

UNIVERSIDADE ESTADUAL DE MARINGÁ  
CENTRO DE TECNOLOGIA  
DEPARTAMENTO DE INFORMÁTICA  
CURSO DE ESPECIALIZAÇÃO  
DESENVOLVIMENTO DE SISTEMAS PARA WEB

Desenvolvimento de Aplicações de Banco de Dados para a Web: PHP e MySQL  
versus PHP e Oracle

ANA CAROLINA DA SILVA PASSOS

Maringá – PR  
Agosto / 2005

ANA CAROLINA DA SILVA PASSOS

Desenvolvimento de Aplicações de Banco de Dados para a Web: PHP e MySQL  
versus PHP e Oracle

Monografia apresentada como requisito parcial à obtenção do grau de Especialista pelo curso de Pós-Graduação em Desenvolvimento de Sistemas para Web, da Universidade Estadual de Maringá.

Orientador: Prof. Dr. Ricardo Rodrigues Ciferri

Co-Orientadora: Prof<sup>ª</sup>. Dra. Cristina Dutra de Aguiar Ciferri

Maringá – PR  
Agosto / 2005

## **TERMO DE APROVAÇÃO**

ANA CAROLINA DA SILVA PASSOS

### **Desenvolvimento de Aplicações de Banco de Dados para a Web: PHP e MySQL versus PHP e Oracle**

Monografia aprovada como requisito parcial para obtenção do grau de Especialista pelo curso de Pós-Graduação em Desenvolvimento de Sistemas para Web, Departamento de Informática, Universidade Estadual de Maringá, pela seguinte banca examinadora:

Elisa Hatsue Moriya Huzita  
Departamento de Informática, UEM.

Tania Fatima Calvi Tait  
Departamento de Informática, UEM.

Flávio Arnaldo Braga da Silva  
Departamento de Informática, UEM.

Maringá – PR  
Agosto / 2005

## AGRADECIMENTOS

Primeiramente agradeço a DEUS pelo dom da vida, da sabedoria e do amor. Por me permitir concluir esse trabalho e pelas pessoas maravilhosas que colocou no meu caminho.

Aos meus queridos pais Emílio e Rosangela, pelo amor e incentivo, sem os quais eu nunca teria chegado até aqui. Pela minha irmã Yanara, pelo carinho e incentivo.

Ao meu namorado Vinícius, pelo amor, compreensão, dedicação e incentivo durante toda a elaboração do trabalho e do curso.

Obrigada a minha orientadora, professora Cristina Dutra de Aguiar Ciferri e Ricardo Rodrigues Ciferri, pelo apoio, conhecimento passado, paciência e principalmente, pela amizade durante esses meses.

Aos companheiros da especialização, com quem muito aprendi, especialmente aos amigos Nelson, César, Fábio e João Carlos que compartilharam das viagens durante o curso.

A todos aqueles que, apesar de não reconhecidos nominalmente aqui, contribuíram de alguma forma para a realização deste trabalho.

# SUMÁRIO

LISTA DE FIGURAS .....	8
LISTA DE TABELAS .....	9
LISTA DE SIGLAS .....	10
RESUMO .....	11
ABSTRACT .....	12
CAPÍTULO 1 .....	13
Introdução.....	13
1.1 Objetivos .....	13
1.2 Metodologia .....	14
1.3 Estruturação da Monografia .....	15
CAPÍTULO 2 .....	16
A Linguagem SQL e os SGBD Oracle e MySQL.....	16
2.1 Aplicação Base.....	16
2.2 Comandos DDL.....	21
2.2.1 CREATE TABLE .....	21
2.2.2 ALTER TABLE .....	23
2.2.3 DROP TABLE .....	24
2.2.4 CREATE VIEW .....	25
2.2.5 DROP VIEW .....	25
2.3 Comandos DML (Data Manipulation Language) .....	25
2.3.1 SELECT .....	26
2.3.2 FUNÇÕES AGREGADAS .....	27
2.3.3 OPERAÇÕES DE CONJUNTOS .....	27
2.3.4 JUNÇÃO (COMPOSIÇÃO DE RELAÇÕES) .....	28
2.3.5 SUBCONSULTAS ANINHADAS .....	28
2.3.6 RELACAO DERIVADAS .....	29
2.3.7 UPDATE .....	29
2.3.8 INSERT .....	30
2.3.9 DELETE.....	30
2.4 Comandos DDL e DML oferecidos pelos SGBD MySQL e Oracle.....	31
2.5 Considerações Finais.....	35
CAPÍTULO 3 .....	36
Funções de Acesso aos SGBD MySQL e Oracle usando a Linguagem PHP .....	36
3.1 Funções de Acesso ao SGBD MySQL usando a Linguagem PHP .....	36
Função mysql_connect() .....	36
Função mysql_select_db( ).....	37
Função mysql_query( ).....	37
Função mysql_fetch_row( ) .....	38
Função mysql_num_fields( ).....	38
Função mysql_close( ) .....	39
Função mysql_data_seek( ) .....	39
Função mysql_fetch_array( ) .....	39
Função mysql_fetch_object( ) .....	40
Função mysql_free_result( ).....	41
Função mysql_num_rows( ).....	41
Função mysql_pconnect( ) .....	41
Função mysql_change_user( ).....	42

Função mysql_create_db( ) .....	42
Função mysql_drop_db( ) .....	43
Função mysql_fetch_field( ) .....	43
Função mysql_list_tables( ) .....	44
Função mysql_tablename( ) .....	44
Função mysql_errno( ) .....	45
Função mysql_error( ) .....	45
Função mysql_affected_rows( ) .....	45
Função mysql_insert_id( ) .....	46
3.2 Funções de Acesso ao SGBD Oracle usando a Linguagem PHP .....	46
Função OCILogon( ) .....	47
Função OCINLogon( ) .....	47
Função OCIPLogon( ) .....	48
Função OCIParse( ) .....	48
Função OCIExecute( ) .....	49
Função OCIFetchInto( ) .....	49
Função OCIFetch( ) .....	50
Função OCIResult( ) .....	50
Função OCIRowCount( ) .....	51
Função OCINumCols( ) .....	51
Função OCICommit( ) .....	51
Função OCIRollback( ) .....	52
Função OCIError( ) .....	52
Função OCILogoff( ) .....	52
Função OCIColumnSize( ) .....	53
Função OCIColumnName( ) .....	53
Função OCIColumnType( ) .....	53
Função OCICancel( ) .....	54
Função OCIFreeCursor( ) .....	54
Função OCIFreeStatement( ) .....	54
Função OCIStatementType( ) .....	55
3.3 Comparação entre as Funções de Acesso PHP-MySQL e PHP-Oracle .....	55
3.4 Considerações Finais .....	57
CAPÍTULO 4 .....	59
Proposta de Funções de Acesso ao SGBD Oracle usando a Linguagem PHP .....	59
4.1 Função OCI_Change_User( ) .....	59
4.2 Função OCI_List_Table( ) .....	60
4.3 Função OCI_Data_Seek( ) .....	61
4.4 Função OCI_Insert_Id( ) .....	61
4.5 Função OCI_Create_Db( ) .....	62
4.6 Função OCI_Drop_Db( ) .....	63
4.7 Comparação entre as Funções de Acesso PHP-MySQL e PHP-Oracle Propostas .....	64
4.8 Considerações Finais .....	64
CAPÍTULO 5 .....	66
Aplicação de Banco de Dados utilizando o SGBD Oracle e a Linguagem de Programação PHP .....	66
5.1 Aplicação Base .....	66
5.2 Telas Principais da Aplicação .....	66
5.3 Script para Consulta Simples .....	69

5.4 Script para Consulta Parametrizada .....	71
5.5 Considerações Finais.....	76
CAPÍTULO 6 .....	77
Conclusão.....	77
REFERÊNCIAS BIBLIOGRÁFICAS.....	79

## LISTA DE FIGURAS

Figura 2.1 - Projeto Conceitual .....	17
Figura 5.1 Tela inicial da aplicação. ....	67
Figura 5.2 Tela intermediária da aplicação. ....	68
Figura 5.3 Opções de consulta disponíveis. ....	68
Figura 5.4 Tela de apresentação dos testes PHP-Oracle. ....	69
Figura 5.5 Tela para consulta parametrizada às tabelas <i>wine</i> e <i>winery</i> . ....	75
Figura 5.6 Tela com os resultados obtidos na consulta parametrizada. ....	75

## LISTA DE TABELAS

Tabela 2.1 - Tipos-Entidade e seus atributos. ....	19
Tabela 2.2 - Comandos SQL DDL oferecidos pelos SGBD MySQL e SGBD Oracle.....	31
Tabela 2.3 - Comandos SQL DML oferecidos pelos SGBD MySQL e SGBD Oracle. ....	32
Tabela 2.4 - Tipos de dados oferecidos pelo SGBD MySQL. ....	33
Tabela 2.5 - Tipos de dados disponíveis pelo SGBD Oracle. ....	35
Tabela 3.1 - Propriedades do objeto retornado da função mysql_fetch_field( ).....	43
Tabela 3.2 Equivalência entre as funções de acesso PHP-MySQL e PHP-Oracle. ....	57
Tabela 4.1 Equivalência entre as funções de acesso PHP-MySQL e PHP-Oracle propostas.....	64

## LISTA DE SIGLAS

<b>SGBD</b>	SISTEMA GERENCIADOR DE BANCO DE DADOS
<b>WEB ou WWW</b>	WORLD WIDE WEB
<b>PHP</b>	PERSONAL HOME PAGE
<b>SQL</b>	STRUCTURED QUERY LANGUAGE
<b>DDL</b>	DATA DEFINITION LANGUAGE
<b>DML</b>	DATA MANIPULATION LANGUAGE

## RESUMO

PASSOS, Ana Carolina da Silva. **Desenvolvimento de Aplicações de Banco de Dados para a Web: PHP e MySQL versus PHP e Oracle**. 2005. 81p. Monografia. (Especialização em Desenvolvimento de Sistemas para WEB). Programa de Pós-Graduação do Departamento de Informática, FUEM, Maringá/PR.

O desenvolvimento de aplicações de banco de dados para a Web tem sido fortemente baseado no uso do sistema gerenciador de banco de dados (SGBD) MySQL e da linguagem de programação PHP. Isto se deve principalmente ao fato de que a linguagem PHP oferece várias funções de fácil utilização que permitem o acesso, a pesquisa e a manipulação dos dados armazenados no SGBD MySQL. Entretanto, o SGBD MySQL possui algumas limitações quando comparado às funcionalidades oferecidas por outros SGBD, como por exemplo o Oracle. O SGBD Oracle é um dos SGBD relacionais mais robustos existentes atualmente no mercado (Navathe, 2003; Ramakrishnan & Gehrke, 2003). Assim, surge a necessidade de se investigar o uso do SGBD Oracle no desenvolvimento de aplicações de banco de dados para a Web. Este trabalho tem como objetivo comparar as funcionalidades da linguagem SQL oferecidas pelos SGBD Oracle e MySQL, além de investigar como o acesso ao SGBD Oracle pode ser realizado por meio da linguagem PHP. Como resultado do trabalho desenvolvido, esta monografia de especialização apresenta tabelas comparativas entre os SGBD sob estudo, propõe novas funções de acesso ao SGBD Oracle usando PHP e descreve uma aplicação de banco de dados para a Web desenvolvida usando a linguagem de programação PHP e o SGBD Oracle.

**Palavras-Chave:** SGBD, WEB, WWW, PHP, SQL, DDL, DML, aplicações, banco de dados, funcionalidades, funções.

## ABSTRACT

PASSOS, Ana Carolina da Silva. **Development of Database Applications for the Web: PHP and MySQL versus PHP and Oracle**. 2005. 81p. Monograph. (Specialization in Web Systems Development). Graduation Program of the Information Science Department, FUEM, Maringá/PR.

The development of web database applications has been widely based on the use of the database management system (DBMS) MySQL and on the PHP programming language, mainly because the PHP language provides many easy-to-use functions, which allow accessing, searching and handling the data stored in the DBMS MySQL. However, the DBMS MySQL is limited when compared to the functionality offered by other DBMSs such as Oracle. The Oracle DBMS is one of the most robust relational DBMSs in the present market (Navathe, 2003; Ramakrishnan & Gehrke, 2003). Thus, it is necessary to investigate the use of the Oracle DBMS in the development of web database applications. This paper aims at comparing the functionalities of the SQL language offered by the Oracle and the MySQL DBMSs, besides investigating how the Oracle DBMS can be accessed through the PHP language. As a result of the work developed this specialization monograph presents comparative tables between the DBMSs studied, proposes new functions to access the Oracle DBMS by using PHP and describes a web database application developed with the use of the programming language PHP and the Oracle DBMS.

**Key words:** DBMS, WEB, WWW, PHP, SQL, DDL, DML, applications, databases, functionalities, functions.

# CAPÍTULO 1

## Introdução

Atualmente, o desenvolvimento de aplicações de banco de dados para a Web tem sido fortemente baseado no uso do sistema gerenciador de banco de dados (SGBD) MySQL e da linguagem de programação PHP. Isto se deve principalmente ao fato de que a linguagem PHP oferece várias funções de fácil utilização que permitem o acesso, a pesquisa e a manipulação dos dados armazenados no SGBD MySQL.

Adicionalmente, outras características da linguagem PHP e do SGBD MySQL também têm contribuído para este cenário. PHP é uma linguagem de *script* eficiente e poderosa, desenvolvida para a criação de *sites* Web dinâmicos. Além de prover grande portabilidade, PHP é totalmente gratuita e oferece mais de 50 bibliotecas que podem ser facilmente utilizadas no desenvolvimento de sistemas complexos (Converse & Park, 2001; Lolis, 2003; Bedin, 2004).

O SGBD relacional MySQL, assim como PHP, também é um produto livre, de código fonte aberto, e que possui grande portabilidade (Miloca, 2004). Este SGBD é voltado ao desenvolvimento de aplicações de banco de dados de porte médio, e oferece a maioria das funcionalidades de um SGBD de grande porte.

Entretanto, o SGBD MySQL possui algumas limitações quando comparado às funcionalidades oferecidas por outros SGBD, como por exemplo o Oracle. O SGBD Oracle é um dos SGBD relacionais mais robustos existentes atualmente no mercado (Navathe, 2003; Ramakrishnan & Gehrke, 2003). Dentre as funcionalidades oferecidas por esse SGBD, pode-se citar aquelas voltadas ao gerenciamento e ao armazenamento de grandes volumes de dados, ao gerenciamento de transações e à recuperação de falhas, à distribuição, à definição e manutenção de restrições de integridade e ao uso de gatilhos. Além disso, existem diversas ferramentas Oracle disponíveis.

Assim, surge a necessidade de se investigar o uso do SGBD Oracle no desenvolvimento de aplicações de banco de dados para a Web.

### 1.1 Objetivos

Esta monografia de especialização tem como objetivo investigar quais funcionalidades adicionais são oferecidas pelo SGBD Oracle com relação ao SGBD MySQL e como o acesso ao SGBD Oracle pode ser realizado por meio da linguagem PHP. Seus objetivos específicos são:

- identificar as características que um SGBD deve oferecer com relação às linguagens de definição e de manipulação dos dados;
- identificar quais dessas características são disponíveis no SGBD MySQL;
- identificar quais dessas características são disponíveis no SGBD Oracle;
- identificar quais as funções necessárias para acesso ao SGBD MySQL usando a linguagem PHP;
- fazer um estudo do estado da arte com a finalidade de identificar quais as funções necessárias para acesso ao SGBD Oracle usando a linguagem PHP;
- desenvolver funções de acesso ao SGBD Oracle usando a linguagem PHP, adicionais às identificadas no item anterior, de forma que estas funções possuam funcionalidades equivalentes às oferecidas para o acesso ao SGBD MySQL; e
- desenvolver uma aplicação de banco de dados para a Web usando a linguagem de programação PHP e o SGBD Oracle.

## **1.2 Metodologia**

Para a realização desse trabalho os métodos de pesquisas utilizados foram a documentação indireta (pesquisa documental e pesquisa bibliográfica) e a pesquisa-ação.

O método de documentação indireta foi utilizado pois toda pesquisa engloba uma etapa de levantamento de dados ou técnicas empregadas. No trabalho desenvolvido, a pesquisa bibliográfica permitiu realizar o levantamento da bibliografia publicada com relação aos conceitos de banco de dados, às características da linguagem SQL, e às funcionalidades dos SGBD MySQL e Oracle. Essa pesquisa garantiu um contato direto do pesquisador com a maioria daquilo que foi escrito sobre o assunto. Adicionalmente, também foram necessárias a leitura de artigos, teses e dissertações, e a pesquisa em manuais.

Já o principal motivo para a escolha da pesquisa-ação é que este tipo de pesquisa é um método com grande capacidade de resolver problemas específicos dentro de um grupo ou organização, além de ser um método adaptável que auxilia pesquisadores e usuários a lidar com inserção de conhecimento na prática. Assim esta monografia de especialização implementa uma aplicação de banco de dados para a Web usando a linguagem de programação PHP e o SGBD Oracle, incluindo nesta aplicação os testes desenvolvidos durante o desenvolvimento da monografia.

### 1.3 Estruturação da Monografia

Além deste capítulo introdutório, esta monografia está estruturada em mais 5 capítulos.

O Capítulo 2 investiga as funcionalidades da linguagem SQL voltadas à definição de dados e à manipulação dos dados. Este capítulo também destaca quais destas funcionalidades podem ser encontradas no SGBD Oracle e no SGBD MySQL.

Funções de acesso aos SGBD MySQL e Oracle usando a linguagem de programação PHP são descritas no Capítulo 3. Cada uma das funções é discutida em termos de sua descrição, de sua sintaxe e de um exemplo de utilização. Este capítulo também apresenta uma tabela que compara as funções de acesso ao SGBD MySQL com as funções de acesso ao SGBD Oracle.

Já o Capítulo 4 tem como objetivo apresentar funções de acesso ao SGBD Oracle usando a linguagem de implementação PHP propostas durante o decorrer deste projeto de especialização. Este capítulo também destaca uma tabela que compara as funções de acesso ao SGBD Oracle propostas com funções de acesso ao SGBD MySQL já existentes.

Uma aplicação de banco de dados para a Web usando a linguagem de programação PHP e o SGBD Oracle é descrita no Capítulo 5. Neste capítulo são destacadas a interface da aplicação, além dos *scripts* desenvolvidos voltados para a realização de consultas simples e de consultas parametrizadas.

O capítulo 6 apresenta as conclusões e as contribuições do trabalho.

## CAPÍTULO 2

### A Linguagem SQL e os SGBD Oracle e MySQL

A linguagem SQL (*Structured Query Language*) é a linguagem padrão para se lidar com banco de dados relacionais, e é aceita por quase todos os produtos relacionais existentes no mercado (Connolly & Begg, 2002; Date, 2004). SQL oferece um conjunto de comandos voltados para a definição da estrutura dos dados e para a consulta e a modificação (i.e., inserção, remoção e atualização) de dados no banco de dados. Além disto, tem facilidades para definir visões, para especificar restrições de segurança e de permissão, para definir restrições de integridade, e para especificar aspectos relacionados ao gerenciamento de transações e à recuperação de falhas (Silberschatz et al., 1999; Navathe, 2003; Ramakrishnan & Gehrke, 2003).

Apesar de todas estas funcionalidades, este capítulo enfoca a SQL sob o ponto de vista de linguagem de definição de dados (DDL – *data definition language*) e de linguagem de manipulação de dados (DML – *data manipulation language*). Comandos relacionados à DDL referem-se àqueles voltados à criação e à remoção de tabelas e de visões, além dos comandos de alteração da estrutura das tabelas. Já os comandos DML dizem respeito aos comandos para consulta, inserção, remoção e atualização dos dados no banco de dados.

O objetivo deste capítulo é investigar as funcionalidades DDL e DML oferecidas pela linguagem SQL, e destacar quais destas funcionalidades podem ser encontradas no SGBD Oracle (Fernandes, 2002; Campos, 1999, Fanderuff, 2000) e no SGBD MySQL. Dentro deste contexto, este capítulo está estruturado como segue. A seção 2.1 descreve a aplicação base que será utilizada ao longo desta monografia. As seções 2.2 e 2.3 listam, respectivamente, os comandos DDL e DML oferecidos pela linguagem SQL padrão. Já a seção 2.4 investiga os comandos DDL e DML oferecidos pelos SGBD Oracle e MySQL, e apresentada uma tabela comparativa entre esses SGBD. O capítulo é finalizado na seção 2.5, com as considerações finais.

#### 2.1 Aplicação Base

Nessa seção é descrita a aplicação a ser utilizada para os testes dos comandos SQL. A aplicação é de uma vinícola, e foi adaptada de Williams & Lane (2002). A Figura 2.1 mostra o projeto conceitual da aplicação, destacando aspectos estruturais do banco de dados. Já a Tabela 2.1 descreve os tipos-entidade e os seus atributos.

# Projeto Conceitual

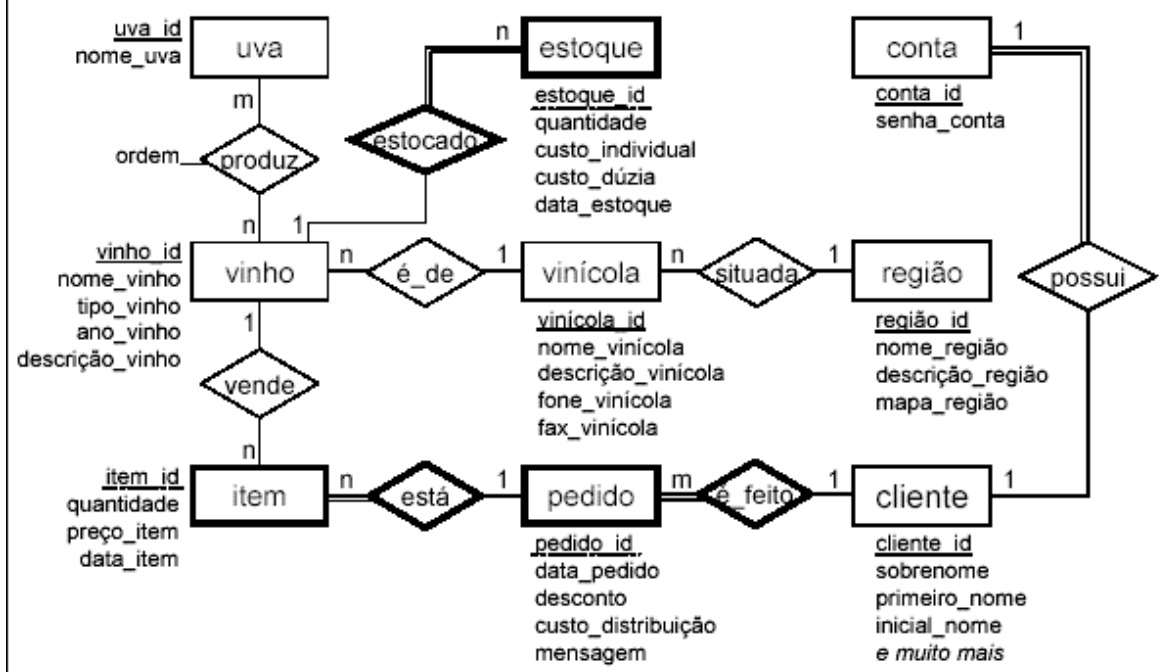


Figura 2.1 - Projeto Conceitual

Português (ER)	Inglês (Aplicação)	Descrição
<b>Uva</b> uva_id nome_uva	<b>Grape_variety</b> variety_id (PK) Variety	chave primária para o tipo-entidade uva nome da uva utilizada na fabricação do vinho; por exemplo: Chardonnay, Semillon, Merlot, Cabernet Sauvignon
<b>Vinho</b> vinho_id nome_vinho tipo_vinho ano_vinho descrição_vinho	<b>Wine</b> wine_id (PK) wine_name Type Year description	chave primária para o tipo-entidade vinho Nome do vinho tipo do vinho; por exemplo: branco, tinto ano da safra de fabricação do vinho revisão do vinho, a qual é similar à encontrada no rótulo do mesmo
<b>Vinicola</b> vinícola_id nome_vinicola descrição_vinicola fone_vinicola fax_vinicola	<b>Winery</b> winery_id (PK) winery_name Description Phone Fax	chave primária para o tipo-entidade vinícola Nome da vinícola descrição da vinícola número do telefone da vinícola número do fax da vinícola
<b>Região</b> região_id nome_região  descrição_região mapa_região	<b>Region</b> region_id (PK) region_name  Description Map	chave primária para o tipo-entidade região nome da região, a qual é uma área na qual a vinícola está localizada descrição da região mapa ou imagem da região

<b>Estoque</b>	<b>Inventory</b>	
estoque_id	inventory_id *PK*)	chave parcial para o tipo-entidade estoque PK = (wine_id, inventory_id) TE Fraca
quantidade	on_hand	quantidade do vinho em estoque
custo_individual	Cost	preço da garrafa de vinho
custo_dúzia	case_cost	preço da caixa de vinho, a qual contém 12 garrafas
data_estoque	Date_added	data na qual a remessa de vinho foi adicionada ao estoque
<b>Pedido</b>	<b>Orders</b>	
pedido_id	order_id (*PK*)	chave parcial para o tipo-entidade pedido PK = (cust_id, order_id) TE Fraca
data_pedido	Date	data na qual o pedido é realizado
desconto	discount	porcentagem de desconto no preço de um pedido; atribuída devido à quantidade comprada, ao dia na qual a compra é realizada ou à regularidade de compra do cliente
custo_distribuição	delivery	taxa cobrada para a entrega dos produtos; depende de onde os produtos serão entregados e de que modo eles serão transportados (i.e., correio convencional, correio aéreo expresso, correio via navio)
mensagem	Note	texto explicativo; por exemplo: “Deixe os vinhos na porta de trás da casa”
<b>Item</b>	<b>Items</b>	
item_id	Item_id (*PK*)	chave parcial para o tipo-entidade item PK = (cust_id, order_id, item_id) TE Fraca
quantidade	Qty	quantidade de garrafas do item a serem compradas
preço_item	price	preço do item por garrafa ou por caixa multiplicado pela quantidade. Caixa ou garrafa dependendo da forma de compra.
data_item	Date	data na qual o item foi adicionado ao carrinho de compras; itens devem ser comprados em até um dia após a sua adição ao carrinho
<b>Cliente</b>	<b>Customer</b>	
cliente_id	Cust_id (PK)	chave primária para o tipo-entidade <i>cliente</i>
sobrenome	surname	sobrenome do cliente
primeiro nome	firstname	primeiro nome do cliente
inicial_nome	initial	letra inicial do nome do meio do cliente
título	title	título do cliente; por exemplo <i>Sir</i>
linha_endereço_1	addressline1	linha de endereço do cliente
linha_endereço_2	addressline2	linha de endereço do cliente
linha_endereço_3	addressline3	linha de endereço do cliente
cidade	city	cidade na qual o cliente mora
estado	state	estado no qual o cliente mora
país	country	país no qual o cliente mora
CEP	zipcode	CEP associado ao endereço do cliente
telefone	phone	número do telefone do cliente
fax	fax	número do fax do cliente
e_mail	email	endereço de correio eletrônico do cliente

data_aniversário	Birth_date	data de aniversário do cliente
salário	salary	salário do cliente
<b>Conta</b>	<b>Users</b>	
conta_id	User_name (PK)	chave primária para o tipo-entidade <i>usuário</i>
senha_conta	password	senha de acesso do usuário

Tabela 2.1 - Tipos-Entidade e seus atributos.

A seguir estão descritos os comandos SQL para a criação do banco de dados *winestore* conforme Williams & Lane (2002). Esses comandos foram estendidos com a inclusão da especificação de cláusulas de chave estrangeira (i.e., cláusula FOREIGN KEY). Visando manter a consistência com a aplicação base, os nomes das tabelas e dos campos são especificados em inglês. A correspondência entre esses nomes em inglês e os nomes utilizados na Figura 2.1 pode ser observada na Tabela 2.1.

```
drop database if exists winestore;
create database winestore;
use winestore;

CREATE TABLE grape_variety (
    variety_id int(3) NOT NULL auto_increment,
    variety varchar(50) DEFAULT '' NOT NULL,
    PRIMARY KEY (variety_id),
    KEY var (variety)
);

CREATE TABLE wine_variety (
    wine_id int(5) DEFAULT '0' NOT NULL,
    variety_id int(3) DEFAULT '0' NOT NULL,
    id int(1) DEFAULT '0' NOT NULL,
    PRIMARY KEY (wine_id,variety_id),
    KEY wine (wine_id,variety_id),
    FOREIGN KEY (wine_id) REFERENCES wine (wine_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (variety_id) REFERENCES grape_variety (variety_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE wine (
    wine_id int(5) NOT NULL auto_increment,
    wine_name varchar(50) DEFAULT '' NOT NULL,
    type varchar(10) DEFAULT '' NOT NULL,
    year int(4) DEFAULT '0' NOT NULL,
    description blob,
    winery_id int(4) DEFAULT '0' NOT NULL,
    PRIMARY KEY (wine_id),
    KEY name (wine_name),
    KEY winery (winery_id),
    FOREIGN KEY (winery_id) REFERENCES winery (winery_id)
);
```

```

        ON UPDATE CASCADE
        ON DELETE CASCADE
    );

CREATE TABLE inventory (
    wine_id int(5) DEFAULT '0' NOT NULL,
    inventory_id int(3) DEFAULT '0' NOT NULL,
    on_hand int(5) DEFAULT '0' NOT NULL,
    cost float(5,2) DEFAULT '0.00' NOT NULL,
    case_cost float(5,2) DEFAULT '0.00' NOT NULL,
    date_added timestamp(12),
    PRIMARY KEY (wine_id,inventory_id),
    FOREIGN KEY (wine_id) REFERENCES wine (wine_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE winery (
    winery_id int(4) NOT NULL auto_increment,
    winery_name varchar(100) DEFAULT '' NOT NULL,
    description blob,
    phone varchar(15),
    fax varchar(15),
    region_id int(4) DEFAULT '0' NOT NULL,
    PRIMARY KEY (winery_id),
    KEY name (winery_name),
    KEY region (region_id),
    FOREIGN KEY (region_id) REFERENCES region (region_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE region (
    region_id int(4) NOT NULL auto_increment,
    region_name varchar(100) DEFAULT '' NOT NULL,
    description blob,
    map mediumblob,
    PRIMARY KEY (region_id),
    KEY region (region_name)
);

CREATE TABLE users (
    user_name int(4) DEFAULT '0' NOT NULL,
    password varchar(15) DEFAULT '' NOT NULL,
    cust_id int(4) DEFAULT '0' NOT NULL,
    PRIMARY KEY (user_name),
    KEY password (password),
    FOREIGN KEY (cust_id) REFERENCES customer (cust_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE customer (
    cust_id int(5) NOT NULL auto_increment,
    surname varchar(50) DEFAULT '' NOT NULL,
    firstname varchar(50) DEFAULT '' NOT NULL,
    initial char(1),
    title varchar(10),
    addressline1 varchar(50) DEFAULT '' NOT NULL,
    addressline2 varchar(50),
    addressline3 varchar(50),

```

```

    city varchar(20) DEFAULT '' NOT NULL,
    state varchar(20),
    zipcode varchar(5),
    country varchar(20),
    phone varchar(15),
    fax varchar(15),
    email varchar(30) DEFAULT '' NOT NULL,
    birth_date date DEFAULT '0000-00-00' NOT NULL,
    salary int(7) DEFAULT '0' NOT NULL,
    PRIMARY KEY (cust_id),
    KEY names (surname,firstname)
);

CREATE TABLE orders (
    cust_id int(5) DEFAULT '0' NOT NULL,
    order_id int(5) DEFAULT '0' NOT NULL,
    date timestamp(12),
    discount float(3,1) DEFAULT '0.0',
    delivery float(4,2) DEFAULT '0.00',
    note varchar(120),
    PRIMARY KEY (cust_id,order_id),
    FOREIGN KEY (cust_id) REFERENCES customer (cust_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE items (
    cust_id int(5) DEFAULT '0' NOT NULL,
    order_id int(5) DEFAULT '0' NOT NULL,
    item_id int(3) DEFAULT '1' NOT NULL,
    wine_id int(4) DEFAULT '0' NOT NULL,
    qty int(3),
    price float(5,2),
    date timestamp(12),
    PRIMARY KEY (cust_id,order_id,item_id),
    FOREIGN KEY (cust_id,order_id) REFERENCES orders (cust_id,order_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (wine_id) REFERENCES wine (wine_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

```

## 2.2 Comandos DDL

Um esquema de dados é especificado por um conjunto de definições expressas por uma linguagem especial chamada linguagem de definição de dados. Esta seção descreve os comandos CREATE TABLE (seção 2.2.1), ALTER TABLE (seção 2.2.2), DROP TABLE (seção 2.2.3), CREATE VIEW (seção 2.2.4) e DROP VIEW (seção 2.2.5).

### 2.2.1 CREATE TABLE

O comando CREATE TABLE tem como objetivo criar a estrutura de uma tabela (a qual é implementada como um arquivo no banco de dados relacional), definido as colunas (i.e., campos) e as chaves primárias e estrangeiras existentes. A sintaxe deste comando é:

```

CREATE TABLE <nome-tabela>
(<nome-coluna> , <tipo-do-dado> [NOT NULL]
                                [NOT NULL WITH DEFAULT] )
PRIMARY KEY (nome-coluna-chave)
FOREIGN KEY (nome-coluna-chave-estrangeira) REFERENCES
            (nome-tabela-pai) ON DELETE [RESTRICT]
                                           [CASCADE]
                                           [SET NULL]

```

onde:

- a) nome-tabela - Representa o nome da tabela que será criada.
- b) nome-coluna - Representa o nome da coluna que será criada. A definição das colunas de uma tabela é feita relacionando-as uma após a outra.
- c) tipo-do-dado - Especifica o tipo e o tamanho dos campos definidos para a tabela. Os tipos de dados mais comuns são definidos na seção 2.4.
- d) NOT NULL - Exige o preenchimento do campo, ou seja, no momento da inclusão é obrigatório que o campo possua um conteúdo.
- e) NOT NULL WITH DEFAULT - Preenche o campo com valores pré-definidos, de acordo com o tipo do campo, caso não seja especificado o seu conteúdo no momento da inclusão do registro. Os valores pré-definidos são:
  - e.1) campos numéricos - valor zero;
  - e.2) campos alfanuméricos - caractere branco;
  - e.3) campo formato *date* - data corrente; e
  - e.4) Campo formato *time* - horário no momento da operação.
- f) PRIMARY KEY (nome-coluna-chave) - Define para o banco de dados a coluna que é a chave primária da tabela. Caso uma tabela tenha mais de um coluna como chave, elas deverão ser relacionadas entre os parênteses.
- g) FOREIGN KEY (nome-coluna-chave-estrangeira) REFERENCES (nome-tabela-pai) - Define para o banco de dados as colunas que são chaves estrangeiras, ou seja, os campos que são chaves primárias de outras tabelas. Na opção REFERENCES deve ser especificada a tabela na qual a coluna é a chave primária.

h) ON DELETE - Especifica os procedimentos que devem ser feitos pelo SGBD em situações nas quais houver uma exclusão de um registro na tabela pai quando existe um registro correspondente nas tabelas filhas. As opções disponíveis são:

h.1) RESTRICT - opção default. Esta opção não permite a exclusão na tabela pai de um registro cuja chave primária exista em alguma tabela filha;

h.2) CASCADE - esta opção realiza a exclusão, em todas as tabelas filhas, de instâncias que possuam o valor da chave que será excluída na tabela pai; e

h.3) SET NULL - Esta opção atribui o valor NULL nas colunas das tabelas filhas que contenham o valor da chave que será excluída na tabela pai.

### 2.2.2 ALTER TABLE

O comando ALTER TABLE tem como objetivo alterar a estrutura de uma tabela (arquivo). É possível acrescentar, alterar e remover os campos, os nomes e os formatos das colunas, além das restrições de integridade referencial definidas em uma determinada tabela. A sintaxe deste comando é descrita a seguir:

```
ALTER TABLE <nome-tabela>
DROP <nome-coluna>
ADD <nome-coluna> <tipo-do-dado> [NOT NULL]
                                     [NOT NULL WITH DEFAULT]
RENAME <nome-coluna> <novo-nome-coluna>
RENAME TABLE <novo-nome-tabela>
MODIFY <nome-coluna> <tipo-do-dado> [NULL]
                                     [NOT NULL]
                                     [NOT NULL WITH DEFAULT]

ADD PRIMARY KEY <nome-coluna>
DROP PRIMARY KEY <nome-coluna>

ADD FOREIGN KEY (nome-coluna-chave-estrangeira) REFERENCES
                 (nome-tabela-pai) ON DELETE [RESTRICT]
                                               [CASCADE]
                                               [SET NULL]
DROP FOREIGN KEY (nome-coluna-chave-estrangeira) REFERENCES
                 (nome-tabela-pai)
```

onde:

a) nome-tabela - Representa o nome da tabela que será atualizada.

b) nome-coluna - Representa o nome da coluna que será criada.

c) tipo-do-dado - Especifica o tipo e o tamanho dos campos definidos para a tabela.

d) DROP <nome-coluna> - Retira a coluna especificada na estrutura da tabela.

e) ADD <nome-coluna> <tipo-do-dado> - Inclui na estrutura da tabela a coluna especificada. Na coluna correspondente a este campo nos registros já existentes será preenchido o valor NULL. As definições NOT NULL e NOT NULL WITH DEFAULT são semelhantes à do comando CREATE TABLE.

f) RENAME <nome-coluna> <novo-nome-coluna> - Altera o nome da coluna especificada.

g) RENAME TABLE <novo-nome-tabela> - Altera o nome da tabela especificada.

h) MODIFY <nome-coluna> <tipo-do-dado> - Altera a característica da coluna especificada. Além das opções existentes na opção ADD (NOT NULL e NOT NULL WITH DEFAULT), existe ainda a opção NULL que altera a característica do campo, passando a permitir o preenchimento com o valor nulo.

i) ADD PRIMARY KEY <nome-coluna> - Esta opção é utilizada quando é acrescentado um novo campo como chave primária da tabela.

j) DROP PRIMARY KEY <nome-coluna> - Esta opção é utilizada quando é retirado um campo como chave primária da tabela.

l) ADD FOREIGN KEY <nome-coluna> - Esta opção é utilizada quando é acrescentado um novo campo que é uma chave estrangeira.

l) DROP FOREIGN KEY <nome-coluna> - Esta opção é utilizada quando é retirada uma chave estrangeira da estrutura da tabela.

### 2.2.3 DROP TABLE

O comando DROP TABLE tem como objetivo remover a estrutura e os dados existentes em uma tabela. Adicionalmente, também estarão removidos os índices de acesso à tabela que estejam a ela associados. A sintaxe deste comando é:

```
DROP TABLE <nome-tabela>
```

onde:

a) nome-tabela - Representa o nome da tabela que será removida.

## 2.2.4 CREATE VIEW

O comando CREATE VIEW tem como objetivo definir uma visão em SQL. Uma visão é uma tabela derivada de outras tabelas ou de outras visões. Diferentemente de uma tabela, uma visão não armazena dados. A definição da visão criada é mantida no banco de dados até que o comando DROP VIEW seja executado para esta visão (Silberschatz et al., 1999).

A sintaxe deste comando é descrita a seguir:

```
CREATE VIEW <nome-visão> as <expressão da consulta>
```

onde:

- a) nome-visão - Representa o nome da visão que será criada.
- b) expressão da consulta - Consiste de qualquer expressão de consulta válida.

## 2.2.5 DROP VIEW

O comando DROP VIEW tem como objetivo remover uma visão. Sua sintaxe é:

```
DROP VIEW <nome-visão>
```

onde:

- a) nome-visão - Representa o nome da visão que será removida.

## 2.3 Comandos DML (Data Manipulation Language)

A DML é a linguagem que viabiliza o acesso ou a manipulação dos dados de forma compatível ao modelo de dados apropriado. Existem basicamente dois tipos: DML procedurais, que obrigam a especificação de quais dados são necessários e como obtê-los; e DML não procedurais, que exigem que o usuário especifique quais dados serão necessários, sem especificar como estes dados serão obtidos (Silberschatz et al., 1999). DML não procedurais são usualmente mais fáceis de aprender e usar do que DML procedurais. A SQL é não procedural e é orientada para conjuntos de informações, ou seja, permite aos usuários especificar o que deve ser feito e processado como conjuntos de elementos em lugar de uma tupla por vez, como ocorre em linguagem procedural.

Esta seção descreve e exemplifica: o comando SELECT (seção 2.3.1), as funções de agregação (seção 2.3.2), as operações que atuam sobre conjuntos (seção 2.3.3), os comandos de junção (seção 2.3.4), subconsultas aninhadas (seção 2.3.5) e relações derivadas (seção 2.3.6), além dos comandos UPDATE (seção 2.3.7), INSERT (seção 2.3.8) e DELETE (seção 2.3.9).

### 2.3.1 SELECT

O comando SELECT tem como objetivo selecionar um conjunto de registros em uma ou mais tabelas que atenda a uma determinada condição definida pelo comando.

A estrutura básica de uma expressão em SQL consiste em três cláusulas: SELECT, FROM e WHERE. A cláusula SELECT corresponde à operação de projeção da álgebra relacional. Ela é usada para relacionar os atributos desejados no resultado de uma consulta. A cláusula FROM corresponde à operação de produto cartesiano da álgebra relacional. Ela associa as relações que serão pesquisadas durante a evolução de uma expressão. A cláusula WHERE corresponde à seleção do predicado da álgebra relacional. Ela consiste em um predicado envolvendo atributos da relação que aparece na cláusula FROM. O resultado de uma consulta SQL é uma relação.

A sintaxe deste comando está descrita a seguir:

```
SELECT ALL / DISTINCT / <nome-coluna> [, <nome-coluna>] ...  
    FROM <nome-tabela> [, <nome-tabela>]  
    WHERE <condição>  
    GROUP BY <nome-coluna>  
    HAVING <condição>  
    ORDER BY <nome-campo> ASC  
            DESC
```

onde:

- a) nome-tabela - Representa os nomes das tabelas que contêm as colunas que serão selecionadas ou que serão utilizadas para a execução da consulta.
- b) condição - Representa a condição para a seleção dos registros. Esta seleção poderá resultar em um ou vários registros.
- c) nome-coluna - Representa as colunas que serão selecionadas ou que serão utilizadas para a execução da consulta.
- d) ALL - Opção default. Mostra todos os valores obtidos na seleção.
- e) DISTINCT - Mostra os valores obtidos na seleção eliminando as duplicidades.
- f) FROM - Especifica as tabelas nas quais serão pesquisados os dados, ou seja, as tabelas que possuem os dados que serão mostrados em resposta a consulta
- g) WHERE - Especifica o critério de seleção dos registros nas tabelas especificadas.
- h) GROUP BY - Especifica os campos que serão agrupados para atender à consulta.

i) HAVING - Especifica uma condição para seleção de um grupo de dados. Esta opção só é utilizada quando combinada com a opção GROUP BY.

j) ORDER BY – Apresenta o resultado da consulta ordenado de forma crescente ou decrescente pelos campos definidos.

### 2.3.2 FUNÇÕES AGREGADAS

Funções agregadas são funções que recebem uma coleção (i.e., um conjunto ou subconjunto) de valores com entrada, e retornam um valor simples. A SQL oferece cinco funções agregadas pré-programadas:

- média: AVG;
- mínimo: MIN;
- máximo: MAX;
- total: SUM; e
- contagem: COUNT.

Segundo Silberschatz et al. (1999), a entrada para SUM e AVG precisa ser um conjunto de números, mas as outras operações podem operar com tipos de dados não-numéricos, como *strings* e semelhantes.

Existem circunstâncias nas quais é necessário aplicar uma função agregada não somente a um conjunto de tuplas, mas também a um grupo de conjunto de tuplas. Isto pode ser feito usando-se a cláusula GROUP BY.

### 2.3.3 OPERAÇÕES DE CONJUNTOS

Os operadores SQL-92 UNION, INTERSECT e EXCEPT operam sobre relações compatíveis e correspondem às operações de  $\cap$ ,  $\cup$  e  $-$  da álgebra relacional. O resultado destas operações de conjunto são conjuntos de tuplas, o que significa que tuplas duplicadas são eliminadas do resultado. Dizer que operações de conjunto só podem ser aplicadas sobre relações compatíveis significa que as duas relações nas quais essas operações são aplicadas têm que ter o mesmo número de atributos, que esses atributos têm que aparecer na mesma ordem em ambas as relações e que atributos correspondentes em cada relação têm que ter o mesmo tipo de dado (Navathe, 2003).

### 2.3.4 JUNÇÃO (COMPOSIÇÃO DE RELAÇÕES)

Além de fornecer o mecanismo básico do produto cartesiano para a composição das tuplas de uma relação disponível nas primeiras versões de SQL, a SQL-92 também oferece diversos outros mecanismos para composição de relações, como as junções condicionais e as junções naturais, assim como várias formas de junções externas. Essas operações adicionais são usadas tipicamente como expressões de subconsultas na cláusula FROM.

Operações de junção recebem duas relações como entrada e geram como resultado uma outra relação. Cada uma das variantes das operações de junção em SQL-92 consiste em um tipo de junção e em uma condição de junção. Os tipos de junções que podem ser utilizados na SQL padrão SQL-92 são: INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN e FULL OUTER JOIN. O primeiro tipo de junção (i.e., INNER JOIN) é a junção interna e os outros três tipos são de junções externas. As condições de junção definem quais tuplas das duas relações apresentam correspondência e quais atributos são apresentados no resultado de uma junção. As três condições de junção são: natural, on e using.

### 2.3.5 SUBCONSULTAS ANINHADAS

Uma subconsulta aninhada consiste em uma expressão da forma SELECT ... FROM ... WHERE ... aninhada dentro de outra consulta. As aplicações mais comuns de subconsultas aninhadas consistem em: membros de conjuntos; comparações de conjuntos; verificação de relações vazias; e teste para ausência de tuplas repetidas.

A SQL moldou no cálculo relacional operações que permitem verificar se uma tupla é membro ou não de um relação. A cláusula IN testa os membros de um conjunto, sendo que o conjunto é a coleção de valores produzidos pela cláusula SELECT. O conectivo NOT IN verifica a ausência de membros de um conjunto. Em adição a esta funcionalidade, a cláusula IN também pode comparar uma tupla de valores entre parênteses com um conjunto ou multiconjunto de tuplas compatíveis para união.

Além da cláusula IN, uma série de outros operadores de comparação podem ser utilizados para comparar um único valor  $v$  (geralmente um nome de atributo) a um conjunto ou multiconjunto  $V$  (geralmente uma subconsulta). Para tanto, são utilizadas as cláusulas SOME e ALL.

A cláusula SOME permite a realização de diversas comparações:

- **< some:** “maior que algum”;
- **<= some:** “maior ou igual a algum”;
- **>= some:** “menor ou igual a algum”;

- = **some**: “igual a algum”; e
- <> **some**: “diferente de algum”.

Como ocorre para **SOME**, SQL também permite comparações:

- < **all**: “maior que todos”;
- <= **all**: “maior ou igual a todos”;
- >= **all**: “menor ou igual a todos”;
- = **all**: “igual a todos”; e
- <> **all**: “diferente de todos”.

A SQL possui também meios para testar se o resultado de uma subconsulta possui alguma tupla. A cláusula **EXISTS** retorna o valor *true* (verdadeiro) se o argumento de uma subconsulta é não-vazio, e *false* caso contrário. Em oposição à cláusula **EXISTS**, a cláusula **NOT EXISTS** retorna *true* se o resultado da consulta for nulo (i.e., não contiver nenhuma tupla).

Alem disso, a SQL contém recursos para testar se uma subconsulta possui alguma tupla repetida em seu resultado. A cláusula **UNIQUE** retorna o valor *true* (verdadeiro) caso o argumento da subconsulta não possua nenhuma tupla repetida. Caso seja utilizada a cláusula **NOT UNIQUE** e a sub-consulta não possuir valores repetidos, o resultado será *false*.

### 2.3.6 RELACAO DERIVADAS

A SQL-92 permite o uso de uma expressão de subconsulta na cláusula **FROM**. Se esse tipo de expressão for usado, a relação resultante deve receber um nome e os atributos precisam ser rebatizados usando-se a cláusula **AS**.

### 2.3.7 UPDATE

O comando **UPDATE** tem como objetivo atualizar os dados de um único registro ou de um grupo de registros em uma tabela do banco de dados. Sua sintaxe é descrita a seguir:

```
UPDATE <nome-tabela>
  SET <nome-coluna> = <novo conteúdo para o campo>
    [<nome-coluna> = <novo conteúdo para o campo>]
  WHERE <condição>
```

onde:

- a) nome-tabela - Representa o nome da tabela cujo conteúdo será alterado.
- b) nome-coluna - Representa os nomes das colunas que terão seus conteúdos alterados com o novo valor especificado.

c) condição - Representa a condição para a seleção dos registros que serão atualizados. Esta seleção poderá resultar na atualização de um ou vários registros.

### 2.3.8 INSERT

O comando INSERT tem como objetivo incluir um novo registro ou um conjunto de registros em uma tabela do banco de dados. Mais especificamente, pode-se inserir dados em uma relação especificando-se somente uma tupla a ser inserida ou escrevendo-se uma consulta cujo resultado é um conjunto de tuplas a serem inseridas. Obviamente, os valores dos atributos para as tuplas a inserir devem pertencer ao domínio desses atributos. Similarmente, tuplas a inserir devem possuir a ordem correta.

A sintaxe do comando INSERT é:

```
INSERT INTO <nome-tabela> [(<nome-coluna>, [<nome-coluna>])]  
VALUES (<relação dos valores a serem incluídos>)
```

onde:

- a) nome-tabela - Representa o nome da tabela onde será incluído o registro.
- b) nome-coluna - Representa os nomes das colunas da tabela sobre a qual a operação de inclusão está sendo realizada.

### 2.3.9 DELETE

O comando DELETE tem como objetivo remover um único registro ou um grupo de registros de uma tabela do banco de dados. Usando-se este comando, pode-se remover somente tuplas inteiras, ou seja, valores individuais de um atributo em particular não podem ser excluídos.

A sintaxe do comando DELETE é descrita a seguir:

```
DELETE FROM <nome-tabela>  
WHERE <condição>
```

onde:

- a) nome-tabela - Representa o nome da tabela onde será removido o registro.

b) condição - Representa a condição para a remoção dos registros. Esta seleção poderá resultar na remoção de um ou vários registros.

## 2.4 Comandos DDL e DML oferecidos pelos SGBD MySQL e Oracle.

Esta seção tem como objetivo comparar o SGBD MySQL e o SGBD Oracle com relação aos comandos SQL oferecidos por estes SGBD. Serão utilizadas as seguintes versões dos SGBD: MySQL 4.0.18 e Oracle 8i Personal Edition 8.1.7.0.0. Os resultados obtidos no desenvolvimento desta monografia podem ser constatados nas tabelas 2.2 e 2.3. O preenchimento destas tabelas envolveu as seguintes atividades:

- criação da base de dados especificada na seção 2.1, tanto no SGBD MySQL quanto no SGBD Oracle;
- povoamento de cada uma das tabelas; e
- execução das mesmas consultas em ambos os SGBD.

A Tabela 2.2 ilustra as funcionalidades oferecidas pelos SGBD MySQL e SGBD Oracle em termos de DDL.

Comandos SQL DDL	SGBD MySQL	SGBD Oracle
<b>CREATE TABLE</b>	Sim	Sim
KEY	Sim	Sim
PRIMARY KEY	Sim	Sim
FOREIGN KEY	Sim	Sim
NOT NULL	Sim	Sim
DEFAULT	Sim	Sim
UNIQUE	Sim	Sim
CHECK	Sim	Sim
<b>DROP TABLE</b>	Sim	Sim
<b>ALTER TABLE</b>	Sim	Sim
<b>CREATE VIEW</b>	Não	Sim
<b>DROP VIEW</b>	Não	Sim

Tabela 2.2 - Comandos SQL DDL oferecidos pelos SGBD MySQL e SGBD Oracle.

A tabela 2.3 ilustra as funcionalidades oferecidas pelos SGBD MySQL e SGBD Oracle em termos de DML.

Comandos SQL DML	SGBD MySQL	SGBD Oracle
<b>SELECT</b>	Sim	Sim
<b>FROM</b>	Sim	Sim
<b>WHERE</b>	Sim	Sim

<b>ORDER BY</b>	Sim	Sim
<b>GROUP BY</b>	Sim	Sim
<b>HAVING</b>	Sim	Sim
<b>NULL</b>	Sim	Sim
<b>UNION / UNION ALL</b>	Sim	Sim
<b>INTERSECT / INTERSECT ALL</b>	Não	Sim
<b>EXCEPT / EXCEPT ALL</b>	Não	Sim
<b>IN / NOT IN</b>	Não	Sim
<b>SOME</b>	Não	Sim
<b>ALL</b>	Não	Sim
<b>EXISTS / NOT EXISTS</b>	Não	Sim
<b>UNIQUE / NOT UNIQUE</b>	Não	Sim
<b>RELAÇÕES DERIVADAS</b>	Não	Sim
<b>DELETE</b>	Sim	Sim
<b>INSERT</b>	Sim	Sim
<b>UPDATE</b>	Sim	Sim
<b>JOIN</b>	Sim	Sim
INNER JOIN	Sim	Sim
LEFT OUTER JOIN	Sim	Sim
RIGHT OUTER JOIN	Sim	Sim
FULL OUTER JOIN	Sim	Sim

Tabela 2.3 - Comandos SQL DML oferecidos pelos SGBD MySQL e SGBD Oracle.

Apenas por questões de completude, as tabelas 2.4 e 2.5 ilustram, respectivamente, os tipos de dados para os quais os SGBD MySQL e SGBD Oracle oferecem suporte.

<b>Tipo de Dados</b>	<b>MySQL</b>
<b>Tipos Numéricos</b>	
TINYINT	Sim
SMALLINT	Sim
MEDIUMINT	Sim
INT	Sim
BIGINT	Sim
FLOAT	Sim
DOUBLE	Sim
DOUBLE PRECISION	Não
REAL	Não
DECIMAL	Sim
SERIAL	Não
MONEY	Não
NUMBER	Não
LONG	Não
<b>Tipos Data e Hora</b>	
DATE	Sim

DATETIME	<b>Sim</b>
TIMESTAMP	<b>Sim</b>
TIME	<b>Sim</b>
YEAR	<b>Sim</b>
INTERVAL	<b>Não</b>
TIME WITH TIME ZONE	<b>Não</b>
<b>Tipos String</b>	
CHAR	<b>Sim</b>
VARCHAR	<b>Sim</b>
VARCHAR2	<b>Não</b>
TINYBLOB / TINYTEXT	<b>Sim</b>
BLOB / TEXT	<b>Sim</b>
MEDIUMBLOB / MEDIUMTEXT	<b>Sim</b>
LOB / LONGTEXT	<b>Sim</b>
ENUM	<b>Sim</b>
SET	<b>Sim</b>
BIT	<b>Não</b>
VARBIT	<b>Não</b>
CHARACTER	<b>Não</b>
CLOB	<b>Não</b>
LONG RAW	<b>Não</b>
LONG VARCHAR	<b>Não</b>
MSL LABEL	<b>Não</b>
NCHAR	<b>Não</b>
NCLOB	<b>Não</b>
NVARCHAR2	<b>Não</b>
RAW MLS LABEL	<b>Não</b>
RAW	<b>Não</b>
<b>Tipos Booleanos</b>	
BOOL	<b>Não</b>
<b>Tipos Geométricos</b>	
POINT	<b>Não</b>
LINE	<b>Não</b>
LSEG	<b>Não</b>
BOX	<b>Não</b>
PATH	<b>Não</b>
POLYGON	<b>Não</b>
CIRCLE	<b>Não</b>
<b>Tipos Endereço de Rede</b>	
CIDR	<b>Não</b>
INET	<b>Não</b>
MASCADDR	<b>Não</b>

Tabela 2.4 - Tipos de dados oferecidos pelo SGBD MySQL.

<b>Tipo de Dados</b>	<b>Oracle</b>
<b>Tipos Numéricos</b>	
TINYINT	<b>Não</b>

SMALLINT	<b>Sim</b>
MEDIUMINT	<b>Não</b>
INT / INTEGER	<b>Sim</b>
BIGINT	<b>Não</b>
FLOAT	<b>Sim</b>
DOUBLE	<b>Não</b>
DOUBLE PRECISION	<b>Não</b>
REAL	<b>Não</b>
DECIMAL	<b>Sim</b>
SERIAL	<b>Não</b>
MONEY	<b>Não</b>
NUMBER	<b>Sim</b>
LONG	<b>Sim</b>
<b>Tipos Data e Hora</b>	
DATE	<b>Sim</b>
DATETIME	<b>Não</b>
TIMESTAMP	<b>Não</b>
TIME	<b>Não</b>
YEAR	<b>Não</b>
INTERVAL	<b>Não</b>
TIME WITH TIME ZONE	<b>Não</b>
<b>Tipos String</b>	
CHAR	<b>Sim</b>
VARCHAR	<b>Sim</b>
VARCHAR2	<b>Sim</b>
TINYBLOB / TINYTEXT	<b>Não</b>
BLOB / TEXT	<b>Sim</b>
MEDIUMBLOB / MEDIUMTEXT	<b>Não</b>
LOB / LONGTEXT	<b>Não</b>
ENUM	<b>Não</b>
SET	<b>Não</b>
BIT	<b>Não</b>
VARBIT	<b>Não</b>
CHARACTER	<b>Sim</b>
CLOB	<b>Sim</b>
LONG RAW	<b>Sim</b>
LONG VARCHAR	<b>Sim</b>
MSL LABEL	<b>Sim</b>
NCHAR	<b>Sim</b>
NCLOB	<b>Sim</b>
NVARCHAR2	<b>Sim</b>
RAW MLS LABEL	<b>Sim</b>
RAW	<b>Sim</b>
<b>Tipos Booleanos</b>	
BOOL	<b>Não</b>
<b>Tipos Geométricos</b>	
POINT	<b>Não</b>
LINE	<b>Não</b>
LSEG	<b>Não</b>

BOX	<b>Não</b>
PATH	<b>Não</b>
POLYGON	<b>Não</b>
CIRCLE	<b>Não</b>
<b>Tipos Endereço de Rede</b>	
CIDR	<b>Não</b>
INET	<b>Não</b>
MASCADDR	<b>Não</b>

Tabela 2.5 - Tipos de dados disponíveis pelo SGBD Oracle.

Note que o objetivo deste trabalho é apresentar e analisar a implementação dos comandos padrão SQL-92. No momento, não pretende-se comparar as funções adicionais que um dos SGBD em estudo oferece em relação ao outro, tais como:

- funções matemáticas: `abs()`, `sin()` e `sqrt()`;
- funções para comparação e operação com strings: `concat()`, `length()` e `lower()`; e
- funções de data e hora: `curdate()`, `dayname()`, e `now()`.

## 2.5 Considerações Finais

Este capítulo descreveu as principais características dos comandos SQL que um SGBD deve oferecer segundo o padrão SQL-92. Esses comandos, relacionados à DDL e à DML, foram enfocados em termos de suas funcionalidades e foram desenvolvidos exemplos de uso de cada comando. Os comandos listados foram testados tanto no SGBD MySQL quanto no SGBD Oracle, gerando uma tabela comparativa entre estes dois SGBD.

Analisando-se as tabelas 2.2 e 2.3, conclui-se que nas versões em estudo, o SGBD MySQL possui limitações quanto ao oferecimento de certos comandos SQL que não são suportados pelo SGBD. Já o SGBD Oracle oferece uma grande quantidade de comandos para manipular e acessar as estruturas de bancos de dados relacionais criadas.

Como contribuição para a autora desta monografia de especialização, o estudo da linguagem SQL permitiu que a mesma aprimorasse o seu conhecimento dessa linguagem, desde que o seu contato com a SQL era pequeno e distante. De um ponto de vista mais genérico, a análise comparativa entre os comandos SQL oferecidos pelos SGBD MySQL e Oracle pode servir de referência para a escolha de um SGBD em detrimento ao outro, de acordo com as necessidades de desenvolvimento da aplicação.

## CAPÍTULO 3

### Funções de Acesso aos SGBD MySQL e Oracle usando a Linguagem PHP

Atualmente, o desenvolvimento de aplicações de banco de dados para a Web tem sido fortemente baseado no uso do SGBD MySQL e da linguagem de programação PHP. Isto se deve ao fato de que a linguagem PHP oferece várias funções de fácil utilização que permitem o acesso, a pesquisa e a manipulação dos dados armazenados no SGBD MySQL. Entretanto, esse SGBD possui algumas limitações quando comparado às funcionalidades oferecidas por outros SGBD, como por exemplo o Oracle. Assim, em adição à necessidade de se investigar quais funcionalidades adicionais são oferecidas pelo SGBD Oracle com relação ao SGBD MySQL, trabalho este apresentado no Capítulo 2, também existe a necessidade de se analisar como o acesso ao SGBD Oracle pode ser realizado por meio da linguagem PHP. Esta segunda perspectiva consiste no objetivo deste capítulo.

A seção 3.1 descreve as funções de acesso ao SGBD MySQL usando a linguagem PHP. Já a seção 3.2 descreve as funções de acesso ao SGBD Oracle usando esta mesma linguagem. Uma tabela comparativa entre as funções estudadas na seção 3.1 e as funções identificadas na seção 3.2 é apresentada na seção 3.3. A seção 3.4 conclui o capítulo.

#### 3.1 Funções de Acesso ao SGBD MySQL usando a Linguagem PHP

Esta seção descreve diversas funções de acesso ao SGBD MySQL, e está organizada da seguinte forma. Para cada função, é apresentada inicialmente a sua descrição básica, e em seguida a sua sintaxe e um exemplo de utilização. O estudo e a descrição das funções desta seção foi fortemente baseado no material de Williams & Lane (2002), Ciferri & Ciferri (2003) e Welling & Thomson (2003).

##### Função `mysql_connect()`

A função `mysql_connect ()` abre uma conexão com o SGBD MySQL. Em caso de sucesso, o valor de retorno pode ser utilizado para acessar a informação associada à conexão. Caso contrário, a função retorna *false*. A sintaxe desta função é:

```
resource mysql_connect ( [string host], [string username],  
                        [string password] )
```

onde:

- a) string host - Nome da máquina servidora do SGBD MySQL;
- b) string username - Nome do usuário; e
- c) string password - Senha do usuário.

A seguir é exemplificado o uso da função `mysql_connect()` que abre uma conexão com o SGBD MySQL, onde o nome da máquina é `localhost`, o nome de usuário é `ana` e a senha é `123456`.

```
$conexão = mysql_connect ("localhost", "ana", "123456");
```

### **Função `mysql_select_db()`**

A função `mysql_select_db()` seleciona o banco de dados a ser utilizado. Em caso de erro, essa função retorna *false*. Sua sintaxe é:

```
int mysql_select_db ( string database, [resource connection] )
```

onde:

- a) string database - Nome do banco de dados; e
- b) resource connection - Nome da conexão com o SGBD, que representa o valor de retorno da função `mysql_connect()`.

A seguir é exemplificado o uso da função `mysql_select_db()` para selecionar o banco de dados `winestore` e utilizar a variável `$conexão` com os dados da conexão com o SGBD.

```
mysql_select_db ("winestore", $conexão);
```

### **Função `mysql_query()`**

A função `mysql_query()` submete uma consulta ao SGBD MySQL. Em caso de sucesso, a função retorna o conjunto resultado contendo as tuplas (i.e., linhas) selecionadas pela consulta SQL. Caso contrário, a função retorna *false*. A sintaxe desta função está descrita a seguir:

```
resource mysql_query ( string command, [resource connection] )
```

onde:

- a) string command - Comando SQL; e
- b) resource connection - Nome da Conexão com o SGBD.

A seguir é exemplificado o uso da função `mysql_query( )`, a qual submete uma consulta ao SGBD MySQL através do comando `SELECT * FROM WINE` e utiliza a variável `$conexão` com os dados da conexão com o SGBD.

```
$resultado = mysql_query ("SELECT * FROM wine", $conexão);
```

### **Função `mysql_fetch_row( )`**

A função `mysql_fetch_row( )` recupera uma tupla do conjunto resultado. Em caso de sucesso, a função retorna uma variável do tipo vetor que armazena a tupla. Caso contrário, a função retorna *false* quando não existirem mais tuplas a serem processadas. Sua sintaxe é:

```
array mysql_fetch_row ( resource result_set )
```

onde:

a) `resource result_set` - Conjunto resultado.

A seguir é exemplificado a função `mysql_fetch_row( )` que recupera os dados através variável `$linha` e apresenta os dados código do cliente e nome do cliente.

```
$linha = mysql_fetch_row ($resultado);  
echo "Código Cliente: $linha[0] <br>";  
echo "Nome Cliente: $linha[1] <br>";
```

### **Função `mysql_num_fields( )`**

A função `mysql_num_fields( )` retorna o número de atributos na tupla. A sintaxe desta função está descrita a seguir:

```
int mysql_num_fields ( resource result_set )
```

onde:

a) `resource result_set` - Conjunto resultado.

A seguir é exemplificado a função `mysql_num_fields( )` que armazena em `$atributos` o número de atributos de `$resultado`.

```
$atributos = mysql_num_fields ($resultado);
```

### **Função `mysql_close()`**

A função `mysql_close()` fecha uma conexão com o SGBD MySQL. A função retorna *false* em caso de erro. As conexões abertas pela função `mysql_connect()` são automaticamente fechadas quando o *script* termina. A sintaxe desta função é:

```
int mysql_close ( [resource connection] )
```

onde:

a) `resource connection` - Nome da conexão com o SGDB.

O uso da função `mysql_close()` pode ser exemplificado pelo comando:

```
mysql_close ($conexão).
```

### **Função `mysql_data_seek()`**

A função `mysql_data_seek()` recupera apenas algumas tuplas para uma consulta. Em caso de erro, a função retorna *false*. Sua sintaxe é:

```
int mysql_data_seek ( resource result_set, int row )
```

onde:

a) `resource result_set` - Conjunto resultado; e

b) `int row` - Linha a partir da qual as tuplas serão recuperadas.

Um exemplo desta função é descrito a seguir, o qual recupera a linha 3 armazenada em `$resultado`.

```
mysql_data_seek ($resultado, 3 )
```

### **Função `mysql_fetch_array()`**

A função `mysql_fetch_array()` é uma versão estendida de `mysql_fetch_row()` que recupera uma tupla do conjunto resultado. A diferença é que `mysql_fetch_array()` permite acesso aos valores do vetor pelos nomes dos atributos da tabela. Em caso de sucesso, a função retorna uma variável do tipo vetor que armazena a tupla. Caso contrário, a função retorna *false* quando não existirem mais tuplas a serem processadas. Sua sintaxe é:

```
array mysql_fetch_array (resource result_set, [int result_type])
```

onde:

- a) resource result\_set - Conjunto resultado; e
- b) int result\_type - Tipo de resultado.
  - acesso aos valores do vetor por nomes de atributos (MYSQL\_ASSOC)
  - acesso numérico aos valores do vetor (MYSQL\_NUM)
  - ambos (*default*, MYSQL\_BOTH)

A seguir é exemplificada a função `mysql_fetch_array( )` que recupera os dados através variável `$linha` e apresenta os dados código do cliente e nome do cliente.

```
$linha = mysql_fetch_array ($resultado);
echo "Código Cliente: $linha[cust_id] <br>";
echo "Nome Cliente: $linha[surname] <br>";
```

### Função `mysql_fetch_object( )`

A função `mysql_fetch_object( )` é uma versão estendida de `mysql_fetch_row( )` que recupera uma tupla do conjunto resultado. A diferença é que `mysql_fetch_object( )` permite acesso aos valores do objeto pelos nomes dos atributos da tabela. Em caso de sucesso, a função retorna variável do tipo objeto que armazena a tupla. Caso contrário, a função retorna *false* quando não existirem mais tuplas a serem processadas. Sua sintaxe é:

```
object mysql_fetch_object(resource result_set, [int result_type])
```

onde:

- a) resource result\_set - Conjunto resultado; e
- b) int result\_type - Tipo de resultado (opcional).
  - acesso aos valores do objeto por nomes de atributos (MYSQL\_ASSOC)
  - acesso numérico aos valores do objeto (MYSQL\_NUM)
  - ambos (*default*, MYSQL\_BOTH)

A seguir é exemplificado a função `mysql_fetch_object( )` que recupera os dados através variável `$objeto` e apresenta os dados código do cliente e nome do cliente.

```
$objeto = mysql_fetch_object ($resultado);
echo "Código Cliente: $objeto->cust_id <br>";
echo "Nome Cliente: $objeto->surname <br>";
```

### **Função `mysql_free_result()`**

A função `mysql_free_result()` libera a memória usada para armazenar o conjunto resultado. Em caso de erro, a função retorna *false*. A sintaxe desta função é descrita a seguir:

```
int mysql_free_result (resource result_set)
```

onde:

a) `resource result_set` - Conjunto resultado.

O comando a seguir libera a memória da variável `$resultado`:

```
mysql_free_result ($resultado);
```

### **Função `mysql_num_rows()`**

A função `mysql_num_rows()` retorna o número de tuplas (i.e., linhas) no conjunto resultado. Esta função deve ser sempre utilizada conjuntamente com comandos `SELECT`. Em caso de erro, a função retorna *false*. Sua sintaxe é:

```
int mysql_num_rows ( resource result_set )
```

onde:

a) `resource result_set` - Conjunto resultado.

O comando a seguir retorna o número de tuplas da variável `$resultado`:

```
$nro_tuplas = mysql_num_rows ($resultado);
```

### **Função `mysql_pconnect()`**

A função `mysql_pconnect()` possui características semelhantes à função `mysql_connect()`. Ela abre uma conexão com o SGBD MySQL e possui os mesmos parâmetros e o mesmo valor de retorno. Porém a função `mysql_pconnect()` abre uma conexão persistente com o SGBD, isto é, a conexão não é fechada quando o *script* termina e não pode ser fechada com a função `mysql_close()`. Sua sintaxe é:

```
resource mysql_pconnect ( [string host], [string username],  
[string password] )
```

onde:

a) `string host` - Nome ou endereço da máquina servidora do SGBD MySQL;

- b) string username - Nome do usuário no SGBD MySQL; e
- c) string password - Senha do usuário no SGBD MySQL.

A seguir é exemplificado o uso da função `mysql_pconnect()`:

```
$conexão = mysql_pconnect ("localhost", "ana", "123456");
```

### **Função `mysql_change_user()`**

A função `mysql_change_user()` altera o usuário corrente para um novo usuário. Em caso de erro, a função retorna *false* e a conexão anterior permanece corrente. A sintaxe desta função está descrita a seguir:

```
int mysql_change_user (string user, string password, [string  
database, [resource connection]] )
```

onde:

- a) string user - Nome do usuário;
- b) string password - Senha de acesso do usuário;
- c) string database - Nome da base de dados; e
- d) resource connection - Nome da conexão.

A seguir é exemplificado o uso da função `mysql_change_user()`:

```
mysql_change_user (root, 123mudar, "winestore", $conexão)
```

### **Função `mysql_create_db()`**

A função `mysql_create_db()` cria um banco de dados no SGBD MySQL. Em caso de erro, a função retorna *false*. Sua sintaxe é:

```
int mysql_create_db (string database, [resource connection] )
```

onde:

- a) string database: Nome do Banco de Dados; e
- b) resource connection: Nome da conexão do SGBD MySQL.

A seguir é exemplificado a função `mysql_create_db()`:

```
mysql_create_db ("exemplo", $conexão);
```

### Função `mysql_drop_db()`

A função `mysql_drop_db()` remove um banco de dados do MySQL. Em caso de erro, a função retorna *false*. A sintaxe desta função está descrita a seguir:

```
int mysql_drop_db (string database, [resource connection])
```

onde:

- a) string database: Nome do banco de dados; e
- b) resource connection: Nome da conexão.

O comando a seguir exemplificada o uso da função `mysql_drop_db()`:

```
mysql_drop_db ("exemplo", $conexão);
```

### Função `mysql_fetch_field()`

A função `mysql_fetch_field()` obtém o metadado para um atributo da relação. Em caso de erro, a função retorna *false*; caso contrário, a função retorna um objeto que armazena o metadado para o atributo.

A Tabela 3.1 mostra as propriedades do objeto retornado por esta função.

Propriedades do Objeto Retornado	
Name	nome do atributo
Table	nome da tabela à qual o atributo pertence
max_length	comprimento máximo do atributo
not_null	1, se o atributo não pode ser NULL
primary_key	1, se o atributo faz parte da chave primária
Unique_key	1, se existe a restrição UNIQUE
multiple_key	1, se o atributo não é uma chave única
numeric	1, se o atributo é do tipo numérico
Blob	1, se o atributo é do tipo BLOB
Type	tipo do atributo
unsigned	1, se o atributo é do tipo numérico sem sinal
Zerofill	1, se a coluna numérica é preenchida com 0

Tabela 3.1 - Propriedades do objeto retornado da função `mysql_fetch_field()`

A sintaxe desta função é:

```
object mysql_fetch_field ( resource result_set, [int  
attribute_name] )
```

onde:

- a) resource *result\_set*: Conjunto Resultado; e
- b) int *attribute\_name*: Número do atributo.

O comando a seguir ilustra o uso de `mysql_fetch_field()`:

```
$info = mysql_fetch_field ($resultado);
```

### **Função `mysql_list_tables()`**

A função `mysql_list_tables()` identifica as tabelas do banco de dados. Em caso de sucesso, a função retorna um conjunto resultado contendo as tabelas do banco de dados. Caso contrário, a função retorna *false*. A sintaxe desta função é:

```
resource mysql_list_tables ( string database, [resource  
connection] )
```

onde:

- a) string *database*: Nome da base de dados; e
- b) resource *connection*: Nome da conexão.

A seguir é exemplificado o uso da função `mysql_list_tables()`:

```
$tabelas = mysql_list_tables ("winestore", $conexão);
```

### **Função `mysql_tablename()`**

A função `mysql_tablename()` retorna o nome de uma tabela. Em caso de sucesso, a função retorna o nome da tabela relativa ao valor numérico. Caso contrário, a função retorna *false*. Esta função deve ser sempre utilizada em conjunto com `mysql_list_tables()`. Sua sintaxe é:

```
string mysql_tablename ( resource result, int table_number)
```

onde:

- a) resource *result*: Conjunto resultado; e
- b) int *table\_number*: Número da tabela.

Os comandos a seguir ilustram o uso da função `mysql_tablename()` conjuntamente com o uso da função `mysql_list_tables`:

```
$tabelas = mysql_list_tables ("winestore", $conexão);  
echo mysql_tablename ($tabelas, 3);
```

### **Função `mysql_errno()`**

A função `mysql_errno()` retorna o número do último erro. A sintaxe desta função está descrita a seguir:

```
int mysql_errno ( [resource connection] )
```

onde:

a) `resource connection`: Nome da conexão com o SGBD.

A seguir é exemplificado o uso da função `mysql_errno()`:

```
mysql_errno ($conexão);
```

### **Função `mysql_error()`**

A função `mysql_error()` retorna uma *string* contendo a descrição do último erro. Esta função tem a seguinte sintaxe:

```
int mysql_error ( [resource connection] )
```

onde:

a) `resource connection`: Nome da conexão com o SGBD.

O comando a seguir ilustra a utilização da função `mysql_error()`:

```
mysql_error ($conexão);
```

### **Função `mysql_affected_rows()`**

Essa função tem a finalidade de retornar a quantidade de linhas que foram alteradas depois da execução de um comando SQL que foi executado de forma correta. Em outras palavras, depois da execução dos comandos INSERT, UPDATE ou DELETE, pode-se utilizar a função `mysql_affected_rows()` para verificar quantas linhas foram alteradas em decorrência do processamento de algum desses comandos. Sua sintaxe é:

```
int mysql_affected_rows ( [resource connection] )
```

onde:

a) `resource connection`: Nome da conexão.

A seguir é exemplificado o uso da função `mysql_affected_rows()`:

```
$operação = "UPDATE customer SET firstname = 'Ricardo' WHERE
cust_id = 2";
mysql_query($operação, $connection);
$alt = mysql_affected_rows($conexão);
if (alt == 1)
    {echo "Alteração feita com Sucesso!";}
else
    {echo "Alteração não realizada!";}
```

### Função `mysql_insert_id()`

Esta função retorna o valor da chave primária da última tupla inserida no banco de dados, e tem como sintaxe:

```
int mysql_insert_id ([resource connection])
```

onde:

a) resource connection: Nome da conexão.

A seguir é exemplificado a função `mysql_insert_id()`:

```
$operação = "INSERT INTO customer (cust_id, firstname)
VALUES (NULL, 'Kelly')";
mysql_query($operação, $conexão);
$alt = mysql_affected_rows($conexão);
if (alt == 1)
    {
        echo "Inclusão realizada com Sucesso!";
        $cod_cli = mysql_insert_id($conexão);
        echo $cod_cli;
    }
else
    {echo "Alteração não realizada!";}
```

## 3.2 Funções de Acesso ao SGBD Oracle usando a Linguagem PHP

Esta seção descreve diversas funções de acesso ao SGBD Oracle utilizando a linguagem PHP. Para cada função, inicialmente será realizada uma breve descrição de sua funcionalidade e em seguida serão apresentados a sua sintaxe e um exemplo de utilização. Essas funções

permitem que seja acessado tanto o banco de dados Oracle7 quanto o banco de dados Oracle8. Com relação a este último, deve ser utilizada a interface de comunicação OCI8.

A investigação das funções desta seção é fortemente baseada no material disponível em (PHP, 2005).

### **Função OCILogon( )**

A função OCILogon( ) abre uma conexão com o SGBD Oracle. Em caso de sucesso, o valor de retorno pode ser utilizado como um identificador de conexão necessária para a grande maioria das outras funções OCI. Caso contrário, a função retorna *false*. Sua sintaxe é:

```
resource OCILogon ( string username, string password [, string db])
```

onde:

- a) string username - Nome do usuário;
- b) string password - Senha do usuário; e
- c) string db – Nome do banco de dados que será utilizado na conexão. É opcional.

A seguir é exemplificado o uso da função OCILogon( ). Neste exemplo, a função abre uma conexão com o SGBD Oracle, sendo que o nome de usuário é ana e a senha é ana.

```
$conexao = @OCILogon("Ana", "ana");
```

### **Função OCINLogon( )**

A função OCINLogon( ) cria uma conexão com o SGBD Oracle. Esta funcionalidade deve ser usada quando é necessário isolar algumas transações. Se existirem múltiplas conexões abertas usando OCINLogon( ), todas as operações de *commit* e de *rollback* serão aplicadas para uma conexão somente.

Em caso de sucesso, o valor de retorno da função OCINLogon( ) pode ser utilizado como um identificador de conexão necessária para a grande maioria das outras funções OCI. Caso contrário, a função retorna *false*. A sintaxe desta função é:

```
resource OCINLogon ( string username, string password [, string db])
```

onde:

- a) string username - Nome do usuário;

- b) string password - Senha do usuário; e
- c) string db – Nome do banco de dados usado na conexão. É opcional.

O comando a seguir abre uma conexão com o SGDB Oracle, sendo que o nome de usuário é ana e a senha é ana.

```
$conexao = @OCILogon("Ana", "ana");
```

### **Função OCIPLogon( )**

A função OCIPLogon( ) conecta no Oracle usando uma conexão persistente. Em caso de sucesso, o valor de retorno pode ser utilizado como um identificador de conexão necessária para a grande maioria das outras funções OCI. Caso contrário, a função retorna *false*. A sintaxe desta função está descrita a seguir:

```
resource OCIPLogon ( string username, string password [, string db])
```

onde:

- a) string username - Nome do usuário;
- b) string password - Senha do usuário; e
- c) string db – Nome do banco de dados utilizado na conexão. É opcional.

O comando a seguir abre uma conexão persistente com o SGDB Oracle. Neste exemplo, o nome de usuário é ana e a senha é ana.

```
$conexao = @OCIPLogon("Ana", "ana");
```

### **Função OCIParse( )**

A função OCIParse( ) interpreta uma consulta e retorna uma declaração Oracle. A função identifica se a declaração retornada na consulta é válida; caso contrário, a função retorna *false*. A sintaxe desta função é:

```
resource OCIParse ( resource conn, string query)
```

onde:

- a) resource conn - Nome da conexão com o SGDB, a qual consiste no valor de retorno da função OCILogon (); e
- b) string query - Comando SQL válido.

A seguir é exemplificado o uso da função `OCI Parse()`.

```
$SQL = @OCI Parse($conexao, "select * from ANA.wine");
```

### **Função OCIExecute()**

A função `OCIExecute()` executa efetivamente um comando SQL. Em caso de sucesso, a função retorna *true*. Caso contrário, a função retorna *false*. Sua sintaxe é:

```
bool OCIExecute ( resource stmt [, int mode])
```

onde:

- a) `resource stmt` - Nome da variável que armazena o comando SQL a ser executado; e
- b) `int mode` – Parâmetro opcional, que permite que seja especificado o modo de execução. O padrão é `OCI_COMMIT_ON_SUCCESS`. Caso não se deseje que os blocos de comando sejam executados automaticamente, deve-se então especificar `OCI_DEFAULT`.

O comando a seguir exemplifica o uso da função `OCIExecute()`:

```
@OCIExecute($SQL, OCI_DEFAULT);
```

### **Função OCIFetchInto()**

A função `OCIFetchInto()` retorna a próxima tupla (i.e., linha) para um comando `SELECT`. Esta linha é armazenada em um array resultado. Por padrão, inicialmente o array contém valores iniciais iguais a zero para todas as colunas que não sejam nulas. Como resultado, a função irá sobrescrever o conteúdo prévio do array. Sua sintaxe é:

```
Int OCIFetchInto ( resource stmt, array & result [, int mode])
```

onde:

- a) `resource stmt` - Nome da variável que contém o SQL a ser executado.
- b) `array & result` - Conjunto resultado.
- c) `int mode` - O parâmetro `mode` permite alterar o comportamento padrão. Pode-se especificar mais do que um flag simplesmente adicionando um sinal positivo. Os flags são:

`OCI_ASSOC` – retorna um array associativo.

`OCI_NUM` – retorna um array indexado iniciando em zero (DEFAULT).

`OCI_RETURN_NULLS` – retorna colunas vazias.

`OCI_RETURN_LOBS` – retorna o valor de um LOB ao invés da descrição.

A seguir é exemplificado o uso da função `OCIFetchInto()`.

```
while (@OCIFetchInto($stmt, $row, OCI_ASSOC))
{
    print $row['WINERY_NAME'] . "<br>";
}
```

### Função `OCIFetch()`

A função `OCIFetch()` retorna a próxima tupla (i.e., linha) para um comando `SELECT`. Esta linha é armazenada em um buffer resultado interno. Em caso de sucesso, a função retorna *true*. Caso contrário, a função retorna *false*. A sintaxe desta função é:

```
bool OCIFetch ( resource stmt )
```

onde:

a) `resource stmt` - Nome da variável que contém o comando SQL a ser executado.

A seguir é exemplificada a função `OCIResult()`.

```
while (@OCIResult($stmt))
{
    echo @OCIResult($stmt, "WINE_NAME") . "<br>";
}
```

### Função `OCIResult()`

A função `OCIResult()` retorna valores de colunas trazidas por uma linha. `OCIResult()` retorna o tipo de dados de uma coluna na linha atual (`OCIFetch()`), com exceção de strings de tipo abstrato (`ROWID`, `LOB` e `FILE`). A sintaxe desta função é:

```
Mixed OCIResult ( resource stmt, mixed col )
```

onde:

a) `resource stmt` - Nome da variável que contém o comando SQL; e

b) `mixed col` – Número da coluna ou o nome da coluna, em letras maiúsculas.

O comando a seguir exemplifica o uso da função `OCIResult()`:

```
@OCIResult($stmt, "WINE_NAME");
```

### **Função OCIRowCount( )**

A função OCIRowCount( ) retorna o número de linhas afetadas. Sua sintaxe é:

```
Int OCIRowCount ( resource stmt )
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL.

O comando a seguir exemplifica o uso da função OCIRowCount( ):

```
@OCIRowCount ($stmt) ;
```

### **Função OCINumCols( )**

A função OCINumCols( ) retorna o número de atributos na tupla. A sintaxe desta função está descrita a seguir:

```
Int OCINumCols ( resource stmt )
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL.

O comando a seguir exemplificada o uso da função OCINumcols( ):

```
@OCINumCols ($stmt) ;
```

### **Função OCICommit( )**

A função OCICommit( ) valida todas as declarações pendentes para as transações ativas na conexão Oracle `connection`. A sintaxe desta função é:

```
bool OCICommit ( resource connection )
```

onde:

a) resource connection - Nome da variável que contém os dados (i.e., usuário e senha) da conexão.

A seguir é exemplificada a função OCICommit( ):

```
@OCICommit ($OCILogon) ;
```

### Função OCIRollback( )

A função OCIRollback( ) retorna todas os blocos de comando pendentes em uma conexão Oracle. Em caso de sucesso, a função retorna *true*. Sua sintaxe é:

```
bool OCIRollback ( resource connection )
```

onde:

a) resource connection - Nome da variável que contém os dados (i.e., usuário e senha) da conexão.

O comando a seguir exemplifica o uso da função OCIRollback( ):

```
@OCIRollback ($OCILogon) ;
```

### Função OCIError( )

A função OCIError( ) retorna o último erro encontrado como um array associativo. Neste array, code consiste no código de erro Oracle e message na string de erro do Oracle. Se nenhum erro for encontrado, OCIError( ) retorna *false*. A sintaxe desta função é:

```
array OCIError ( resource stmt|conn|global )
```

onde:

a) resource stmt|conn|global - Nome da variável que contém os dados (i.e., usuário e senha) da conexão e o comando SQL solicitado.

O exemplo a seguir ilustra o uso da função OCIError( ):

```
@OCIError ($OCILogon) ;
```

### Função OCILogoff( )

A função OCILogoff( ) fecha uma conexão Oracle. A chamada à função OCILogoff( ) não é usualmente necessária. Por exemplo, quando conexões não persistentes são abertas, elas são automaticamente fechadas no final da execução de um *script*. A sintaxe da função é:

```
bool OCILogoff ( resource connection )
```

onde:

a) resource connection - Nome da conexão Oracle.

A seguir é exemplificada a função `OCILogoff()`:

```
@OCILogoff($OCILogon);
```

### Função `OCIColumnSize()`

A função `OCIColumnSize()` retorna o tamanho de uma coluna. Este tamanho é aquele definido na criação da tabela. A sintaxe desta função é descrita a seguir:

```
int OCIColumnSize ( resource stmt, mixed column )
```

onde:

- a) `resource stmt` - Nome da variável que contém o comando SQL; e
- b) `mixed column` – Nome da variável que contém o número da coluna.

No exemplo a seguir, o tamanho da coluna impresso é 50, desde que o campo `wine_name` foi definido como do tipo `varchar(50)`.

```
$t = @OCIParse($OCILogon, "select * from ANA.wine");  
@OCIExecute($t, OCI_DEFAULT);  
$OCIColumnSize = @OCIColumnSize($t, 2);  
print "Tamanho da coluna ".$OCIColumnSize."<p>";
```

### Função `OCIColumnName()`

A função `OCIColumnName()` retorna o nome de uma coluna. Sua sintaxe é:

```
string OCIColumnName ( resource stmt, int col )
```

onde:

- a) `resource stmt` - Nome da variável que contém o comando SQL; e
- b) `int col` – Nome da variável que contém o número da coluna desejada.

O comando a seguir retorna o nome da coluna `$i` para o comando SQL armazenado na variável em `$sql`:

```
@OCIColumnName($sql, $i);
```

### Função `OCIColumnType()`

A função `OCIColumnType()` retorna o tipo de dado de uma coluna. A sintaxe desta função é descrita a seguir:

```
mixed OCIColumnType ( resource stmt, int col )
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL;

b) int col – Nome da variável que contém o número da coluna.

O comando a seguir retorna o tipo de dado da coluna \$i para o comando SQL armazenado na variável em \$sql:

```
@OCIColumnType ($sql, $i);
```

### **Função OCICancel()**

A função OCICancel() cancela a leitura de um cursor. Sua sintaxe é:

```
bool OCICancel ( resource stmt)
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL.

A seguir é exemplificada a função OCICancel():

```
@OCICancel ($sql);
```

### **Função OCIFreeCursor()**

A função OCIFreeCursor() libera todos os recursos associados a um cursor. Esta função tem a seguinte sintaxe:

```
bool OCIFreeCursor (resource stmt)
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL.

A seguir é exemplificada a função OCIFreeCursor():

```
@OCIFreeCursor ($sql);
```

### **Função OCIFreeStatement()**

A função OCIFreeStatement() libera todos os recursos associados a uma declaração. Sua sintaxe é:

```
bool OCIFreeStatement (resource stmt)
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL.

O comando a seguir exemplifica o uso da função OCIFreeStatement():

```
@OCIFreeStatement ($sql);
```

### Função OCIStatementType( )

A função OCIStatementType( ) retorna o tipo de uma declaração OCI, que pode ser um dos seguintes valores: SELECT, UPDATE, DELETE, INSERT, CREATE, DROP, ALTER, BEGIN, DECLARE ou UNKNOWN. A sintaxe desta função é:

```
string OCIStatementType (resource stmt)
```

onde:

a) resource stmt - Nome da variável que contém o comando SQL.

O comando a seguir exemplifica o uso desta função:

```
@OCIStatementType ($stmt);
```

### 3.3 Comparação entre as Funções de Acesso PHP-MySQL e PHP-Oracle

Esta seção tem como objetivo fazer uma comparação (Tabela 3.2) entre as funções de acesso ao SGBD MySQL usando a linguagem PHP estudadas e descritas na seção 3.1 e as funções de acesso ao SGBD Oracle usando a linguagem PHP identificadas e apresentadas na seção 3.2.

Funcionalidade	Funções de Acesso PHP – MySQL	Funções de Acesso PHP – Oracle
abrir conexão	mysql_connect( )	OCILogon() OCINLogon()
abrir conexão persistente	mysql_pconnect( )	OCIPLogon( )
selecionar o banco de dados	mysql_select_db( )	OCILogon() OCINLogon() OCIPLogon( )

interpretar uma consulta		OCI Parse( )
submeter uma consulta ao SGBD	mysql_query( )	OCIExecute( )
retornar valores de colunas trazidas por uma tupla		OCIResult( )
recuperar uma tupla do conjunto resultado	mysql_fetch_row( ) mysql_fetch_array( ) mysql_fetch_object( )	OCIFetchInto( ) OCIFetch( )
retornar o número de tuplas no conjunto resultado	mysql_num_rows( )	OCIRowCount( )
retornar o número de atributos (i.e., colunas) na tupla	mysql_num_fields( )	OCI NumCols( )
fechar uma conexão	mysql_close( )	OCI Logoff( )
liberar memória usada para armazenar o conjunto resultado	mysql_free_result( )	OCICancel( ) OCI FreeCursor( ) OCI FreeStatement( )
validar as declarações pendentes		OCI Commit( )
retornar os blocos de comando pendentes		OCI Rollback( )
retornar a descrição do último erro	mysql_error( )	OCI Error( )
retornar o número do último erro	mysql_errno( )	OCI Error( )
obter o metadado para um atributo da relação	mysql_fetch_field( )	OCI ColumnSize( ) OCI ColumnName( ) OCI ColumnType( )
alterar o usuário corrente para um novo usuário	mysql_change_user( )	
criar um banco de dados	mysql_create_db( )	

banco de dados		
remover um banco de dados	mysql_drop_db()	
identificar as tabelas do banco de dados	mysql_list_tables()	
retornar o nome de uma tabela	mysql_tablename()	
retornar a quantidade de linhas alteradas após a execução de um comando SQL	mysql_affected_rows()	OCIRowCount()
retornar o valor da chave primária da última tupla inserida no banco de dados	mysql_insert_id()	
recuperar algumas tuplas para uma consulta	mysql_data_seek()	
retornar o tipo de uma declaração OCI		OCIStatementType()

Tabela 3.2 Equivalência entre as funções de acesso PHP-MySQL e PHP-Oracle.

### 3.4 Considerações Finais

Este capítulo descreveu as funções de acesso ao SGBD MySQL usando a linguagem PHP. O capítulo também apresentou o resultado de uma pesquisa bibliográfica que teve como objetivo identificar as funções de acesso ao SGBD Oracle disponíveis usando a linguagem PHP. Todas as funções foram comparadas em termos de funcionalidades, resultado este ilustrado na Tabela 3.2.

Como pode ser observado na Tabela 3.2, a grande maioria das funções de acesso oferecidas pelo SGBD MySQL e pelo SGBD Oracle possuem funcionalidades equivalentes. Entretanto, existem algumas funções que encontram-se disponíveis para o acesso ao SGBD MySQL, mas que não encontram-se disponíveis para acesso ao SGBD Oracle. A proposta de desenvolvimento de funções equivalentes a estas para acesso ao SGBD Oracle é apresentada no Capítulo 4.

Além de identificar as funções de acesso ao SGBD Oracle disponíveis, este capítulo também contribui no sentido que permite que usuários comumente acostumados a utilizar o SGBD MySQL podem desenvolver aplicações de banco de dados utilizando o SGBD Oracle com base na tabela comparativa apresentada na seção 3.3.

## CAPÍTULO 4

### Proposta de Funções de Acesso ao SGBD Oracle usando a Linguagem PHP

Este capítulo tem como objetivo apresentar as funções de acesso ao SGBD Oracle usando a linguagem de implementação PHP propostas durante o decorrer deste projeto de especialização. O desenvolvimento destas funções enfocou o oferecimento de funcionalidades semelhantes às seguintes funções de acesso ao SGBD MySQL: `mysql_change_user( )`, `mysql_list_tables( )`, `mysql_tablename( )`, `mysql_data_seek( )`, `mysql_insert_id( )`, `mysql_create_db( )` e `mysql_drop_db( )`.

As seções 4.1, 4.2, 4.3, 4.4, 4.5 e 4.6 apresentam, respectivamente, a proposta das seguintes funções de acesso ao SGBD Oracle: `OCI_Change_User( )`, `OCI_List_Table( )`, `OCI_Data_Seek( )`, `OCI_Insert_ID( )`, `OCI_Create_Db( )` e `OCI_Drop_Db( )`. O capítulo é concluído na seção 4.7, com as considerações finais.

#### 4.1 Função `OCI_Change_User( )`

A função `OCI_Change_User( )` altera o usuário corrente para um novo usuário. Ela foi desenvolvida de forma a possuir funcionalidade semelhante à função `mysql_change_user( )`. Sua sintaxe é:

```
@OCI_Change_User(string user, string senha, resource conexao);
```

onde:

- a) `string user` - Nome do novo usuário;
- b) `string senha` - Nova senha para conexão; e
- c) `resource conexao` - Nome da variável que contém os dados (i.e., usuário e senha) da conexão atual.

Um exemplo de utilização da função proposta é ilustrado a seguir:

```
$user = "SYSTEM";  
$senha = "manager";  
@OCI_Change_User($user, $senha, $conexao);
```

A função `OCI_Change_User( )` foi desenvolvida da seguinte forma:

```

function OCI_Change_User($user, $senha, $conexao)
{
@OCILogOff($conexao);
@OCIFreeStatement($conexao);
$nova_conexao = @OCILogon($user,$senha);
if ($nova_conexao)
    echo "Conexão bem sucedida."."<p>";
else
    echo "Erro na conexão com o Oracle."."<p>";
}

```

## 4.2 Função OCI\_List\_Table( )

A função OCI\_List\_Table( ) lista as tabelas do banco de dados. Ela foi desenvolvida de forma a possuir funcionalidade semelhante às funções mysql\_list\_tables( ) e mysql\_tablename( ).

Sua sintaxe é:

```
OCI_List_Table (resource conexao)
```

onde:

c) resource conexao - Nome da variável que contém os dados (i.e., usuário e senha) da conexão atual.

Um exemplo de utilização da função proposta é ilustrado a seguir:

```
@OCI_List_Table($conexao);
```

A função OCI\_List\_Table( ) foi desenvolvida da seguinte forma:

```

function OCI_List_Table($conexao)
{
$sql = @OCIParse($conexao, "select table_name from user_tables");
@OCIExecute($sql, OCI_DEFAULT);
    while (@OCIFetch($sql)) {
        echo @OCIResult($sql, "TABLE_NAME") . "<br>"; }
}

```

### 4.3 Função OCI\_Data\_Seek( )

A função OCI\_Data\_Seek( ) recupera apenas algumas tuplas para uma consulta. Ela foi desenvolvida de forma a possuir funcionalidade semelhante à função mysql\_data\_seek( ). Sua sintaxe é:

```
OCI_Data_Seek(string sql, int linhas);
```

onde:

- a) string sql - Nome da variável que contém o comando SQL; e
- b) int linhas - Quantidade de linhas que serão recuperadas.

Um exemplo de utilização da função proposta é ilustrado a seguir:

```
@OCI_Data_Seek($sql, $linhas);
```

A função OCI\_Data\_Seek( ) foi desenvolvida da seguinte forma:

```
function OCI_Data_Seek($sql, $linhas)
{
    $colunas = @OCINumCols($sql);
    while (@OCIFetchInto($sql,$row,OCI_NUM) and ($linhas > 0))
    {
        $contador=1;
        while ($contador <= $colunas)
        {
            print $row[$contador]."<br>";
            $contador = $contador +1;
        }
        $linhas = $linhas - 1;
    }
}
```

### 4.4 Função OCI\_Insert\_Id( )

A função OCI\_Insert\_Id( ) retorna o valor da chave primária da última tupla inserida no banco de dados. Ela foi desenvolvida de forma a possuir funcionalidade semelhante à função mysql\_insert\_id( ). Sua sintaxe é:

```
OCI_Insert_Id(string nome_tabela, resource conexao);
```

onde:

- a) string nome\_tabela - Nome da tabela do banco de dados; e
- b) resource conexao - Nome da variável que contém os dados (i.e., usuário e senha) da conexão atual.

O exemplo a seguir ilustra o uso desta função:

```
@OCI_Insert_Id($nome_tabela, $conexao);
```

A função OCI\_Insert\_Id( ) foi desenvolvida da seguinte forma:

```
function OCI_Insert_Id($nome_tabela,$conexao)
{
    $sql = @OCIParse($conexao, "select * from ". $nome_tabela);
    @OCIExecute($sql, OCI_DEFAULT);
    $campo_chave = @OCIColumnName($sql,1);
    $max = @OCIParse($conexao, "select max(".$campo_chave.") as COD from
". $nome_tabela);
    @OCIExecute($max, OCI_DEFAULT);
    while (@OCIFetch($max)) {
        echo @OCIResult($max, "COD") . "<br>"; }
}
```

#### 4.5 Função OCI\_Create\_Db()

A função OCI\_Create\_Db( ) cria um banco de dados no SGBD Oracle. Ela foi desenvolvida de forma a possuir funcionalidade semelhante à função mysql\_create\_db( ). Sua sintaxe é:

```
OCI_Create_Db(string nome_banco, resource conexao);
```

onde:

- a) string nome\_banco - Nome do banco de dados; e
- b) resource conexao - Nome da variável que contém os dados (i.e., usuário e senha) da conexão atual.

A seguir é exemplificado o uso da função OCI\_Create\_Db( ):

```
@OCI_Create_Db($nome_banco, $conexao);
```

A função `OCI_Create_Db()` foi desenvolvida da seguinte forma:

```
function OCI_Create_Db($nome_banco, $conexao)
{
    $sql = @OCIParse($conexao, "Create database ". $nome_banco);
    @OCIExecute($sql, OCI_DEFAULT);
    echo "Banco de dados criado = ". $nome_banco."<p>";
}
```

## 4.6 Função `OCI_Drop_Db()`

A função `OCI_Drop_Db()` remove um banco de dados no SGBD Oracle. Ela foi desenvolvida de forma a possuir funcionalidade semelhante à função `mysql_drop_db()`. Sua sintaxe é:

```
OCI_Drop_Db(string nome_banco, resource conexao);
```

onde:

- a) `string nome_banco` - Nome do banco de dados; e
- b) `resource conexao` - Nome da variável que contém os dados (i.e., usuário e senha) da conexão atual.

A seguir é exemplificado o uso da função proposta:

```
@OCI_Drop_Db($nome_banco, $conexao);
```

A função `OCI_Drop_Db()` foi desenvolvida da seguinte forma:

```
function OCI_Drop_Db($nome_banco, $conexao)
{
    $sql = @OCIParse($conexao, "Drop database ". $nome_banco);
    @OCIExecute($sql, OCI_DEFAULT);
    echo "Banco de dados excluído = ". $nome_banco."<p>";
}
```

## 4.7 Comparação entre as Funções de Acesso PHP-MySQL e PHP-Oracle

### Propostas

Funcionalidade	Funções de Acesso PHP – MySQL	Funções de Acesso PHP – Oracle
alterar o usuário corrente para um novo usuário	<code>mysql_change_user( )</code>	<code>OCI_Change_User( )</code>
criar um banco de dados	<code>mysql_create_db( )</code>	<code>OCI_Create_DB( )</code>
remover um banco de dados	<code>mysql_drop_db( )</code>	<code>OCI_Drop_DB( )</code>
identificar as tabelas do banco de dados	<code>mysql_list_tables( )</code>	<code>OCI_List_Table( )</code>
retornar o nome de uma tabela	<code>mysql_tablename( )</code>	<code>OCI_List_Table( )</code>
retornar o valor da chave primária da última tupla inserida no banco de dados	<code>mysql_insert_id( )</code>	<code>OCI_Insert_Id( )</code>
recuperar algumas tuplas para uma consulta	<code>mysql_data_seek( )</code>	<code>OCI_Data_Seek( )</code>

Tabela 4.1 Equivalência entre as funções de acesso PHP-MySQL e PHP-Oracle propostas.

## 4.8 Considerações Finais

Este capítulo apresentou as seguintes funções de acesso ao SGBD Oracle usando a linguagem de programação PHP: `OCI_Change_User( )`, `OCI_List_Table( )`, `OCI_Data_Seek( )`, `OCI_Insert_ID( )`, `OCI_Create_Db( )` e `OCI_Drop_Db( )`. Essas funções foram desenvolvidas durante o decorrer deste projeto de especialização. Cada uma das funções foi apresentada em termos de sua sintaxe, de um exemplo de utilização e de sua implementação.

A principal contribuição deste capítulo é a proposta de funções de acesso ao SGBD Oracle que sejam equivalentes às funções de acesso ao SGBD MySQL já existentes. Assim, usuários comumente acostumados a utilizar este segundo SGBD encontram disponíveis funções similares para acesso ao SGBD Oracle, facilitando o desenvolvimento de aplicações de banco de dados que utilizem este SGBD para armazenar e manipular seus dados.

As funções de acesso descritas no Capítulo 3 e as funções de acesso propostas neste capítulo serão utilizadas no Capítulo 5 para a implementação de uma aplicação.

## CAPÍTULO 5

### Aplicação de Banco de Dados utilizando o SGBD Oracle e a Linguagem de Programação PHP

Este capítulo utiliza a aplicação da vinícola *winestore* descrita na seção 2.1 para aplicar as funções de acesso ao SGBD Oracle usando a linguagem PHP. Em Williams & Lane (2002) foram desenvolvidos *scripts* para a realização de consultas contra a base de dados dessa vinícola, porém considerando-se que os dados estavam armazenados no SGBD MySQL.

Este capítulo apresenta os *scripts* desenvolvidos durante esta monografia de especialização. Esses *scripts* são voltados à realização de consultas simples (seção 5.3) e à realização de consultas parametrizadas (seção 5.4) contra a base de dados da vinícola *winestore* armazenada no SGBD Oracle. Informações complementares são discutidas na seção 5.1, a qual destaca alguns detalhes adicionais relativos à descrição da aplicação. Já a seção 5.2 apresenta as telas principais da aplicação. A seção 5.5, por sua vez, conclui o capítulo.

#### 5.1 Aplicação Base

Na seção 2.1 foram apresentados os comandos SQL para a criação do banco de dados da vinícola *winestore*. Esses comandos foram submetidos ao SGBD Oracle, criando a base de dados a ser utilizada no desenvolvimento da aplicação descrita neste capítulo.

Em adição aos comandos descritos na seção 2.1, alguns comandos adicionais também foram executados. Antes da criação da base de dados *winestore*, foram executados comandos adicionais para a criação do usuário *ana* e para a atribuição de direitos a este usuário:

```
CREATE USER ANA IDENTIFIED BY ana;  
  
GRANT CONNECT TO ANA;  
GRANT DBA TO ANA;  
GRANT RESOURCE TO ANA;  
GRANT EXP_FULL_DATABASE, IMP_FULL_DATABASE TO ANA;
```

#### 5.2 Telas Principais da Aplicação

A Figura 5.1 ilustra a tela inicial da aplicação desenvolvida. Esta tela é exibida ao se executar a aplicação.

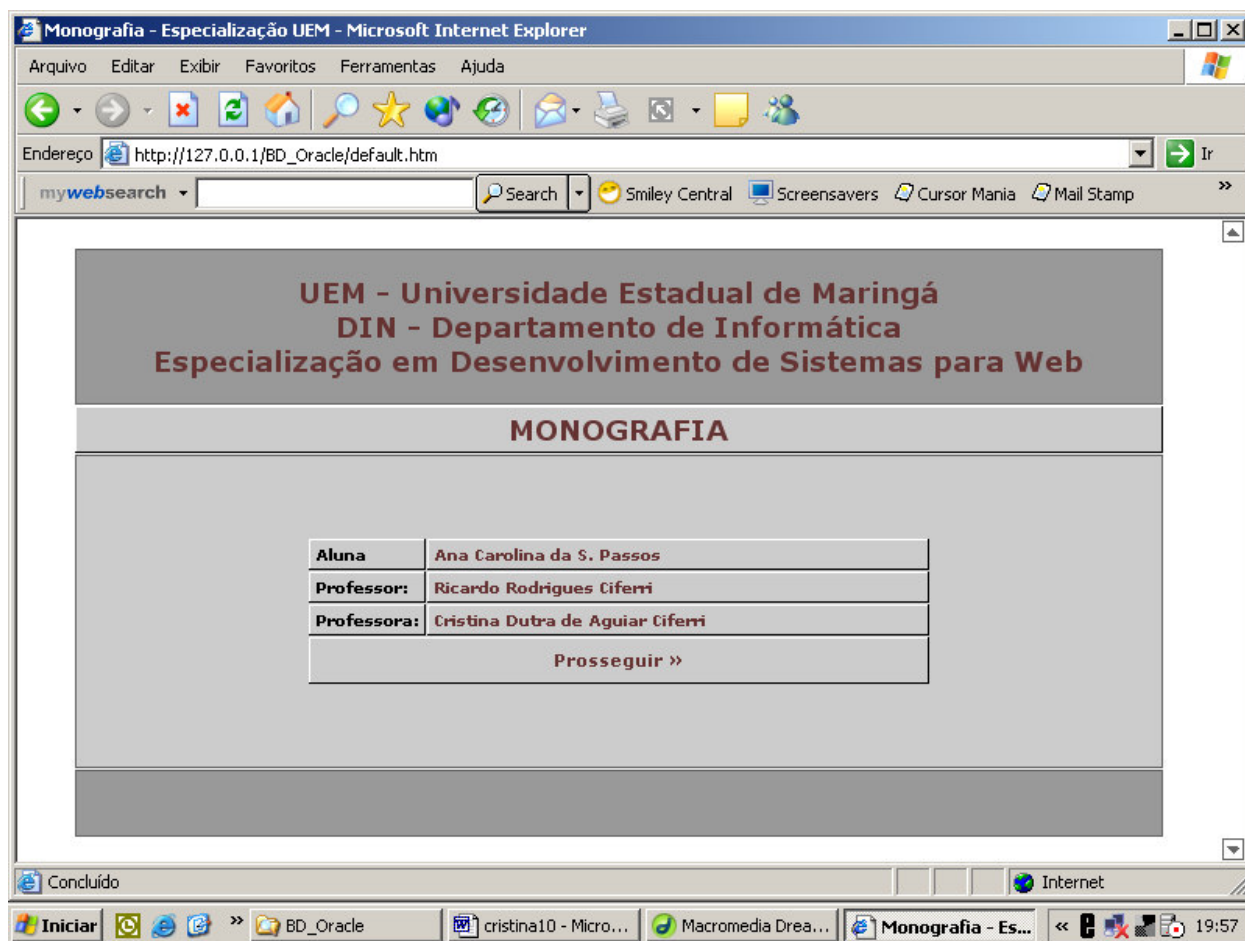


Figura 5.1 Tela inicial da aplicação.

A Figura 5.2 mostra uma tela intermediária do sistema. Nesta tela são exibidas duas opções. A Parte I consiste em um *link* para a tela de consultas. A Parte II, por sua vez, é um *link* para a tela de Inserção e Alteração. Apesar do *link* representado pela Parte II, os *scripts* de inserção e alteração dos dados não foram desenvolvidos como parte desta monografia. O desenvolvimento destes *scripts*, assim como outros para a remoção de dados são destacados como trabalhos futuros na conclusão.

Já a Figura 5.3 exibe quatro (4) opções para consulta ao SGBD:

- Consulta 01: utiliza apenas uma tabela, com parâmetros;
- Consulta 02: utiliza duas tabelas, com parâmetros;
- Consulta 03: utiliza apenas uma tabela, sem parâmetros; e
- Consulta 04: mostra todos os testes realizados com as funções Oracle.

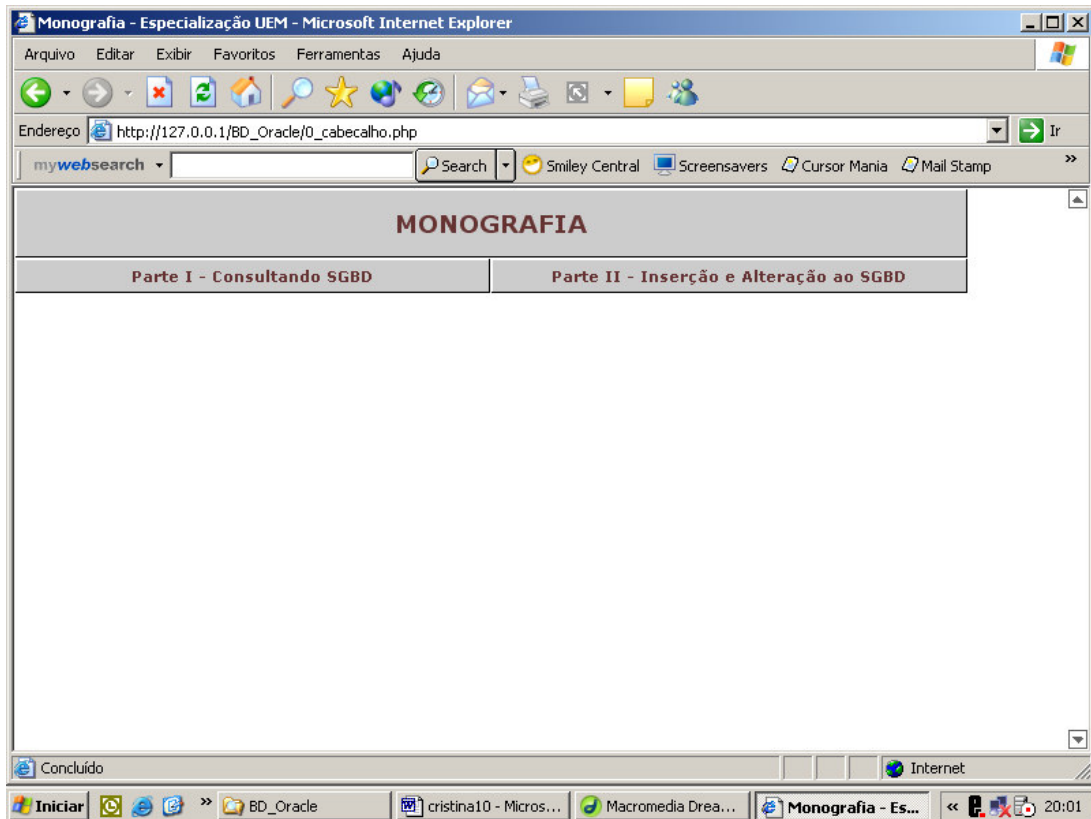


Figura 5.2 Tela intermediária da aplicação.

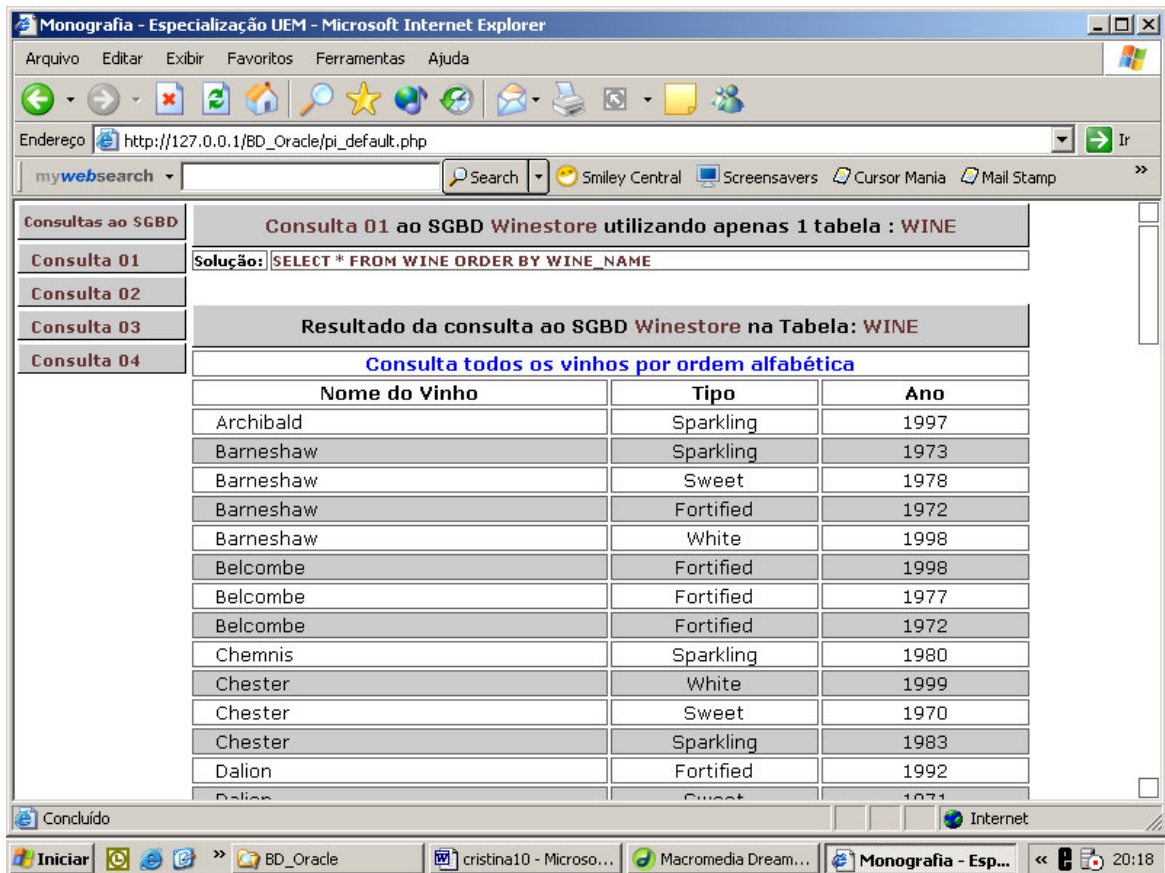


Figura 5.3 Opções de consulta disponíveis.

A Figura 5.4 exibe o resultado da Consulta 4, ou seja, apresenta os testes das funções PHP-Oracle.

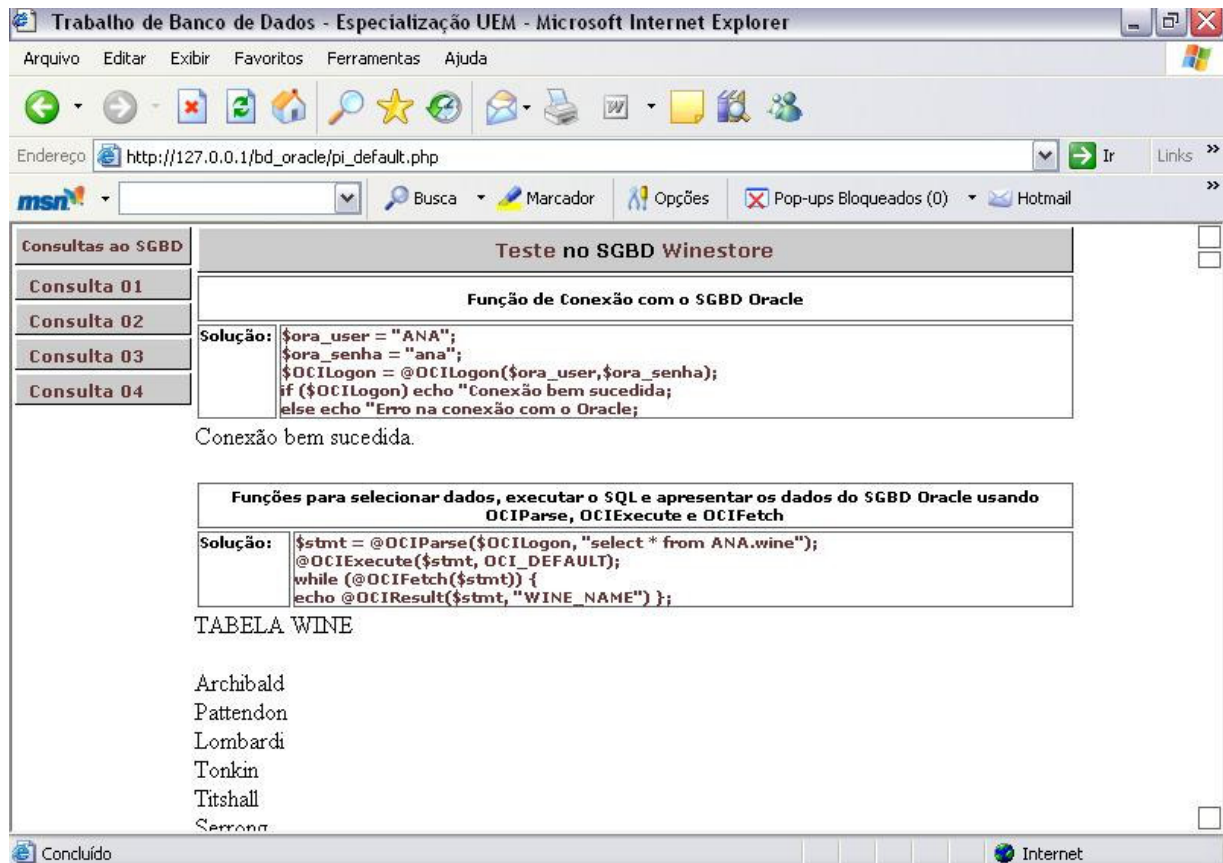


Figura 5.4 Tela de apresentação dos testes PHP-Oracle.

### 5.3 Script para Consulta Simples

Esta seção descreve um *script* para uma consulta simples. Esta consulta tem como objetivos pesquisar no banco de dados todos os vinhos por ordem alfabética utilizando apenas uma tabela sem parametrização e apresentar os resultados obtidos. A tela desenvolvida para a consulta está ilustrada na Figura 5.3.

```
<!--
PI_GERAL_BANCO.PHP
PI_GERAL_BANCO.PHP
Prof. Dr. Ricardo Rodrigues Ciferri e
    Dra. Cristina Dutra de Aguiar Ciferri
Aluna: Ana Carolina da Silva Passos
-->
<?
```

```

include "conexao.php"; /* ARQUIVO DE CONEXÃO COM O BANCO DE DADOS */
/* BLOCO DA CONSULTA À TABELA WINE (VINHO) */
$SQLselect="select * from wine order by wine_name";
$stmt = @OCIParse($OCILogon, $SQLselect);
@OCIExecute($stmt, OCI_DEFAULT);
$titulo = "Consulta todos os vinhos por ordem alfabética"
?>
<html>
<head>
<title>Monografia - Especialização UEM</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="config_links.css" rel="stylesheet" type="text/css">
</head>
<body topmargin="0" bottommargin="0" leftmargin="0" rightmargin="0">
<form name="formulario" action="pi_geral.php" method="post" onSubmit="return validab()">
  <table width="580" border="0" cellspacing="2" cellpadding="0">
    <tr>
      <td height="30" colspan="2" align="center" valign="middle" bgcolor="#CCCCCC"
class="botaof"><strong><font color="#663333" size="2" face="Verdana, Arial, Helvetica, sans-
serif">Consulta
      04</font><font color="#000000" size="2" face="Verdana, Arial, Helvetica, sans-serif">
      ao SGBD <font color="#663333">Winestore</font> utilizando apenas 1 tabela
      : <font color="#663333">WINE</font></font></strong></td>
    </tr>
    <tr>
      <td width="50" align="left" valign="top" class="borda"><font color="#000000" size="1"
face="Verdana, Arial, Helvetica, sans-
serif"><strong>Solú&ccedil;&atilde;o:</strong></font></td>
      <td width="530" align="left" valign="top" class="borda"><font color="#000000" size="1"
face="Verdana, Arial, Helvetica, sans-serif">
      <strong> <font color="#663333">SELECT * FROM WINE ORDER BY
WINE_NAME</font></strong></font></td>
    </tr>
  </table>

</form>
<table width="580" border="0" cellspacing="2" cellpadding="0">
  <tr>
    <td height="30" colspan="3" align="center" valign="middle" bgcolor="#CCCCCC"
class="botaof"><strong><font color="#000000" size="2" face="Verdana, Arial, Helvetica, sans-
serif">Resultado
    da consulta ao SGBD <font color="#663333">Winestore</font> na Tabela: <font
color="#663333">WINE</font></font></strong></td>
  </tr>
  <tr>
    <td colspan="3" align="center" valign="middle" class="borda"><strong><font
color="#0000FF" size="2" face="Verdana, Arial, Helvetica, sans-serif"><? echo $titulo;
?></font></strong></td>
  </tr>
  <tr align="center">

```



```

<!--
  PI_GERAL_1.PHP
  Prof. Dr. Ricardo Rodrigues Ciferri
    Dra. Cristina Dutra de Aguiar Ciferri e
  Aluna: Ana Carolina da Silva Passos
-->
<?
  include "conexao.php"; /* ARQUIVO DE CONEXÃO COM O BANCO DE DADOS */
  /* BLOCO PARA EXIBIR OS TITULOS CONFORME OS PARÂMENTROS DE
CONSULTA SUBMETIDAS */
  if ($fclausula == "winery_name"){ $titulo = "Consulta por vinícola";}
  if ($fclausula == "type"){ $titulo = "Consulta por tipo de vinho";}
  /* BLOCO DA CONSULTA À TABELA WINE (VINHO) - SQL=MONTAGEM DINÂMICA
  ATRAVÉS DA VARIÁVEL $fdesc e $fclausula */
  $fdesc = trim($fdesc);
  $SQLselect="select w.winery_id,w.winery_name,n.wine_id,n.type,n.wine_name from
winery w, wine n where $fclausula like '%$fdesc%' and w.winery_id = n.winery_id order
by $fclausula";
  $stmt = @OCIParse($OCILogon, $SQLselect);
  @OCIExecute($stmt, OCI_DEFAULT);
?>
<html>
<head>
<title>Monografia - Especialização UEM</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="config_links.css" rel="stylesheet" type="text/css">
<!-- BLOCO DE VALIDAÇÃO DA ENTRADAS DE DADOS DO LADO DO CLIENTE
UTILIZANDO JAVASCRIPT -->
<script language="JavaScript">
function validab ()
{
  if (document.formulario.fdesc.value == "")
  {
    alert ("Por favor, digite uma descrição e escolha um parâmetro (Vinícola ou Tipo));
    return(false);
  }
  return(true);
}
</script>
</head>

<body topmargin="0" bottommargin="0" leftmargin="0" rightmargin="0">
<form name="formulario" action="pi_geral_1.php" method="post" onSubmit="return
validab()">
  <table width="580" border="0" cellspacing="2" cellpadding="0">
    <tr>
      <td height="30" colspan="2" align="center" valign="middle" bgcolor="#CCCCCC"
class="borda"><strong><font color="#663333" size="2" face="Verdana, Arial, Helvetica, sans-
serif">Consulta

```

```

02</font><font color="#000000" size="2" face="Verdana, Arial, Helvetica, sans-serif">
ao SGBD <font color="#663333">Winestore</font> utilizando 2 tabelas: <font
color="#663333">WINE
e WINERY</font></font></strong></td>
</tr>
<tr>
<td width="50" align="left" valign="top" class="borda"><strong><font color="#000000"
size="1" face="Verdana, Arial, Helvetica, sans-
serif">Soluc&cedil;&atilde;o:</font></strong></td>
<td width="530" align="left" valign="middle" class="borda"><font color="#663333"
size="1" face="Verdana, Arial, Helvetica, sans-serif"><strong>
SELECT w.winery_id,w.winery_name,n.wine_id,n.type,n.wine_name <br>
FROM winery w, wine n <br>
WHERE $fclausula like '%$fdesc%' and w.winery_id = n.winery_id order by
$fclausula</strong></font></td>
</tr>
<tr>
<td height="30" colspan="2" align="center" valign="middle" class="borda"><strong><font
color="#000000" size="1" face="Verdana, Arial, Helvetica, sans-serif">Consulte
por
<input type="radio" name="fclausula" value="winery_name" onClick="javascript:
document.formulario.fdesc.value='Digite o nome da vin&iacute;cula'">
<font color="#0000FF">VIN&Iacute;COLA</font> ou por
<input type="radio" name="fclausula" value="type" onClick="javascript:
document.formulario.fdesc.value='Digite o tipo de vinho'">
<font color="#0000FF">TIPO DE VINHO</font>:
<input name="fdesc" type="text" class="input" id="fdesc" onClick="javascript:
document.formulario.fdesc.value=' ' style="width:200px">
<input name="Submit" type="submit" class="botao" value="OK">
</font></strong></td>
</tr>
<tr>
<td colspan="2">&nbsp;</td>
</tr>
</table>

</form>
<? if ($fdesc != "" && $fclausula != "") { ?>
<table width="580" border="0" cellspacing="2" cellpadding="0">
<tr>
<td height="30" colspan="3" align="center" valign="middle" bgcolor="#CCCCCC"
class="borda"><strong><font color="#000000" size="2" face="Verdana, Arial, Helvetica, sans-
serif">Resultado
da consulta ao SGBD <font color="#663333">Winestore</font> nas Tabelas:
<font color="#663333">WINE e WINERY</font> </font></strong></td>
</tr>
<tr>
<td colspan="3" align="center" valign="middle" class="borda"><strong><font
color="#0000FF" size="2" face="Verdana, Arial, Helvetica, sans-serif">
<? echo $titulo; ?>

```



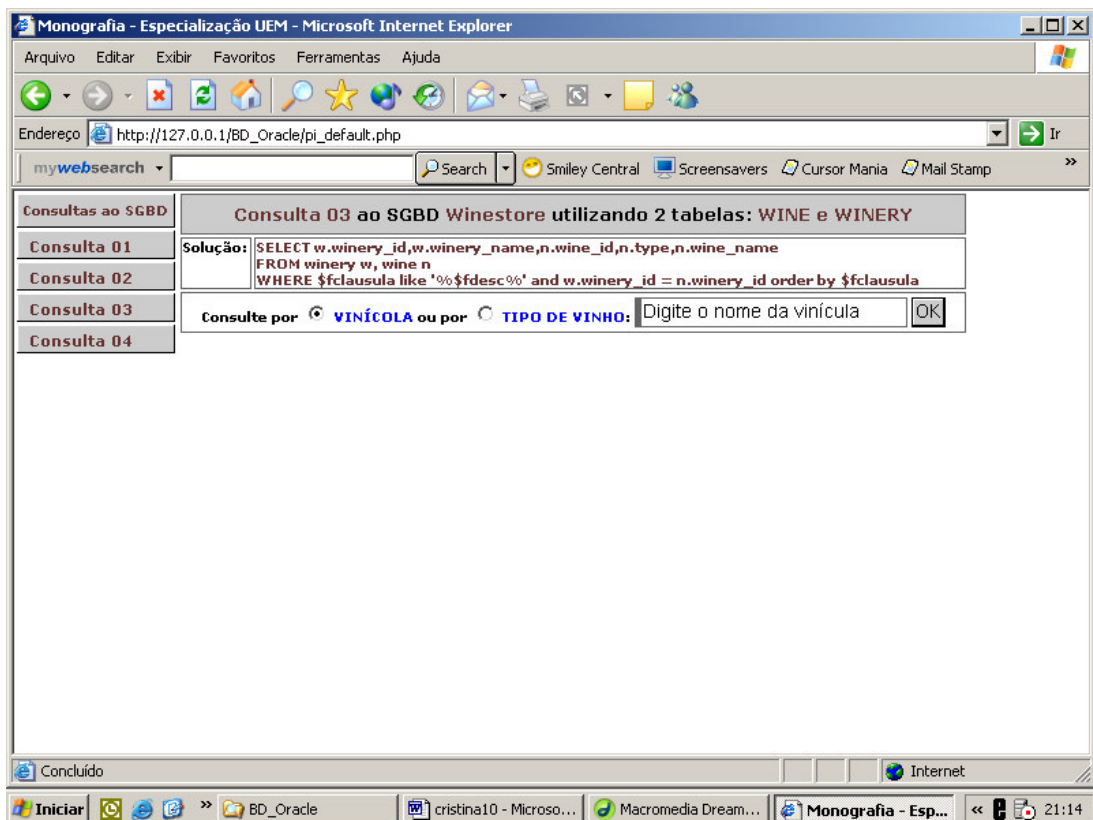


Figura 5.5 Tela para consulta parametrizada às tabelas wine e winery.

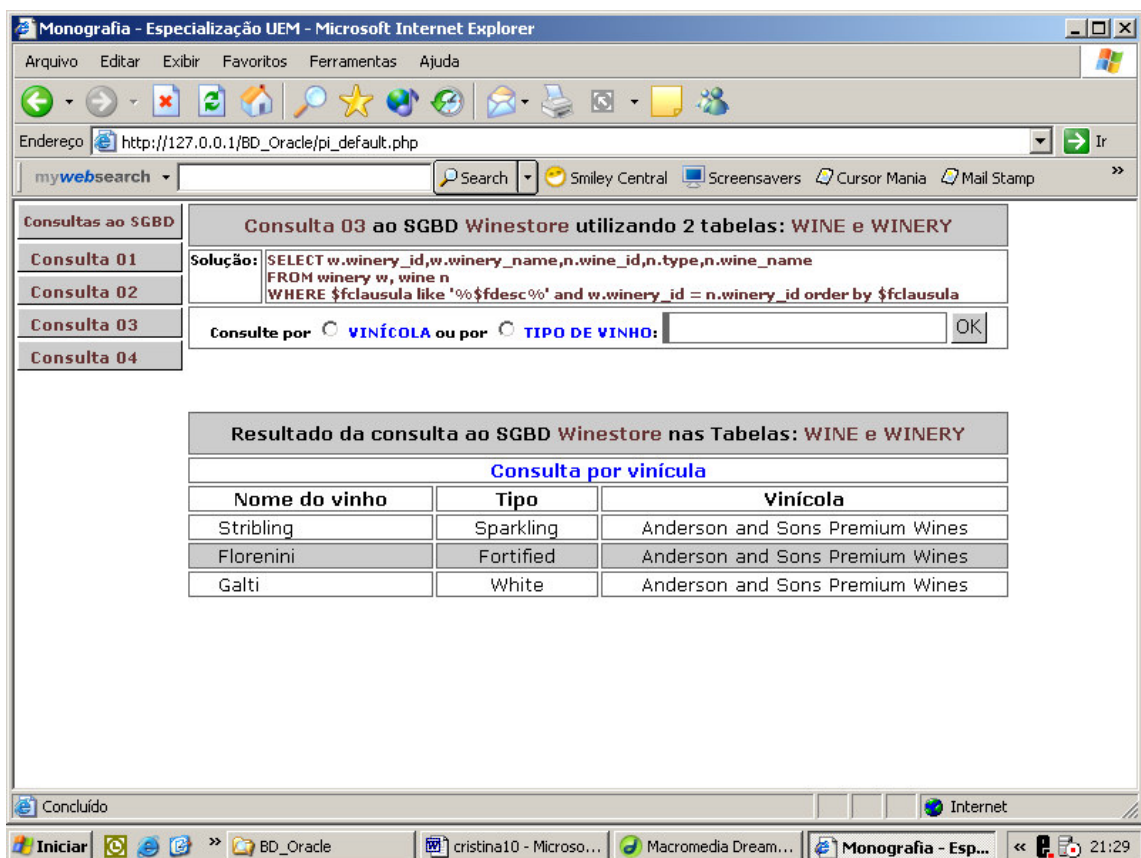


Figura 5.6 Tela com os resultados obtidos na consulta parametrizada.

## 5.5 Considerações Finais

Este capítulo apresentou uma aplicação de banco de dados utilizando o SGBD Oracle e a linguagem de programação PHP. A aplicação utilizou como fundamentação a base de dados *winestore*, proposta por Williams & Lane (2002). Para o desenvolvimento da aplicação, esta base de dados foi criada e povoada no SGBD Oracle.

A aplicação desenvolvida foi descrita em termos de suas principais telas e de dois *scripts*: um voltado para consulta simples e outro voltado para consulta parametrizada. Para o desenvolvimento da aplicação não houve grandes problemas na utilização das funções de acesso ao SGBD Oracle usando a linguagem PHP. A investigação da sintaxe dessas funções e a realização de testes durante a identificação destas funções facilitaram o desenvolvimento da aplicação apresentada neste capítulo.

## CAPÍTULO 6

### Conclusão

Esta monografia de especialização teve como objetivo investigar quais funcionalidades adicionais são oferecidas pelo SGBD Oracle com relação ao SGBD MySQL e como o acesso ao SGBD Oracle pode ser realizado por meio da linguagem PHP. Por um lado, a investigação das funcionalidades oferecidas pelo SGBD Oracle e pelo SGBD MySQL permitiu a realização de um estudo comparativo entre esses dois SGBD. Baseado nos resultados obtidos, desenvolvedores de aplicações de banco de dados para a Web podem escolher qual o melhor SGBD a ser utilizado de acordo com as necessidades de suas aplicações. Por outro lado, a investigação de como o acesso ao SGBD Oracle pode ser realizado por meio da linguagem PHP originou uma lista de funções equivalentes de acesso PHP–MySQL e PHP–Oracle, além da proposta de novas funções de acesso ao SGBD Oracle. Conseqüentemente, desenvolvedores de aplicações de banco de dados para a Web usando o SGBD Oracle são capazes de acessar, pesquisar e manipular os dados armazenados nesse SGBD por meio de funções PHP de fácil utilização.

Mais detalhadamente, as contribuições do trabalho desenvolvido são:

- identificação das características que um SGBD deve oferecer com relação às linguagens de definição e de manipulação dos dados;
- identificação de quais dessas características são disponíveis no SGBD MySQL;
- identificação de quais dessas características são disponíveis no SGBD Oracle;
- investigação das funções de acesso ao SGBD MySQL usando a linguagem PHP;
- investigação das funções de acesso ao SGBD Oracle usando a linguagem PHP;
- proposta de funções de acesso ao SGBD Oracle usando a linguagem PHP, adicionais às identificadas no item anterior, de forma que estas funções possuam funcionalidades equivalentes às oferecidas para o acesso ao SGBD MySQL; e
- desenvolvimento de uma aplicação de banco de dados para a Web usando a linguagem de programação PHP e o SGBD Oracle, com destaque para a descrição de *scripts* voltados à realização de consultas simples e à realização de consultas parametrizadas.

Trabalhos futuros relacionados a esta monografia de especialização incluem o desenvolvimento completo da aplicação descrita no Capítulo 5, com destaque para funcionalidades adicionais tais como rotinas para a inserção, a remoção e a atualização dos dados da base `winestore`, com enfoque no gerenciamento de transações e na recuperação de falhas.

Ainda com relação à aplicação, outra atividade refere-se à verificação e validação de dados de entrada, tanto do lado cliente usando-se a linguagem JavaScript, quanto do lado servidor usando-se a linguagem PHP. Outros trabalhos futuros consistem na comparação das funcionalidades e das funções de acesso oferecidas por outros SGBD e considerando-se diferentes linguagens de programação.

## REFERÊNCIAS BIBLIOGRÁFICAS

BEDIN, R. *Desenvolvimento de uma Aplicação de Banco de Dados para a Web usando PHP e MySQL: Enfoque Consultas e Operações de Atualização*. Trabalho de Final de Curso. Departamento de Informática (DIN), Universidade Estadual de Maringá (UEM), Maringá, PR, BR, 2004.

CAMPOS, LUÍS MORENO. *Oracle 8i*, FCA Editora, 2ª edição, 1999.

CONNOLLY, T.M., BEGG, C.E. *Database Systems – A Practical Approach to Design, Implementation, and Management*, 3<sup>rd</sup> edition. Addison-Wesley, 2002.

CONVERSE, T., PARK, J., *PHP 4: A Bíblia*, Campus, 2001.

DATE, C.J. *Introduction to Database Systems*, 8<sup>th</sup> edition. Addison-Wesley, 2003.

ELMASRI, R., NAVATHE, S. *Fundamentals of Database Systems*, 4<sup>th</sup> edition. Addison-Wesley, 2003.

FANDERUFF, DAMARIS. *Oracle8i utilizando SQL\*PLUS e PL/SQL*, editora Makron Books, 2000.

FERNANDES, L. *Oracle 9i: para Desenvolvedores Oracle Developer 6i – Curso Completo*. Axcel Books, 2002.

LOLIS, G. V. *Segurança em Aplicações de Banco de Dados para Web*. Trabalho de Graduação. Departamento de Informática (DIN), Universidade Estadual de Maringá (UEM), Maringá, PR, BR, 2003.

MILOCA, K. C. *Desenvolvimento de uma Aplicação de Banco de Dados para a Web usando PHP e MySQL: Enfoque Consultas e Operações de Inserção e Remoção*. Trabalho de Final de

Curso. Departamento de Informática (DIN), Universidade Estadual de Maringá (UEM), Maringá, PR, BR, 2004.

PHP. *Manual do PHP*. Disponível na URL <http://php.planetmirror.com/manual>. Última visita em 24/07/2005.

RAMAKRISHNAN, R.; GEHRKE, J. *Database Management Systems*, McGraw Hill, 2003.

SILBERSCHATZ, A., KORTH, H. F., SUDARSHAN, S., *Sistema de Banco de Dados*, MAKRON Books, 1999.

WELLING, L., THOMSON, L. *PHP and MySQL Web Development*, 2<sup>nd</sup> edition. Sams Publishing, 2003.

WILLIAMS, H. E.; LANE, D. *Web Database Applications with PHP and MySQL*. O'Reilly & Associates, 2002.