Revisão Banco de Dados

Principais Tipos de Dados Estruturados

Tipos de dados	Descrição
CHARACTER(n)	Cadeia (String) de caracteres. Tamanho fixo n
VARCHAR(n) or	Cadeia de caracteres. Comprimento variável máximo n
CHARACTER VARYING(n)	
BINARY(n)	Cadeia (String) binária. Comprimento fixo n
BOOLEAN	Armazena valores TRUE ou FALSE
VARBINARY(n) or	Cadeia Binária. Comprimento variável máximo n
BINARY VARYING(n)	
INTEGER(p) OU INT	Numérico inteiro com precisão p, INT valores inteiros até 32.657
FLOAT	Aproximação numérica, precisão da mantissa 16
DOUBLE PRECISION	Aproximação numérica, precisão da mantissa 16

Comando SELECT

- Facilidades para projeção de informações
- Não há eliminação de duplicatas no Select
  - tabela ≡ coleção
- retorno de valores calculados
  - uso de operadores aritméticos (+,-,*,/)
- invocação de funções de agregação
  - COUNT (contador de ocorrências [de um atributo])

- MAX / MIN (valores máximo / mínimo de um atributo)
- SUM (somador de valores de um atributo)
- AVG (média de valores de um atributo)

- **Eliminação de duplicatas**

select [distinct] lista_atributos
...

- Exemplo
  - buscar as especialidades dos médicos

```
select distinct especialidade
from Médicos
```

- **Retorno de valores calculados - Exemplos**
  - quantos grupos de 5 leitos podem ser formados em cada ambulatório?

```
select nroa, capacidade/5 as grupos5
from Ambulatórios
```

$\equiv \rho_{(nroa,\ grupo5)}(\pi_{\ nroa,\ capacidade/5}(Ambulatórios))$

  - qual o salário líquido dos funcionários (desc. 10%)?

select CPF, salário – (salário * 0.1) as líquido from Funcionários

- **Função COUNT - Exemplos**
  - 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 - Exemplo**
  - informar a capacidade total dos ambulatórios do primeiro andar

```
select sum(capacidade) as TotalAndar1
from Ambulatórios
where andar = 1
```

- **Função AVG - Exemplo**
  - informar a média de idade dos pacientes de Florianópolis

```
select avg(idade) as MediaPacFpolis
from Pacientes
where cidade = ´Florianópolis´
```

- **Funções MAX / MIN - Exemplo**
  - informar o menor e o maior salário pagos aos Funcionários do departamento pessoal com mais de 50 anos

```
select min(salário) as mínimo,
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
```

Cláusula WHERE

- **Busca por padrões**

where atributo like ´padrão´
% : casa com qq cadeia de caracteres

- Exemplos
  - buscar CPF e nome dos médicos com inicial M

```
select CPF, nome
from Médicos
where nome like ´M%´
```

- Exemplos
  - buscar nomes de pacientes cujo CPF termina com 20000 ou 30000

```
select nome
from Pacientes
```

```
where CPF like '%20000‘
or CPF like '%30000‘
```

- Observações
  - em alguns dialetos SQL, ´*´ é usado invés de ´%´

- **Busca por intervalos 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 elemento-conjunto - Exemplo**

  - buscar os dados das médicos ortopedistas, traumatologistas e cardiologistas de Florianópolis

```
select *
from Médicos
where cidade = ´Florianópolis´
and especialidade in (´cardiologia´,
                      ´traumatologia´,
                      ´pediatria´)
```

Junção

- **Sintaxe**

```
select lista_atributos
from tabela1 [inner] join tabela2 on condição_junção [join tabela3 on ...]
[where condição]
```

CROSS JOIN

O tipo mais simples de join que podemos fazer é a CROSS JOIN, ou "produto cartesiano".
Essa join pega cada linha de uma tabela e a combina com cada linha da outra tabela.
Se tivéssemos duas listas – uma contendo 1, 2, 3 e a outra contendo A, B, C — o produto cartesiano das duas seria:
1A, 1B, 1C

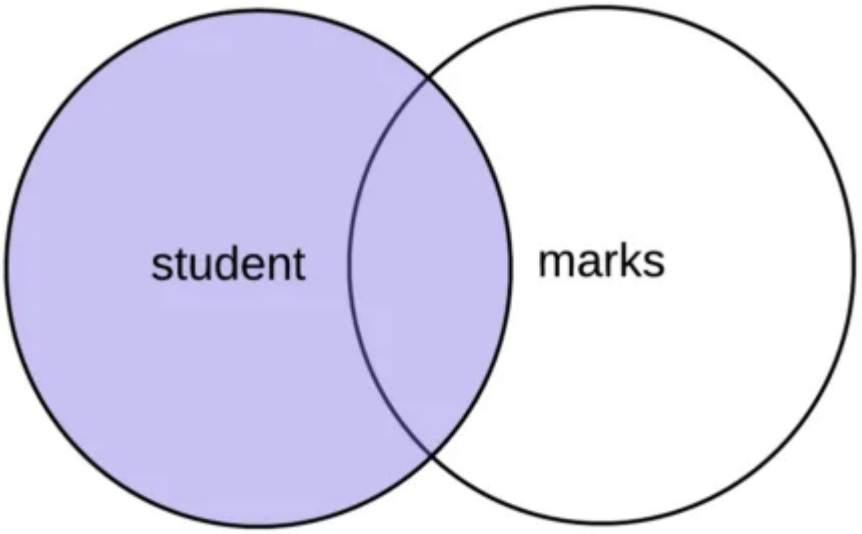
2A, 2B, 2C
3A, 3B, 3C

INNER JOIN

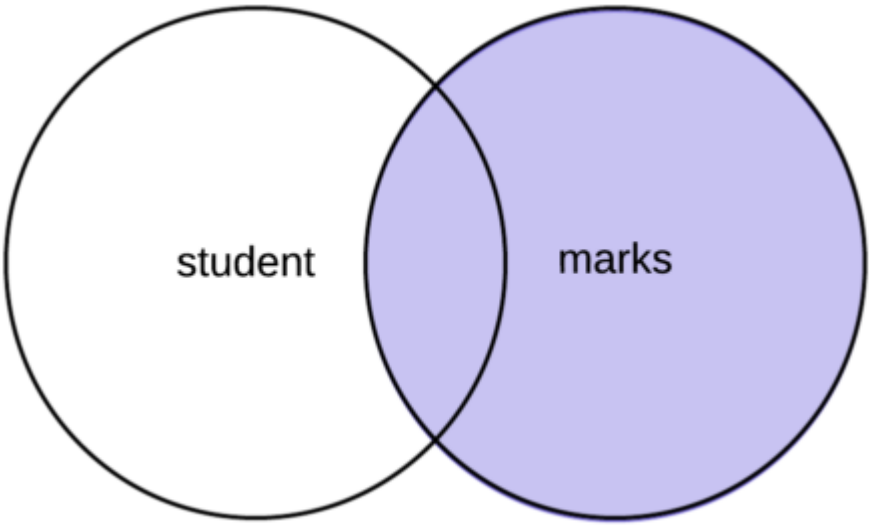
A palavra-chave INNER JOIN seleciona todas as linhas das duas tabelas, desde que haja correspondência entre as colunas. Se houver algum registro em uma das tabelas que não tenha correspondência na outra tabela, esses registros não serão exibidos.

LEFT JOIN

Usado quando queremos recuperar tuplas (linhas) sem correspondência de apenas uma única tabela, neste caso a tabela da esquerda. Aqui, todas as linhas da tabela esquerda são exibidas junto com as linhas da tabela direita que possuem linhas correspondentes às da tabela esquerda. Se não houver linhas correspondentes na tabela correta, essas entradas receberão um valor NULL por padrão.



RIGHT JOIN



FULL OUTER JOIN

- A palavra-chave FULL JOIN ou FULL OUTER JOIN é usada para selecionar todos os registros da tabela esquerda e da tabela direita. Ele combina ambas as tabelas em um conjunto de resultados e o retorna ao usuário. Observe que o MySQL FULL JOIN é conhecido por criar grandes conjuntos de dados. Funciona como a operação de união que você veria na teoria dos conjuntos em matemática. Uma junção completa é essencialmente uma união de uma junção à esquerda com uma junção à direita.
- ESSE COMANDO NÃO EXISTE NO MYSQL. E AGORA?

Subconsultas com IN

- Testam a relação de pertinência ou não-pertinência elemento-conjunto

```
select lista_atributos
from tabela1 [...]
where atributo_ou_expressão [NOT] IN
(consulta_SQL)
```

Subconsultas com ANY

- Permitem outras comparações do tipo elemento-conjunto

– testa se um valor é >, <, =, ... que algum valor em um conjunto

```
select lista_atributos
from tabela1 [, ...]
where atributo_ou_expressão [=|<|<=|>|>=|<>| !=] ANY
(consulta_SQL)
```

Subconsultas com ALL

- Realiza uma comparação de igualdade ou desigualdade de um elemento com todos os elementos de um conjunto

```
select lista_atributos
from tabela1 [, ...]
where atributo_ou_expressão [=|<|<=|>|>=|<>| !=] ALL(consulta_SQL)
```

Ordenação de Resultados

- **Cláusula ORDER BY**

```
select lista_atributos
from lista_tabelas
[where condição]
[order by nome_atributo 1 [desc] {[, nome_atributo n [desc]]} ]
```

- Exemplos

```
select *                    select salário, nome
from Pacientes      from Funcionários
order by nome               order by salário desc, nome
```

- É possível determinar a quantidade de valores ordenados a retornar

```
select ...
limit valor1 [,valor2]
```

- Exemplos

```
select *
from Pacientes
order by nome
limit 5
```

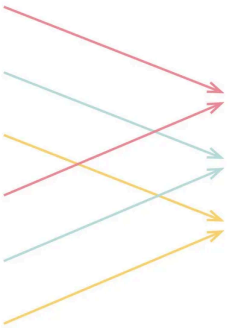
Definição de Grupos

- **Cláusula GROUP BY**

select lista_atributos
from lista_tabelas
[where condição]
[group by lista_atributos_agrupamento
[having condição_para_agrupamento]]

- **GROUP BY**
  - define grupos para combinações de valores dos atributos definidos em lista_atributos_agrupamento
  - apenas atributos definidos em lista_atributos_agrupamento podem aparecer no resultado da consulta
  - geralmente o resultado da consulta possui uma função de agregação

title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1



genre	total
adventure	7
fantasy	8
romance	3