SQL - Manipulação de Dados

Mestrado em BioInformática, 1º ano

Unidade Curricular de …. Bases de Dados

Aula 4

António Abelha

Departamento de Informática

Escola de Engenharia

SQL - Manipulação de Dados

- A finalidade e importância do *Structured Query Language* (SQL).
- Como escrever um comando SQL.
- Como interrogar uma base de dados utilizando a instrução SELECT.
    - - Usar a cláusula WHERE para obter linhas que satisfazem várias condições;
    - - Resultados da consulta utilizando ORDER BY;
    - - Utilizar as funções agregação em SQL;
    - - Dados de grupo usando GROUP BY;
    - - Utilização subqueries;
    - - Unir tabelas: Executar operações set (UNION, INTERSECT, EXCEPT).
- Como alterar os dados de uma base de dados utilizando INSERT, UPDATE e DELETE.

SQL - Manipulação de Dados

A linguagem surgiu a partir do desenvolvimento do modelo relacional, a ***Structured Query Language***, ou **SQL**. Ao longo dos anos, o **SQL** tornou-se a linguagem de base de dados relacional padrão. Em 1986, foi definida uma norma para o **SQL**, pelo American National Standards Institute (ANSI), que foi adotado em 1987 como padrão internacional pela Organização Internacional for Standardization (ISO, 1987).

Sistemas de Gestão de mais de uma centena de Bases de Dados suportam o **SQL**, funcionando em várias plataformas de hardware desde PCs a mainframes.

SQL - Manipulação de Dados

Objetivos do SQL

Uma linguagem de bases de dados deve permitir a um utilizador:

- criar a base de dados e as estruturas das relações;
- executar tarefas de gestão de dados básicos, tais como a inserção, alteração e exclusão de dados das relações;
- realizar consultas simples e complexas.

Uma linguagem de bases de dados deve executar essas tarefas com esforço mínimo do utilizador, e a sua estrutura e sintaxe devem ser fáceis de aprender.

SQL - Manipulação de Dados

```
SELECT      [DISTINCT | ALL] {* | [columnExpression [AS newName]] [, . . . ]}
FROM        TableName [alias] [, . . . ]
[WHERE      condition]
[GROUP BY   columnList] [HAVING condition]
[ORDER BY   columnList]
```

SQL - Manipulação de Dados

Como linguagem, a norma ISO SQL tem dois componentes principais:

Data Definition Language (DDL) - para definir a estrutura da base de dados e controlar o acesso aos dados;

Data Manipulation Language (DML) - para recuperação e atualização de dados.

É uma linguagem não-procedimental, especifica-mos quais os dados necessários, ao invés de como obtê-los. O SQL não requer a especificação dos métodos de acesso aos dados.

Como a maioria das linguagens modernas o SQL é praticamente isento de formato, o que significa que as declarações não são necessárias.

SQL - Manipulação de Dados

A estrutura dos comandos é composta por palavras em inglês, como CREATE TABLE, INSERT, SELECT. Por exemplo:

- CREATE TABLE Staff (staffNo VARCHAR (5), lName VARCHAR (15), salary DECIMAL (7,2));
- INSERT INTO Staff VALUES ('SG16', 'Brown', 8300);
- SELECT staffNo , lName , salary FROM Staff WHERE salary > 10000;

SQL pode ser usado por uma variedade de utilizadores, incluindo administradores da base de dados (DBA), gestão pessoal, programadores, e muitos outros tipos de utilizadores finais.

SQL - Manipulação de Dados

Data Manipulation Language (DML)

- **SELECT -** para consultar dados na base de dados;
- **INSERT -** para inserir dados numa tabela;
- **UPDATE -** para atualizar dados de uma tabela;
- **DELETE -** para apagar os dados de uma tabela.

SQL - Manipulação de Dados

Literais

Distinguir entre literais que são colocados entre plicas e aqueles que não são. Todos os valores de dados não-numéricos devem ser colocados entre plicas; todos os dados numéricos não devem ser colocados entre plicas. Por exemplo, podemos usar literais para inserir dados numa tabela:

INSERT INTO PropertyForRent(propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo) VALUES (‘PA14’, ‘16 Holhead’, ‘Aberdeen’, ‘AB7 5SU’, ‘House’, 6, 650.00,‘CO46’, ‘SA9’, ‘B007’);

SQL - Manipulação de Dados

A sequência de processamento de uma instrução **SELECT** é:

FROM - especifica a tabela ou tabelas a serem utilizadas

WHERE - filtra as linhas sujeitas a alguma condição

GROUP BY forma grupos de linhas com o(s) mesmo(s) valor(es) da(s) coluna(s)

HAVING - filtros de grupos sujeitos a alguma condição

SELECT - especifica que colunas devem aparecer na saída

ORDER BY - especifica a ordem de saída

A ordem das cláusulas na instrução SELECT não pode ser alterado. A apenas as duas primeiras cláusulas são obrigatórias: SELECT e FROM; as restantes são opcionais.

SQL - Manipulação de Dados

Apresentar todas as colunas e todas as linhas

Listar todos os detalhes de todos os funcionários.

Como não há restrições nesta consulta, a cláusula WHERE é desnecessária e temos de apresentar todas as colunas.

```
SELECT staffNo, fName, lName, position, sex, DOB, salary, branchNo FROM
Staff;
```

utilizando um asterisco (*) no lugar dos nomes das colunas.

```
SELECT * FROM Staff;
```

SQL - Manipulação de Dados

DISTINCT

Listar os números de todas as propriedade que foram vistas.

SELECT propertyNo FROM Viewing;

No resultado aparecem valores duplicados, ao contrário da operação Projeção da álgebra relacional o **SELECT não elimina duplicados** quando se projeta sobre uma ou mais colunas. Para eliminar os duplicados utilizamos o DISTINCT:

SELECT DISTINCT propertyNo FROM Viewing;

propertyNo
PA14
PG4
PG4
PA14
PG36

propertyNo
PA14
PG4
PG36

SQL - Manipulação de Dados

Campos calculados

Apresente uma lista dos salários mensais para todos os funcionários, que mostre o número de funcionário, o primeiro e último nome, e os detalhes do salário.

```
SELECT staffNo, fName, lName, salary/12 FROM Staff;
```

A norma ISO permite que a coluna possa ser renomeada usando a cláusula AS

```
SELECT staffNo, fName, lName, salary/12  AS monthlySalary  FROM Staff;
```

SQL - Manipulação de Dados

Seleção de linhas (cláusula WHERE)

- **Comparação -** Compara o valor de uma expressão com o valor de uma outra expressão.
- **Conjunto -** verifica se o valor de uma expressão cai dentro de um intervalo especificado de valores.
- **Pertence -** se o valor de uma expressão pertence a um conjunto de valores.
- **Pattern match -** se uma cadeia de caracteres corresponde a um padrão especificado.
- **Null -** testar se uma coluna tem um valor nulo (desconhecido).

A cláusula **WHERE** é equivalente à Seleção da álgebra relacional

SQL - Manipulação de Dados

Condição de pesquisa - Comparação

Listar todos os funcionários com um salário superior a 10.000.

SELECT staffNo, fName, lName, position, salary FROM Staff WHERE salary > 10000;

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

SQL - Manipulação de Dados

Operadores disponíveis:

= equals

<> diferente(padrão ISO) ! = diferente (permitido em SGBD

< é inferior a <= é inferior ou igual a

> é maior que > = é maior do que ou igual a

operadores lógicos AND, OR, e NOT, com parênteses (se necessário ou desejado) para alterar a precedência normal:

- avaliar uma expressão da esquerda para a direita;
- entre parênteses são avaliadas em primeiro lugar;
- NOTs são avaliadas antes operadores AND e OR;
- ANDs são avaliadas antes dos ORs.

SQL - Manipulação de Dados

Listar os endereços de todos os escritórios em Londres ou Glasgow.

```
SELECT * FROM Branch  WHERE city = 'London'  OR city = 'Glasgow';
```

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

SQL - Manipulação de Dados

Condição de pesquisa - Intervalo (BETWEEN / NOT BETWEEN)

Listar todos os funcionários com um salário entre 20.000 e 30.000.

```
SELECT staffNo, fName, lName, position, salary FROM Staff WHERE
salary BETWEEN 20000 AND 30000;

SELECT staffNo, fName, lName, position, salary FROM Staff
WHERE salary > = 20000  AND salary < = 30000;
```

SQL - Manipulação de Dados

Condição de pesquisa - Pertence (IN / NOT IN)

Listar todos os gerentes e supervisores.

SELECT staffNo, fName, lName, position FROM Staff WHERE position IN ('Manager', 'Supervisor');

SELECT staffNo, fName, lName, position FROM Staff WHERE position = 'Manager' OR position = 'Supervisor';

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

SQL - Manipulação de Dados

Condição de pesquisa - correspondencia a um padrão (LIKE / NOT LIKE)

O SQL tem dois símbolos especiais de correspondência de padrões:

% - representa qualquer sequência de zero ou mais caracteres (wildcard).

_ - representa qualquer caracter único.

Por exemplo:

LIKE 'H%' - o primeiro caracter deve ser H, mas o resto da cadeia pode ser vazia.

LIKE 'H_ _ _' - significa que deve haver exatamente quatro caracteres na sequência, a primeira das quais tem de ser um H.

LIKE '%e' - qualquer sequência de caracteres, de comprimento pelo menos 1, a última letra um *e*.

endereço **LIKE '% Glasgow%'**, uma sequência de caracteres de qualquer comprimento contendo *Glasgow*.

endereço **NOT LIKE 'H%',** o primeiro caracter não pode ser um H.

SQL - Manipulação de Dados

Condição de pesquisa - correspondência a um padrão (LIKE / NOT LIKE)

Listar todos os proprietários com a string 'Glasgow "em seu endereço.

SELECT ownerNo, fName, lName, address, telNo FROM PrivateOwner WHERE address LIKE '%Glasgow%';

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

SQL - Manipulação de Dados

Condição de pesquisa NULL (IS NULL / IS NOT NULL)

Listar os detalhes de todas as visitas à propriedade PG4 onde não tenha sido registado qualquer comentário.

- (propertyNo = 'PG4' AND comment = ' ') ou
- (propertyNo = 'PG4' AND comment < > 'too remote')

- SELECT clientNo, viewDate FROM Viewing WHERE propertyNo = 'PG4' AND comment IS NULL;

SQL - Manipulação de Dados

Ordenar resultados (ORDER BY)

A cláusula ORDER BY permite que as linhas sejam apresentadas por ordem ascendente (ASC) ou decrescente (DESC) de qualquer coluna ou combinação de colunas, independentemente do que aparece no resultado. Em algumas implementações o ORDER BY só permite elementos da lista SELECT. Em ambos os casos, a ORDER BY deve ser sempre a última cláusula da instrução SELECT.

SQL - Manipulação de Dados

Ordenar resultados (ORDER BY)

Apresente uma lista dos salários de todos os funcionários, dispostos em ordem decrescente de salário.

SELECT staffNo, fName, lName, salary FROM Staff ORDER BY salary DESC;

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

SQL - Manipulação de Dados

Ordenar resultados (ORDER BY)

Apresente uma lista abreviada das propriedades dispostas por ordem do tipo de propriedade.

```
SELECT propertyNo, type, rooms, rent FROM PropertyForRent  ORDER BY type, rent  DESC;
```

SQL - Manipulação de Dados

Funções de agregação em SQL

A norma ISO define cinco funções agregadas:

- **COUNT** - retorna o número de valores de uma coluna especificada;
- **SUM** - retorna a soma dos valores de uma coluna especificada;
- **AVG** - retorna a média dos valores de uma coluna especificada;
- **MIN** - retorna o menor valor de uma coluna especificada;
- **MAX** - retorna o maior valor de uma coluna especificada.

Estas funções operaram numa única coluna de uma tabela e retornam um único valor. **COUNT**, **MIN**, **MAX** e aplicam-se a colunas numéricas e não numéricas, o **SUM** e **AVG** apenas podem ser utilizados em campos numéricos. Cada função elimina nulos primeiro e opera apenas sobre os valores não nulos restantes. **COUNT** (*) tem uma utilização especial, que conta todas as linhas de uma tabela.

SQL - Manipulação de Dados

Funções de agregação em SQL

- COUNT(*)

Quantas propriedades têm um aluguer mensal superior a 350?

SELECT COUNT(*) AS myCount FROM PropertyForRent WHERE rent > 350;

myCount
5

SQL - Manipulação de Dados

Funções de agregação em SQL

COUNT(DISTINCT)

Quantas propriedades diferentes foram vistas em Maio de 2004?

```
SELECT COUNT( DISTINCT propertyNo)  AS myCount  FROM Viewing

WHERE viewDate  BETWEEN '1-May-04'  AND '31-May-04';
```

myCount
2

SQL - Manipulação de Dados

Funções de agregação em SQL

COUNT e SUM

Apresente o número de gerentes e a soma dos seus salários.?

SELECT COUNT(staffNo) AS myCount, SUM(salary) AS mySum FROM Staff WHERE position = 'Manager';

myCount	mySum
2	54000.00

SQL - Manipulação de Dados

Funções de agregação em SQL

MIN, MAX e AVG

Apresente o mínimo, máximo e média salarial dos funcionários?

```
SELECT MIN(salary) AS myMin, MAX(salary) AS myMax, AVG(salary) AS myAvg
FROM Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

SQL - Manipulação de Dados

Agrupar resultados (Cláusula GROUP BY)

As colunas referenciadas no GROUP BY são designadas colunas de agrupamento. Quando GROUP BY é utilizado, cada item da lista de seleção deve ser de valor único por grupo. Além disso, a cláusula SELECT só pode conter:

- nomes de coluna;
- funções de agregação;
- constantes;
- uma expressão envolvendo combinações dos anteriores.

SQL - Manipulação de Dados

Agrupar resultados (Cláusula GROUP BY)

Qual o numero de funcionários que trabalham em cada filial e a soma de seus salários.

SELECT branchNo, COUNT(staffNo) AS myCount, SUM(salary) AS mySum FROM Staff GROUP BY branchNo ORDER BY branchNo;

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

branchNo	staffNo	salary
B003	SG37	12000.00
B003	SG14	18000.00
B003	SG5	24000.00
B005	SL21	30000.00
B005	SL41	9000.00
B007	SA9	9000.00

COUNT(staffNo)	SUM(salary)
3	54000.00
2	39000.00
1	9000.00

SQL - Manipulação de Dados

Agrupar resultados (Cláusula GROUP BY)

```
SELECT branchNo, ( SELECT COUNT(staffNo)  AS myCount FROM Staff s WHERE
s.branchNo = b.branchNo),
( SELECT SUM(salary)  AS mySum FROM Staff s WHERE s.branchNo = b.branchNo)
 FROM Branch b ORDER BY branchNo;
```

Com esta versão do SELECT, os dois valores agregados são produzidos para cada filial, **apresentando se for o caso valores a zero**.

SQL - Manipulação de Dados

Restrições em agrupamentos (cláusula HAVING)

Para cada filial com mais do que um funcionário, apresente o número de funcionários e a soma dos seus salários.

```
 SELECT branchNo,  COUNT(staffNo)  AS myCount,  SUM(salary)  AS mySum
FROM Staff
GROUP BY branchNo
HAVING COUNT(staffNo) > 1
ORDER BY branchNo;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

SQL - Manipulação de Dados

Subqueries

Uma instrução SELECT pode ser usada dentro outra instrução SELECT. Os resultados dessa instrução SELECT interna (ou subselect) são utilizados na instrução externa para ajudar a determinar o conteúdo do resultado final. Um subselect pode ser usado no WHERE e HAVING. Neste caso é chamado de uma subquery ou query aninhado. Subselects podem ser utilizados no INSERT,UPDATE, e DELETE.

Existem três tipos de subquerys:

Um subquery escalar retorna uma única coluna, um único valor. Pode ser usado sempre que um valor único é necessário.

Um subquery de linha retorna várias colunas, apenas uma única linha. Um subquery de linha pode ser usado sempre que é necessário um construtor de linha,.

Um subquery tabela retorna uma ou mais colunas e várias linhas. Um subquery tabela pode ser utilizado sempre que é necessário uma tabela, por exemplo, para o predicado IN.

SQL - Manipulação de Dados

Subqueries

Apresente os funcionários cujo salário é superior ao salário médio, e mostrar a diferença do seu salário para a média.

```
 SELECT staffNo, fName, lName, position, salary – ( SELECT AVG(salary)  FROM Staff)  AS salDiffFROM Staff

WHERE salary > ( SELECT AVG(salary)  FROM Staff);
```

SQL - Manipulação de Dados

Subqueries aninhados (IN)

Apresente as propriedades que são geridas por funcionários que trabalham na filial de '163 Main St ".

```
SELECT propertyNo, street, city, postcode, type, rooms, rent  FROM PropertyForRent

WHERE staffNo  IN ( SELECT staffNo FROM StaffWHERE branchNo = ( SELECT
branchNo  FROM BranchWHERE street = ‘163 Main St’));
```

SQL - Manipulação de Dados

Queries - Multi Tabela

Apresente uma lista dos nomes de todos os clientes que visualizaram uma propriedade e o comentário registado.

```
SELECT c.clientNo, fName, lName, propertyNo, comment

FROM Client c, Viewing v

WHERE c.clientNo = v.clientNo;
```

SQL - Manipulação de Dados

Queries - Multi Tabela

O SQL possuiu as seguintes formas alternativas para especificar esta junção:

FROM Client c JOIN Viewing v ON c.clientNo = v.clientNo

FROM Client JOIN Viewing USING clientNo

FROM Client NATURAL JOIN Viewing

SQL - Manipulação de Dados

Queries - Multi Tabela

Para cada filial, apresente o números e o nome dos funcionários que gerem propriedades, incluindo a cidade em que se situa a filial e as propriedades geridas.

```
SELECT b.branchNo, b.city, s.staffNo, fName, lName, propertyNo
 FROM Branch b, Staff s, PropertyForRent p
 WHERE b.branchNo = s.branchNo  AND s.staffNo = p.staffNo
 ORDER BY b.branchNo, s.staffNo, propertyNo;
```

SQL - Manipulação de Dados

Outer joins

A operação de junção combina dados de duas tabelas, formando pares de linhas relacionadas onde as colunas relacionadas em cada tabela tem o mesmo valor.

Se uma linha de uma tabela não se relaciona, a linha é omitida da tabela de resultados.

A norma ISO fornece uma outra forma de operação de junção chamada **outer joins**

O **outer join** mantém as linhas que não satisfazem a condição de junção.

SQL - Manipulação de Dados

Outer joins

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

```
SELECT b.*, p.*
 FROM Branch1 b, PropertyForRent1 p
 WHERE b.bCity = p.pCity;
```

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

SQL - Manipulação de Dados

Outer joins

Apresentar todas as filiais e as propriedades que estão na mesma cidade.

```
SELECT b.*, p.*

FROM Branch1 b  LEFT JOIN PropertyForRent1 p  ON b.bCity = p.pCity;
```

ou

```
SELECT b.*, p.* FROM Branch1 b, PropertyForRent1 p

WHERE b.bCity = p.pCity (+);
```

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

SQL - Manipulação de Dados

Outer joins

Apresentar todas as filiais e as propriedades que estão na mesma cidade.

```
SELECT b.*, p.*

FROM Branch1 b   RIGHT JOIN PropertyForRent1 p  ON b.bCity = p.pCity;
```

ou

```
SELECT b.*, p.* FROM Branch1 b, PropertyForRent1 p

WHERE b.bCity(+) = p.pCity;
```

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

SQL - Manipulação de Dados

Outer joins

Apresentar todas as filiais e as propriedades que estão na mesma cidade.

```
SELECT b.*, p.*

FROM Branch1 b    FULL JOIN PropertyForRent1 p  ON b.bCity = p.pCity;
```

ou

```
SELECT b.*, p.* FROM Branch1 b, PropertyForRent1 p

WHERE b.bCity(+) = p.pCity(+);
```

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

SQL - Manipulação de Dados

EXISTS e NOT EXISTS

Apenas podes ser usadas em subquerys e produzem um resultado verdadeiro ou falso. EXISTS é verdadeira se, e só se existir pelo menos uma linha na tabela retornada pelo subquery; é falsa se o subquery retornar vazio. NOT EXISTS é o oposto do EXISTS.

EXISTS e NOT EXISTS verificam apenas a existência ou não existência de linhas no resultado do subquery.

A subquery pode conter qualquer número de colunas. Para simplificar, usar a seguinte forma:

(SELECT * FROM...)

SQL - Manipulação de Dados

EXISTS e NOT EXISTS

Apresente todos os funcionários que trabalham na filial de Londres.

```
SELECT staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS ( SELECT *
FROM Branch b
WHERE s.branchNo = b.branchNo  AND city = ‘London’);
```

ou

```
SELECT staffNo, fName, lName, position
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo  AND city = ‘London’;
```

SQL - Manipulação de Dados

EXISTS e NOT EXISTS

Identificar todos os clientes que visualizaram todos os imóveis com três quartos.

```
SELECT clientNo, fName, lName FROM Client c
WHERE NOT EXIST (SELECT * FROM PropertyForRent p WHERE rooms = 3
AND NOT EXISTS (SELECT * FROM Viewing v WHERE
c.clientNo = v.clientNo AND p.propertyNo = v.propertyNo))
```

SQL - Manipulação de Dados

Combinar Tabelas (UNION, INTERSECT, EXCEPT)

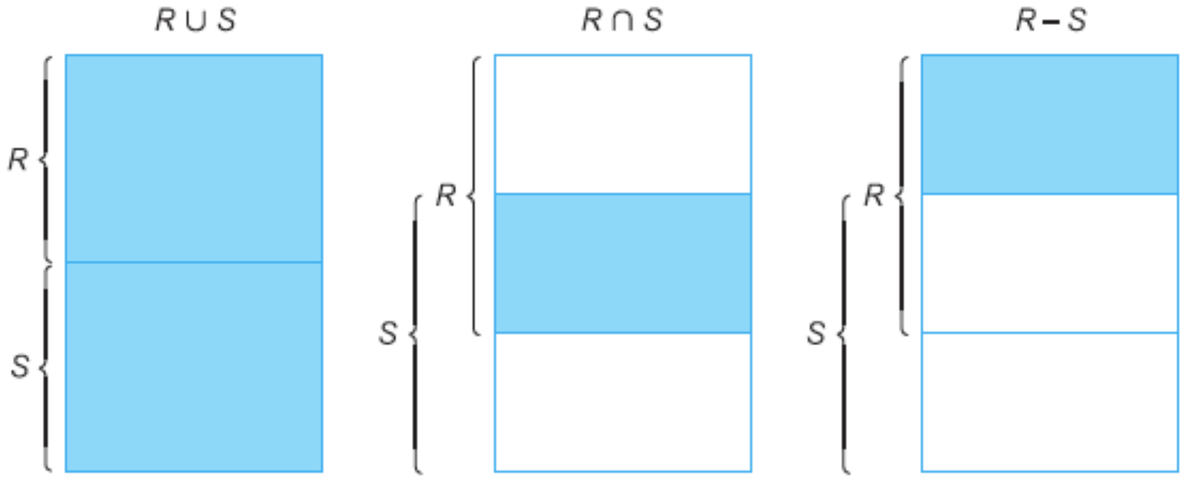
Em SQL, podemos usar as operações de **União, Interseção e Diferença** para combinar os resultados de dois ou mais querys num único resultado:

- A União de duas tabelas, A e B, é uma tabela com todas as linhas que estão, na primeira tabela A ou na segunda tabela B ou em ambas.
- A Interseção de duas tabelas, A e B, é uma tabela com todas as linhas que são comuns às duas tabelas A e B.
- A Diferença de duas tabelas, A e B, é uma tabela com todas as linhas que estão na tabela A mas não estão na tabela B.

SQL - Manipulação de Dados

Combinar Tabelas (UNION, INTERSECT, EXCEPT)

Isto implica que as duas tabelas deve conter o mesmo número de colunas, e que as colunas possuem os mesmos tipos de dados e comprimentos.



operator [ALL] - Se **ALL** for utilizado, o resultado pode incluir linhas duplicadas.

SQL - Manipulação de Dados

Uso da UNION

Apresentar uma lista de todas as cidades em que não há nem uma filial ou um imóvel.

(SELECT city **FROM** Branch **WHERE** city **IS NOT NULL)**

UNION

(SELECT city **FROM** PropertyForRent **WHERE** city **IS NOT NULL);**

city
London
Glasgow
Aberdeen
Bristol

SQL - Manipulação de Dados

Uso da INTERSECT

Apresentar uma lista de todas as cidades em que existe uma filial e um imóvel.

```
(SELECT city FROM Branch)
INTERSECT
(SELECT city FROM PropertyForRent);
```

city
Aberdeen
Glasgow
London

```
SELECT DISTINCT b.city
FROM Branch b, PropertyForRent p
WHERE b.city = p.city;
```

or

```
SELECT DISTINCT city
FROM Branch b
WHERE EXISTS (SELECT *
              FROM PropertyForRent p
              WHERE b.city = p.city);
```

SQL - Manipulação de Dados

Uso da EXCEPT

Apresentar uma lista de todas as cidades em que existe uma filial e não existe um imóvel.

```
(SELECT city FROM Branch)
 EXCEPT
(SELECT city FROM PropertyForRent);
```

city
Bristol

```
SELECT DISTINCT city
FROM Branch
WHERE city NOT IN (SELECT city
                   FROM PropertyForRent);
```

or

```
SELECT DISTINCT city
FROM Branch b
WHERE NOT EXISTS
          (SELECT *
           FROM PropertyForRent p
           WHERE b.city = p.city);
```

SQL - Manipulação de Dados

Alterações nas Bases de Dados

O SQL é uma linguagem completa para manipulação de dados e pode ser utilizada para alterara informação da base de dados, além das operações de consulta. São três os comandos SQL que estão disponíveis para modificar o conteúdo das tabelas na base de dados:

INSERT - acrescenta novas linhas numa tabela;

UPDATE - modifica os dados de uma tabela;

DELETE - remove linhas de uma tabela.

SQL - Manipulação de Dados

Adicionar dados na base de dados (INSERT)

Há duas formas de utilizar a instrução **INSERT**. A primeira permite que uma única linha seja inserida numa tabela:

INSERT INTO TableName [(columnList)] **VALUES** (dataValueList)

columnList é opcional; se omitido o SQL assume a lista de todas as colunas na versão original do CREATE TABLE da tabela.

Quando o *columnList* é apresentado, as colunas que são omitidas devem ter sido declaradas como permitindo o NULL quando a tabela foi criada.

SQL - Manipulação de Dados

Adicionar dados na base de dados (INSERT)

O *dataValueList* deve corresponder ao *columnList* da seguinte forma:

- o número de itens em cada lista deve ser o mesmo;
- deve haver uma correspondência directa na posição de itens nas duas listas, de modo que o primeiro item no *dataValueList* corresponde ao primeiro item da *columnList*, o segundo item *dataValueList* corresponde ao segundo item *columnList*, etc;
- o tipo de cada item de dados da *dataValueList* deve ser compatível com o tipo de dados da coluna correspondente.

SQL - Manipulação de Dados

INSERT . . . VALUES

Inserir uma nova linha na tabela Staff carregando valores em todas as colunas.

INSERT INTO Staff **VALUES** (‘SG16’, ‘Alan’, ‘Brown’, ‘Assistant’, ‘M’, date_format(‘1957-05-25’,’YYYY-mm-dd’), 8300,‘B003’);

SQL - Manipulação de Dados

INSERT using defaults

Inserir uma nova linha na tabela de Staff indicando as seguintes colunas: staffNo, fName, lName, position, salary, e branchNo.

INSERT INTO Staff (staffNo, fName, lName, position, salary, branchNo) **VALUES** ('SG44', 'Anne', 'Jones', 'Assistant', 8100, 'B003');

Ao inserir dados em algumas colunas, temos de especificar os seus nomes. A ordem para os nomes das colunas não é relevante, o mais normal é especificá-los na ordem em que aparecem na tabela. Em alternativa podemos representar o INSERT como:

INSERT INTO Staff **VALUES** ('SG44', 'Anne', 'Jones', 'Assistant', NULL, NULL, 8100, 'B003');

SQL - Manipulação de Dados

Adicionar dados na base de dados (INSERT)

A segunda forma da instrução INSERT permite que várias linhas sejam copiadas de uma ou mais tabelas, tem o seguinte formato:

INSERT INTO TableName [(ColumnList)] **SELECT. . .**

As linhas inseridas na tabela resultam do resultado produzido pelo *subquery*.

SQL - Manipulação de Dados

INSERT . . . SELECT

Considere a tabela StaffPropCount que contém o nome dos funcionários e o numero de propriedades que gerem:

StaffPropCount(staffNo, fName, lName, propCount)

Complete a tabela StaffPropCount com a informação das tabelas Staff e PropertyForRent.

INSERT INTO StaffPropCount
(**SELECT** s.staffNo, fName, lName, **COUNT**(*) **FROM** Staff s, PropertyForRent p **WHERE** s.staffNo = p.staffNo **GROUP BY** s.staffNo, fName, lName)
UNION
(**SELECT** staffNo, fName, lName, 0 **FROM** Staff s **WHERE NOT EXISTS** (**SELECT** * **FROM** PropertyForRent p **WHERE** p.staffNo = s.staffNo));

SQL - Manipulação de Dados

Alteração de dados em bases de dados (UPDATE)

A instrução UPDATE permite que o conteúdo de linhas existentes numa tabela possa ser alterado. O formato do comando é:

UPDATE TableName **SET** columnName1 = dataValue1 [, columnName2 = dataValue2. . .] [**WHERE** searchCondition]

Os novos *datavalue(s)* devem ser compatíveis com os tipo(s) de dados da coluna correspondente.

SQL - Manipulação de Dados

UPDATE alterar todas as linhas

Aumentar o salário de todos os funcionários em 3%.

UPDATE Staff **SET** salary = salary*1.03;

Um **UPDATE** afeta todas as linhas da tabela quando a cláusula WHERE é omitida.

SQL - Manipulação de Dados

UPDATE para linhas especificadas

Aumentar o salário de todos os Managers em 5%.

UPDATE Staff SET salary = salary*1.05 WHERE position = ‘Manager’;

UPDATE multiplas colunas

Promover David Ford (staffNo = ‘SG14’) a Manager e aumentar o seu salário para 18000.

UPDATE Staff **SET** position = ‘Manager’, salary = 18000 **WHERE** staffNo = ‘SG14’;

SQL - Manipulação de Dados

Eliminar dados de uma base de dados (DELETE)

A instrução **DELETE** permite linhas excluir de uma tabela. O formato do comando é:

DELETE FROM TableName [**WHERE** searchCondition]

A *searchCondition* é opcional; se omitida, todas as linhas da tabela são apagadas. Isso não apaga a tabela - para apagar o conteúdo e a definição de uma tabela utilizamos o comando **DROP TABLE**.

SQL - Manipulação de Dados

DELETE para linhas espenicadas

Apagar todas as visitas para a propriedade PG4.

DELETE FROM Viewing **WHERE** propertyNo = ‘PG4’;

DELETE para todas as linhas

Apagar todas as linhas da tabela Viewing.

DELETE FROM Viewing;

SQL - Manipulação de Dados

- Connolly, T., Begg, C., **Database Systems, A Practical Approach to Design, Implementation, and Management** , Addison-Wesley, 4a Edição, 2004