Desenvolvimento de Sistemas de Informação baseados em PHP e MySQL, e Java e Oracle

Francisco M. Couto
Emanuel Santos

DI–FCUL–LO–2011

DOI:10455/3167

(http://hdl.handle.net/10455/3167)

30 de Dezembro de 2011

Desenvolvimento de Sistemas de Informação baseados em PHP e MySQL, e Java e Oracle

Francisco M. Couto

Emanuel Santos

30 de Dezembro de 2011

Conteúdo

1

Introdução

Este manual tem como objectivo apoiar o desenvolvimento de sistemas de informação baseados nas linguagens de programação *PHP* e *Java* no uso dos sistemas de gestão de bases de dados relacionais *MySQL* e *Oracle*, respectivamente.

Este documento introduz os conceitos básicos explicados através de exemplos, que embora tenham sido testados na infra-estrutura informática do Departamento de Informática da FCUL podem ser fácilmente adaptados a qualquer outra infra-estrutura que possuia a mesma tecnologia.

1.1 PHP

O *PHP*[1] é uma linguagem de programação que é especialmente direccionada para o desenvolvimento de aplicações Web e que pode ser directamente integrada com código *HTML*. Da mesma forma que se pode criar um ficheiro *HTML* e disponibilizá-lo na Web através de um servidor Web, também se pode criar um ficheiro *PHP* e disponibilizá-lo na Web. A diferença reside no facto que no ficheiro *HTML* a informação a disponibilizar é estática enquanto que o ficheiro *PHP* irá disponibilizar informação dinâmica. Ou seja, o resultado de aceder ao ficheiro *HTML* através da Web é sempre o conteúdo do

[1] `http://www.php.net/`

próprio ficheiro que só muda quando o ficheiro é modificado. Pelo contrário, o resultado de aceder ao ficheiro *PHP* através da Web é o *HTML* produzido ao executar os comandos contidos nesse ficheiro. Esse resultado pode variar de acordo com os dados de entrada, conteúdo de uma base de dados, hora a que se acede, etc. Por exemplo, o ficheiro pode conter um comando que muda a cor de fundo da página de acordo com a hora de acesso ao ficheiro.

1.1.1 Servidor Web

Para disponibilizar ficheiros *PHP* terá de utilizar um servidor Web que suporte *PHP*, ou seja em que todos os ficheiros cuja extensão seja *.php* sejam executados pelo interpretador de *PHP*. Um servidor Web[2] é uma aplicação informática responsável por aceitar pedidos HTTP e devolver as respectivas respostas, que normalmente são documentos *HTML* ou objectos tais como imagens, documentos PDF, etc. Um dos mais populares servidores Web que suportam *PHP* é o Apache[3], que pode ser instalado em diversos sistemas operativos.

1.2 MySQL

O *MySQL*[4] é um sistema de gestão de bases de dados relacional, que pode ser usado pelos ficheiros *PHP* para guardar, gerir, actualizar e recolher os dados. Por exemplo, um ficheiro *PHP* pode guardar numa base de dados *MySQL* as datas de quando este foi acedido pela Web, enquanto outro ficheiro *PHP* acede a estes dados para indicar as datas onde existiram mais acessos. A

[2] `http://en.wikipedia.org/wiki/Web_server`
[3] `http://www.apache.org/`
[4] `http://www.mysql.com/`

comunicação entre o *PHP* e o *MySQL* é feita através de comandos SQL[5], que permitem criar, remover e gerir estruturas de dados assim como inserir, actualizar e apagar dados.

1.3 Oracle

O *Oracle*[6], tal como *MySQL*, é um sistema de gestão de bases de dados relacional, que pode também ser usado pelos ficheiros *PHP* para guardar, gerir, actualizar e recolher os dados. Neste manual, estas funcionalidades vão ser executadas através de um programa *Java*, via JDBC, através de comandos SQL.

1.4 Próximos Capítulos

Os capítulos seguintes irão apresentar através de exemplos a forma como construir um sistema de informação baseado na tecnologia aqui referida. O Capítulo 2 irá apresentar exemplos de ficheiros *PHP*. O capítulo 3 irá apresentar exemplos de como usar o *MySQL* através de comandos SQL e de sistema. O capítulo 4 irá apresentar exemplos de como aceder ao *MySQL* através de um ficheiro *PHP*. O capítulo 5 irá apresentar exemplos de como usar o *Oracle* através de comandos SQL e de sistema. O capítulo 6 irá apresentar um exemplo de um programa *Java* que acede ao *Oracle*.

[5] `http://en.wikipedia.org/wiki/SQL`
[6] `http://www.oracle.com/`

2

PHP

2.1 Acesso à Máquina com o Servidor Web

Para executar ficheiros *PHP* deverá identificar o local[1] que tem disponível para colocar ficheiros, onde o servidor Web irá aceder sempre que receber um pedido HTTP para esse ficheiro.

No caso da infra-estrutura do DI esse local está disponível nas directorias *public_html* de cada utilizador da máquina *appserver.di.fc.ul.pt*, onde está instalado o servidor Web. A acesso à máquina deverá ser feita através de SSH[2] ou Samba[3].

Assim, em Linux poderá escrever num terminal local (*Applications* → *System Tools* → *Terminal*) o seguinte comando (substitua *utilizador* pelo o que lhe foi fornecido):

shell

```
ssh utilizador@appserver.di.fc.ul.pt
```

No Windows poderá usar aplicações como o *SSH Secure Shell* ou o *PuTTY*[4] para aceder à máquina remota. Note que só pode aceder à máquina *appser-*

[1] `http://en.wikipedia.org/wiki/Webserver_directory_index`
[2] `http://en.wikipedia.org/wiki/Secure_Shell`
[3] `http://en.wikipedia.org/wiki/Samba_(software)`
[4] `http://en.wikipedia.org/wiki/PuTTY`

ver dentro da rede DI, ou seja num computador de um laboratório do DI ou através de VPN[5].

O comando irá pedir a senha do respectivo *utilizador*, que terá sido fornecida ou seleccionada previamente. De seguida pode executar o seguinte comando para alterar a sua senha.

shell

```
passwd
```

Este terminal remoto[6] permiti-lhe executar programas[7] directamente na máquina *appserver* através do seu computador pessoal. Para transferir ficheiros entre o seu computador pessoal e a máquina remota pode usar o comando SCP[8].

Por exemplo para copiar um ficheiro *index.php* que esteja na directoria */tmp* do computador pessoal para a directoria *public_html* do utilizador *utilizador* na máquina *appserver* deverá executar o seguinte comando a partir do terminal local (não do terminal remoto):

shell

```
scp /tmp/index.php utilizador@appserver.di.fc.ul.pt:~/public_html
```

Para fazer a transferência do ficheiro no sentido inverso deve executar o seguinte comando:

shell

```
scp utilizador@appserver.di.fc.ul.pt:~/public_html/index.php /tmp/
```

[5] `https://admin.di.fc.ul.pt/main/servicosVPN.php`
[6] `http://en.wikipedia.org/wiki/Unix_shell`
[7] `http://en.wikipedia.org/wiki/List_of_Unix_programs`
[8] `http://en.wikipedia.org/wiki/Secure_copy`

2.2 Acesso ao Servidor Web

O servidor Web do DI disponibiliza por acesso HTTP todos os ficheiros dentro da directoria *public_html* de cada utilizador. Esta é normalmente uma configuração por pré-definição dos servidores Web. Assim, o primeiro passo deve ser criar esta directoria e dar as permissões para que o servidor Web consiga aceder aos ficheiros.

Através do terminal remoto execute os seguintes comandos:

shell

```
cd ~
mkdir public_html
chmod -R 755 public_html
```

O primeiro comando altera a directoria actual do terminal remoto para a directoria raiz do utilizador. O segundo comando cria a directoria *public_html*. Os restantes comandos dão permissões[9] de acesso, de leitura e de execução a todos os outros utilizadores, incluindo o utilizador associado ao servidor Web. Considera-se que a directoria utilizador pode ser lida pelo Apache.

Após executar os comando anteriores, pode então criar um ficheiro *HTML* e colocá-lo dentro da directoria *public_html*. Para isso pode executar os seguintes comandos:

shell

```
cd public_html
echo '<html> Hello World! </html>' > index.html
```

Agora pode abrir no seu computador pessoal o link `http://appserver.di.fc.ul.pt/~utilizador/index.html` e verá que o resultado é o ficheiro

[9] `http://en.wikipedia.org/wiki/Chmod`

que criou atrás (substitua *utilizador*). No caso de estar a usar um servidor Web instalado localmente no seu computador pessoal deverá usar *localhost* em vez de *appserver.di.fc.ul.pt*. Este resultado é estático e por isso não irá mudar até que o ficheiro *index.html* na máquina *appserver* seja alterado.

Para criar uma página dinâmica pode então criar um ficheiro *PHP* e colocá-lo dentro da directoria *public_html*. Para isso pode executar os seguintes comandos:

shell

```
cd public_html
echo '<html> <?php echo date(DATE_RFC822); ?> </html>' > index.php
```

Agora ao abrir o link `http://appserver.di.fc.ul.pt/~utilizador/index.php`, verá que o resultado não é o conteúdo do ficheiro que criou atrás, mas sim o resultado de executar os comandos *PHP*, ou seja irá mostrar a data em que foi efectuado o acesso. Este resultado é dinâmico e por isso irá mudar em cada acesso sem que o ficheiro *index.php* na máquina *appserver* tenha sido alterado.

Para testar de uma forma mais rápida os ficheiros *PHP*, pode também executar o comando *php*[10] através do terminal remoto:

shell

```
php index.php
```

Para criar e editar os ficheiros *PHP* deverá usar um editor de texto[11], como o emacs[12], e não com o comando *echo* que serve apenas para criar ficheiros de pequena dimensão.

[10] `http://pt.php.net/features.commandline`
[11] `http://en.wikipedia.org/wiki/List_of_text_editors`
[12] `http://en.wikipedia.org/wiki/Emacs`

2.3 Conceitos Básicos de PHP

2.3.1 Dados de Entrada

Como qualquer linguagem de programação os ficheiros de *PHP* terão de receber dados dos utilizadores para os depois tratarem. No caso do *PHP* este pode receber dados por POST ou GET[13].

Por exemplo, crie com o seu editor de texto o ficheiro *get.php* com o seguinte conteúdo:

PHP

```
<?php
echo 'Hello ' . htmlspecialchars($_GET['name']) . '!';
?>
```

Agora ao abrir o link `http://appserver.di.fc.ul.pt/~utilizador/get.php?name=FCUL` irá ver que o resultado inclui *FCUL*.

Pode também criar o ficheiro *form.html* com o seguinte código *HTML* para criar um formulário:

PHP

```
<html>
<form action='action.php' method='post'>
 <p>Your name: <input type='text' name='name' /></p>
 <p>Your age: <input type='text' name='age' /></p>
 <p><input type='submit' /></p>
</form>
</html>
```

E crie também o ficheiro *action.php* que irá receber os resultados do *form.html* através do seguinte código:

[13] `http://en.wikipedia.org/wiki/HTTP` e `http://us3.php.net/manual/en/reserved.variables.php`

PHP

```
<html>
Hi <?php echo htmlspecialchars($_POST['name']); ?>.
You are <?php echo (int)$_POST['age']; ?> years old.
</html>
```

Agora ao submeter o formulário do link `http://appserver.di.fc.ul.pt/~utilizador/form.html` irá ver o resultado da execução do ficheiro *action.php*.

2.3.2 Acesso à Web

O *PHP* permite tratar páginas Web da mesma forma que ficheiros locais. Por exemplo, crie o ficheiro *elgoog.php* com o seguinte código:

PHP

```
<html>
<?php
$html1 = implode('', file('http://www.google.com/'));
$html2 = str_ireplace('google', 'elgoog', $html1);
echo $html2;
?>
</html>
```

Agora ao aceder ao link `http://appserver.di.fc.ul.pt/~utilizador/elgoog.php` irá ver a página do *Google* adulterada.

2.3.3 Manuais

Este capítulo tem como objectivo apenas introduzir os conceitos básicos de *PHP* de maior interesse para o projecto da disciplina em questão. Para mais informação sobre o PHP deverá consultar os seguintes manuais online:

- http://www.php.net/docs.php

- http://www.w3schools.com/php/

- http://www.biophp.org/

3

MySQL

3.1 Acesso ao Servidor MySQL

O servidor *MySQL* é um programa que gere várias bases de dados e tal como o servidor Web recebe pedidos e devolve respostas. Neste caso, o servidor *MySQL* recebe comandos SQL e de sistema e devolve os resultados respectivos. No caso da infra-estrutura do DI o servidor de *MySQL* está instalado na máquina *appserver.di.fc.ul.pt*.

Para aceder ao servidor podemos por exemplo executar o cliente de *MySQL*[1] directamente de um terminal local:

shell

```
mysql −u utilizador −p −h appserver basedados
```

Substitua *utilizador* e *basedados* pelos que lhe foram fornecidos, e indique a respectiva senha assim que for pedida.

A chamada ao cliente *MySQL* usa os seguintes parâmetros:

- *-u utilizador* : representa o utilizador para aceder ao servidor *MySQL*

- *-p* : indica que o este utilizador necessita de introduzir password

[1] `http://dev.mysql.com/doc/refman/5.1/en/mysql.html`

- *-h appserver* : indica o endereço da máquina onde está a correr o servidor *MySQL*, neste caso na máquina appserver que só está disponível dentro da rede do Departamento de Informática. Caso queira usar o servidor de *MySQL* que instalou no mesmo computador onde está a usar o cliente *MySQL* então deve omitir esta opção ou então substituir *appserver* por *localhost*.

- *basedados* : indica a base de dados que queremos aceder no servidor *MySQL*

Para saber mais sobre os parâmetros do cliente *MySQL* pode inserir o seguinte comando no terminal local (não dentro do cliente *MySQL*, execute este comando noutro terminal caso já tenha entrado no cliente *MySQL*):

shell

```
mysql -?
```

Caso o comando *mysql* não seja reconhecido certifique-se que este está instalado no seu computador[2] e que se encontra na PATH[3].

Pode também aceder à sua base de dados através da interface gráfica phpMyAdmin[4] instalada no *appserver* e disponível em: `http://appserver.di.fc.ul.pt/phpmyadmin/`.

3.2 Comandos de Sistema

O cliente de *MySQL* assemelha-se gráficamente ao terminal local, apenas se diferencia pelo texto *mysql>* antes da posição actual do cursor. Para testar

[2] `http://dev.mysql.com/doc/refman/5.1/en/installing.html`
[3] `http://en.wikipedia.org/wiki/Path_(variable)`
[4] `http://en.wikipedia.org/wiki/PhpMyAdmin`

a comunicação com o servidor *MySQL* pode introduzir o seguinte comando de sistema no cliente *MySQL*:

shell

```
show tables;
```

Não se esqueça do ponto e virgula, que é obrigatório no final de cada comando introduzido no cliente *MySQL*.

Para saber todos os comandos de sistema disponíveis use o seguinte comando:

shell

```
help;
```

O primeiro passo será agora mudar a sua senha no servidor *MySQL* através do comando:

shell

```
SET PASSWORD = PASSWORD ('senha');
```

Em que deve substituir *senha* por uma outra palavra à sua escolha.

Para sair do cliente *MySQL* e voltar ao terminal local entre o seguinte comando (sem ponto e vírgula):

shell

```
\q
```

3.3 Comandos SQL

O armazenamento e recolha de dados de uma base de dados gerida pelo servidor *MySQL* é feita através de comandos SQL. Estes comandos são executados

directamente a partir do cliente de *MySQL* ou através de uma linguagem de programação como o *PHP* (ver capítulo 4).

Deverá começar por criar um ficheiro *teste.sql* com o seu editor de texto que contenha os seguintes comandos SQL:

SQL

```
DROP TABLE IF EXISTS teste;

CREATE TABLE teste (
    x INT PRIMARY KEY,
    y VARCHAR(255));

INSERT INTO teste (x,y) VALUES (1,2),(2,4),(3,6);
```

O comnado *DROP*[5] apaga a tabela *teste* caso ela já exista na base de dados. O comando *CREATE*[6] cria a tabela *teste* com duas colunas x e y. O comando *INSERT*[7] insere três linhas na tabela *teste*.

Para executar este comandos SQL, abra agora um cliente *MySQL* como fez anteriormente e execute o seguinte comando de sistema:

shell

```
source teste.sql;
```

O resultado do executar o comando anterior deverá ser:

shell

```
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

[5] `http://dev.mysql.com/doc/refman/5.1/en/drop-table.html`
[6] `http://dev.mysql.com/doc/refman/5.1/en/create-table.html`
[7] `http://dev.mysql.com/doc/refman/5.1/en/insert.html`

```
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Caso tenha obtido um erro verifique que:

- gravou o ficheiro *teste.sql*
- o nome do ficheiro é *teste.sql* e não *teste.sql.txt*
- a directoria onde está o ficheiro *teste.sql* é a mesma onde chamou o cliente *MySQL*

Caso queira usar um ficheiro de uma outra directoria, por exemplo */tmp* pode executar:

shell

```
source /tmp/teste.sql;
```

Pode agora verificar o resultado de ter executado os comandos SQL que estavam no ficheiro, através dos seguintes comandos:

SQL

```
show tables;
desc teste;
SELECT * FROM teste;
```

O comando de sistema *show tables*[8] indica as tabelas na base de dados, e no nosso caso deverá devolver o seguinte resultado:

shell

[8] `http://dev.mysql.com/doc/refman/5.1/en/show-tables.html`

```
+--------------------+
| Tables_in_utilizador |
+--------------------+
| teste |
+--------------------+
```

O comando de sistema *desc teste*[9] descreve a estrutura da tabela, e no nosso caso deverá devolver o seguinte resultado:

shell

```
+-------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+

| x | int(11) | NO | PRI | | |
| y | varchar(255) | YES | | NULL | |
+-------+--------------+------+-----+---------+-------+

2 rows in set (0.01 sec)
```

O comando SQL *SELECT*[10] recolhe os dados da base de dados, e no nosso caso deverá devolver o seguinte resultado:

shell

```
+---+-------+
| x | y |
+---+-------+
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
```

[9] `http://dev.mysql.com/doc/refman/5.1/en/describe.html`
[10] `http://dev.mysql.com/doc/refman/5.1/en/select.html`

```
+---+-------+
3 rows in set (0.00 sec)
```

3.4 Comandos Directamente do Terminal

Os ficheiros de comandos SQL podem também ser executados directamente a partir do terminal através da redirecção[11] do STDIN e do STDOUT[12].

Por exemplo, pode executar o ficheiro *teste.sql* a partir do terminal local da seguinte forma:

shell

```
mysql -u utilizador --password=senha -h appserver basedados < teste.sql
```

O cliente *MySQL* lê os comandos directamente do ficheiro *teste.sql*, executa-os e termina sem entrar em modo interactivo.

Crie um novo ficheiro *interrogacoes.sql* com os seguintes comandos SQL:

SQL

```
SELECT * FROM teste WHERE x > 1;

SELECT '-';

SELECT y FROM teste WHERE x < 4;
```

A execução do ficheiro *interrogacoes.sql* com o seguinte comando:

shell

```
mysql -u utilizador --password=senha -h appserver basedados < interrogacoes.sql
```

irá resultar na seguinte informação:

[11] `http://en.wikipedia.org/wiki/Redirection_(computing)`
[12] `http://en.wikipedia.org/wiki/Standard_streams`

shell

```
x y
2 4
3 6
−
−
y
2
4
6
```

Pode guardar o resultado num ficheiro com formato *tsv* (tab separeted values)[13] acrescentado > *resultado.tsv* no final do comando:

shell

```
mysql −u utilizador −−password=senha −h appserver basedados < interrogacoes.sql >
    resultado.tsv
```

Pode agora abrir o ficheiro *resultado.tsv* num editor de texto ou numa folha de cálculo[14], como por exemplo o *Microsoft Excel*.

Pode também guardar num ficheiro com formato *html* usando o seguinte comando:

shell

```
mysql −−html −u utilizador −−password=senha −h appserver basedados <
    interrogacoes.sql > resultado.html
```

[13] `http://en.wikipedia.org/wiki/Delimiter-separated_values`
[14] `http://en.wikipedia.org/wiki/Spreadsheet`

3.5 Manuais de SQL

Para mais informações sobre o *Mysql* e SQL deve consultar os seguintes manuais online:

- `http://dev.mysql.com/doc/`

- `http://www.w3schools.com/sql/`

4

PHP e MySQL

O servidor *MySQL* também pode ser contactado directamente através de um programa informático usando bibliotecas de ligação a bases de dados[1]. O *PHP* é uma das linguagens de programação que pode ser configurada de forma a aceder a um servidor *MySQL*[2].

4.1 Exemplos

Nesta secção são apresentados exemplos simples de como aceder ao *MySQL* através de um ficheiro *PHP*.

Para usar um servidor *MySQL*, uma aplicação *PHP* deve começar por estabelecer uma ligação ao servidor que é determinada pelos mesmos parâmetros da chamada ao cliente *MySQL*. Desta forma deverá criar um ficheiro *config.php* com o seguinte conteúdo:

PHP

```
<?php
$dbhost = 'appserver.di.fc.ul.pt';
$dbuser = 'utilizador';
$dbpass = 'senha';
$dbname = 'basedados';
```

[1] `http://en.wikipedia.org/wiki/Odbc`
[2] `http://dev.mysql.com/doc/refman/5.1/en/apis-php.html`

```
?>
```

Não se esqueça de substituir *utilizador* pelo seu nome de utilizador, *senha* pela senha escolhida anteriormente, e *basedados* pelo nome da sua base de dados.

De seguida crie o ficheiro *opendb.php* com os seguintes comandos para estabelecer uma ligação ao servidor *MySQL*:

PHP

```
<?php
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to
    mysql');
mysql_select_db($dbname);
?>
```

E crie o ficheiro *closedb.php* com os seguintes comandos para terminar a ligação ao servidor *MySQL*:

PHP

```
<?php
mysql_close($conn);
?>
```

De seguida crie o ficheiro *students.sql* com os seguintes comandos SQL:

SQL

```
DROP TABLE IF EXISTS students;

CREATE TABLE students(
        id INT NOT NULL AUTO_INCREMENT,
        fname VARCHAR(15) NOT NULL,
        lname VARCHAR(15) NOT NULL, PRIMARY KEY(id)
);
```

```
INSERT INTO students (fname, lname) VALUES('Harry', 'Potter');

INSERT INTO students (fname, lname) VALUES('Ron', 'Wesley');

INSERT INTO students (fname, lname) VALUES('Hermione', 'Granger');

INSERT INTO students (fname, lname) VALUES('Chidori', 'Kaname');

INSERT INTO students (fname, lname) VALUES('Sagara', 'Sousuke');

INSERT INTO students (fname, lname) VALUES('Monkey D.', 'Luffy');

INSERT INTO students (fname, lname) VALUES('Roronoa', 'Zoro');

INSERT INTO students (fname, lname) VALUES('Toni − Toni', 'Chopper');

INSERT INTO students (fname, lname) VALUES('Robin', 'Nico');
```

Crie agora o ficheiro *initdb.php* com o seguintes comandos *PHP*, que estabelecem a ligação ao servidor *MySQL* e executa os comandos SQL do ficheiro *students*.

PHP

```
<?php
include 'config.php';
include 'opendb.php';
$queryFile = 'students.sql';
$queries = implode('', file($queryFile));
$arr = split(';', $queries);
array_pop($arr);
foreach ($arr as $query) {
  $result = mysql_query($query) or die('Error, query failed: '.$query);
```

```
}
include 'closedb.php';
?>
```

Para executar o ficheiro basta abri-lo através da Web, tal como foi feito no capítulo 2.

Para recolher informação dos dados armazenados pela execução do programa anterior, pode criar e executar o ficheiro *select.php* com os seguintes comandos:

PHP

```
<?php
include 'config.php';
include 'opendb.php';
$query = 'SELECT fname, lname FROM students';
$result = mysql_query($query) or die('Error, query failed');
while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
        echo 'First Name: {$row['fname']} <br>' . 'Last Name: {$row['lname']} <br
            ><br>';
}
include 'closedb.php';
?>
```

Pode inserir também mais valores na base de dados criando e executando o ficheiro *insert.php* com os seguintes comandos:

PHP

```
<?
include 'config.php';
include 'opendb.php';
$query = "INSERT INTO students (fname, lname) VALUES ('Peter', 'Pan')";
mysql_query($query) or die('Error, insert query failed'.$query);
include 'closedb.php';
```

```
?>
```

Por exemplo, pode disponibilizar os dados num ficheiro *xls* criando e executando o ficheiro *convert.php* com os seguintes comandos:

PHP

```
<?
include 'config.php';
include 'opendb.php';
$query = "SELECT fname, lname FROM students";
$result = mysql_query($query) or die('Error, query failed'.$query);
$tsv = array();
$html = array();
while($row = mysql_fetch_array($result, MYSQL_NUM)) {
    $tsv[] = implode("\t", $row);
    $html[] = "<tr><td>" .implode("</td><td>", $row) . "</td></tr>";
}

if ($_GET['format']=='tsv') {
  $tsv = implode("\r\n", $tsv);
  $fileName = 'mysql-to-excel.xls';
  header("Content-type: application/vnd.ms-excel");
  header("Content-Disposition: attachment; filename=$fileName");
  echo $tsv;
} else {
  $html = "<table>" . implode("\r\n", $html) . "</table>";
  echo $html;
}
include 'closedb.php';
?>
```

4.2 Manuais

Este capítulo exemplificou como se pode aceder a um servidor *MySQL* através de *PHP*. Existem muitas mais funcionalidades que aqui não foram abordadas e que devem ser consultados em:

- `http://pt2.php.net/mysql`

- `http://www.w3schools.com/php/php_mysql_intro.asp`

5

Oracle

5.1 Acesso ao Servidor Oracle

Tal como o *MySQL* o servidor *Oracle* é um programa que gere várias bases de dados e recebe comandos SQL e de sistema e devolve os resultados respectivos. No caso da infra-estrutura do DI o servidor de *Oracle* está instalado na máquina *luna.di.fc.ul.pt*.

Uma das possíveis formas de aceder ao servidor *Oracle* é através da aplicação *SQL PLUS**[1]. Esta aplicação permite acesso ao servidor *Oracle* usando um terminal. No entanto, a forma mais fácil de aceder ao servidor é através do *IDE SQLDeveloper*[2], que é usado como referência neste manual.

Para aceder ao servidor *Oracle* usando o *SQLDeveloper* basta criar uma nova ligação (*File* → *New* → *Database Connection*) e definir:

- um nome (*connection name*);

- o *username*: sibdXX, onde XX corresponde ao número do respectivo grupo ;

[1] `http://download.oracle.com/docs/cd/B19306_01/server.102/b14357/toc.htm`

[2] `http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/`

- a *password*: sibdXX, onde XX corresponde ao número do respectivo grupo;

- o *hostname*: **luna.di.fc.ul.pt**;

- o *port*: **1521**;

- o *service identifier* (*SID*): **difcul**.

Para guardar a ligação criada clique em *Save* e para testar a ligação clique em *Test*. Se o resultado do teste for insucesso verifique o código do erro devolvido e a sua descrição. Note que para aceder ao servidor *Oracle* do DI fora do campus da Faculdade de Ciências terá de usar uma *VPN*.

5.2 Comandos de Sistema

Após estabelecer ligação com o servidor *Oracle*, o *SQLDeveloper* disponibiliza *Worksheets tabs* onde podem ser executados comandos como se de um terminal se tratasse. Utilize um *Worksheet* para mudar a senha no servidor *Oracle* através do comando:

worksheet

```
PASSWORD;
```

Para executar este comando seleccione o comando anterior (ou apenas a linha) no *Worksheet* e clique no botão *Run Statement* ou *Run Script*. A distinção destes dois comandos reside na forma como são apresentados os resultados dos comandos executados.

Os comandos *SET AUTOCOMMIT* [3] e *COMMIT*[4] são utilizados quando se pretende especificar o *commit* às alterações feitas na base de dados.

Existe uma grande diversidade de comandos que podem ser executados utilizando a interface gráfica. Explore a lista de componentes associada à sua conta *Oracle* e as suas respectivas opções que se encontram à esquerda do *Worksheet*.

5.3 Comandos SQL

O armazenamento e recolha de dados de uma base de dados gerida pelo servidor *Oracle* é feita através de comandos SQL. Estes comandos são executados directamente a partir do cliente de *Oracle* ou através de uma linguagem de programação como o *Java* (ver capítulo 6).

Usando o *SQLDeveloper* a execução dos comandos SQL pode ser feita directamente num *worksheet* ou através da importação de um ficheiro *SQL*.

A título de exemplo, crie um ficheiro *teste.sql* utilizando o seu editor de texto ou através do *SQLDeveloper* (*File* → *New* → *SQL File*) com os seguintes comandos SQL:

SQL

```
DROP TABLE clientes;

CREATE TABLE clientes (
Cid int,
Apelido varchar(50) NOT NULL,
Nome varchar(50) NOT NULL,
```

[3] http://download.oracle.com/docs/cd/B19306_01/server.102/b14357/ch12040.htm#i2698639

[4] http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/commit_statement.htm

```
Morada varchar(255),
Cidade varchar(50) NOT NULL,
PRIMARY KEY (Cid),
CONSTRAINT chk_cid CHECK (Cid>0 AND (Cidade='Lisboa' OR Cidade='
    Porto'))
);

INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade) VALUES (101, 'Silva'
    , 'Paula', null, 'Lisboa');
INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade) VALUES (102, '
    Ribeiro', 'Paulo', 'Rua X', 'Lisboa');
INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade) VALUES (103, '
    Santos', 'Joao', 'Rua Y', 'Porto');
```

O comando *DROP*[5] apaga a tabela *clientes* caso ela exista na base de dados. Se a tabela não existir na base de dados, o comando retorna uma mensagem de erro.

O comando *CREATE TABLE*[6] cria a tabela *clientes* com cinco atributos/colunas *Cid*, *Apelido*, *Nome*, *Morada* e *Cidade*. O atributo *Cid* corresponde a um inteiro e os restantes atributos a strings com uma dimensão máxima de 255 caracteres. O atributo *Cid* corresponde à chave primária da tabela.

Para além da restrição[7] associada à chave primária, foram também definidas restrições NOT NULL para os atributos *Apelido*, *Nome* e *Cidade* e uma restrição CHECK que garante que o atributo *Cid* só poderá tomar valores positivos e que o atributo *Cidade* só poderá tomar os valores "Lisboa"e

[5] http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_9003.htm

[6] http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_7002.htm

[7] http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/clauses002.htm

"Porto".

Finalmente, os comandos *INSERT*[8] permitem a inserção de três tuplos na tabela *clientes*.

Ao executar estes comandos SQL clicando no botão *Run Script*, o resultado obtido é o seguinte:

script output

```
DROP TABLE Clientes succeeded.
CREATE TABLE succeeded.
1 rows inserted
1 rows inserted
1 rows inserted
```

Experimente agora executar o seguinte comando *INSERT*:

SQL

```
INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade) VALUES (-1, '
    Ribeiro', 'Paulo', 'Rua XPTO', 'Lisboa');
```

Neste caso, o tuplo a inserir viola a restrição *CHECK* da tabela clientes porque o valor do atributo Cid é negativo. Portanto, ao executar este comando obtém-se o seguinte erro:

script output

```
Error starting at line 17 in command:
INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade) VALUES (-1, 'Pereira',
    'Catarina', 'Rua XPTO', 'Lisboa')
Error report:
SQL Error: ORA-02290: restricao de verificacao (SIBD000.CHK_CID) violada
02290. 00000 - "check constraint (%s.%s) violated"
*Cause: The values being inserted do not satisfy the named check
```

[8] `http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_9014.htm`

∗Action: **do** not insert values that violate the constraint.

O conjunto dos nomes das tabelas criadas pode ser obtido através do seguinte comando SQL:

SQL

```
SELECT table_name FROM USER_TABLES;
```

A tabela *USER_TABLES*[9] contém a descrição de todas as tabelas criadas pelo utilizador. Se a tabela clientes for a única tabela criada o resultado da execução do comando anterior é o seguinte:

script output

```
TABLE_NAME
------------------------------
CLIENTES


1 rows selected
```

O comando de sistema *DESC*[10] descreve a estrutura de uma tabela. Por exemplo, a estrutura da tabela clientes pode ser obtida executando o seguinte comando:

worksheet

```
DESC clientes;
```

Para visualizar o conteúdo da tabela clientes basta executar o comando SQL correspondente:

[9] http://download.oracle.com/docs/cd/B12037_01/server.101/b10755/statviews_2666.htm

[10] http://download.oracle.com/docs/cd/B19306_01/server.102/b14357/ch12019.htm

SQL

```
SELECT * FROM clientes;
```

O comando SQL *SELECT*[11] permite fazer uma consulta aos dados da base de dados. No exemplo corrente, o resultado da consulta anterior é o seguinte:

script output

```
CID    APELIDO    NOME    MORADA    CIDADE
------------------------------------------
101    Silva      Paula             Lisboa
102    Ribeiro    Paulo   Rua X     Lisboa
103    Santos     Joao    Rua Y     Porto
```

Os comandos seguintes procedem à alteração do esquema da tabela clientes e à actualização dos tuplos na mesma tabela.

SQL

```
ALTER TABLE clientes ADD pais varchar(50);

UPDATE clientes SET pais='Portugal' WHERE pais IS NULL;

SELECT * FROM clientes;
```

O comando SQL *ALTER TABLE*[12] permite alterar o esquema de uma tabela. Por exemplo, adicionar ou apagar uma coluna. No exemplo corrente, o comando *ALTER TABLE* acrescenta a coluna país à tabela clientes.

O comando SQL *UPDATE*[13] permite actualizar valores dos tuplos de uma tabela. No presente exemplo, o comando *UPDATE* actualiza o valor

[11] http://dev.mysql.com/doc/refman/5.1/en/select.html

[12] http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/statements_3001.htm

[13] http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_10007.htm

do atributo país dos tuplos da tabela clientes, que têm a *null* o atributo país, para Portugal. O resultado da execução dos comandos anteriores é o seguinte:

script output

ALTER TABLE clientes succeeded.

2 rows updated

CID	APELIDO	NOME	MORADA	CIDADE	PAIS
101	Silva	Paula		Lisboa	Portugal
102	Ribeiro	Paulo	Rua X	Lisboa	Portugal
103	Santos	Joao	Rua Y	Porto	Portugal

5.4 Manuais de SQL

Para mais informações sobre o *Oracle* e SQL deve consultar os seguintes manuais online:

- `http://www.oracle.com/pls/db102/homepage`

- `http://www.w3schools.com/sql/`

6

Java e Oracle

O servidor *Oracle* pode ser acedido através de uma aplicação *Java*. Esta ligação é feita via JDBC[1]. Para isso, o programa *Java* a desenvolver deverá incluir o driver JDBC correspondente à versão da base de dados *Oracle*. A versão actual da base de dados *Oracle* do DI é a 11.2.0.1.0. O driver poderá ser obtido através do site[2] da *Oracle*. A escolha da versão do driver JDBC deve ter em conta a versão do JDK usada.

6.1 Exemplo

Nesta secção, baseado nos exemplos apresentados na Secção 5.3, é apresentado um simples programa em JAVA que efectua uma ligação a um base de dados *Oracle* e executa comandos SQL e de sistema. Este programa é composto por dois ficheiros: *DatabaseConnection.java* e *Exemplo.java*. No primeiro, é definida a classe e respectivos métodos que permitem efectuar uma ligação à base de dados *Oracle*. No segundo, são executados exemplos de comandos SQL e de sistema na base da dados.

[1] `http://download.oracle.com/javase/tutorial/jdbc/`

[2] `http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html`

DatabaseConnection.java

```
1  package sibd1112;
2
3  import java.sql.Connection;
4  import java.sql.SQLException;
5  import java.sql.DriverManager;
6
7  /**
8   * Manages a Database Connection
9   */
10 public class DatabaseConnection {
11
12     // Variables needed for connecting to the Database
13     private String url;
14     private String user;
15     private String password;
16     private Connection theConnection;
17
18
19     private static DatabaseConnection dbc;
20
21     private DatabaseConnection() {
22         //Gets the parameters for the connection to the Database
23         this.user = "sibdXX";
24         this.password = "password";
25         this.url = "oracle:thin:@luna.di.fc.ul.pt:1521:difcul";
26     }
27
28     /**
29      * Singleton
30      */
31     public static DatabaseConnection databaseConnection() {
32         if (dbc == null)
33             dbc = new DatabaseConnection();
34         return dbc;
35     }
```

A classe *DatabaseConnection* é uma classe *singleton* onde é definida e criada a ligação à base de dados *Oracle*. No construtor desta classe são definidas as variáveis necessárias para estabelecer a ligação, nomeadamente, o *username* sibdXX (onde XX corresponde ao número do grupo) e a *password* (previamente modificada) da conta *Oracle*, e a *url* do servidor *Oracle*, **oracle:thin:@luna.di.fc.ul.pt:1521:difcul** (linhas 23-25).

A classe *DatabaseConnection* disponibiliza dois métodos para abertura, obtenção e fecho da ligação ao servidor *Oracle*.

DatabaseConnection.java (cont.)

```
37      /**
38       * Opens the jbdc connection
39       */
40      public void openConnection() {
41
42          try {
43
44              //Driver registry
45              DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver
                    ());
46
47              //Gets the jdbc connection
48              this.theConnection = DriverManager.getConnection("jdbc:" + this.
                    url,this.user,this.password);
49
50          } catch (Exception e) {
51              System.out.println("Could not load the driver::problem when
                    opening the connection");
52              e.printStackTrace();
53              System.exit(-1);
54          }
55
56          System.out.println("Connection established.");
57
58      }
59
60      /**
61       * Closes the jbdc connection
62       */
63      public void closeConnection() {
64
65          System.out.println("Closing Connection...");
66          try {
67              this.theConnection.close();
68              System.out.println("Connection closed.");
69          } catch (SQLException e) {
70              System.out.println("DataBase.closeConnection :" + e.getMessage()
                    );
71              e.printStackTrace();
72              System.exit(-1);
73          }
74      }
75
76      /**
77       * Returns the connection to the database
78       */
79      public Connection getConnection() {
80          if (this.theConnection == null) System.out.println("Error
                getConnection: the connection is null");
81          return this.theConnection;
82      }
83  }
```

O método *openConnection* cria uma ligação. Para isso, em primeiro lugar, é feito o registro do *driver Oracle* no *DriverManager* (linha 45). A ligação é

estabelecida através do método *getConnection* do *DriverManager* que recebe como argumentos os dados da conta e do servidor *Oracle*, nomeadamente, o *username*, *password* e a *url* (linha 48).

Os métodos *closeConnection* e *getConnection()* permitem, respectivamente, fechar e obter a ligação já previamente estabelecida. Note que é necessário tratar eventuais excepções que poderão ocorrer durante a criação e o fecho de uma ligação.

Na classe *Example* são executados alguns comandos SQL e de sistema descritos na Secção 5.3. Os resultados das consultas efectuadas são também tratados visualizados no *output*.

Example.java

```
1  package sibd1112;
2  import java.sql.*;
3
4
5  /**
6   * DB example
7   */
8  public class Example{
9
10     /**
11      * Displays the given result set
12      */
13     private static void displayResultSet(ResultSet rs) throws SQLException{
14
15         ResultSetMetaData rsmd = rs.getMetaData();
16         int numberOfColumns = rsmd.getColumnCount();
17
18
19         // Iterate through the data in the result set and display it.
20         System.out.println("--------------------");
21
22         for(int i = 1; i <= numberOfColumns; i++){
23             System.out.print(rsmd.getColumnLabel(i) + " | ");
24         }
25
26         System.out.println("\n--------------------");
27
28         while(rs.next()) {
29             for(int i = 1; i <= numberOfColumns; i++){
30                 Object a = rs.getObject(i);
31                 if(a==null){
32                     System.out.print("<null>" + " | ");
33                 }else{
34                     System.out.print(a.toString() + " | ");
35                 }
```

```
36              }
37              System.out.println();
38          }
39
40          System.out.println("-------------------\n");
41      }
```

O método *displayResultSet* formata os resultados retornados por uma query, uma instância de *ResultSet*, de forma a serem visualizados pelo utilizador. Para isso são utilizados alguns métodos disponíbilizados pela classe *ResultSet*, nomeadamente, *getMetaData()*, *getColumnCount()*, *getColumnLabel()* e *getObject(i)*.

Example.java (cont.)

```
43      public static void main(String[] args) {
44
45          // Declare the JDBC objects.
46          Statement stmt = null;
47          ResultSet rs = null;
48          Connection con = null;
49
50          int upd = 0;
51          String nr_upd = null;
52
53
54          try {
55              // Established and get a connection
56              DatabaseConnection dbcon = DatabaseConnection.databaseConnection
                    ();
57              dbcon.openConnection();
58              con = dbcon.getConnection();
59
60              // Sets auto commit false
61              con.setAutoCommit(false);
62
63              // Creates an SQL statement
64              stmt = con.createStatement();
```

No método *main*, antes de executar as várias operações na base de dados *Oracle*, é estabelecida a respectiva ligação ao servidor *Oracle* utilizando os métodos da classe *DatabaseConnection* (linhas 56-58). A seguir, a opção *auto commit* é desactivada (linha 61). Desta forma, as alterações efectuadas na base de dados só são tornadas permanentes quando é feito *commit* (linha 122).

Através da criação de uma *Statement*[3] (linha 64) torna-se possível a execução (via JDBC) de comandos SQL. Em alternativa à criação de uma *Statement*, a criação de uma *PreparedStatement*[4] permite pré-compilar comandos SQL com ou sem parâmetros. Esta alternativa torna-se mais eficiente quando o respectivo comando SQL é executado múltiplas vezes (por exemplo, na inserção de tuplos numa tabela).

Example.java (cont.)

```
66              // Executes an SQL statement (CREATE TABLE) that returns nothing
                    (0).
67              String SQL = "CREATE TABLE clientes (Cid INT, Apelido varchar
                    (50) NOT NULL, Nome varchar(50) NOT NULL, Morada varchar(50)
                    , Cidade varchar(50) NOT NULL, Primary Key(Cid), CONSTRAINT
                    chk_cid CHECK (Cid>0 AND (Cidade='Lisboa' OR Cidade='Porto')
                    ))";
68              stmt.executeUpdate(SQL);
69              System.out.println("\nTable created.\n");
70
71
72              // Execute an SQL statement (SELECT) that returns some data.
73              SQL = "SELECT * FROM clientes";
74              rs = stmt.executeQuery(SQL);
75              System.out.println(SQL);
76              // Displays the result set returned.
77              displayResultSet(rs);
```

A criação da tabela clientes é feita através do método *executeUpdate* da *Statement* criada anteriormente, cujo argumento corresponde a uma string com o respectivo comando SQL *CREATE TABLE* (linhas 67-68) (ver Secção 5.3). Note-se que o método *executeUpdate* devolve sempre zero para comandos SQL DDL. Se ocorrer algum erro na criação da tabela, uma excepção SQL é lançada.

A seguir, nas linhas 73-74, é executada uma consulta *SELECT* através do método *executeQuery* da *Statement* criada anteriormente. Este método retorna um *ResultSet* que corresponde ao resultado da consulta. Este re-

[3] http://download.oracle.com/javase/1.4.2/docs/api/java/sql/Statement.html

[4] http://download.oracle.com/javase/1.4.2/docs/api/java/sql/PreparedStatement.html

sultado é visualizado através do método *displayResultSet* (linha 77). Neste caso, o resultado corresponde a um conjunto vazio de tuplos porque ainda não foram inseridos tuplos na tabela clientes.

Example.java (cont.)

```
80              // Executes an SQL statement (INSERT) that returns the number of
                    inserted rows.
81              SQL = "INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade)
                    VALUES (101, 'Silva', 'Paula', null, 'Lisboa')";
82              upd = stmt.executeUpdate(SQL);
83              SQL = "INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade)
                    VALUES (102, 'Ribeiro', 'Paulo', 'Rua X', 'Lisboa')";
84              upd += stmt.executeUpdate(SQL);
85              SQL = "INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade)
                    VALUES (103, 'Santos', 'Joao', 'Rua Y', 'Porto')";
86              upd += stmt.executeUpdate(SQL);
87              SQL = "INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade)
                    VALUES (104, 'Pereira', 'Tiago', 'Rua A', 'Porto')";
88              upd += stmt.executeUpdate(SQL);
89              SQL = "INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade)
                    VALUES (105, 'Garcia', 'Ana', 'Rua B', 'Porto')";
90              upd += stmt.executeUpdate(SQL);
91              SQL = "INSERT INTO clientes (Cid, Apelido, Nome, Morada, Cidade)
                    VALUES (106, 'Sousa', 'Pedro', 'Rua C', 'Porto')";
92              upd += stmt.executeUpdate(SQL);
93              nr_upd = upd + " tuples inserted.\n";
94              System.out.println(nr_upd);
95
96
97              SQL = "SELECT * FROM clientes";
98              rs = stmt.executeQuery(SQL);
99              System.out.println(SQL);
100             displayResultSet(rs);
```

A inserção de tuplos é também feita através do método *executeUpdate* (linhas 81-92). No entanto, tratando-se de um comando SQL DML, o *executeUpdate* retorna o número de tuplos envolvidos, ou seja, neste caso, o número de tuplos inseridos. Após a inserção dos tuplos, a consulta feita anteriormente (linhas 97-98) retorna os tuplos inseridos.

Example.java (cont.)

```
102             // Executes an SQL statement (ALTER) that that returns nothing
                    (0).
103             SQL = "ALTER TABLE clientes ADD pais varchar(50)";
104             stmt.executeUpdate(SQL);
105             System.out.println("Table altered.\n");
106
```

```
107
108             // Executes an SQL statement (UPDATE) that that returns the
                    number of updated rows.
109             SQL = "UPDATE clientes SET pais='Portugal' WHERE pais IS NULL";
110             upd= stmt.executeUpdate(SQL);
111             nr_upd = upd + " tuples updated.\n";
112             System.out.println(nr_upd);
113
114
115             SQL = "SELECT * FROM clientes";
116             rs = stmt.executeQuery(SQL);
117             System.out.println(SQL);
118             displayResultSet(rs);
119
120
121             // Applies commit.
122             con.commit();
123
124             //Closes connection
125             dbcon.closeConnection();
126
127         }
128         catch (Exception e) {
129             e.printStackTrace();
130         }
131         finally {
132             if (rs != null) try { rs.close(); } catch(Exception e) {}
133             if (stmt != null) try { stmt.close(); } catch(Exception e) {}
134             if (con != null) try { con.close(); } catch(Exception e) {}
135         }
136     }
137 }
```

Os comandos *ALTER TABLE* e *UPDATE* descritas na Secção 5.3 são executadas através do método *executeUpdate* (linhas 103-104 e 109-110, respectivamente).

Por fim é feito o commit das operações anteriores (linha 122) e é fechada a ligação criada (linha 125). As excepções, são tratadas nas linhas 128-134.

A seguir, é apresentado o output da execução do programa anterior.

java output

```
Connection established.


Table created.


SELECT * FROM clientes
--------------------
```

```
CID | APELIDO | NOME | MORADA | CIDADE |
--------------------
--------------------

6 tuples inserted.

SELECT * FROM clientes
--------------------
CID | APELIDO | NOME | MORADA | CIDADE |
--------------------
101 | Silva | Paula | <null> | Lisboa |
102 | Ribeiro | Paulo | Rua X | Lisboa |
103 | Santos | Joao | Rua Y | Porto |
104 | Pereira | Tiago | Rua A | Porto |
105 | Garcia | Ana | Rua B | Porto |
106 | Sousa | Pedro | Rua C | Porto |
--------------------

Table altered.

6 tuples updated.

SELECT * FROM clientes
--------------------
CID | APELIDO | NOME | MORADA | CIDADE | PAIS |
--------------------
101 | Silva | Paula | <null> | Lisboa | Portugal |
102 | Ribeiro | Paulo | Rua X | Lisboa | Portugal |
103 | Santos | Joao | Rua Y | Porto | Portugal |
104 | Pereira | Tiago | Rua A | Porto | Portugal |
105 | Garcia | Ana | Rua B | Porto | Portugal |
106 | Sousa | Pedro | Rua C | Porto | Portugal |
--------------------
```

```
Closing Connection...
Connection closed.
```

6.2 Manuais

Neste capítulo apresentou-se um exemplo de ligação a um servidor *Oracle*, e de execução de comandos SQL e de sistema através de Java. Só uma pequena parte das funcionalidades disponíveis é que foi abordada. Estas e as restantes funcionalidades disponíveis podem ser consultadas em:

- `http://download.oracle.com/javase/tutorial/jdbc/`

- `http://www.jdbc-tutorial.com/`