

Resumo de conceitos de DB

Uma base de dados relacional é tipicamente constituída por várias tabelas
Exemplo

Alunos

IDaluno	nome
1	João
2	Marta
3	Raquel
4	Pedro

Notas

IDaluno	IDdisciplina	nota
1	2	12
1	3	14
2	3	15
3	1	14
4	2	16

Disciplinas

IDdisciplina	disciplina
1	Matemática
2	Português
3	TIC
4	História

Mesmo sem saber nada sobre BD consegue deduzir as respostas a

- Que nota teve o João a Português?
- Qual o aluno que teve nota mais alta a TIC?

- É mais ou menos claro como reconstituir a informação colocada em várias tabelas
- No entanto os **SGBD** (Sistemas de Gestão de Bases de Dados) não compreendem os dados e é necessário dar-lhes uma ajuda.

Que nota teve o João a Português?

Alunos

IDaluno	nome
1	João
2	Marta
3	Raquel
4	Pedro

Notas

IDaluno	IDdisciplina	nota
1	2	12
1	3	14
2	3	15
3	1	14
4	2	16

Disciplinas

IDdisciplina	disciplina
1	Matemática
2	Português
3	TIC
4	História

- Procurando o João na tabela Alunos descobrimos que tem IDaluno=1
- Procurando Português na tabela Disciplinas descobrimos que IDdisciplina=2
- Procurando os registos da tabela Notas que têm IDaluno=1 e IDdisciplina=2 encontramos um único registo com a nota 12

Queries em SQL

Para extrair informação de bases de dados usamos as consultas (queries)

No caso da consulta para obter a nota que o João teve a Português pedimos

- a informação das tabelas Alunos, Notas e Disciplinas
- os registos são combinados de forma a que o campo IDaluno nas tabelas Alunos e Notas sejam iguais
- e que o campo IDdisciplina nas tabelas Notas e Disciplinas sejam iguais
- seleccionamos apenas os casos em que disciplina é Português e o nome é João

em SQL fazemos

```
SELECT nota FROM Alunos, Notas, Disciplinas
WHERE Alunos.IDaluno=Notas.IDaluno
      AND Notas.IDdisciplina=Disciplinas.IDdisciplina
      AND nome="João" AND disciplina="Português";
```

- SELECT diz os campos que pretendemos visualizar
- FROM indica as tabelas que precisamos para fazer a pesquisa
- WHERE indica as condições que os registos devem respeitar

Qual a nota do João a Português - SQL - condições (cont.)

```
SELECT nota FROM Alunos, Notas, Disciplinas
WHERE Alunos.IDaluno=Notas.IDaluno
      AND Notas.IDdisciplina=Disciplinas.IDdisciplina
      AND nome="João" AND disciplina="Português";
```

- A condição *Alunos.IDaluno=Notas.IDaluno AND Notas.IDdisciplina=Disciplinas.IDdisciplina* serve para restringir a combinação de registos das tabelas apenas nos casos em que faz sentido (um registo dos Alunos só pode ser combinado com uma nota desse aluno ...)
- A condição sobre João e Português serve para que apenas os registos deste aluno a essa disciplina seja visualizado

Esquema sintático de uma consulta simples em SQL

```
SELECT <lista de campos ou expressões>  
[ FROM <lista de tabelas ou subconsultas>  
  [WHERE <condição>]  
  [ORDER BY <lista de campos>]  
];
```

- os elementos entre "[" e "]" são opcionais
- uma consulta sem a parte do FROM só permite fazer cálculos (exemplo `SELECT 3*4+1`)
- uma consulta sem WHERE não precisa de respeitar condições (não aconselhado se o FROM tiver várias tabelas ou subconsultas)
- a ordenação também não é obrigatória

Junções

- Podemos usar junções para que a combinação de registos de tabelas diferentes faça sentido
- No exemplo anterior usámos condições de combinação na cláusula WHERE
- Podemos usar JOIN em vez de colocar essas condições no WHERE

A consulta anterior usando JOIN

```
SELECT nota
FROM Alunos JOIN Notas ON Alunos.IDaluno=Notas.IDaluno
      JOIN Disciplinas ON Notas.IDdisciplina=Disciplinas.IDdisciplina
WHERE nome="João" AND disciplina="Português";
```

Em algumas implementações da especificação SQL podemos usar NATURAL JOIN. Com NATURAL JOIN Todos os campos de igual nome deve ter os mesmo valores em todas as tabelas (o sqlite tem NATURAL JOIN).

```
SELECT nota
FROM Alunos NATURAL JOIN Notas NATURAL JOIN Disciplinas
WHERE nome="João" AND disciplina="Português";
```

Consultas com subconsultas

- Podemos colocar subconsultas nas **listas de tabelas e consultas** usadas no FROM
- Podemos colocar subconsultas também em condições do WHERE

Exemplos de consultas com subconsultas

```
SELECT S1.nota
FROM Alunos JOIN
    (SELECT Notas.* FROM Notas NATURAL JOIN Disciplinas
     WHERE disciplina="Português") AS S1
  ON Alunos.IDaluno=S1.IDaluno
WHERE nome="João";
```

```
SELECT nota FROM Notas
WHERE
  IDaluno=(SELECT IDaluno FROM Alunos WHERE nome="João")
AND
  IDdisciplina=
    (SELECT IDdisciplina FROM Disciplinas
     WHERE disciplina="Português");
```

Expressões Calculadas

Para além de campos na cláusula SELECT podemos usar também expressões

Exemplos

```
SELECT nota+1 FROM Notas;
```

```
SELECT avg(nota) FROM Notas NATURAL JOIN Disciplinas  
WHERE disciplina="Matemática";
```

Neste caso **avg** dá a média de todas as notas na disciplina de Matemática. Esta é uma função denominada de agregação dá a nota média de todos os registos seleccionados pelo **WHERE**. Resulta num único número.

Outras funções de agregação (principais):

count max min sum

Algumas funções que não são de agregação:

abs length round lower upper

Exemplo:

```
SELECT upper(nome) FROM Alunos;
```


Consultas com Agrupamentos

Neste tipo de consultas os registos são agrupados segundo os valores de campos escolhidos para definir o grupo (colocamos GROUP BY).

Por exemplo suponhamos que pretendemos saber qual a média das notas para cada disciplina. Neste caso não queremos a média global mas sim uma média para cada disciplina específica.

A ideia é formar grupos de modo que cada grupo contém notas da mesma disciplina.

Finalmente para cada grupo calculamos a média das notas

Em SQL escrevemos assim:

```
SELECT disciplina, avg(nota)
FROM Notas NATURAL JOIN Disciplinas
GROUP BY disciplina;
```

outro exemplo: (nº de notas por disciplina)

```
SELECT disciplina, count(nota)
FROM Notas NATURAL JOIN Disciplinas
GROUP BY disciplina;
```

Como será a consulta que dá o número de notas que cada aluno tem atribuídas?

Condições nos valores agregados

As condições que vimos até agora (utilizados no WHERE) são baseadas nos campos das tabelas.

Por vezes queremos colocar condições não nos campos das tabelas mas sim no resultado de valores agregados.

Considere-se o exemplo em que queremos calcular a média por disciplina mas ver apenas os resultados com média superior a 13.

Em SQL escrevemos:

```
SELECT disciplina, avg(nota)
FROM Notas NATURAL JOIN Disciplinas
GROUP BY disciplina
HAVING avg(nota)>12;
```

Neste caso como vemos temos que usar a cláusula **HAVING**.

O que pretendemos é mostrar apenas aqueles em que a média das nota é superior a 12 e não aqueles com nota>12.

Outros tipos de junções e aplicação

Por vezes queremos combinar registos de várias tabelas mas sem obrigar a que hajam correspondências completas.

Imagine-se o caso em que se pretende listar os nomes dos alunos e colocar se existir a nota de Matemática.

Queremos listar os alunos todos não só os com nota a Matemática.

Neste caso Um JOIN normal ou NATURAL JOIN não serve.

Temos que usar um LEFT OUTER JOIN ou RIGHT OUTER JOIN (o sqlite tem apenas o LEFT OUTER JOIN).

A consulta pretendida é

```
SELECT nome, nota
FROM Alunos LEFT OUTER JOIN
(
  SELECT IDaluno, nota
  FROM Notas NATURAL JOIN Disciplinas
  WHERE disciplina="Matemática"
) AS N1
ON Alunos.IDaluno=N1.IDaluno;
```

Outros tipos de junções e aplicação

Estas junções permitem também responder a questões do tipo quais os alunos sem nota a Matemática.

Neste caso fica:

```
SELECT nome
FROM Alunos LEFT OUTER JOIN
(
  SELECT IDaluno, nota
  FROM Notas NATURAL JOIN Disciplinas
  WHERE disciplina="Matemática"
) AS N1
ON Alunos.IDaluno=N1.IDaluno
WHERE nota IS NULL;
```

SQLITE

- é um sistema de gestão de bases de dados simples
- implementa bastantes especificações do SQL (NATURAL JOIN, TRIGGERS ...)
- não precisa de correr um servidor para correr SQL
- funciona como uma aplicação *standalone* e não necessita de instalação
- basta fazer o download e está pronta a usar

Colocar o SQLITE no computador ... descarregar do Moodle o arquivo comprimido SQLIntro.zip para uma pasta que pretenda (no interior de uma pasta a que tenha acesso de escrita).

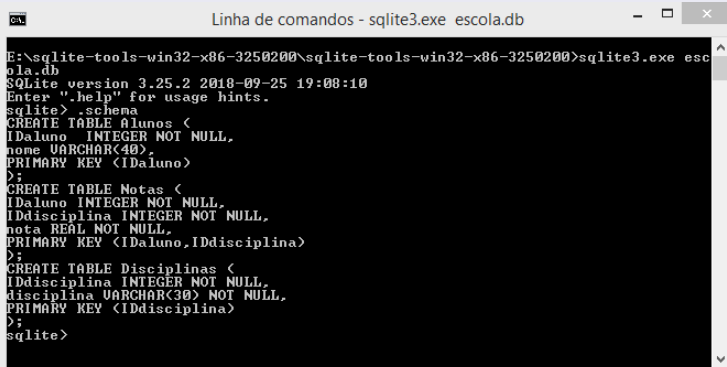
Descomprimir o arquivo ZIP nessa pasta.

Para começar a trabalhar com a base de exemplo (Alunos, Notas e Disciplinas) executar a janela de linha de comandos e escrever **sqlite3.exe escola.bd** (ENTER).

SQLITE

Podemos usando o teclado executar consultas em SQL (que terminam em ;) ou introduzir comandos especiais do SQLITE (que começam por .)

Para ver as tabelas que existem na BD usamos o comando `.schema` (ENTER)



```
CA. Linha de comandos - sqlite3.exe escola.db
E:\sqlite-tools-win32-x86-3250200\sqlite-tools-win32-x86-3250200>sqlite3.exe escola.db
SQLite version 3.25.2 2018-09-25 19:08:10
Enter ".help" for usage hints.
sqlite> .schema
CREATE TABLE Alunos (
IDaluno INTEGER NOT NULL,
nome VARCHAR(40),
PRIMARY KEY (IDaluno)
);
CREATE TABLE Notas (
IDaluno INTEGER NOT NULL,
IDdisciplina INTEGER NOT NULL,
nota REAL NOT NULL,
PRIMARY KEY (IDaluno,IDdisciplina)
);
CREATE TABLE Disciplinas (
IDdisciplina INTEGER NOT NULL,
disciplina VARCHAR(30) NOT NULL,
PRIMARY KEY (IDdisciplina)
);
sqlite>
```

Para sair do SQLITE fazemos `.quit` (ENTER)

SQLITE

Vejamos como executar uma consulta simples na BD escola.db

```
sqlite> SELECT nome, nota
...> FROM Alunos NATURAL JOIN Notas NATURAL JOIN DISCIPLINAS
...> WHERE disciplina="Português";
João:12.0
Pedro:16.0
sqlite>
```

O resultado não aparece com os nomes dos campos e desalinhado.

Podemos usar os comandos

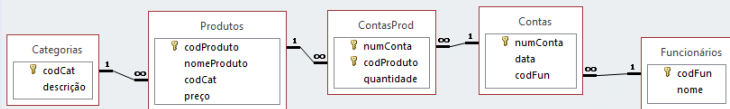
.mode tab (ENTER) e .head on (ENTER) para alinhar e colocar os nomes dos campos respetivamente.

```
sqlite> .mode tab
sqlite> .head on
sqlite>
sqlite> SELECT nome, nota
...> FROM Alunos NATURAL JOIN Notas NATURAL JOIN DISCIPLINAS
...> WHERE disciplina="Português";
nome      nota
João      12.0
Pedro     16.0
sqlite>
```

Embora a aparência do resultado não seja muito cativante é o suficiente para ver a informação pretendida.

SQLITE - Exercícios com uma base de dados maior

Na seguinte figura mostra-se a esquema de tabelas (ao estilo do ACCESS) de uma base de dados sobre informação de uma pizaria.



- a pizzaria vende vários produtos que têm uma categoria específica (ex. pizzas, bebidas, sobremesas ...)
- cada conta, que agrega os consumos do cliente, é registada por um funcionário
- cada conta pode conter vários produtos e isso encontra-se registado na tabela ContasProd.

SQLITE - Exercícios com a BD pizzaria

Para usar esta BD executar da linha de comandos `sqlite3.exe pizzaria.db` (ENTER)

Exercícios

- 1 Quais os produtos que custam mais do que 7 euros?
- 2 Quais os produtos da categoria bebidas? (por ordem do nome do produto)
- 3 Quais as contas que foram processadas pelo funcionário José Silva e em que foram registados o produto com o código 3?
- 4 Que contas têm o produto com a designação "sumo de laranja"?
- 5 Qual o produto mais caro da categoria "pizzas"?
- 6 Quantos produtos tem cada categoria? (ordenado pela designação da categoria)
- 7 Quantas contas foram processadas por cada funcionário?
- 8 Qual o valor a pagar por cada conta?
- 9 Qual o produto mais caro vendido pelo funcionário "José Silva"?
- 10 Qual a quantidade total vendida por cada produto?
- 11 Qual o produto menos vendido?
- 12 Que contas têm o produto com código 2 mas não têm o produto com código 4?
- 13 Que produto nunca foi vendido pelo funcionário "José Silva"?