Dirección Xeral de Educación, Formación Profesional e Innovación Educativa

Material para

a formación profesional inicial

A01. Uso de bases de datos obxecto-relacionais.

|  |  |  |
| --- | --- | --- |
| Familia profesional | IFC | Informática e comunicacións |
| Ciclo formativo | CSIFC03  CSIFC02 | Desenvolvemento de aplicacións web  Desenvolvemento de aplicacións multiplataforma |
| Grao |  | Superior |
| Módulo profesional | MP0484 | Bases de datos |
| Unidade didáctica | UD09 | Bases de datos obxecto-relacionais |
| **Actividade** | **A01** | **Uso de bases de datos obxecto-relacionais** |
| Autores |  | Marta Fernández García  María del Carmen Fernández Lameiro  Miguel Fraga Vila  María Carmen Pato González  Andrés del Río Rodríguez |
| Nome do arquivo |  | CSIFC02\_MP0484\_V000901\_UD09\_A01\_Uso\_de\_BDs\_Obx-relacs.docx |
| © 2015 Xunta de Galicia.  Consellería de Cultura, Educación e Ordenación Universitaria.  Este traballo foi realizado durante unha licenza de formación retribuída pola Consellería de Cultura, Educación e Ordenación Universitaria e ten licenza CreativeCommons BY-NC-SA (recoñecemento - non comercial - compartir igual). Para ver unha copia desta licenza, visitar a ligazón http://creativecommons.org/licenses/by-nc-sa/3.0/es/. | | |

Índice

[1. Ficha técnica 5](#_Toc435722804)

[Contexto da actividade 5](#_Toc435722805)

[Título da actividade 6](#_Toc435722806)

[Resultados de aprendizaxe do currículo 6](#_Toc435722807)

[Obxectivos didácticos e título e descrición da actividade 6](#_Toc435722808)

[Criterios de avaliación 6](#_Toc435722809)

[Contidos 6](#_Toc435722810)

[Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación 8](#_Toc435722811)

[2. A01. Uso de bases de datos obxecto-relacionais 10](#_Toc435722812)

[2.1 Introdución 10](#_Toc435722813)

[2.2 Actividade 10](#_Toc435722814)

[2.2.1 Características das bases de datos obxecto-relacionais 10](#_Toc435722815)

[2.2.1.1 Tecnoloxía BDOR de Oracle 10](#_Toc435722816)

[2.2.2 Tipo de datos obxecto 11](#_Toc435722817)

[2.2.2.1 Declaración de atributos 11](#_Toc435722818)

[2.2.2.2 Definición de métodos 12](#_Toc435722819)

[2.2.2.3 O parámetro SELF 13](#_Toc435722820)

[2.2.2.4 Sobrecarga 14](#_Toc435722821)

[2.2.2.5 Métodos construtores 14](#_Toc435722822)

[2.2.3 Utilización de obxectos 15](#_Toc435722823)

[2.2.3.1 Declaración de obxectos 15](#_Toc435722824)

[2.2.3.2 Inicialización de obxectos 16](#_Toc435722825)

[2.2.3.3 Acceso aos atributos de obxectos 16](#_Toc435722826)

[2.2.3.4 Chamada aos métodos dos obxectos 17](#_Toc435722827)

[2.2.3.5 Herdanza 17](#_Toc435722828)

[2.2.4 Métodos MAP e ORDER 18](#_Toc435722829)

[2.2.4.1 Métodos ORDER 19](#_Toc435722830)

[2.2.5 Tipos de datos colección 19](#_Toc435722831)

[2.2.5.1 Declaración e uso de coleccións. 20](#_Toc435722832)

[2.2.6 Táboas de obxectos 21](#_Toc435722833)

[2.2.6.1 Táboas con columnas tipo obxecto 22](#_Toc435722834)

[2.2.6.2 Uso da sentenza Select 22](#_Toc435722835)

[2.2.6.3 Inserción de obxectos 23](#_Toc435722836)

[2.2.6.4 Modificación de obxectos 23](#_Toc435722837)

[2.2.6.5 Borrado de obxectos 24](#_Toc435722838)

[2.2.6.6 Consultas coa función VALUE 25](#_Toc435722839)

[2.2.6.7 Referencias a obxectos. 26](#_Toc435722840)

[2.2.6.8 Navegación a través de referencias 27](#_Toc435722841)

[2.3 Tarefas 27](#_Toc435722842)

[2.3.1 Tarefa 1. Instalación e configuración do contorno de traballo 27](#_Toc435722843)

[Solución 28](#_Toc435722844)

[Descarga e instalación 28](#_Toc435722845)

[Configuración do contorno de traballo 35](#_Toc435722846)

[2.3.2 Tarefa 2. Creación de tipos de dato obxecto 38](#_Toc435722847)

[Solución 38](#_Toc435722848)

[2.3.3 Tarefa 3. Creación de tipos de datos colección 39](#_Toc435722849)

[Solución 39](#_Toc435722850)

[2.3.4 Tarefa 4. Creación de táboas de obxectos e táboas de columnas tipo obxecto 40](#_Toc435722851)

[Apartado a 40](#_Toc435722852)

[Apartado b 40](#_Toc435722853)

[Solución 40](#_Toc435722854)

[Apartado a 40](#_Toc435722855)

[Apartado b 40](#_Toc435722856)

[2.3.5 Tarefa 5. Modificación de obxectos almacenados na base de datos 40](#_Toc435722857)

[Apartado a 40](#_Toc435722858)

[Apartado b 40](#_Toc435722859)

[Solución 41](#_Toc435722860)

[Apartado a 41](#_Toc435722861)

[Apartado b 41](#_Toc435722862)

[2.3.6 Tarefa 6. Consultas sobre táboas de obxectos e táboas de columnas tipo obxecto 41](#_Toc435722863)

[Apartado a 41](#_Toc435722864)

[Apartado b 41](#_Toc435722865)

[Solución 41](#_Toc435722866)

[Apartado a 41](#_Toc435722867)

[Apartado b 41](#_Toc435722868)

[3. Materiais 42](#_Toc435722869)

[3.1 Documentos de apoio ou referencia 42](#_Toc435722870)

[3.2 Recursos didácticos 42](#_Toc435722871)

[4. Avaliación 43](#_Toc435722872)

[4.1 Criterios de avaliación 43](#_Toc435722873)

[4.2 Modelos de probas 43](#_Toc435722874)

[Modelo de proba combinada para CA9.1, CA9.2, CA9.3, CA9.4, CA9.5 e CA9.6 43](#_Toc435722875)

[Texto da proba 44](#_Toc435722876)

[Parte práctica 44](#_Toc435722877)

[Parte teórica 44](#_Toc435722878)

[Exemplo de solucións 45](#_Toc435722879)

[Apartado a 45](#_Toc435722880)

[Apartado b 45](#_Toc435722881)

[Apartado c 45](#_Toc435722882)

[Apartado d 45](#_Toc435722883)

[Apartado e 45](#_Toc435722884)

[Apartado f 45](#_Toc435722885)

[Apartado g 46](#_Toc435722886)

[Exemplo de listas de cotexo LC.1, LC.2, LC.3, LC.4 e LC.5 46](#_Toc435722887)

[Exemplo de lista de cotexo para CA9.2 46](#_Toc435722888)

[Exemplo de lista de cotexo para CA9.3 46](#_Toc435722889)

[Exemplo de lista de cotexo para CA9.4 47](#_Toc435722890)

[Exemplo de lista de cotexo para CA9.5 47](#_Toc435722891)

[Exemplo de lista de cotexo para CA9.6 47](#_Toc435722892)

1. Ficha técnica

Contexto da actividade

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Módulo | Duración  horas | Unidade didáctica. | Sesións 50´ | Actividades | Sesións 50´ |
| MP0484. Bases de datos | 187 | UD01. Bases de datos e sistemas de almacenamento da información | 12 | A01. Bases de datos e sistemas de almacenamento da información | 7 |
| A02. Introdución aos sistemas xestores de bases de datos | 5 |
| UD02. Deseño conceptual de bases de datos | 38 | A01. Fases do deseño de bases de datos | 3 |
| A02.Descrición e representación gráfica dos elementos do Modelo Entidade-Interrelación (MER) | 10 |
| A03. Descrición e representación gráfica dos elementos do Modelo Entidade-Interrelación Estendido(MERE) | 9 |
| A04. Construción e validación do modelo de datos | 11 |
| A05. Notación Martin e ferramentas CASE | 5 |
| UD03. Deseño lóxico de base de datos | 32 | A01. O deseño lóxico empregando o modelo relacional | 15 |
| A02. Normalización de relacións | 10 |
| A03. Operacións do MR: álxebra e cálculo | 7 |
| UD04. Deseño físico de bases de datos | 28 | A01. Creación da estrutura de bases de datos relacionais | 13 |
| A02. Modificación da estrutura de bases de datos relacionais | 9 |
| A03. Verificación e probas | 3 |
| A04. Utilización de ferramentas gráficas para a creación da estrutura de bases de datos relacionais | 3 |
| UD05. Consultas de datos | 39 | A01. Consultas simples cunha táboa. | 12 |
| A02. Consultas con datos de máis dunha táboa | 10 |
| A03. Consultas resumo con agrupamentos | 7 |
| A04. Consultas con subconsultas | 7 |
| A05. Optimización de consultas | 3 |
| UD06. Tratamento de datos | 24 | A01. Manipulación de datos | 12 |
| A02. Vistas | 5 |
| A03. Transaccións e bloqueos | 7 |
| UD07. Programación de bases de datos | 22 | A01. Introdución á programación con SQL | 4 |
| A02. Rutinas almacenadas | 9 |
| A03. Disparadores e eventos | 5 |
| A04. Cursores | 4 |
| UD08. Administración de bases de datos | 19 | A01. Copias de seguridade e intercambio de datos entre SXBDs | 10 |
| A02. Índices e xestión de usuarios | 9 |
| **UD09. Bases de datos obxecto-relacionais** | **10** | **A01. Uso de bases de datos obxecto-relacionais** | **10** |

NOTA: Esta actividade está vinculada á programación recollida no arquivo CSIFC02\_MP0484\_V000900\_ UD09\_BDs\_Obxecto\_relacionais.pdf

Título da actividade

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Título | Descrición | Duración |
| A01 | Uso de bases de datos obxecto-relacionais | Describiranse as características das bases de datos obxecto-relacionais e aplicaranse técnicas para o manexo básico das mesmas | 10 |

Resultados de aprendizaxe do currículo

|  |  |
| --- | --- |
| Resultados de aprendizaxe do currículo | Completo |
| * RA9 - Xestiona a información almacenada en bases de datos obxecto-relacionais, para o que utiliza e avalía as posibilidades que proporciona o sistema xestor. | Si |

Obxectivos didácticos e título e descrición da actividade

| Obxectivos específicos | | Actividade | | Descrición básica | Duración |
| --- | --- | --- | --- | --- | --- |
| O1.1 | Identificar as características das bases de datos obxecto-relacionais | A01 | Uso de bases de datos obxecto-relacionais | Describiranse as características das bases de datos obxecto-relacionais e aplicaranse técnicas para o manexo básico das mesmas | 10 |
| O1.2 | Crear estruturas para o almacenamento de datos en bases de datos obxecto-relacionais (tipos de datos obxecto, tipos de datos colección, táboas de obxectos e táboas de columnas tipo obxecto) |
| O1.3 | Consultar e modificar a información almacenada en bases de datos obxecto-relacionais |

Criterios de avaliación

|  |
| --- |
| Criterios de avaliación |
| * CA9.1 Identificáronse as características das bases de datos obxecto-relacionais. * CA9.2 Creáronse tipos de datos obxecto, os seus atributos e os seus métodos. * CA9.3 Creáronse táboas de obxectos e táboas de columnas tipo obxecto. * CA9.4 Creáronse tipos de datos colección. * CA9.5 Realizáronse consultas. * CA9.6 Modificouse a información almacenada mantendo a integridade e a consistencia dos datos. |

Contidos

| Contidos |
| --- |
| * Características das bases de datos obxecto-relacionais. * Tipos de datos colección. * Declaración e iniciación de obxectos. * Uso da sentenza SELECT. * Navegación a través de referencias. * Chamadas a métodos. * Inserción, modificación e borrado de obxectos. * Tipos de datos obxecto. * Atributos e métodos. * Sobrecarga. * Construtores. * Definición de tipos de obxecto. * Definición de métodos. * Herdanza. * Identificadores e referencias. |

Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación

| Que e para que | Como | | | Con que | Como e con que se valora | Duración (sesións) |
| --- | --- | --- | --- | --- | --- | --- |
| Actividade (título e descrición) | Profesorado (en termos de tarefas) | Alumnado (tarefas) | Resultados  ou produtos | Recursos | Instrumentos e procedementos de avaliación |
| A01. Uso de bases de datos obxecto-relacionais   * Describiranse as características das bases de datos obxecto-relacionais e aplicaranse técnicas para o manexo básico das mesmas. | * **Tp1.1** - Exposición sobre as características das bases de datos obxecto-relacionais. | * **Ta1.1** - Toma de notas e consulta de dúbidas. |  | * Proxector. * Material didáctico subministrado polo profesorado. * Ordenadores con sistemas operativos Linux e Windows e conexión a Internet. |  | 1 |
| * **Tp1.2** - Exposición e demostración práctica de sentenzas de creación de tipos de datos obxecto, os seus atributos e os seus métodos. | * **Ta1.2** - Instalación e configuración do contornos de traballo * **Ta1.3**.-.Realización de exercicios de sentenzas de creación de tipos de datos obxecto, atributos e métodos. | * Contorno de traballo instalado e configurado. * Exercicios resoltos. |  | 3 |
| * **Tp1.3** - Exposición e demostración práctica de sentenzas de creación de táboas de obxectos e táboas de columnas tipo obxecto. | * **Ta1.4** - Realización de exercicios de sentenzas de creación de táboas de obxectos e táboas de columnas tipo obxecto. | * Exercicios resoltos. |  | 2 |
| * **Tp1.4** - Exposición e demostración práctica de sentenzas de creación de tipos de datos colección. | * **Ta1.5** - Realización de exercicios de sentenzas de creación de tipos de datos colección. | * Exercicios resoltos. |  | 1 |
| * **Tp1.5** - Exposición e demostración práctica de sentenzas de consulta de información e sentenzas de modificación da información almacenada en táboas de obxectos e táboas de columnas tipo obxecto. | * **Ta1.6** - Realización de exercicios de sentenzas consulta de información e sentenzas de modificación da información almacenada en táboas de obxectos e táboas de columnas tipo obxecto. | * Exercicios resoltos. |  | 2 |
|  | * **Ta1.7 -** Tarefa de avaliación combinada con PE.1, LC.1, LC.2, LC.3, LC.4 e LC.5 | * Exame |  | * **PE.1**- Cuestionario de preguntas cortas sobre as características das bases de datos obxecto-relacionais. * **LC.1.-** Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de tipos de datos obxecto, atributos e métodos. * **LC.2**.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de táboas de obxectos e táboas de columnas tipo obxecto. * **LC.3**- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de tipos de datos colección. * **LC.4**- Documento de rexistro de proba práctica consistente na elaboración de sentenzas para a consulta de información alma-cenada en táboas de obxectos e táboas de columnas tipo obxecto. * **LC.5**- Documento de rexistro de proba práctica consistente na elaboración de sentenzas para modificación da información al-macenada en táboas de obxectos e táboas de columnas tipo obxecto. | 1 |

1. A01. Uso de bases de datos obxecto-relacionais
   1. Introdución

Na actividade que nos ocupa farase unha descrición das características principais das bases de datos obxecto-relacionais e aprenderase a crear e manexar as estruturas básicas para o almacenamento de datos neste tipo de bases de datos. Así mesmo, aprenderase a consultar e modificar os datos almacenados en ditas estruturas.

* 1. Actividade
     1. Características das bases de datos obxecto-relacionais

O térmo Base de Datos Obxecto Relacional (BDOR) úsase para describir unha base de datos que evolucionou desde o modelo relacional cara a outra máis ampla que incorpora conceptos do paradigma orientado a obxectos. Xa que logo, un Sistema de Xestión Obxecto-Relacional (SXBDOR) contén ambas tecnoloxías: relacional e de obxectos.

Unha idea básica das BDOR é que o usuario poida crear os seus propios tipos de datos para ser utilizados naquelas tecnoloxías que permitan a creación de tipos de datos predefinidos. Ademais, as BDOR permiten crear métodos para eses tipos de datos. Con iso, este tipo de SXBD fai posible a creación de funcións membro usando tipos de datos definidos polo usuario, o que proporciona flexibilidade e seguridade.

Os SXBDOR permiten importantes melloras en moitos aspectos con respecto ós SXBDR tradicionais. Estes sistemas xestionan tipos de datos complexos cun esforzo mínimo e albergan parte da aplicación no servidor de base de datos. Permiten almacenar datos complexos dunha aplicación dentro da BDOR sen necesidade de forzar os tipos de datos tradicionais. Son compatibles en sentido ascendente coas bases de datos relacionais tradicionais, tan familiares a multitude de usuarios. É dicir, pódense pasar as aplicacións sobre bases de datos relacionais ao novo modelo sen ter que reescribilas. Posteriormente, si así se quixera, poderíanse ir adaptando as aplicacións e bases de datos para que utilicen as funcións orientadas a obxectos.

* + - 1. Tecnoloxía BDOR de Oracle

Para ver con máis detalle o funcionamento e as características dunha BDOR utilizarase Oracle como SXBDR.

Debido aos requirimentos das novas aplicacións, o sistema de xestión de bases de datos relacionais de Oracle, desde a versión 8.i, foi estendido con conceptos do modelo de bases de datos orientadas a obxectos. Deste xeito, aínda que as estruturas de datos que se utilizan para almacenar a información seguen sendo táboas, os usuarios poden utilizar moitos dos mecanismos de orientación a obxectos para definir e acceder aos datos.

Oracle proporciona mecanismos para que o usuario poida definir os seus propios tipos de datos, cuxa estrutura pode ser complexa, e permítese a asignación dun tipo complexo (dominio complexo) a unha columna dunha táboa. Ademais, recoñécese o concepto de obxectos, de tal xeito que un obxecto ten un tipo, almacénase en certa fila de certa táboa e ten un identificador único (OID). Estes identificadores pódense utilizar para referenciar a outros obxectos e así representar relacións de asociación e de agregación.

Oracle tamén proporciona mecanismos para asociar métodos a tipos, e construtores para deseñar tipos de datos multivaluados (coleccións) e táboas aniñadas.

* + 1. Tipo de datos obxecto

Un tipo de dato define unha estrutura e un comportamento común para un conxunto de datos. Para o traballo con BDOR, Oracle permite definir os seus propios tipos de datos mediante dúas categorías: tipo de datos obxecto (object type) e tipo de datos colección (collection type).

Un tipo de obxecto representa unha entidade do mundo real e componse dos seguintes elementos:

* **Nome**, que serve para identificar o tipo dos obxectos.
* **Atributos**, que modelan a estrutura e os valores dos datos dese tipo. Cada atributo pode ser dun tipo de datos básico ou dun tipo definido polo usuario.
* **Métodos** (procedementos ou funcións) escritos en linguaxe PL/SQL (almacenados na BDOR), ou escritos en C (almacenados externamente).

Os tipos de obxectos actúan como padróns para os obxectos de cada tipo.

Para definir un obxecto en Oracle debese utilizar a sentenza CREATE TYPE que ten o seguinte formato:

CREATE TYPE nome\_tipo AS OBJECT ( Declaración\_atributos Declaración\_métodos );

Sendo *nome\_tipo* o nome desexado para o novo tipo de obxecto. A forma de declarar os atributos e os métodos verase nos seguintes apartados.

No caso de que o nome do tipo de obxecto xa estivese sendo usado para outro tipo de obxecto obteríase un erro. Si se desexa trocar o tipo anteriormente creado polo novo que se vai a declarar, pódese engadir a cláusula OR REPLACE na declaración do tipo de obxecto:

CREATE OR REPLACE TYPE nome\_tipo AS OBJECT

Por exemplo, para crear o tipo de obxecto *Usuario*, trocando a declaración que tivese anteriormente poderíase facer algo similar ao seguinte:

CREATE OR REPLACE TYPE Usuario AS OBJECT ( Declaración\_atributos Declaración\_métodos );

Se nalgún momento se desexa eliminar o tipo de obxecto creado pódese utilizar a sentenza

DROP TYPE nome\_tipo;

Onde *nome\_tipo* debe ser o nome do tipo de dato obxecto que se desexa eliminar. Por exemplo, para o tipo de obxectos anterior, deberíase indicar:

DROP TYPE Usuario

* + - 1. Declaración de atributos

A declaración dos atributos pódese realizar de forma moi similar a unha declaración de variable, é dicir, utilizando un nome e un tipo de dato. O devandito nome debe ser único dentro do tipo de obxecto, aínda que pode ser reempregado noutros tipos de obxecto.

O tipo de dato que pode almacenar un determinado atributo pode ser calquera dos tipos de Oracle excepto os seguintes:

* LONG e LONG RAW.
* ROWID e UROWID.
* Os tipos específicos PL/SQL BINARY\_INTEGER (e os seus subtipos), BOOLEAN, PLS\_INTEGER, RECORD, REF CURSOR, %TYPE, e %ROWTYPE.
* Os tipos definidos dentro dun paquete PL/SQL.

Débese ter en conta que non se poden inicializar os atributos usando o operador de asignación, nin a cláusula DEFAULT, nin asignar a restrición NOT NULL.

O tipo de dato dun atributo pode ser outro tipo de obxecto, polo que a estrutura de datos pode ser tan complicada como sexa necesario. Por exemplo:

CREATE OR REPLACE TYPE Usuario AS OBJECT (

login VARCHAR2(10),

nome VARCHAR2(30),

f\_ingreso DATE,

credito NUMBER );

Logo de ser creado o tipo de obxecto, pódense modificar os seus atributos utilizando a sentenza ALTER TYPE. Para engadir novos atributos utilízase a cláusula ADD ATTRIBUTE seguida da lista de novos atributos cos seus correspondentes tipos de dato. Utilizando MODIFY ATTRIBUTE poderanse modificar os atributos existentes, e para eliminar atributos disponse de xeito similar de DROP ATTRIBUTE.

Exemplos de modificación do tipo de obxecto *Usuario* creado anteriormente:

ALTER TYPE Usuario DROP ATTRIBUTE f\_ingreso;

ALTER TYPE Usuario ADD ATTRIBUTE (

apellidos VARCHAR2(40),

localidad VARCHAR2(50));

ALTER TYPE Usuario

ADD ATTRIBUTE cp VARCHAR2(5),

MODIFY ATTRIBUTE nome VARCHAR2(35);

* Na seguinte ligazón pódese atopar a sintaxe completa da instrución CREATE TYPE: [http://download.oracle.com/docs/cd/B13789\_01/server.101/b10759/statements\_8001.htm](http://download.oracle.com/docs/cd/B13789_01/server.101/b10759/statements_8001.htm%20)
* Na seguinte ligazón pódese atopar a sintaxe completa da instrución ALTER TYPE. <http://docs.oracle.com/cd/B14117_01/server.101/b10759/statements_4002.htm>
* Tarefa 1. Instalación e configuración do contorno de traballo.
  + - 1. Definición de métodos

Un método é un subprograma que se declara na especificación dun tipo de obxecto usando as palabras clave MEMBER ou STATIC. Débese ter en conta que o nome dun determinado método non pode ser o mesmo nome que o tipo de obxecto nin o de ningún dos seus atributos. Como se verá máis adiante, pódense crear métodos co mesmo nome que o tipo de obxecto, pero devanditos métodos terán unha función especial.

Os métodos teñen dous partes: unha **especificación** e un **corpo**.

* Na **especificación** ou declaración atópase o nome do método, unha lista opcional de parámetros, e, no caso das funcións, un tipo de dato de retorno.

A especificación dun método pode levar asociada unha directiva de compilación (PRAGMA RESTRICT\_REFERENCES), para evitar que os métodos manipulen a base de datos ou as variables do paquete PL/SQL. Os parámetros da directiva teñen o seguinte significado:

* WNDS: non se permite ao método modificar as táboas da base de datos.
* WNPS: non se permite ao método modificar as variables do paquete PL/SQL.
* RNDS: non se permite ao método ler as táboas da base de datos.
* RNPS: non se permite ao método ler as variables do paquete PL/SQL.

No seguinte exemplo vese a especificación dos métodos *incrementoCredito* e *ret\_credito* detrás das declaracións dos atributos:

CREATE OR REPLACE TYPE Usuario AS OBJECT (

login VARCHAR2(10),

nome VARCHAR2(30),

f\_ingreso DATE,

credito NUMBER,

MEMBER PROCEDURE incrementoCredito(inc NUMBER),

MEMBER FUNCTION ret\_credito RETURN NUMBER,

PRAGMA RESTRICT\_REFERENCES(ret\_credito, WNDS));

* No **corpo** débese indicar o código que se debe executar para realizar unha determinada tarefa cando o método é invocado. No seguinte exemplo desenvólvese o corpo do método que se declarou antes:

CREATE OR REPLACE TYPE BODY Usuario AS

MEMBER PROCEDURE incrementoCredito(inc NUMBER) IS

BEGIN

credito := credito inc;

END;

MAP MEMBER FUNCTION ret\_credito RETURN NUMBER IS

BEGIN

RETURN credito;

END;

END;

Por cada especificación de método que se indique no bloque de especificación do tipo de obxecto, debe existir o seu correspondente corpo do método, ou ben, o método debe declararse como NOT INSTANTIABLE, para indicar que o corpo do método atoparase nun subtipo dese tipo de obxecto. Ademais, débese ter en conta que as cabeceiras dos métodos deben coincidir exactamente na especificación e no corpo.

Do mesmo xeito que os atributos, os parámetros formais decláranse cun nome e un tipo de dato. Con todo, o tipo de dato dun parámetro non pode ter restricións de tamaño. O tipo de datos pode ser calquera dos empregados por Oracle salvo os indicados anteriormente para os atributos. As mesmas restricións aplícanse para os tipos de retorno das funcións.

O código fonte dos métodos non só pode escribirse na linguaxe PL/SQL. Tamén con outras linguaxes de programación como Java ou C.

Pódese usar a sentenza ALTER TYPE para engadir, modificar ou eliminar métodos dun tipo de obxecto existente, de xeito similar á utilizada para modificar os atributos dun tipo de obxecto.

* Tarefa 2. Creación de tipos de dato obxecto.
  + - 1. O parámetro SELF

Un parámetro especial que se pode utilizar cos métodos MEMBER é o que recibe o nome SELF. Este parámetro fai referencia a unha instancia (obxecto) do mesmo tipo de obxecto. Aínda que non se declare explicitamente, este parámetro sempre se declara automaticamente.

O tipo de dato correspondente ao parámetro SELF será o mesmo que o do obxecto orixinal.

Cando se fai referencia ao parámetro SELF dentro do corpo dun método, realmente estase facendo referencia ao obxecto que invocou a devandito método. Xa que logo, cando se utiliza SELF.nome\_atributo ou SELF.nome\_método, estarase utilizando un atributo ou un método do mesmo obxecto que chamou ao método onde se atopa utilizado o parámetro SELF.

MEMBER PROCEDURE setNome(Nome VARCHAR2) IS

BEGIN /\* O primeiro elemento (SELF.Nome) fai referencia ao atributo do tipo de obxecto mentres que o segundo (Nome) fai referencia ao parámetro do método \*/

SELF.Nome := Nome;

END setNome;

* + - 1. Sobrecarga

Do mesmo xeito que ocorre cos subprogramas empaquetados, os métodos poden ser sobrecargados, é dicir, pódese utilizar o mesmo nome para métodos diferentes sempre que os seus parámetros formais sexan diferentes (en cantidade ou tipo de dato).

Cando se fai unha chamada a un método, compáranse os parámetros actuais cos parámetros formais dos métodos que se declararon, e executarase aquel no que haxa unha coincidencia entre ambos tipos de parámetros.

Hai que ter en conta que non é válida a sobrecarga de dous métodos cuxos parámetros formais se diferencian unicamente no seu nome, así como tampouco en funcións que se diferencien unicamente no valor de retorno.

Estes dous exemplos son correctos, xa que se diferencian no número de parámetros:

MEMBER PROCEDURE setNome(Nome VARCHAR2)

MEMBER PROCEDURE setNome(Nome VARCHAR2, Apelidos VARCHAR2)

Non é válido crear un novo método no que se utilicen os mesmos tipos de parámetros formais aínda que se diferencien os nomes dos parámetros:

MEMBER PROCEDURE setNome(Name VARCHAR2, Surname VARCHAR2)

* + - 1. Métodos construtores

Cada tipo de obxecto ten un método construtor, que se trata dunha función co mesmo nome que o tipo de obxecto e que se encarga de inicializar os atributos e retornar unha nova instancia dese tipo de obxecto.

Oracle crea un método construtor por defecto para cada tipo de obxecto declarado, cuxos parámetros formais coinciden en orde, nomes e tipos de datos cos atributos do tipo de obxecto.

Tamén se poden declarar métodos construtores propios, reescribindo o método declarado polo sistema, ou ben, definindo un novo método con outros parámetros. Unha das vantaxes de crear un novo método construtor personalizado é que se pode facer unha verificación de que os datos que se van a asignar aos atributos son correctos (por exemplo, que cumpren unha determinada restrición).

Si se desexa cambiar o método construtor por defecto, débese utilizar a sentenza CONSTRUCTOR FUNCTION seguida do nome do tipo de obxecto no que se atopa (hai que recordar que os métodos construtores teñen o mesmo nome que o tipo de obxecto). A continuación débense indicar os parámetros que sexan necesarios do xeito habitual. Para rematar, debese indicar que o valor de retorno da función é o propio obxecto utilizando a cláusula RETURN SELF AS RESULT.

Pódense crear varios métodos construtores seguindo as restricións indicadas para a sobrecarga de métodos.

No seguinte exemplo pódese ver a declaración e o corpo dun método construtor para o tipo de obxecto *Usuario*. Como se pode comprobar, utiliza dous parámetros: *login* e *credito*. O corpo do método realiza un pequeno control para que no caso de que o crédito indicado sexa negativo, déixese en cero:

CONSTRUCTOR FUNCTION Usuario(login VARCHAR2, credito NUMBER)

RETURN SELF AS RESULT

CREATE OR REPLACE TYPE BODY Usuario AS

CONSTRUCTOR FUNCTION Usuario(login VARCHAR2, credito NUMBER)

RETURN SELF AS RESULT

IS

BEGIN

IF (credito >= 0) THEN

SELF.credito := credito;

ELSE

SELF.credito := 0;

END IF;

RETURN;

END;

END;

* + 1. Utilización de obxectos

Unha vez que se dispón dos coñecementos necesarios para ter declarado un tipo de dato obxecto, pódese coñecer a forma de utilizar os obxectos que se vaian a crear dese tipo.

Nos seguintes apartados verase como se poden declarar variables que permitan almacenar obxectos, darlle uns valores iniciais aos seus atributos, acceder ao contido de devanditos atributos en calquera momento, e chamar aos métodos que ofrece o tipo de obxecto utilizado.

* + - 1. Declaración de obxectos

Unha vez que o tipo de obxecto foi definido, este pode ser utilizado para declarar variables de obxectos dese tipo en calquera bloque PL/SQL, subprograma ou paquete. Ese tipo de obxecto pódese utilizar como tipo de dato para unha variable, atributo, elemento dunha táboa, parámetro formal, ou resultado dunha función, de igual xeito que se utilizan os tipos de datos habituais como VARCHAR ou NUMBER.

Por exemplo, para declarar unha variable denominada *u1*, que vai permitir almacenar un obxecto do tipo *Usuario*, debese facer a seguinte declaración:

u1 Usuario;

Na declaración de calquera procedemento ou función, incluídos os métodos do mesmo tipo de obxecto ou doutro, pódese utilizar o tipo de dato obxecto definido para indicar que debe pasarse como parámetro un obxecto de devandito tipo na chamada. Por exemplo, un procedemento ao que se lle debe pasar como parámetro un obxecto do tipo *Usuario*:

PROCEDURE setUsuario(u IN Usuario)

A chamada a este método realizaríase utilizando como parámetro un obxecto, como o que podemos ter na variable declarada anteriormente:

setUsuario(u1);

De xeito semellante unha función pode retornar obxectos:

FUNCTION getUsuario(codigo INTEGER) RETURN Usuario

Os obxectos créanse durante a execución do código como instancias do tipo de obxecto, e cada un deles pode conter valores diferentes nos seus atributos.

O ámbito dos obxectos segue as mesmas regras habituais en PL/SQL, é dicir, nun bloque ou subprograma os obxectos son creados (instanciados) cando se entra en devandito bloque ou subprograma, e destrúense automaticamente cando se sae deles. Nun paquete, os obxectos son instanciados no momento de facer referencia ao paquete, e deixan de existir cando se finaliza a sesión na base de datos.

* + - 1. Inicialización de obxectos

Para crear ou instanciar un obxecto dun determinado tipo de obxecto, débese facer unha chamada ao seu método construtor. Isto pódese realizar empregando a instrución NEW seguido do nome do tipo de obxecto como unha chamada a unha función na que se indican como parámetros os valores que se desexan asignar inicialmente aos atributos. Nunha asignación tamén se pode optar por facer iso mesmo omitindo a palabra NEW.

A orde dos parámetros debe coincidir coa orde no que están declarados os atributos, así como os tipos de datos. O formato sería como o seguinte:

variable\_obxecto := NEW Nome\_Tipo\_Obxecto (valor\_atributo1, valor\_atributo2, ...);

Por exemplo, no caso do tipo de obxecto *Usuario*:

u1 := NEW Usuario('luitom64', 'LUÍS ', 'TOMAS BRUÑA', '24/10/07', 100);

Nese momento inicializase o obxecto. Ata que non se inicializa o obxecto chamando ao seu constructor, o obxecto ten o valor NULL.

É habitual inicializar os obxectos na súa declaración:

u1 Usuario := NEW Usuario('luitom64', 'LUÍS ', 'TOMAS BRUÑA', '24/10/07', 100);

A chamada ao método construtor pódese realizar en calquera lugar no que se poida facer unha chamada a unha función da forma habitual. Por exemplo, a chamada ao método construtor pode ser utilizada como parte dunha expresión.

Os valores dos parámetros que se pasan ao construtor cando se fai a chamada, son asignados aos atributos do obxecto que está sendo creado. Se a chamada é ao método construtor que incorpora Oracle por defecto, débese indicar un parámetro para cada atributo, no mesmo orde en que están declarados os atributos. Hai que ter en conta que os atributos, en contra do que ocorre con variables e constantes, non poden ter valores por defecto asignados na súa declaración. Xa que logo, os valores que se desexe que teñan inicialmente os atributos dun obxecto instanciado deben indicarse como parámetros na chamada ao método construtor.

Existe a posibilidade de utilizar os nomes dos parámetros formais na chamada ao método construtor, en lugar de utilizar o modelo posicional dos parámetros. Deste xeito non é obrigatorio respectar a orde na que se atopan os parámetros reais respecto dos parámetros formais, que como se comentou antes coincide coa orde dos atributos.

DECLARE

u1 Usuario;

BEGIN

u1 := NEW Usuario('user1', -10);

/\* Mostrarase o crédito como cero, ao intentar asignar un crédito negativo \*/

dbms\_output.put\_line(u1.credito);

END;

* + - 1. Acceso aos atributos de obxectos

Para facer referencia a un atributo dun obxecto débese utilizar o nome de devandito atributo, utilizando o punto para acceder ao valor que contén ou ben para modificalo. Antes debe ir precedido do obxecto cuxo atributo se desexa coñecer ou modificar.

nome\_obxecto.nome\_atributo

Por exemplo, a consulta do valor dun atributo pode utilizarse como parte dunha asignación ou como parámetro na chamada a unha función:

unNome := usuario1.nome;

dbms\_output.put\_line(usuario1.nome);

A modificación do valor contido no atributo pode ser similar á seguinte:

usuario1.nome:= 'Novo Nome';

Os nomes dos atributos poden ser encadeados, o que permite o acceso a atributos de tipos de obxectos aniñados. Por exemplo, se o obxecto *sitio1* ten un atributo do tipo de obxecto *Usuario*, accederíase ao atributo do nome do usuario con:

sitio1.usuario1.nome

Cando se utiliza nunha expresión, o acceso a un atributo dun obxecto que non foi inicializado, se avalía como NULL. Doutra banda, cando se intenta asignar valores a un obxecto non inicializado, este lanza unha excepción ACCESS\_INTO\_NULL.

Para comprobar se un obxecto é NULL pódese utilizar o operador de comparación IS NULL co que se obtén o valor TRUE se é así.

De xeito similar, ao intentar facer unha chamada a un método dun obxecto que non foi inicializado, lánzase unha excepción NULL\_SELF\_DISPATCH. Cando se pasa como parámetro de tipo IN, os atributos do obxecto NULL avalíanse como NULL, e se o parámetro é de tipo OUT ou IN OUT lánzase unha excepción ao intentar modificar o valor dos seus atributos.

* + - 1. Chamada aos métodos dos obxectos

Do mesmo xeito que nas chamadas a subprogramas, pódese invocar aos métodos dun tipo de obxectos utilizando un punto entre o nome do obxecto e o do método. Os parámetros reais que se pasen ao método indicaranse separados por comas, entre parénteses, despois do nome do método.

usuario1.setNomeCompleto('Juan', 'García Fernández');

Se o método non ten parámetros, indicarase a lista de parámetros reais baleira (pódense incluír ou non os parénteses).

credito := usuario1.getCredito();

As chamadas a métodos poden encadearse. Nese caso, a orde de execución dos métodos é de dereita a esquerda. Débese ter en conta que o método da esquerda debe retornar un obxecto do tipo correspondente ao método da dereita.

Por exemplo, si se dispón dun obxecto *sitio1* que ten declarado un método *getUsuario* o cal retorna un obxecto do tipo *Usuario*, pódese realizar con ese valor retornado unha chamada a un método do tipo de obxecto *Usuario*:

sitio1.getUsuario.setNomeCompleto('Juan', 'García Fernández');

Os métodos MEMBER son invocados utilizando unha instancia do tipo de obxecto:

nome\_obxecto.metodo()

En cambio, os métodos STATIC invócanse usando o tipo de obxecto, en lugar dunha das súas instancias:

nome\_tipo\_obxecto.metodo()

* + - 1. Herdanza

A linguaxe PL/SQL admite a herdanza simple de tipos de obxectos, mediante a cal, pódense definir subtipos dos tipos de obxecto. Estes subtipos, ou tipos herdados, conteñen todos os atributos e métodos do tipo pai, pero ademais poden conter atributos e métodos adicionais, ou ata sobrescribir métodos do tipo pai.

Para indicar que un tipo de obxecto é herdado doutro, hai que usar a palabra reservada UNDER, e ademais hai que ter en conta que o tipo de obxecto do que herda debe ter a propiedade NOT FINAL. Por defecto, os tipos de obxecto decláranse como FINAL, é dicir, que non se pode crear un tipo de obxecto que herde del.

Se non se indica o contrario, sempre se poden crear obxectos (instancias) dos tipos de obxecto declarados. Indicando a opción NOT INSTANTIABLE pódense declarar tipos de obxecto dos que non se poden crear obxectos. Estes tipos terán a función de ser pais doutros tipos de obxecto.

No seguinte exemplo pódese ver como se crea o tipo de obxecto *Acode*, que se utilizará herdado no tipo de obxecto *UsuarioPersona*. Deste xeito, este último terá os atributos de *Persoa* máis os atributos declarados en *UsuarioPersona*. Na creación do obxecto pódese observar que se deben asignar os valores para todos os atributos, incluíndo os herdados.

CREATE TYPE Acode AS OBJECT (

nome VARCHAR2(20),

apelidos VARCHAR2(30)

) NOT FINAL;

CREATE TYPE UsuarioPersona UNDER Acode (

login VARCHAR(30),

f\_ingreso DATE,

credito NUMBER

);

DECLARE

u1 UsuarioPersona;

BEGIN

u1 := NEW UsuarioPersona('nome1', 'apelidos1', 'user1', '01/01/2001', 100);

dbms\_output.put\_line(u1.nome);

END;

* + 1. Métodos MAP e ORDER

As instancias dun tipo de obxecto non teñen unha orde determinada. Para establecer unha orde nelas, co fin de facer unha ordenación ou unha comparación, débese crear un método MAP.

Por exemplo, para facer unha comparación entre dous obxectos *Usuario*, e obter se un é maior que outro, en base a que criterio faise esa comparación?, pola orde alfabética de apelidos e nome?, pola data de alta?, polo crédito?. Débese establecer cun método MAP cal vai ser o valor que se vai a utilizar nas comparacións.

Para crear un método MAP debese declarar un método que retorne o valor que se vai a utilizar para facer as comparacións. O método que se declare para iso debe empezar a súa declaración coa palabra MAP:

CREATE OR REPLACE TYPE Usuario AS OBJECT (

login VARCHAR2(30),

nome VARCHAR2(30),

apelidos VARCHAR2(40),

f\_ingreso DATE,

credito NUMBER,

MAP MEMBER FUNCTION ordenarUsuario RETURN VARCHAR2

);

No corpo do método débese retornar o valor que se utilizará para realizar as comparacións entre as instancias do tipo de obxecto. Por exemplo, para establecer que as comparacións entre obxectos do tipo *Usuario* se realicen considerando a orde alfabética habitual de apelidos e nome:

CREATE OR REPLACE TYPE BODY Usuario AS

MAP MEMBER FUNCTION ordenarUsuario RETURN VARCHAR2 IS

BEGIN

RETURN (apelidos || ' ' || nomee);

END ordenarUsuario;

END;

A linguaxe PL/SQL utiliza os métodos MAP para avaliar expresións lóxicas que dan como resultado valores booleanos como *obxecto1 > obxecto2*, e para realizar as comparacións implícitas nas cláusulas DISTINCT, GROUP BY e ORDER BY.

Cada tipo de obxecto só pode ter un método MAP declarado, e só pode retornar algún dos seguintes tipos: DATE, NUMBER, VARCHAR2, CHARACTER ou REAL.

* + - 1. Métodos ORDER

De forma similar ao método MAP, pódese declarar en calquera tipo de obxecto un método ORDER que permitirá establecer unha orde entre os obxectos instanciados de devandito tipo.

Cada tipo de obxecto só pode conter un método ORDER, o cal debe retornar un valor numérico que permita establecer a orde entre os obxectos. Para indicar que un obxecto sexa menor que outro pódese retornar, por exemplo, o valor -1. Para determinar que sexan iguais, pódese devolve 0, e si vai a ser maior, retornar 1. Deste xeito, considerando o valor retornado polo método ORDER, pódese establecer se un obxecto é menor, igual ou maior que outro no momento de facer unha ordenación entre unha serie de obxectos do mesmo tipo.

Para declarar que método vai realizar esta operación, débese indicar a palabra ORDER diante da súa declaración. Débese ter en conta que vai retornar un valor numérico (INTEGER), e que necesita que se indique un parámetro que será do mesmo tipo de obxecto. O obxecto que se indique nese parámetro será o que se compare co obxecto que utilice este método.

No seguinte exemplo, o sistema que se vai a establecer para ordenar aos usuarios, realízase en función dos díxitos que se atopan a partir da posición 7 do atributo *login*.

CREATE OR REPLACE TYPE BODY Usuario AS

ORDER MEMBER FUNCTION ordenUsuario(u Usuario) RETURN INTEGER IS

BEGIN

/\* A función substr obtén unha subcadena desde a posición indicada ata o final\*/ IF substr(SELF.login, 7) < substr(u.login, 7) THEN

RETURN -1;

ELSIF substr(SELF.login, 7) > substr(u.login, 7) THEN

RETURN 1;

ELSE

RETURN 0;

END IF;

END;

END;

Con ese exemplo, un usuario con login *'luitom64'* considérase maior que un usuario *'caragu62'*, xa que *'64'* é maior que *'62'*.

O método debe retornar un número negativo, cero, ou un número positivo que significará que o obxecto que utiliza o método (SELF) é menor, igual, ou maior que o obxecto pasado por parámetro. Débese ter en conta que se pode declarar un método MAP ou un método ORDER, pero non os dous. Cando se vaia a ordenar ou mesturar un alto número de obxectos, é preferible usar un método MAP, xa que neses casos un método ORDER é menos eficiente.

* + 1. Tipos de datos colección

Unha colección é un conxunto de elementos do mesmo tipo. Pode compararse cos vectores e matrices que se utilizan en moitas linguaxes de programación. Neste caso, as coleccións só poden ter unha dimensión e os elementos se indexan mediante un valor de tipo numérico ou cadea de carácteres.

A base de datos de Oracle proporciona os tipos VARRAY e NESTED TABLE (táboa aniñada) como tipos de datos colección.

* Un VARRAY é unha colección de elementos á que se lle establece unha dimensión máxima que debe indicarse ao declarala. Ao ter unha lonxitude fixa, a eliminación de elementos non aforra espazo na memoria do ordenador.
* Unha NESTED TABLE (táboa aniñada) pode almacenar calquera número de elementos. Teñen, xa que logo, un tamaño dinámico, e non teñen que existir forzosamente valores para todas as posicións da colección.
* Unha variación das táboas aniñadas son os arrays asociativos, que utilizan valores arbitrarios para os seus índices. Neste caso, os índices non teñen que ser necesariamente consecutivos.

Cando se necesite almacenar un número fixo de elementos, ou facer un percorrido entre os elementos de forma ordenada, ou cando se necesita obter e manipular toda a colección como un valor, deberíase utilizar o tipo VARRAY.

En cambio, cando se necesita executar consultas sobre unha colección de xeito eficiente, ou manipular un número arbitrario de elementos, ou ben realizar operacións de inserción, actualización ou borrado de forma masiva, deberíase usar unha NESTED TABLE.

As coleccións poden ser declaradas como unha instrución SQL ou no bloque de declaracións dun programa PL/SQL. O tipo de dato dos elementos que pode conter unha colección declarada en PL/SQL é calquera tipo de dato PL/SQL, excepto REF CURSOR. Os elementos das coleccións declaradas en SQL, ademais non poden ser dos tipos: BINARY\_INTEGER, PLS\_INTEGER, BOOLEAN, LONG, LONG RAW, NATURAL, NATURALN, POSITIVE, POSITIVEN, REF CURSOR, SIGNTYPE, STRING.

Calquera tipo de obxectos declarado previamente pode ser utilizado como tipo de elemento para unha colección.

Unha táboa da base de datos pode conter columnas que sexan coleccións. Sobre unha táboa que contén coleccións poderanse realizar operacións de consulta e manipulación de datos do mesma xeito que se realiza con táboas cos tipos de datos habituais.

* + - 1. Declaración e uso de coleccións.

A declaración destes tipos de coleccións segue o formato seguinte:

TYPE nome\_tipo IS VARRAY (tamaño\_max) OF tipo\_elemento;

TYPE nome\_tipo IS TABLE OF tipo\_elemento;

TYPE nome\_tipo IS TABLE OF tipo\_elemento INDEX BY tipo\_índice;

Onde *nome\_tipo* representa o nome da colección, *tamaño\_max* é o número máximo de elementos que poderá conter a colección, e *tipo\_elemento* é o tipo de dato dos elementos que forman a colección. O tipo de elemento utilizado pode ser tamén calquera tipo de obxecto declarado previamente.

No caso de que a declaración se faga en SQL, fóra dun subprograma PL/SQL, débese declarar co formato CREATE [OR REPLACE] TYPE:

CREATE TYPE nome\_tipo AS ...

O *tipo\_índice* representa o tipo de dato que se vai a utilizar para o índice. Pode ser PLS\_INTEGER, BINARY\_INTEGER ou VARCHAR2. Neste último tipo débese indicar entre parénteses o tamaño que se vai a utilizar para o índice, por exemplo, VARCHAR2(5).

Ata que non sexa inicializada unha colección, esta é NULL. Para inicializar unha colección débese utilizar o construtor, que é unha función co mesmo nome que a colección. A esta función pásaselle como parámetros os valores iniciais da colección. Por exemplo:

DECLARE

TYPE Cores IS TABLE OF VARCHAR2(10);

misColores Cores;

BEGIN

misColores := Cores('Vermello', 'Laranxa', 'Amarelo', 'Verde', 'Azul');

END;

A inicialización pódese realizar no bloque de código do programa, ou ben, directamente no bloque de declaracións como se pode ver neste exemplo:

DECLARE

TYPE Cores IS TABLE OF VARCHAR2(10);

misColores Cores := Cores('Vermello', 'Laranxa', 'Amarelo', 'Verde', 'Azul');

Para obter un dos elementos da colección ou modificar o seu contido débese indicar o nome da colección seguido, entre parénteses, do índice que ocupa o elemento desexado. Tanto nos VARRAY como en NESTED TABLE, o primeiro elemento ten o índice 1.

Por exemplo, para mostrar en pantalla o segundo elemento (*'Laranxa'*) da colección *Cores*:

dbms\_output.put\_line(misColores(2));

No seguinte exemplo modifícase o contido da posición 3:

misColores(3) := 'Gris';

No seguinte exemplo pódese comprobar como poden utilizarse as coleccións para almacenar os seus datos nunha táboa da base de datos, así como a utilización con varias sentenzas de consulta e manipulación dos datos da colección que se atopa na táboa.

CREATE TYPE ListaColores AS TABLE OF VARCHAR2(20);

/

CREATE TABLE flores (nome VARCHAR2(20), coloresFlor ListaColores)

NESTED TABLE coloresFlor STORE AS cores\_tab;

DECLARE

cores ListaColores;

BEGIN

INSERT INTO flores VALUES('Rosa', ListaColores('Vermello','Amarelo','Branco'));

cores := ListaColores('Vermello','Amarelo','Branco','Rosa Claro');

UPDATE flores SET coloresFlor = cores WHERE nome = 'Rosa';

SELECT coloresFlor INTO cores FROM flores WHERE nome = 'Rosa';

END;/

* Para saber máis sobre os tipos de dato VARRAY e NESTED TABLE pódese consultar a seguinte ligazón: https://docs.oracle.com/cd/B10501\_01/appdev.920/a96624/05\_colls.htm
* Tarefa 3. Creación de tipos de datos colección.
  + 1. Táboas de obxectos

Logo de ver que un grupo de obxectos pódese almacenar en memoria mediante coleccións, verase neste apartado que tamén se poden almacenar os obxectos en táboas de igual xeito que os tipos de datos habituais das bases de datos.

Os tipos de datos obxecto pódense usar para formar unha táboa exclusivamente formada por elementos deste tipo, ou ben, para usarse como un tipo de columna máis entre outras columnas doutros tipos de datos.

No caso de que se desexe crear unha táboa formada exclusivamente por un determinado tipo de dato obxecto (táboa de obxectos), débese utilizar a sentenza CREATE TABLE xunto co tipo de obxecto do seguinte xeito:

CREATE TABLE NomeTabla OF TipoObjeto;

Sendo *NomeTabla* o nome que se desexa dar á táboa que vai almacenar os obxectos do tipo *TipoObjeto*. Por exemplo, para crear a táboa *UsuariosObj*, que almacene obxectos do tipo *Usuario*:

CREATE TABLE UsuariosObj OF Usuario;

Débese ter en conta que se unha táboa fai uso dun tipo de obxecto, non se poderá eliminar nin modificar a estrutura de devandito tipo de obxecto. Xa que logo, desde o momento en que o tipo de obxecto sexa utilizado nunha táboa, non se poderá volver a definir.

Para poder crear este exemplo, previamente débese ter declarado o tipo de obxecto *Usuario*, que se utilizou en apartados anteriores.

Ao crear unha táboa deste xeito, estase a conseguir que se poida almacenar obxectos do tipo *Usuario* nunha táboa da base de datos, quedando así os seus datos persistentes mentres non sexan eliminados da táboa. Anteriormente instanciáronse obxectos que se gardaron en variables, polo que ao terminar a execución, os obxectos, e a información que conteñen, desaparecen. Se eses obxectos se almacenan en táboas, non desaparecen ata que se eliminen da táboa na que se atopen.

Cando se instancia un obxecto co fin de almacenalo nunha táboa, o obxecto non ten identidade fóra da táboa da base de datos. Con todo, o tipo de obxecto existe independentemente de calquera táboa, e pode ser usado para crear obxectos de calquera modo.

As táboas que só conteñen filas con obxectos, reciben o nome de táboas de obxectos.

* + - 1. Táboas con columnas tipo obxecto

Pódese usar calquera tipo de obxecto, que se declare previamente, para utilizalo como un tipo de dato dunha columna máis nunha táboa da base de datos. Así, unha vez creada a táboa, pódese utilizar calquera das sentenzas SQL para inserir un obxecto, seleccionar os seus atributos e actualizar os seus datos.

Para crear unha táboa na que algunha das súas columnas sexa un tipo de obxecto, simplemente débese facer coma se fose unha columna como as que se utilizaron ata agora, pero no tipo de dato débese especificar o tipo de obxecto.

Por exemplo, pódese crear unha táboa que conteña, entre outras columnas, unha columna de obxectos do tipo *Usuario* que se utilizou anteriormente.

CREATE TABLE Xente (

dni VARCHAR2(10),

unUsuario Usuario,

partidasJugadas SMALLINT

);

* Tarefa 4. Creación de táboas de obxectos e táboas de columnas tipo obxecto.
  + - 1. Uso da sentenza Select

De xeito similar ás consultas que se realizan sobre táboas sen tipos de obxectos, pódese utilizar a sentenza SELECT para obter datos das filas almacenadas en táboas de obxectos ou táboas con columnas de tipos de obxectos.

O uso máis sinxelo sería para mostrar todas as filas contidas na táboa:

SELECT \* FROM NomeTabla;

Nas sentenzas SELECT que se utilizan con obxectos, pódese incluír calquera das cláusulas e funcións de agrupamento para a sentenza SELECT que se usan coas táboas que conteñen columnas de tipos básicos. Por exemplo, pódese utilizar: SUM, MAX, WHERE, ORDER, JOIN, etc.

Por exemplo, na seguinte consulta se desexa obter o nome e os apelidos dos usuarios que teñen algo de crédito:

SELECT ou.nome, ou.apelidos FROM UsuariosObj ou WHERE ou.credito > 0

Si se trata dunha táboa con columnas de tipo obxecto, o acceso aos atributos do obxecto débese realizar indicando previamente o nome asignado á columna que contén os obxectos:

SELECT g.unUsuario.nome, g.unUsuario.apelidos FROM Xente g;

* + - 1. Inserción de obxectos

Evidentemente, non servirá de nada unha táboa que poida conter obxectos senón se coñece o xeito de inserir obxectos na táboa.

O xeito para facelo é o mesmo que se utiliza para introducir datos de calquera tipo habitual nas táboas da base de datos: usando a sentenza INSERT de SQL.

Nas táboas habituais, cando se quere engadir unha fila a unha táboa que teña un campo VARCHAR engadíaselle á sentenza INSERT un dato dese tipo. Pois si a táboa é dun determinado tipo de obxectos, ou si posúe un campo dun determinado tipo de obxecto, terase que fornecer á sentenza INSERT un obxecto instanciado do seu tipo de obxecto correspondente.

Por exemplo, para inserir un *Usuario* na táboa *UsuariosObj* creada no apartado anterior, previamente débese crear o obxecto ou os obxectos que se desexen inserir e a continuación poderase utilizar dentro da sentenza INSERT.

DECLARE

u1 Usuario;

u2 Usuario;

BEGIN

u1 := NEW Usuario('luitom64', 'LUÍS', 'TOMAS BRUNA', '24/10/2007', 50);

u2 := NEW Usuario('caragu72', 'CARLOS', 'AGUDO SEGURA', '06/07/2007', 100);

INSERT INTO UsuariosObj VALUES (u1);

INSERT INTO UsuariosObj VALUES (u2);

END;

Dun xeito máis directo, pódese crear o obxecto dentro da sentenza INSERT directamente, sen necesidade de gardar o obxecto previamente nunha variable:

INSERT INTO UsuariosObj VALUES (Usuario('luitom64', 'LUÍS', 'TOMAS BRUNA', '24/10/2007', 50));

Pódense comprobar os resultados facendo unha consulta SELECT sobre a táboa do xeito habitual:

SELECT \* FROM UsuariosObj;

De xeito similar pódese realizar a inserción de filas en táboas con columnas de tipo obxecto. Por exemplo, para a táboa *Xente* que posúe entre as súas columnas unha de tipo obxecto *Usuario*, poderíase usar o formato de instanciar o obxecto directamente na sentenza INSERT, ou ben, indicar unha variable que almacena un obxecto que se instanciou anteriormente:

INSERT INTO Xente VALUES ('22900970P', Usuario('luitom64', 'LUÍS', 'TOMAS BRUNA',

'24/10/2007', 50), 54);

INSERT INTO Xente VALUES ('62603088D', u2, 21);

* + - 1. Modificación de obxectos

Para modificar un obxecto almacenado nunha táboa, tan só se teñen que utilizar as mesmas sentenzas SQL das que se dispón para modificar filas dunha táboa.

A sentenza UPDATE agora pódese volver a utilizar para modificar tamén os obxectos da táboa, de igual xeito que calquera outro tipo de dato.

Hai unha pequena diferenza na forma de especificar os nomes dos campos afectados, en función do tipo de táboa: segundo sexa unha táboa de obxectos, ou ben unha táboa con algunha columna de tipo obxecto.

Se é unha táboa de obxectos, farase referencia aos atributos dos obxectos xusto detrás do nome asignado á táboa. Sería algo similar ao formato seguinte:

UPDATE NomeTabla

SET NomeTabla.atributoModificado = nuevoValor

WHERE NomeTabla.atributoBusqueda = valorBusqueda;

Continuando co exemplo empregado anteriormente, supoñamos que se desexan modificar os datos dun determinado *usuario*. Por exemplo, o crédito do *usuario* identificado polo login *'luitom64'*, asignándolle o valor 0.

UPDATE UsuariosObj

SET UsuariosObj.credito = 0

WHERE UsuariosObj.login = 'luitom64';

É moi habitual abreviar o nome da táboa cun alias:

UPDATE UsuariosObj u

SET u.credito = 0

WHERE u.login = 'luitom64';

Pero non só se pode cambiar o valor dun determinado atributo do obxecto. Pódese cambiar un obxecto por outro como se pode ver no seguinte exemplo, no que se substitúe o *usuario* con login *'caragu72'* por outro usuario novo.

UPDATE UsuariosObj u SET u = Usuario('juaesc82', 'JUAN', 'ESCUDERO LARRASA', '10/04/2011', 0)

WHERE u.login = 'caragu72';

Si se trata dunha táboa con columnas de tipo obxecto, débese facer referencia ao nome da columna que contén os obxectos:

UPDATE NomeTabla

SET NomeTabla.colObjeto.atributoModificado = nuevoValor

WHERE NomeTabla.colObjeto.atributoBusqueda = valorBusqueda;

A continuación pódese ver un exemplo de actualización de datos da táboa que se creou cunha columna do tipo de obxecto *Usuario*. Hai que recordar que á columna na que se almacenaban os obxectos de tipo *Usuario* asignáraselle o nome *unUsuario*:

UPDATE Xente g

SET g.unUsuario.credito = 0

WHERE g.unUsuario.login = 'luitom64';

Ou ben, pódese cambiar todo un obxecto por outro, mantendo o resto dos datos da fila sen modificar, como no seguinte exemplo, onde os datos de DNI e *partidasJugadas* non se cambian, só se cambia un usuario por outro.

UPDATE Xente g

SET g.unUsuario = Usuario('juaesc82', 'JUAN', 'ESCUDERO LARRASA', '10/04/2011', 0)

WHERE g.unUsuario.login = 'caragu72';

* + - 1. Borrado de obxectos

Por suposto, non pode faltar unha sentenza que permita eliminar determinados obxectos almacenados en táboas. Do mesmo xeito que nos casos anteriores, pódese utilizar a mesma sentenza que nas operacións habituais sobre táboas. Neste caso de borrado de obxectos deberase utilizar a sentenza DELETE.

O modo de uso de DELETE sobre obxectos almacenados en táboas é moi similar ao utilizado ata agora:

DELETE FROM NomeTablaObjetos;

Se non se indica ningunha condición, eliminaranse todos os obxectos da táboa, polo que adoita ser habitual utilizar a sentenza DELETE cunha condición detrás da cláusula WHERE. Os obxectos ou filas da táboa que cumpran coa condición indicada serán os que se eliminen.

DELETE FROM NomeTablaObjetos WHERE condición;

Obsérvese o seguinte exemplo no que se borrarán da táboa *UsuariosObj*, que é unha táboa de obxectos, os usuarios cuxo crédito sexa 0. Obsérvese que se utiliza un alias para o nome da táboa:

DELETE FROM UsuariosObj u WHERE u.credito = 0;

De xeito similar pódese realizar o borrado de filas en táboas nas que algunha das súas columnas son obxectos. Pódese comprobar co seguinte exemplo, onde se utiliza a táboa *Xente*, na que unha das súas columnas (*unUsuario*) é do tipo de obxecto *Usuario* que se utiliza noutros apartados anteriores.

DELETE FROM Xente g WHERE g.unUsuario.credito = 0;

Esta sentenza, do mesmo xeito que as anteriores, pódese combinar con outras consultas SELECT, de maneira que no canto de realizar o borrado sobre unha determinada táboa, fágase sobre o resultado dunha consulta, ou ben que a condición que determina as filas que deben ser eliminadas sexa tamén o resultado dunha consulta. É dicir, todo o aprendido sobre as operacións de manipulación de datos sobre as táboas habituais, pódese aplicar sobre táboas de tipos de obxectos, ou táboas con columnas de tipos de obxectos.

* Tarefa 5. Modificación de obxectos almacenados na base de datos.
* Tarefa 6. Consultas sobre táboas de obxectos e táboas de columnas tipo obxecto.
  + - 1. Consultas coa función VALUE

Cando se teña a necesidade de facer referencia a un obxecto en lugar dalgún dos seus atributos, pódese utilizar a función VALUE xunto co nome da táboa de obxectos ou o seu alias, dentro dunha sentenza SELECT. Pódese ver a continuación un exemplo de uso de devandita función para facer insercións noutra táboa (*Favoritos*) do mesmo tipo de obxectos:

INSERT INTO Favoritos SELECT VALUE(u) FROM UsuariosObj u WHERE u.credito >= 100;

Esa mesma función VALUE pódese utilizar para facer comparacións de igualdade entre obxectos. Por exemplo, para obter datos dos usuarios que se atopen nas táboas *Favoritos* e *UsuariosObj*.:

SELECT u.login FROM UsuariosObj u JOIN Favoritos f ON VALUE(u)=VALUE(f);

Obsérvese a diferenza no uso cando se fai a comparación cunha columna de tipo de obxectos. Nese caso a referencia que se fai á columna (*g.unUsuario*), permite obter directamente un obxecto, sen necesidade de utilizar a función VALUE.

SELECT g.dni FROM Xente g JOIN Favoritos f ON g.unUsuario=VALUE(f);

Usando a cláusula INTO poderase gardar en variables o obxecto obtido nas consultas usando a función VALUE. Unha vez que se teña asignado o obxecto á variable, poderase facer uso dela de calquera das formas que se viron anteriormente na manipulación de obxectos. Por exemplo, pódese acceder aos seus atributos, formar parte de asignacións, etc.

No seguinte exemplo realízase unha consulta da táboa *UsuariosObj* para obter un determinado obxecto de tipo *Usuario*. O obxecto resultante da consulta gárdase na variable *u1*. Esa variable utilízase para mostrar en pantalla o nome do usuario, e para ser asignada a unha segunda variable, que conterá os mesmos datos que a primeira.

DECLARE

u1 Usuario;

u2 Usuario;

BEGIN

SELECT VALUE(u) INTO u1 FROM UsuariosObj u WHERE u.login = 'luitom64';

dbms\_output.put\_line(u1.nome);

u2 := u1;

dbms\_output.put\_line(u2.nome);

END;

* + - 1. Referencias a obxectos.

O paso de obxectos a un método resulta ineficiente cando se trata de obxectos de gran tamaño, polo que é máis conveniente pasar un punteiro a devandito obxecto, o que permite que o método que o recibe poida facer referencia a devandito obxecto sen que sexa necesario que se pase por completo. Ese punteiro é o que se coñece en Oracle como unha referencia (REF).

Ao compartir un obxecto mediante a súa referencia, os datos non son duplicados, polo que cando se fai calquera cambio nos atributos do obxecto, prodúcense nun único lugar.

Cada obxecto almacenado nunha táboa ten un identificador de obxecto que identifica de forma única ao obxecto gardado nunha determinada fila e serve como unha referencia a devandito obxecto.

As referencias créanse utilizando o modificador REF diante do tipo de obxecto, e pódese usar con variables, parámetros, campos, atributos, e ata como variables de entrada ou saída para sentenzas de manipulación de datos en SQL.

CREATE OR REPLACE TYPE Partida AS OBJECT (

codigo INTEGER,

nome VARCHAR2(20),

usuarioCreador REF Usuario

);

/

DECLARE

u\_ref REF Usuario;

p1 Partida;

BEGIN

SELECT REF(u) INTO u\_ref FROM UsuariosObj u WHERE u.login = 'luitom64';

p1 := NEW Partida(1, 'partida1', u\_ref);

END;

/

Hai que ter en conta que só se poden usar referencias a tipos de obxectos que foron declarados previamente. Seguindo o exemplo anterior, non se podería declarar o tipo *Partida* antes que o tipo *Usuario*, xa que dentro do tipo *Partida* utilízase unha referencia ao tipo *Usuario*. Xa que logo, primeiro debe estar declarado o tipo *Usuario* e logo o tipo *Partida*.

O problema xorde cando se teñan dous tipos que utilizan referencias mutuas. É dicir, un atributo do primeiro tipo fai referencia a un obxecto do segundo tipo, e viceversa. Isto pódese solucionar facendo unha declaración de tipo anticipada. Realízase indicando unicamente o nome do tipo de obxecto que se detallará máis adiante:

CREATE OR REPLACE TYPE tipo2;

/

CREATE OR REPLACE TYPE tipo1 AS OBJECT (

tipo2\_ref REF tipo2

/\*Declaración do resto de atributos do tipo1\*/

);

/

CREATE OR REPLACE TYPE tipo2 AS OBJECT (

tipo1\_ref REF tipo1

/\*Declaración do resto de atributos do tipo2\*/

);

/

* + - 1. Navegación a través de referencias

Débese ter en conta que non se pode acceder directamente aos atributos dun obxecto referenciado que se atope almacenado nunha táboa. Para iso, pódese utilizar a función DEREF.

Esta función toma unha referencia a un obxecto e retorna o valor dese obxecto.

Verase nun exemplo supoñendo que se dispoñen das seguintes variables declaradas

u\_ref REF Usuario;

u1 Usuario;

Se *u\_ref* fai referencia a un obxecto de tipo *Usuario* que se atopa na táboa *UsuariosObj*, para obter información sobre algún dos atributos de devandito obxecto referenciado, hai que utilizar a función DEREF.

Esta función utilízase como parte dunha consulta SELECT, polo que hai que utilizar unha táboa trala cláusula FROM. Isto pode resultar algo confuso, xa que as referencias a obxectos apuntan directamente a un obxecto concreto que se atopa almacenado nunha determinada táboa. Xa que logo, non debería ser necesario indicar de novo en que táboa atópase. Realmente é así.

Pódese facer referencia a calquera táboa na consulta, e a función DEREF devolverá o obxecto referenciado que se atopa na súa táboa correspondente.

A base de datos de Oracle ofrece a táboa DUAL para este tipo de operacións. Esta táboa é creada de forma automática pola base de datos, é accesible por todos os usuarios, e ten un só campo e un só rexistro. Xa que logo, é como unha táboa comodín.

SELECT DEREF(u\_ref) INTO u1 FROM Dual;

dbms\_output.put\_line(u1.nome);

Xa que logo, para obter o obxecto referenciado por unha variable REF, débese utilizar unha consulta sobre calquera táboa, independentemente da táboa na que se atope o obxecto referenciado. Só existe a condición de que sempre se obteña unha soa fila como resultado. O máis cómodo é utilizar esa táboa DUAL. Aínda que se use esa táboa comodín, o resultado será un obxecto almacenado na táboa *UsuariosObj*.

* 1. Tarefas

As tarefas propostas son as seguintes:

* Tarefa 1. Instalación e configuración do contorno de traballo.
* Tarefa 2. Creación de tipos de dato obxecto.
* Tarefa 3. Creación de tipos de datos colección.
* Tarefa 4. Creación de táboas de obxectos e táboas de columnas tipo obxecto.
* Tarefa 5. Consultas sobre táboas de obxectos e táboas de columnas tipo obxecto.
* Tarefa 6. Modificación de obxectos almacenados na base de datos.
  + 1. Tarefa 1. Instalación e configuración do contorno de traballo

Oracle ofrece unha versión gratuíta da súa base de datos, co principal obxectivo de introducir a desarrolladores, DBAs, estudantes, formadores, e vendedores de hardware e software que queiran distribuír xunto cos seus produtos unha base de datos sen custos de licenza. O produto é Oracle Express Edition (Oracle Database XE) e, obviamente, ten as súas limitacións.

A súa versión actual (Novembro 2015) é a 11g e só pode utilizar 1 procesador do equipo onde estea instalada, un máximo de 1 Gb de RAM, e ten limitado o almacenamento a 4 Gb de datos de usuario. Cando se superen estas limitacións, Oracle ofrece un sinxelo proceso de actualización a outras versións máis completas, e nas que hai que pagar licenzas.

De todas as maneiras é unha opción moi válida para quen queira practicar con bases de datos Oracle, ou para quen necesite unha base de datos limitada en canto a tamaño, pero fiable, eficiente, e gratuíta.

Coa base de datos, Oracle inclúe tamén Oracle Application Express, un contorno sinxelo e intuitivo que permite realizar graficamente as tarefas básicas de administración da base de datos.

A tarefa consistirá, polo tanto, en descargar, instalar e configurar Oracle Express Edition 11g no equipo de traballo ou máquina virtual.

Solución

A continuación descríbense os pasos necesarios para a descarga do paquete, a súa instalación e posterior configuración para dispor dun entorno operativo co que desenrolar o resto de tarefas da actividade.

Os pasos seguidos correspóndese coa instalación nun equipo con sistema operativo Windows 7.

Descarga e instalación[[1]](#footnote-3)

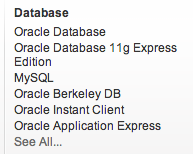
Dirixímonos ao sitio oficial de Oracle: http://www.oracle.com

|  |
| --- |
| http://4.bp.blogspot.com/-MMIkjcBuJAI/UXHpoE0QFpI/AAAAAAAAAeY/UbrAxaX8eEs/s1600/Captura+de+pantalla+2013-04-19+a+la%28s%29+19.42.51.png |
|  |

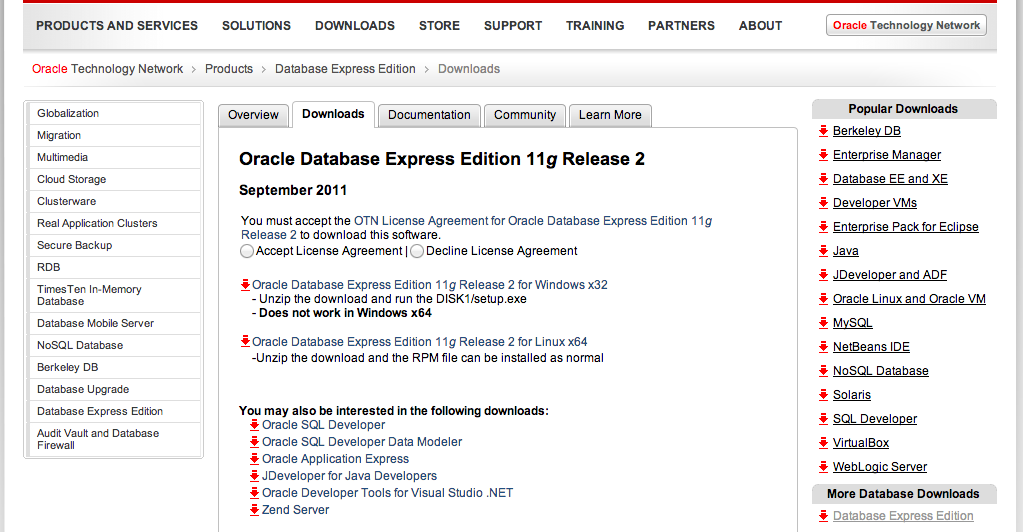
No menú de opcións pasamos a frecha do rato sobre a opción *dowloads* e despregaranse outras opcións categorizadas.

|  |
| --- |
| http://4.bp.blogspot.com/-cU0qW7kzvGk/UXHqPhS3ofI/AAAAAAAAAeg/cyKRgXYTz_4/s1600/Captura+de+pantalla+2013-04-19+a+la%28s%29+19.43.36.png |
|  |

Na categoría *Database* seleccionaremos a versión actual do Oracle database Express Edition, neste caso atopamos a versión 11g. Pulsamos sobre a opción que di *Oracle Database 11g Express Edition*.



Aceptamos os termos da licenza marcando na opción *Accept License Agreement*.



O sistema operativo debe ser de 32Bits e é claro cando di que non traballa en Windows de 64Bits. Seleccionamos o link para descargar onde di *Oracle Database Express Edition 11g Release 2 for Windows x32.*

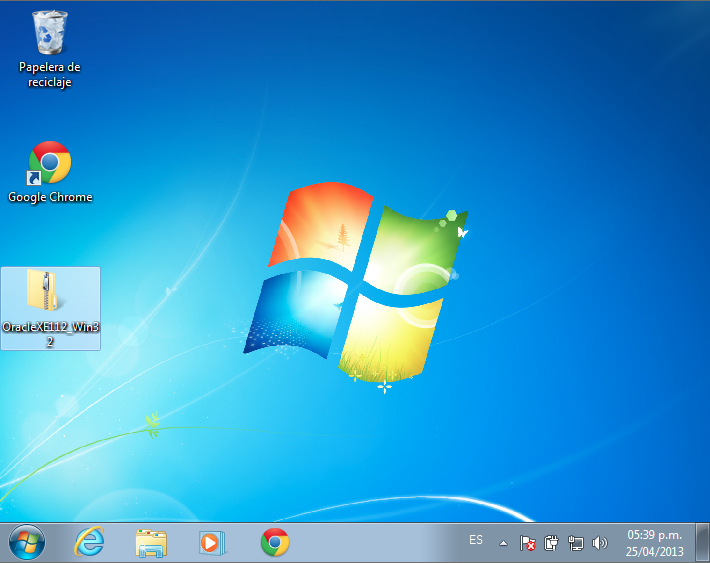
[http://1.bp.blogspot.com/-BGKnpfhN-zY/UXHrZ6UTn3I/AAAAAAAAAfI/RH02qDdFcsc/s1600/Captura+de+pantalla+2013-04-19+a+la%28s%29+19.54.06.png](http://1.bp.blogspot.com/-BGKnpfhN-zY/UXHrZ6UTn3I/AAAAAAAAAfI/RH02qDdFcsc/s1600/Captura+de+pantalla+2013-04-19+a+la(s)+19.54.06.png)

Para poder descargar Oracle Database XE 11g, hai que ter unha conta Oracle; este procedemento tamén é necesario no caso de que se queiran descargar outras aplicacións de Oracle. Se xa se dispón dunha conta, teremos que introducir as nosas credenciais e seleccionar "Inicio de Sesión". En caso contrario, hai que pulsar en "Inscríbase" e seguir os pasos que alí nos describen.



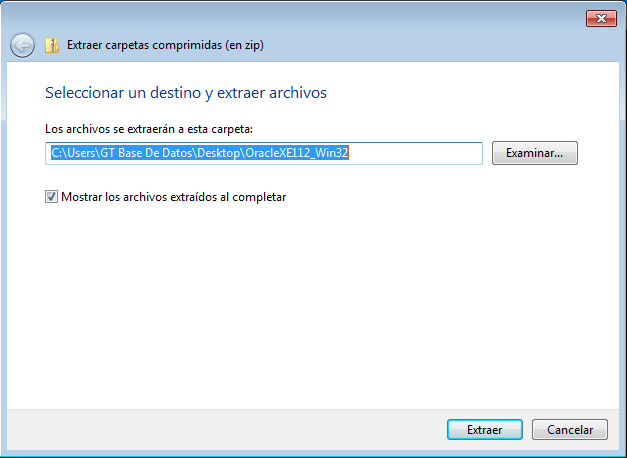
E por ultimo pregunta onde queremos situar o instalador de Oracle Database 11g XE. Indicámoslle a carpeta destino e dámoslle a "*Gardar*".

Finalmente comezará a descarga do ficheiro na carpeta que se seleccionou.

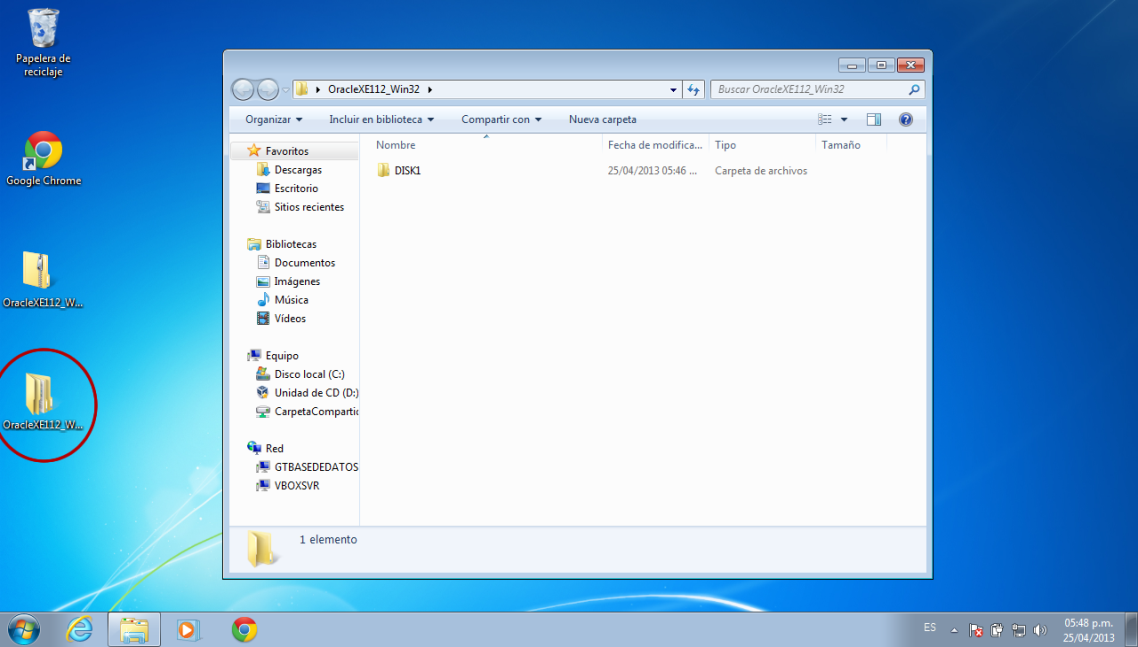


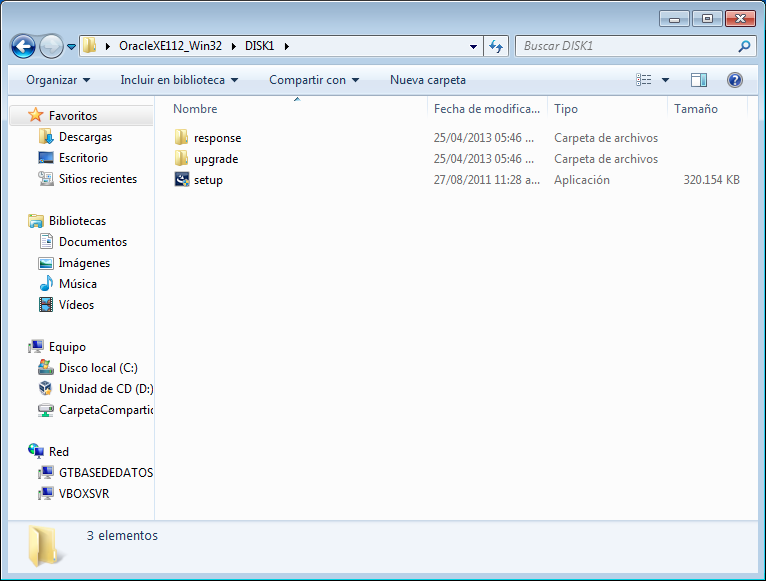
Antes de comezar coa instalación hai que asegurarse que esta instalada, como mínimo, a versión 6 do JDK de Java. Unha vez confirmado que está instalado o JDK, comezase coa instalación propiamente dita.

O primeiro que hai que facer é descomprimir o ficheiro descargado, pois como se ve, trátase dun ficheiro .zip.

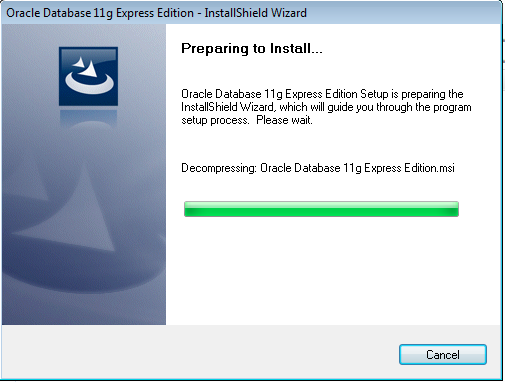


Logo de descomprimir o arquivo vese que hai unha nova carpeta co mesmo nome do arquivo descargado. Dentro dela atópase outra carpeta "*Disk1"* e dentro atopase un ficheiro executable de nome *"setup"*. Faise doble-click en dito ficheiro.

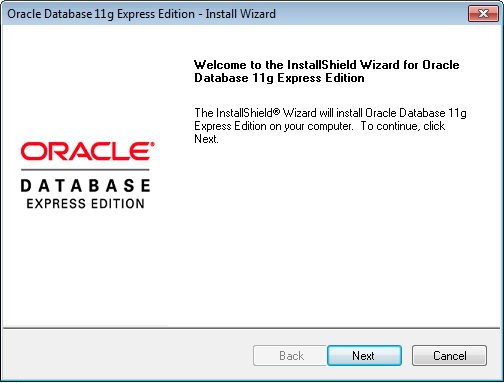


****

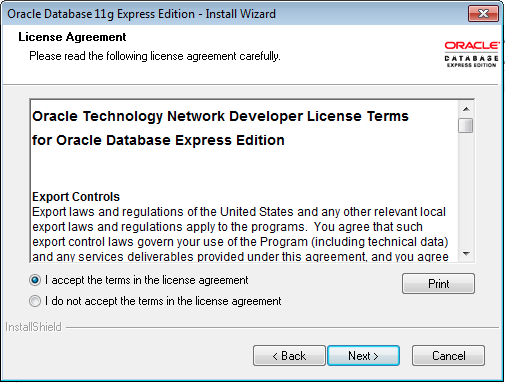
Iniciarase un proceso de descompresión para preparar e iniciar a instalación.



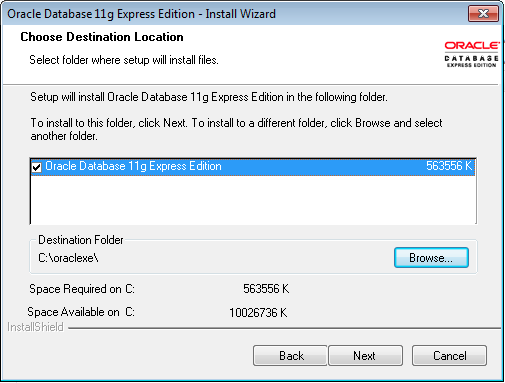
Ao terminar o proceso de preparación, aparece a primeira pantalla do proceso de instalación dando a benvida. Para continuar selecciónase *"Next"* ou *"Seguinte".*



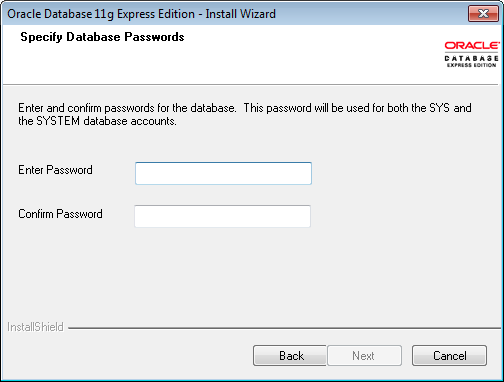
A seguinte pantalla mostra os acordos de licenza, é dicir, as condicións coas que Oracle permitirá o uso de Oracle Database Express Edition. Para continuar acéptanse os términos da licenza marcando *"I accept the terms in the license agreement"* e logo en *"Next"* ou *"Seguinte".*



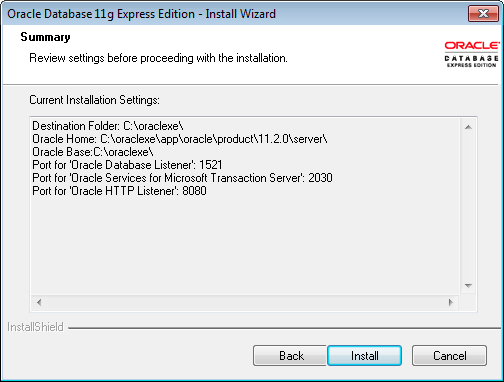
Débese indicar agora en que carpeta do equipo se quere instalar Oracle XE. A situación por defecto é en *"C:"* nun cartafol chamado *"oraclexe".* Se non se quere esta situación, pélsase no botón *"Browse..."* e selecciónase a situación desexada. Para este caso deixarase a situación por defecto. Para continuar faise click en "*Next*" ou en "*Seguinte*".



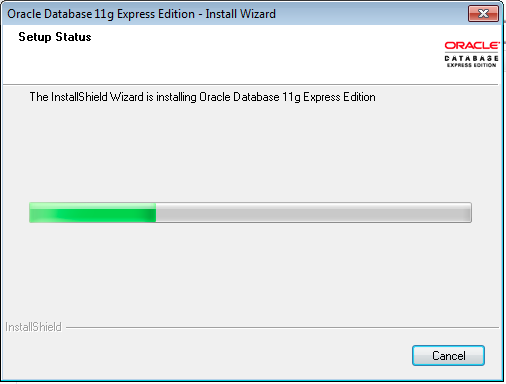
A seguinte pantalla pide un contrasinal para os usuarios SYS e SYSTEM da base de datos.



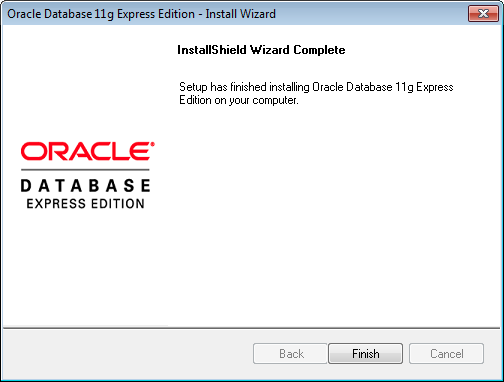
A continuación aparece a pantalla para comezar o proceso final de instalación. Prémese no botón *"Install".*



Iníciase o proceso de instalación. O proceso pode durar uns minutos.



Ao terminar o proceso aparece unha pantalla coa confirmación da instalación. Para terminar co proceso selecciónase "*Finish*".

[](http://1.bp.blogspot.com/--uWbZ1WQNv0/UXnDEsxappI/AAAAAAAAAmo/-DNVjEFET0I/s1600/Captura+de+pantalla+2013-04-25+a+la(s)+18.57.05.png)

Configuración do contorno de traballo

Para configurar un contorno de traballo no cal desenvolver as distintas tarefas da actividade hai varias posibilidades:

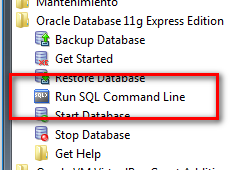
* **Crear unha nova base de datos**. Oracle XE crea por defecto a base de datos "*XE*". Se non se quere traballar con dita base de datos poderiase crear unha nova.

Oracle XE non conta con asistente de creación de bases de datos que traen outras versións de Oracle polo que habería que creala manualmente.

Nos seguintes enlaces podense ver os pasos para a creación manual dunha nova base de datos:

* https://docs.oracle.com/cd/B28359\_01/server.111/b28310/create003.htm
* http://oracleenespanol.blogspot.com.es/2012/02/rac-y-como-crear-manualmente-una-base.html
* **Traballar dentro dun *tablespace* da base de datos por defecto**.- Un *tablespace* é unha unidade lóxica de almacenamento dentro de unha base de datos Oracle. Para traballar cun *tablespace* específico o mellor é crear un novo usuario e asignarlle un *tablespace* por defecto.

Para poder usar unha consola para comandos SQL dentro de Oracle XE, podemos acceder á aplicación *Run SQL Command Line*



A sintaxe para crear un usuario en Oracle é:

create user NOME\_CUENTA

identified by CONTRASEÑA

default tablespace NOME\_ESPACIO\_DE\_TABLA;

Por exemplo, mediante :

create user MUSICAL

identified by MIMUSICA

default tablespace USERS;

Especifica o *tablespace* *USERS*; si se omitira, os obxectos creados polo usuario *MUSICAL*, almacenaríanse por defecto no tablespace *System* que é o lugar onde se atopa almacenado o dicionario de datos.

Agora hai que darlle privilexios ao usuario *MUSICAL* que lle permitan conectarse ao servidor Oracle. Para iso, úsase a sentenza:

grant connect to MUSICAL;

Agora, *MUSICAL* pode conectarse á instancia Oracle, pero non poderá crear ningún tipo de obxectos como táboas, índices ou vistas. Para elo, úsase a sentenza:

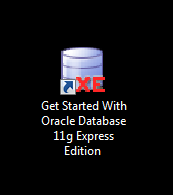
grant resource to MUSICAL;

Agora si se pode traballar coa conta *MUSICAL* e crear obxectos de base de datos.

En vez de traballar có *tablespace USERS* poderiase crear un *tablespace* novo. Na seguinte ligazón hai un exemplo cos pasos que se terían que seguir: http://www.tuinformaticafacil.com/oracle-10g/blog/oracle-10g/como-crear-un-tablespace-en-oracle-10g

* **Traballar con *Workspaces***.- Oracle XE incorpora o concepto de *workspaces*. Os *workspaces* pódense definir como bases de datos virtuais independentes e multiusuario, que garanten a privacidade dos obxectos e datos nelas almacenados respecto de outros *workspaces* existentes na mesma base de datos real. Cada *workspace* ten os seus propios *tablespaces* e posúe almacenamento lóxico (ficheiro de datos) independente.

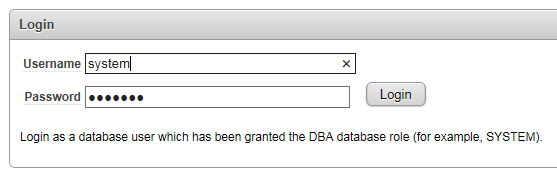
Para crear un novo *Workspace* utilizarase a ferramenta "*Oracle Aplication Express"* que ven incorporada coa versión Xpress Edition de Oracle. Para acceder a ela farase click na seguinte icona do Escritorio:

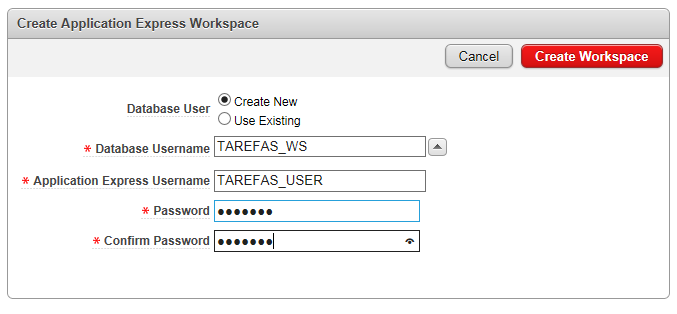


Unha vez dentro da aplicación, farase click en *"Aplication Express":*



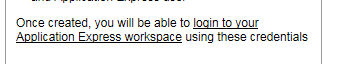
Na seguinte pantalla accédese co usuario *system* utilizando o contrasinal que se tecleou durante a instalación do software.

  
Introdúcense os parámetros do novo *workspace*:

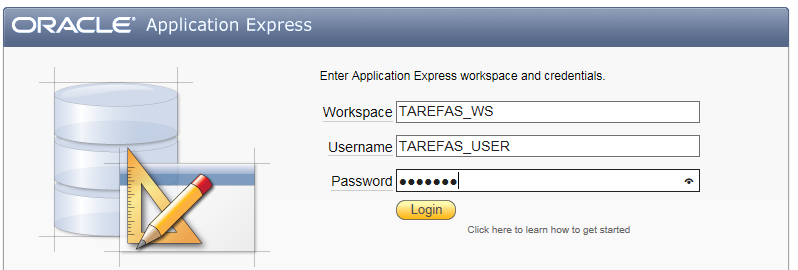


Sendo *database username* o nome do workspace e *application express username* o nome de usuario para o acceso ao *workspace* creado.

Para acceder ao novo espazo de traballo, faise click na seguinte ligazón no lado dereito da pantalla:

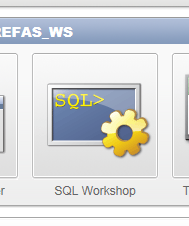


Aparecerá a seguinte pantalla na cal se introducen os parámetros anteriores:

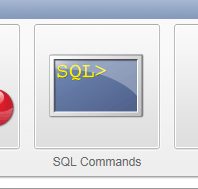


Xa dentro do *workspace* téñense varias utilidades gráficas que facilitan o traballo co mesmo.

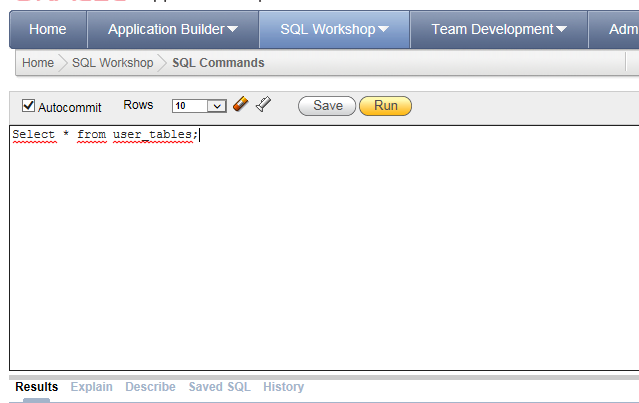
Para poder dispoñer dunha consola de SQL, farase click en *SQL Workshop:*



E posteriormente en *SQL Commands:*



Abrirase unha consola de SQL para poder traballar no *workspace*



* + 1. Tarefa 2. Creación de tipos de dato obxecto

Dada a seguinte descrición de tipos de datos empregando unha linguaxe de definición de obxectos (ODL) do estándar ODMG[[2]](#footnote-4).

define type Direccion\_T:

tuple [calle:string,

ciudad:string,

prov:string,

codpos:string]

define class Cliente\_T

type tuple [clinum: integer,

clinomb:string,

direccion:Direccion\_T,

telefono: string,

fecha-nac:date]

operations edad():integer

Utiliza a cláusula CREATE TYPEpara crear os correspondentes tipos de dato obxecto *direccion\_t* e *cliente\_t* nunha base de datos Oracle. Inclúe tamén o corpo da función *edad().*

Solución

CREATE TYPE direccion\_t AS OBJECT (

calle VARCHAR2(200),

ciudad VARCHAR2(200),

prov CHAR(2),

codpos VARCHAR2(20));

CREATE TYPE cliente\_t AS OBJECT (

clinum NUMBER,

clinomb VARCHAR2(200),

direccion direccion\_t,

telefono VARCHAR2(20),

fecha\_nac DATE,

MEMBER FUNCTION edad RETURN NUMBER,

PRAGMA RESTRICT\_REFERENCES(edad,WNDS));

/

CREATE OR REPLACE TYPE BODY cliente\_t AS

MEMBER FUNCTION edad RETURN NUMBER IS

a NUMBER;

d DATE;

BEGIN

d:= today();

a:= d.año – fecha\_nac.año;

IF (d.mes < fecha\_nac.mes) OR

((d.mes = fecha\_nac.mes) AND (d.dia < fecha\_nac.dia))

THEN a:= a-1;

END IF;

RETURN a;

END;

END;

* + 1. Tarefa 3. Creación de tipos de datos colección

Nunha base de datos Oracle, crea os seguintes tipos de datos colección:

* Utilizando un VARRAY, crea un tipo de dato *nome\_rios* que nos permita almacenar os nomes dos ríos galegos. O tipo de dato dos elementos que forman a colección será VARCHAR2 de tamaño 50 e o tamaño da colección, xa que din que Galicia é o país dos mil ríos, será de 1000. Crea unha táboa *Rios* cunha columna que utilice o tipo de dato *nome\_rios* e introduce dous elementos nela.
* Utilizando agora unha NESTED TABLE, crea un tipo de datos *nome\_montañas* que nos permita almacenar os nomes das montañas galegas. O tipo de datos dos elementos volverá a ser VARCHAR2 de tamaño 50. Crea unha táboa *Montañas* cunha columna que utilice o tipo de dato *nome\_montañas* e introduce dous elementos nela.

Unha vez creados, inserta en ámbalas dúas coleccións un par de elementos.

Solución

CREATE TYPE nome\_rios AS VARRAY (1000) OF VARCHAR2(50);

CREATE TABLE Rios (ríos nome\_rios);

INSERT INTO Rios VALUES (nome\_rios('Eume','Fray Bermuz'));

CREATE TYPE nome\_montañas AS TABLE OF VARCHAR2(50);

CREATE TABLE Montañas (montañas nome\_montañas)

NESTED TABLE montañas STORE AS montañas\_tab;

INSERT INTO Montañas VALUES (nome\_montañas('Penafesa','Pena da Uz'));

* + 1. Tarefa 4. Creación de táboas de obxectos e táboas de columnas tipo obxecto

Apartado a

Crea un novo tipo de datos obxecto chamado *coches* que permita almacenar a seguinte información: número de matrícula, marca, modelo e Kms percorridos. Despois crea unha táboa chamada *Taller* que sexa de tipo *coches.*

Apartado b

Crea un táboa *debedas* que teña as seguintes dúas columnas:

* *Datos\_cliente* que será do tipo *cliente\_*t creado na tarefa 2.
* *Cantidade* que será de tipo NUMBER.

Solución

Apartado a

CREATE TYPE coches AS OBJECT (

num\_matricula VARCHAR2(10),

marca VARCHAR2(20),

modelo VARCHAR2(20),

kms NUMBER);

CREATE TABLE Taller OF coches;

Apartado b

CREATE TABLE debedas (

datos\_cliente cliente\_t,

cantidade NUMBER);

* + 1. Tarefa 5. Modificación de obxectos almacenados na base de datos

Apartado a

Realiza as seguintes accións sobre a táboa *Taller* creada na tarefa anterior.

* Introduce dous novos coches con tódolos seus datos.
* Modifica os Kms dun dos coches anteriores.
* Borra un dos coches creados.

Apartado b

Realiza as seguintes accións sobre a táboa *debedas* creada na tarefa anterior.

* Introduce un novo cliente con tódolos seus datos.
* Modifica a dirección do cliente introducido.
* Borra o cliente creado.

Solución

Apartado a

INSERT INTO Taller VALUES (coches('9898GHT','Audi','A6','56789'));

INSERT INTO Taller VALUES (coches ('4498HTB','Renault','Clio','156895'));

/

UPDATE Taller t

SET t.kms='80000'

WHERE t.num\_matricula='9898GHT';

/

DELETE FROM Taller t

WHERE t.num\_matricula='9898GHT';

Apartado b

INSERT INTO debedas VALUES (cliente\_t('12', 'Pedro', direccion\_t('c/Real, nº5','Fene','CO','15456'), '981313131', '12/12/1983'),'250000');

/

UPDATE debedas d

SET d.datos\_cliente.direccion=direccion\_t('c/Vigo, nº5','Fene','CO','15456')

WHERE d.datos\_cliente.clinum='12';

/

DELETE FROM debedas d

WHERE d.datos\_cliente.clinum='12';

* + 1. Tarefa 6. Consultas sobre táboas de obxectos e táboas de columnas tipo obxecto

Apartado a

Realiza unha consulta para seleccionar as matrículas de todos os coches *Renault* da táboa *Taller*.

Apartado b

Realiza unha consulta para seleccionar os nomes de todos clientes (*clinomb*) da táboa *debedas* que deban unha *cantidade* maior de 5000.

Solución

Apartado a

SELECT t.num\_matricula FROM Taller t

WHERE t.marca='Renault';

Apartado b

SELECT d.datos\_cliente.clinomb FROM debedas d

WHERE d.cantidade>5000;

1. Materiais
   1. Documentos de apoio ou referencia

* VELA, Belén. *Diseño de bases de datos objeto-relacionales con UML*. LIBROS DYKINSON, 2005. 9788499823461.
* https://es.wikipedia.org/wiki/Base\_de\_datos\_objeto-relacional
* http://juancarlosgarcia.es/gsdaw/BBDD/BD07.pdf
* http://www.v-espino.com/~chema/daw1/tutoriales/oracle/BDO-R.pdf
* http://www.xtec.cat/~iguixa/materialsGenerics/DAMDAW\_M02\_UF4\_UV\_BDOR.pdf
* http://personales.unican.es/zorrillm/BDAvanzadas/Teoria/bda-t3-trans-ObjetoRelacional.pdf
* http://informatica.uv.es/iiguia/DBD/Teoria/capitulo\_4.pdf
* http://exa.unne.edu.ar/depar/areas/informatica/dad/DAD/Presentaciones/BD\_Objeto\_Relacionales\_Relacionales\_Extendidas\_y\_Otros\_Tipos\_Especiales\_de\_BD.pdf
  1. Recursos didácticos
* Material didáctico proporcionado polo profesorado.
* Proxector.
* Ordenadores con sistemas operativos Linux e/ou Windows e conexión a Internet.

1. Avaliación
   1. Criterios de avaliación

|  |  |  |  |
| --- | --- | --- | --- |
| Criterios de avaliación seleccionados  para esta actividade | Tipo de  evidencia de aprendizaxe | Instrumento de avaliación | Peso na cualificación  da UD |
| * CA9.1 Identificáronse as características das bases de datos obxecto-relacionais. | * Exame de resposta preguntas cortas. | * **PE.1.**- Cuestionario de preguntas de resposta breve sobre as características das bases de datos obxecto-relacionais. | 15% |
| * CA9.2 Creáronse tipos de datos obxecto, os seus atributos e os seus métodos. | * Documento de rexistro de operacións. | * **LC.1**.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de tipos de datos obxecto, atributos e métodos. | 25% |
| * CA9.3 Creáronse táboas de obxectos e táboas de columnas tipo obxecto. | * Documento de rexistro de operacións. | * **LC.2**.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de táboas de obxectos e táboas de columnas tipo obxecto. | 15% |
| * CA9.4 Creáronse tipos de datos colección. | * Documento de rexistro de operacións. | * **LC.3**.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de tipos de datos colección. | 10% |
| * CA9.5 Realizáronse consultas. | * Documento de rexistro de operacións. | * **LC.4**.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas para a consulta de información almacenada en táboas de obxectos e táboas de columnas tipo obxecto. | 15% |
| * CA9.6 Modificouse a información almacenada mantendo a integridade e a consistencia dos datos. | * Documento de rexistro de operacións. | * **LC.5**.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas para modificación da información almacenada en táboas de obxectos e táboas de columnas tipo obxecto. | 20% |

* 1. Modelos de probas

Modelo de proba combinada para CA9.1, CA9.2, CA9.3, CA9.4, CA9.5 e CA9.6

Proponse unha proba combinada para os instrumentos de avaliación PE.1, LC.1, LC.2, LC.3, LC.4 e LC.5. A correspondencia dos distintos apartados da proba cós criterios de avaliación é a seguinte:

* CA9.2: Apartado a.
* CA9.4: Apartado b.
* CA9.3: Apartados c, d.
* CA9.5: Apartado f.
* CA9.6: Apartado e.
* CA9.1: Apartado g.

Texto da proba

Parte práctica

Utilizando o Oracle XE instalado no teu equipo, realiza as accións solicitadas nos seguintes apartados (a - f).

Nun documento aparte captura e pega as sentenzas empregadas en cada un dos apartados.

Apartado a

Crea un novo tipo de dato obxecto *persoa* para almacenar a información relativa a varias persoas. Os datos que interesa almacenar son: DNI, Nome, Apelidos, Idade, Dirección, Data de nacemento, Sexo e Estado Civil.

Ademais deberás definir a especificación e o corpo dun método *saber\_idade* que devolverá a idade da persoa.

Utiliza os tipos de datos que consideres oportunos para cada un dos atributos do obxecto.

Apartado b

* Utilizando un VARRAY, crea un tipo de dato *nome\_xogadores* que nos permita almacenar os nomes dos xogadores da alineación inicial dun equipo de fútbol. O tipo de dato dos elementos que forman a colección será VARCHAR2 de tamaño 20 e o tamaño da colección será de 11.
* Crea unha táboa *Equipos\_futbol* con dúas columnas: unha para almacenar o nome do equipo (VARCHAR2(20) e outra columna que utilice o tipo de dato *nome\_xogadores* para indicar a alineación inicial do equipo.
* Introduce un equipo na táboa *Equipo\_futbol.*

Apartado c

Crea unha táboa *Familiares* que teña as seguintes dúas columnas:

* *datos\_familiar*  que será do tipo *persoa* creado no apartado a.
* *parentesco* que será de tipo VARCHAR2(20).

Apartado d

Crea unha táboa *Amigos* do tipo *persoa* creado no apartado a.

Apartado e

* Na táboa *Familiares* creada no apartado C:
* Introduce dous novos familiares con tódolos seus datos.
* Modifica o estado civil dun dos familiares introducido.
* Borra un dos familiares introducidos.

Apartado f

* Executa unha sentenza SQL para seleccionar o nome e apelidos de tódolos tíos e tías existentes na táboa *Familiares* creada no apartado C.

Parte teórica

Apartado g

Cita as principais características e melloras que aportan as Bases de Datos Obxecto-Relacionais.

Exemplo de solucións

Apartado a

CREATE TYPE persoa AS OBJECT (

dni VARCHAR2(9),

nome VARCHAR2(40),

apelidos VARCHAR2(100),

direccion VARCHAR2(300),

sexo VARCHAR2(8),

idade NUMBER,

data\_nac DATE,

estado\_civil VARCHAR2(10),

MEMBER FUNCTION saber\_idade RETURN NUMBER,

PRAGMA RESTRICT\_REFERENCES(saber\_idade,WNDS));

CREATE OR REPLACE TYPE BODY persoa AS

MEMBER FUNCTION saber\_idade RETURN NUMBER IS

BEGIN

RETURN idade;

END;

END;

Apartado b

CREATE TYPE nome\_xogadores AS VARRAY (11) OF VARCHAR2(20);

CREATE TABLE Equipos\_futbol (

equipo VARCHAR2(20),

alineacion nome\_xogadores);

INSERT INTO Equipos\_futbol VALUES (

'Depor',

nome\_xogadores ('Jose', 'Pepe', 'Dani', 'Maki', 'Peter', 'Fran', 'Pedro', 'Johnny', 'Mark', 'Felgosa', 'Drene'));

Apartado c

CREATE TABLE Familiares (

datos\_familiar persoa,

parentesco VARCHAR2(20));

Apartado d

CREATE TABLE Amigos OF persoa;

Apartado e

INSERT INTO Familiares VALUES (persoa('12345678Z', 'Pedro', 'Rodríguez López', 'C/real nº5. A Coruña', 'home', '34', '12/12/1970','solteiro'),'tío');

INSERT INTO Familiares VALUES (persoa('87654321Z', 'María', 'López Pérez', 'C/real nº8. A Coruña', 'muller', '54', '12/12/1950','casada'),'aboa');

/

UPDATE Familiares f

SET f.datos\_familiar.estado\_civil='casado'

WHERE f.datos\_familiar.dni='12345678Z';

/

DELETE FROM Familiares f

WHERE f.datos\_familiar.dni='87654321Z';

Apartado f

SELECT f.datos\_familiar.nome, f.datos\_familiar.apelidos

FROM Familiares f

WHERE f.parentesco='tío' OR f.parentesco='tía';

Apartado g

As principais características e melloras que aportan as Bases de Datos Obxecto-Relacionais son:

* O usuario pode crear os seus propios tipos de datos.
* Permiten crear métodos para os tipos de datos definidos polos usuarios proporcionando flexibilidade e seguridade.
* Xestionan tipos de datos complexos cun esforzo mínimo.
* Poden albergar parte da aplicación software no servidor de base de datos.
* Permiten almacenar datos complexos dunha aplicación dentro da BDOR sen necesidade de forzar os tipos de datos tradicionais.
* Son compatibles en sentido ascendente coas bases de datos relacionais tradicionais. É dicir, pódense pasar as aplicacións sobre bases de datos relacionais ao novo modelo sen ter que reescribilas. Posteriormente, si así se quixera, poderíanse ir adaptando as aplicacións e bases de datos para que utilicen as funcións orientadas a obxectos.

Exemplo de listas de cotexo LC.1, LC.2, LC.3, LC.4 e LC.5

Exemplo de lista de cotexo para CA9.2

Proponse a seguinte lista de cotexo para o instrumento de avaliación LC.1.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de tipos de datos obxecto, atributos e métodos.

| Nome | Data | |
| --- | --- | --- |
| CA9.2 Creáronse tipos de datos obxecto, os seus atributos e os seus métodos. |  | |
| Indicadores | SI | Non |
| As sentenzas execútanse correctamente e sen erros |  |  |
| Creouse o tipo de dato obxecto solicitado |  |  |
| Creáronse os atributos solicitados |  |  |
| Creouse o método solicitado |  |  |
| Seleccionáronse adecuadamente os tipos de atributos en función do seu contido |  |  |
| A nomenclatura dos distintos elementos creados coinciden cós solicitados. |  |  |

Exemplo de lista de cotexo para CA9.3

Proponse a seguinte lista de cotexo para o instrumento de avaliación LC.2.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de táboas de obxectos e táboas de columnas tipo obxecto.

| Nome | Data | |
| --- | --- | --- |
| CA9.3 Creáronse táboas de obxectos e táboas de columnas tipo obxecto. |  | |
| Indicadores | SI | Non |
| As sentenzas execútanse correctamente e sen erros |  |  |
| Creouse a táboa de obxectos solicitada |  |  |
| Creouse a columna tipo obxecto dentro da táboa segundo se solicitou |  |  |
| Os tipos de datos dos elementos coinciden cós solicitados |  |  |
| A nomenclatura dos distintos elementos creados coinciden cós solicitados. |  |  |

Exemplo de lista de cotexo para CA9.4

Proponse a seguinte lista de cotexo para o instrumento de avaliación LC.3.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas de creación de tipos de datos colección.

| Nome | Data | |
| --- | --- | --- |
| CA9.4 Creáronse tipos de datos colección. |  | |
| Indicadores | SI | Non |
| As sentenzas execútanse correctamente e sen erros |  |  |
| Creouse o tipo de dato solicitado |  |  |
| Creouse a táboa solicitada |  |  |
| Introduciuse un elemento na táboa |  |  |
| Os tipos de datos dos elementos coinciden cós solicitados |  |  |
| A nomenclatura dos distintos elementos creados coinciden cós solicitados. |  |  |

Exemplo de lista de cotexo para CA9.5

Proponse a seguinte lista de cotexo para o instrumento de avaliación LC.4.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas para a consulta de información almacenada en táboas de obxectos e táboas de columnas tipo obxecto.

| Nome | Data | |
| --- | --- | --- |
| CA9.5 Realizáronse consultas. |  | |
| Indicadores | SI | Non |
| A sentenza execútase correctamente e sen erros |  |  |
| A sentenza execútase sobre a táboa indicada |  |  |
| A sentenza devolve a información solicitada |  |  |

Exemplo de lista de cotexo para CA9.6

Proponse a seguinte lista de cotexo para o instrumento de avaliación LC.5.- Documento de rexistro de proba práctica consistente na elaboración de sentenzas para modificación da información almacenada en táboas de obxectos e táboas de columnas tipo obxecto.

| Nome | Data | |
| --- | --- | --- |
| CA9.6 Modificouse a información almacenada mantendo a integridade e a consistencia dos datos. |  | |
| Indicadores | SI | Non |
| As sentenzas execútanse correctamente e sen erros |  |  |
| Introducíronse novos datos na táboa |  |  |
| Modificáronse os datos da táboa segundo se solicitou |  |  |
| Borráronse datos da táboa |  |  |

1. http://gtbasededatos.blogspot.com.es/2013/04/como-descargar-oracle-xe-11g.html [↑](#footnote-ref-3)
2. https://en.wikipedia.org/wiki/Object\_Data\_Management\_Group [↑](#footnote-ref-4)