPF,nome **FROM** Médicos **WHERE** nroaisnull

**Busca por intervalor de valores:**

**-Exemplo:**

**-Buscar os dados das consultas marcadas para o período da tarde**

**SELECT** \* **FROM** Consultas **WHERE** hora **BETWEEN** ´14:00´ AND ´18:00´

**Teste de pertinência sobre um conjunto de valores:**

**-Exemplo:**

**– Buscar os dados das médicos ortopedistas, traumatologistas e cardiologistas de Florianópolis**

**SELECT \* FROM** médicos **WHERE** cidade **= ‘**Floripa’ **AND** especialidade **in ({**ortopedista**}, {**cardiologista**}, {**traumatologista**})**

UNIÃODETABELAS **(UNION):**

O **UNION** implementa a união de tabelas.

**Sintaxe**: Consulta1SQL union Consulta2SQL

Exemplo:

– Buscar o nome e o CPF dos médicos e pacientes

**SELECT** CPF, nome **FROM** Médicos **UNION** **SELECT** CPF, nome **FROM** Pacientes

**CONSULTAS ENVOLVENDO MAIS DE UMA TABELA,**

**Exemplo de produto Cartesiano (Combinação todas com todas).**

SELECT \* FROM Pacientes, Consultas

Percebe-se que essa virgula entre as tabelas Paciente e Consultas é o nosso Produto Cartesiano.

Exemplo de como juntar chaves estrangeiras com Produto Cartesiano (Sintaxe).

SELECT CPF, nome, data FROM Pacientes, Consultas WHERE hora > ‘12:00’ AND Pacientes.codp = Consultas.codp

**Exemplo de junção (lembrar aquele famoso θ JOIN).**

**SELECT** \* **FROM** Paciente **JOIN** Consultas **on** Pacientes.codp = Consulta.codp

Isso é equivalente ao (Pacientes θ X Consultas) θ = Pacientes.codp = Consultas.codp da álgebra relacional.

**Exemplo de Junção Natural (Natural Join).**

SELECT \* FROM Pacientes NATURAL JOIN Consultas

Isso é equivalente ao (Pacientes ⨝ Consultas) da álgebra relacional.

Ele compara os atributos de mesmo nome das duas tabelas, e só traz se eles derem match.

**Exemplos de junções externas (ou não naturais).**

**SELECT** \* **FROM** Pacientes **left** **join** Consultas on Pacientes.codp = Consultas.codp

Vai selecionar tudo dos pacientes e onde Pacientes.codp der match com Consultas.codp ele vai adicionar na tabela da esquerda, se não bater fica null.

Poder ter a variação para o right join, mesma ideia.