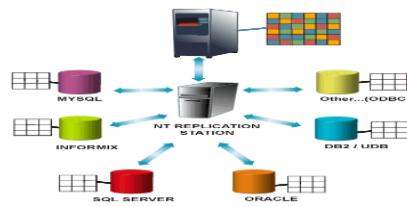


SISTEMAS DE BASES DE DATOS II

Ing. Henry J. Lezcano

Departamento de Sistemas de Información

CAPITULO I
DESARROLLO DE MODELOS DE BASES DE DATOS



SISTEMAS DE BASES DE DATOS II

CONTENIDO

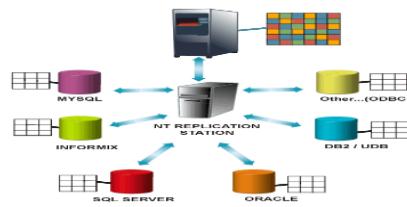
CAPITULO I. DESARROLLO DEL MODELO DE BASE DE DATOS

1.1- DESARROLLO DE PROBLEMAS DEL MODELOS E/R VS EL MODELO LOGICO RELACIONAL.

1.2- DESARROLLO DE PROBLEMAS DEL MODELOS E/R EXTENDIDO Y SU CORRESPONDIENTE MODELO LOGICO RELACIONAL

1.3 PROCESO DE NORMALIZACION PARA EL MODELO LOGICO RELACIONAL

MODELADO DE DATOS

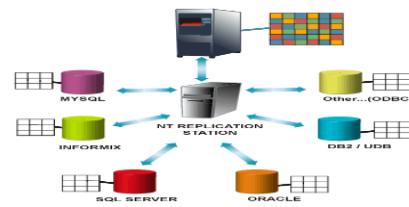


Los **modelos** se utilizan en todo tipo de ciencias. Su finalidad es la de simbolizar una parte del mundo real de forma que sea más fácilmente manipulable. En definitiva es un esquema mental (conceptual) en el que se intentan reproducir las características de una realidad específica.

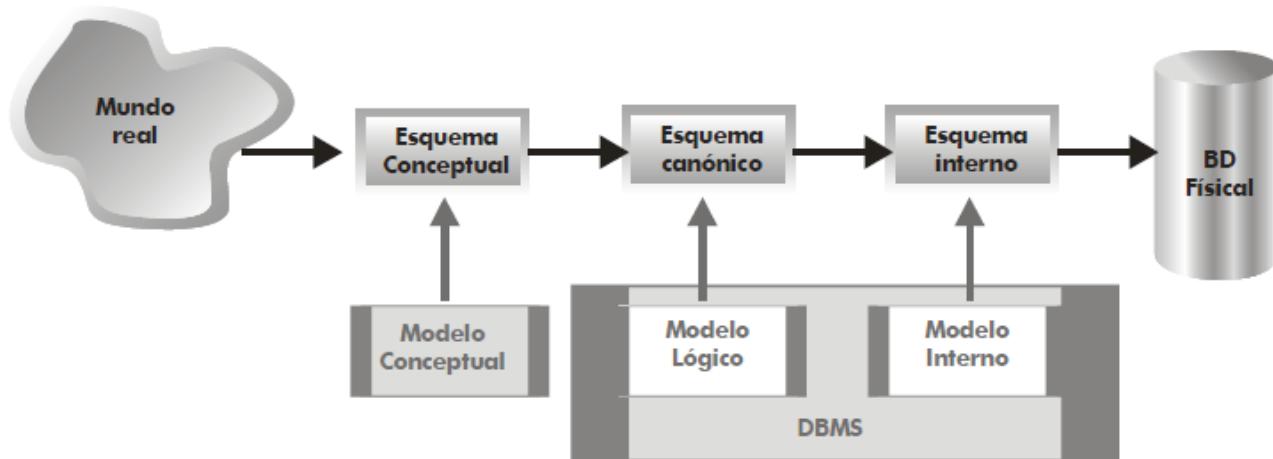
En el caso de los **modelos de datos**, lo que intentan reproducir es una información real que deseamos almacenar en un sistema informático.

Se denomina **esquema** a una descripción específica en términos de un modelo de datos. El conjunto de datos representados por el esquema forma la base de datos.

Un **Modelo de Datos** no es más que una colección de herramientas conceptuales que se utilizan para describir los datos, las relaciones existentes entre ellos, la semántica asociada a los mismos y las restricciones de consistencia.



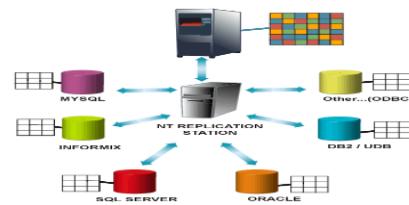
CLASIFICACION DE LOS MODELOS DE DATOS



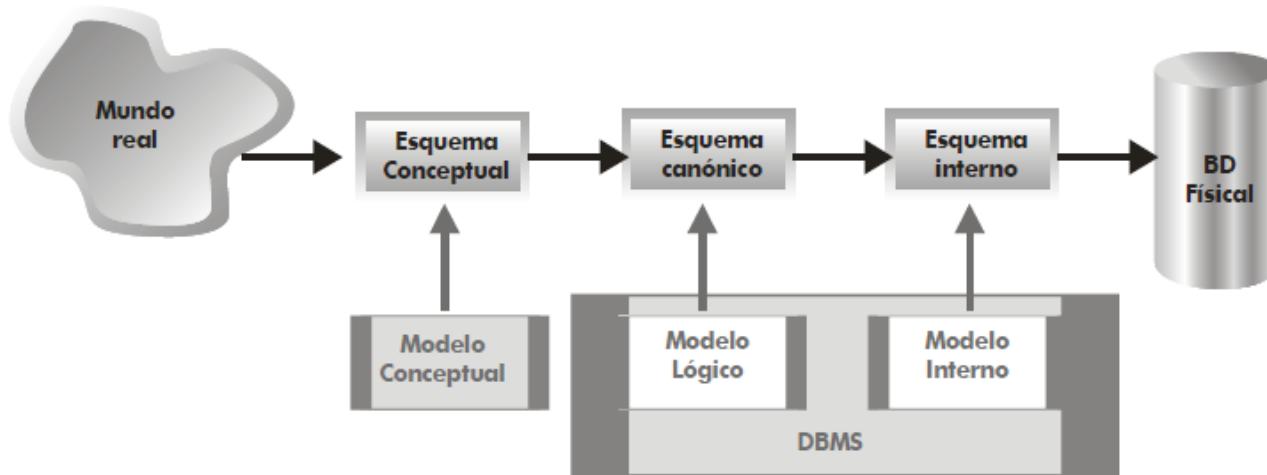
En la ilustración mostrada aparecen los distintos esquemas que llevan desde el mundo real a la base de datos física.

Los elementos de ese esquema son:

- **Mundo real.** Contiene la información tal cual la percibimos como seres humanos. Es el punto de partida
- **Esquema conceptual.** Representa el modelo de datos de forma independiente del DBMS que se utilizará.

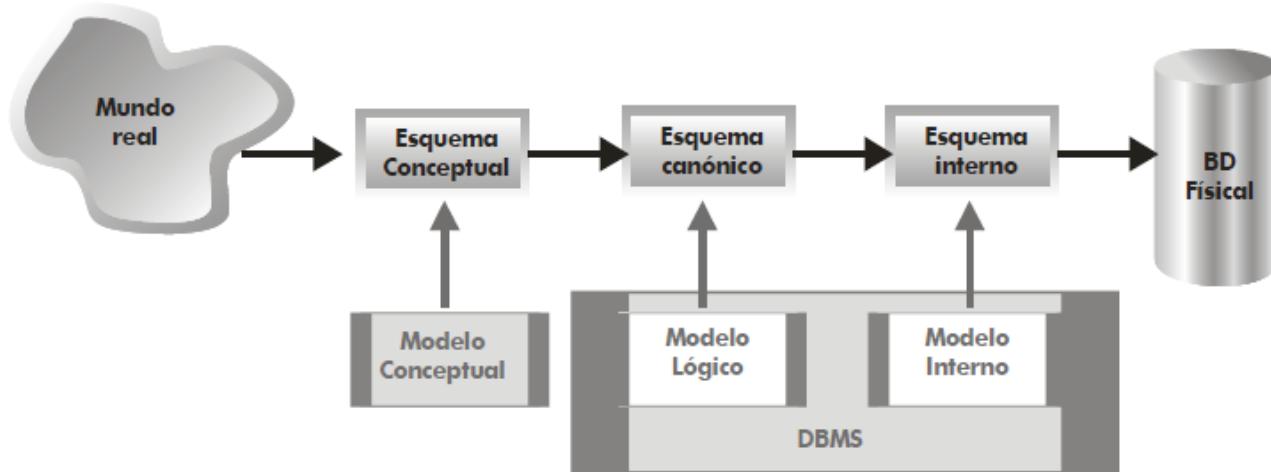


CLASIFICACION DE LOS MODELOS DE DATOS



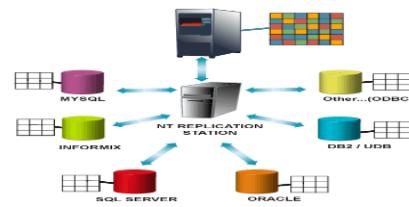
- **Esquema canónico (o de base de datos).** Representa los datos en un formato más cercano al del computador
- **Esquema interno.** Representa los datos según el modelo concreto de un sistema gestor de bases de datos (por ejemplo Oracle)
- **Base de datos física.** Los datos tal cual son almacenados en disco.

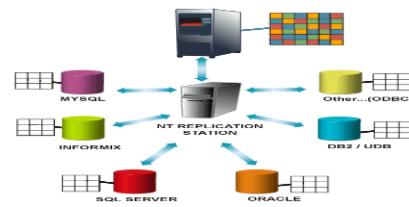
CLASIFICACION DE LOS MODELOS DE DATOS



Para conseguir estos esquemas se utilizan modelos de datos. El paso entre cada esquema se sigue con unas directrices concretas. Estas directrices permiten adaptar un esquema hacia otro.

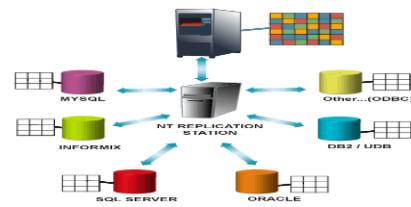
Los dos modelos fundamentales de datos son el **conceptual** y el **lógico**. Ambos son conceptuales en el sentido de que convierten parámetros del mundo real en abstracciones que permiten entender los datos sin tener en cuenta la física de los mismos.





DIFERENCIA ENTRE EL MODELO LOGICO Y EL MODELO CONCEPTUAL

- **El modelo conceptual** es independiente del DBMS que se vaya a utilizar. El lógico depende de un tipo de SGBD en particular
- **El modelo lógico** es más cercano al computador
- **El Modelo Conceptual** es más cercano al usuario, el lógico forma el paso entre el informático y el sistema.



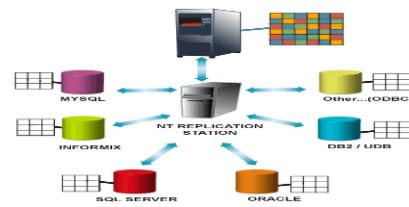
EJEMPLOS DE MODELOS DE DATOS

Algunos ejemplos de modelos conceptuales son:

- ❖ **Modelo E/R**
- ❖ **Modelo RM/T**
- ❖ **Modelos semántico**

Ejemplos de modelos lógicos son:

- ❖ **Modelo relacional**
- ❖ **Codasyl**
- ❖ **Jerárquico**

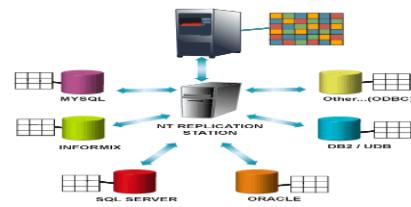


MODELO ENTIDAD RELACION

El Modelo Entidad Relación sirve para crear **esquemas conceptuales** de bases de datos. De hecho es prácticamente un estándar para crear esta tarea.

Se le llama modelo E/R e incluso EI (Entidad / Interrelación). Sus siglas más populares son las E/R por que sirven para ambos idiomas, el inglés y el español.

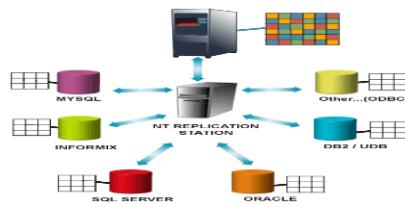
Inicialmente sólo se incluían los conceptos de entidad, relación y atributos. Después se añadieron otras propuestas (atributos compuestos, generalizaciones,...) que forman el llamado **modelo entidad relación extendido** (se conoce con las siglas ERE)



MODELO ENTIDAD RELACION

El Modelo de Base de Datos que nos permite obtener un esquema conceptual basado en diagrama, el cual utiliza los siguientes elementos:

- 1 • Entidad
- 2 • Relación
- 3 • Cardinalidad
- 4 • Roles
- 5 • Atributos
- 6 • Identificadores

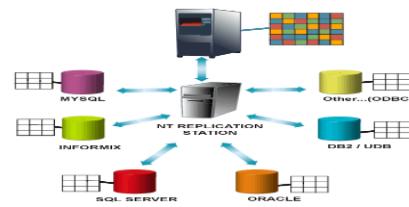


MODELO ENTIDAD RELACION ENTIDADES

ENTIDAD

Corresponde a cualquier objeto u elemento (real o abstracto) acerca del cual se pueda almacenar información en la base de datos. Ejemplos de entidades son Manuel, el numero de factura 42456, el matricula de un auto 3452BCW.

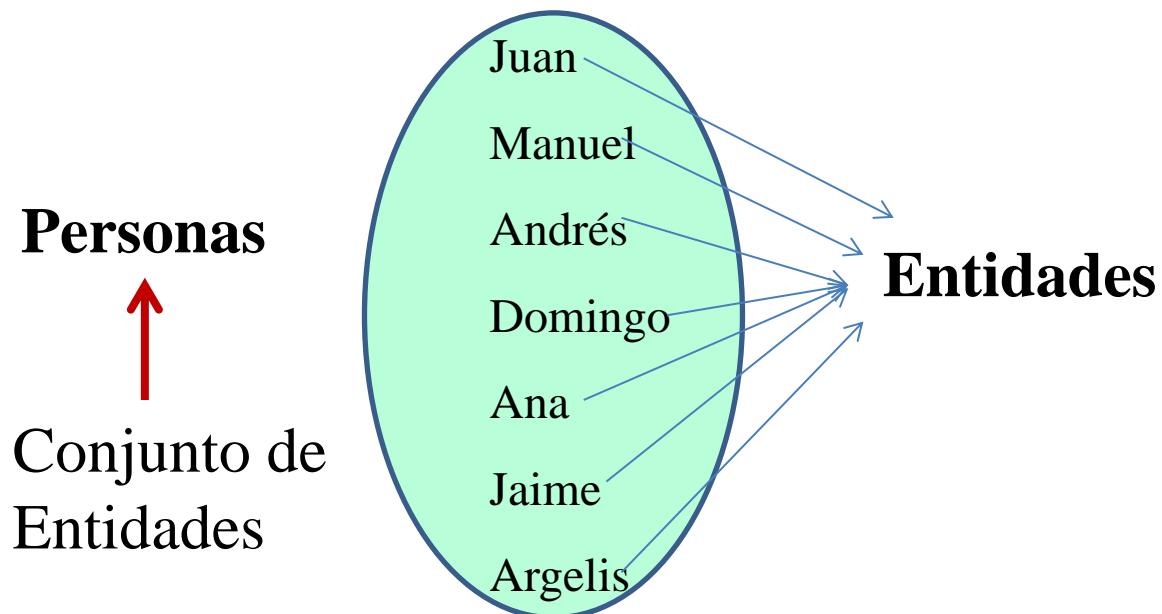
Una entidad no es un propiedad concreta sino un objeto que puede poseer múltiples propiedades (atributos).



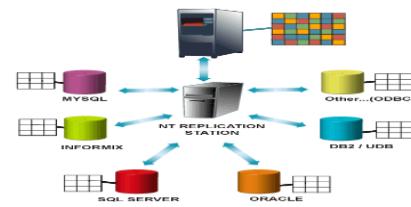
MODELO ENTIDAD RELACION ENTIDADES

CONJUNTO DE ENTIDADES

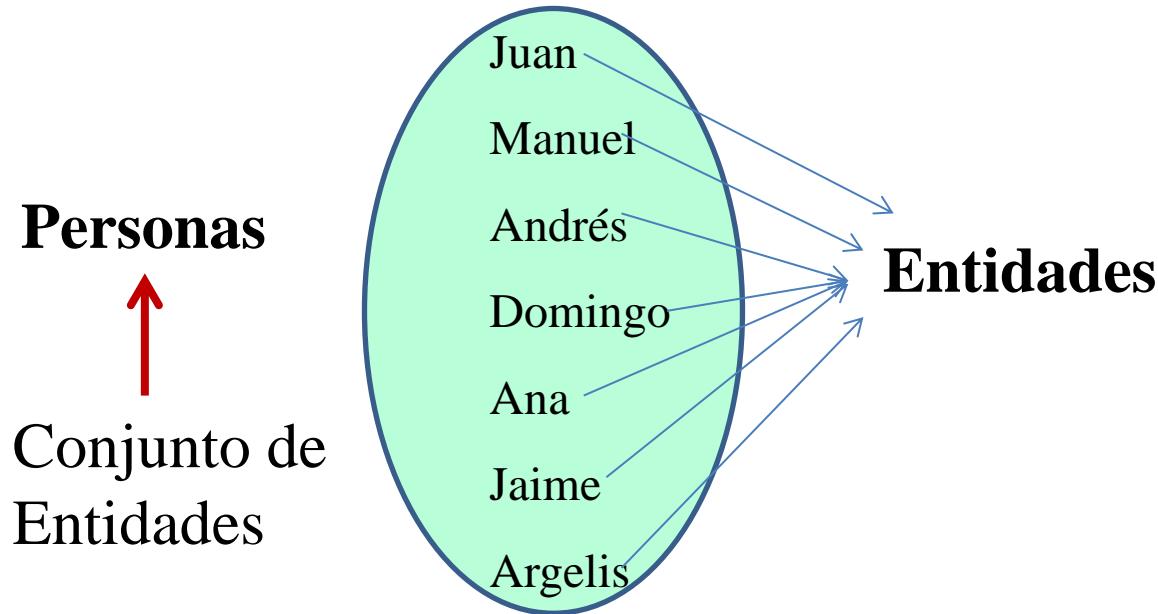
Las entidades que poseen las mismas propiedades forman conjuntos de entidades. Ejemplos de conjuntos de entidades son los conjuntos: **personas, facturas, autos,...**



MODELO ENTIDAD RELACION ENTIDADES

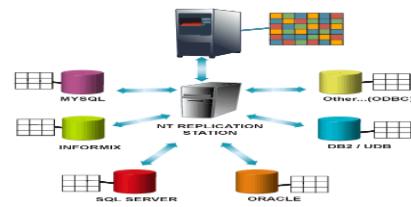


CONJUNTO DE ENTIDADES



En la actualidad se suele llamar **entidad** a lo que anteriormente se ha definido como conjunto de entidades. De este modo hablaríamos de la entidad PERSONAS. Mientras que cada persona en concreto sería una **ocurrencia** o un **ejemplar** de la entidad **persona**.

MODELO ENTIDAD RELACION ENTIDADES



REPRESENTACION GRAFICA DE LAS ENTIDADES

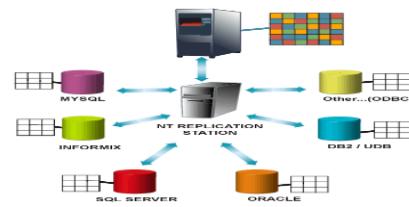
En el modelo entidad relación los conjuntos de entidades se representan con un rectángulo dentro del cual se escribe el nombre de la entidad. Por ejemplo **Personas, facturas, Autos**

PERSONAS

FACTURAS

AUTOS

MODELO ENTIDAD RELACION ENTIDADES



TIPOS DE ENTIDADES

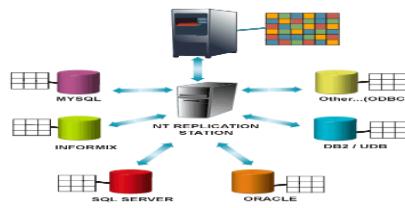
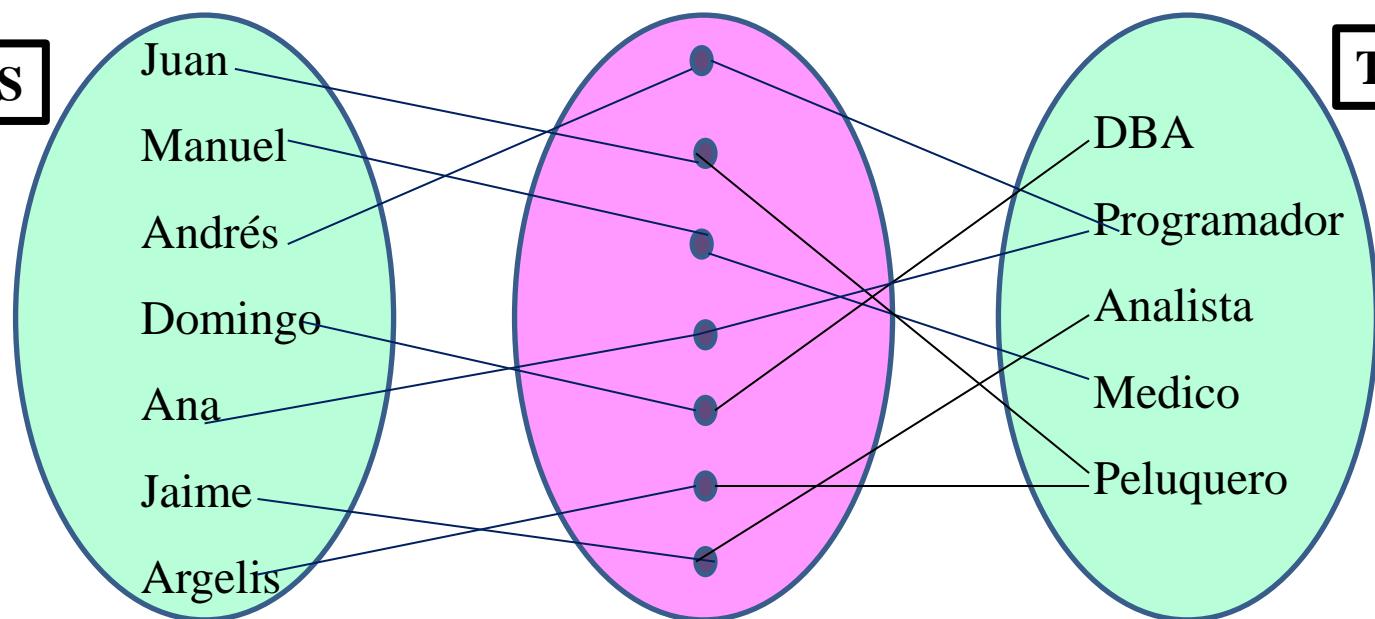
- **Entidades Fuertes.** Son las entidades normales que tienen existencia por sí mismas sin depender de otras. Su representación gráfica es la indicada en la PPT anterior.
- **Entidades Débiles.** Su existencia depende de otras. Por ejemplo la entidad **tarea laboral** sólo podrá tener existencia si existe la entidad **trabajo**. Las entidades débiles se presentan de esta forma:

TAREAS LABORALES

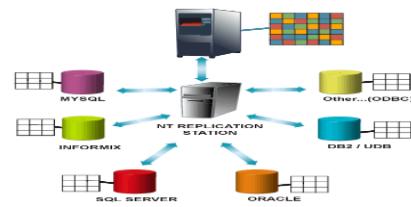
MODELO ENTIDAD RELACION RELACIONES

RELACION

Representan asociaciones entre entidades. Es el elemento del modelo que permite relacionar en sí los datos del modelo. Por ejemplo, en el caso de que tengamos una entidad personas y otra entidad trabajos. Ambas se realizan ya que las personas trabajan y los trabajos son realizados por personas:



MODELO ENTIDAD RELACION RELACIONES



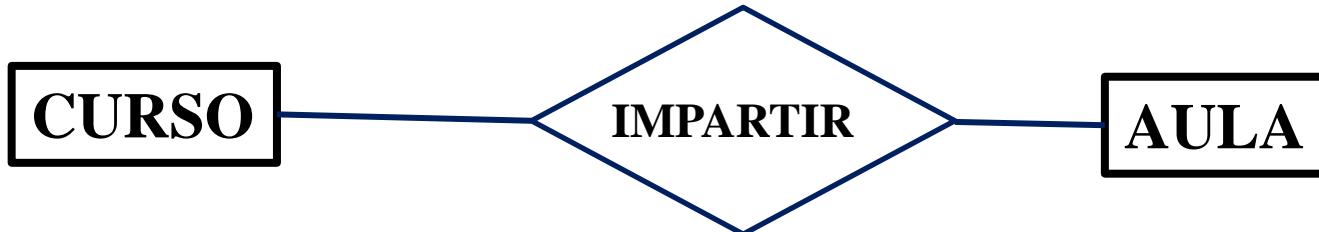
REPRESENTACION GRAFICA DE RELACIONES

La representación gráfica de las relaciones se realiza con un rombo al que se le unen líneas que se dirigen a las entidades, las relaciones tienen nombre (se suele usar un **verbo**). En el ejemplo anterior podría usarse como nombre de relación, trabajar:

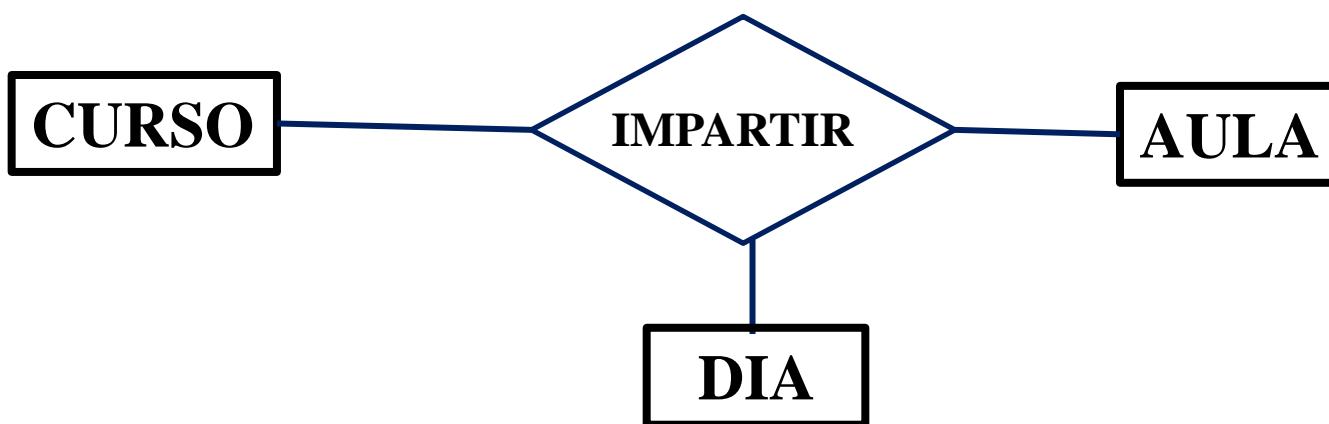


MODELO ENTIDAD RELACION RELACIONES

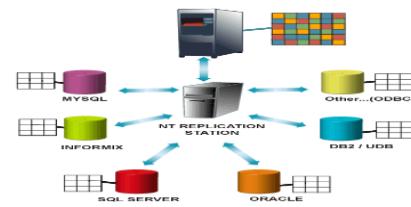
GRADO DE LAS RELACIONES ' Numero entidades involucradas'



RELACION BINARIA

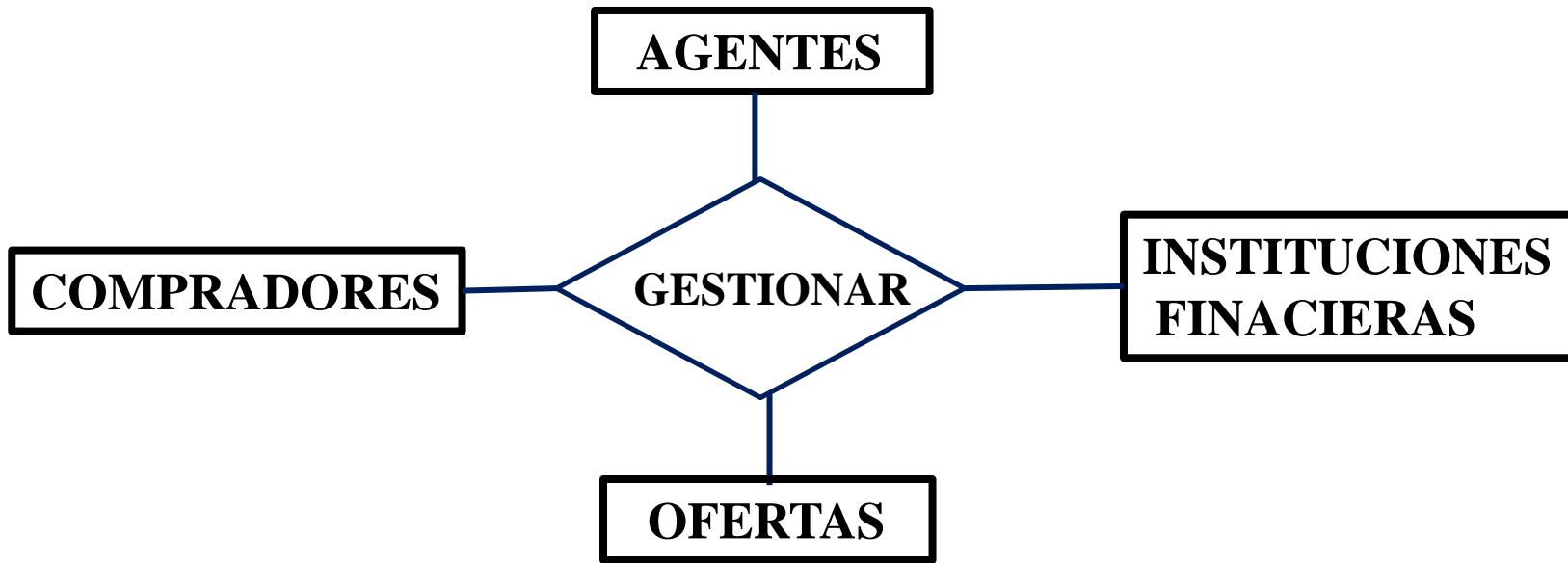


RELACION TERNARIA

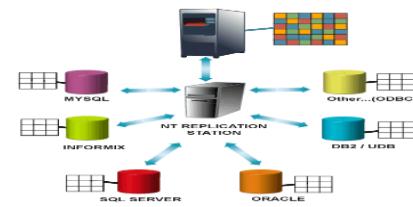


MODELO ENTIDAD RELACION RELACIONES

GRADO DE LAS RELACIONES ' Numero entidades involucradas'

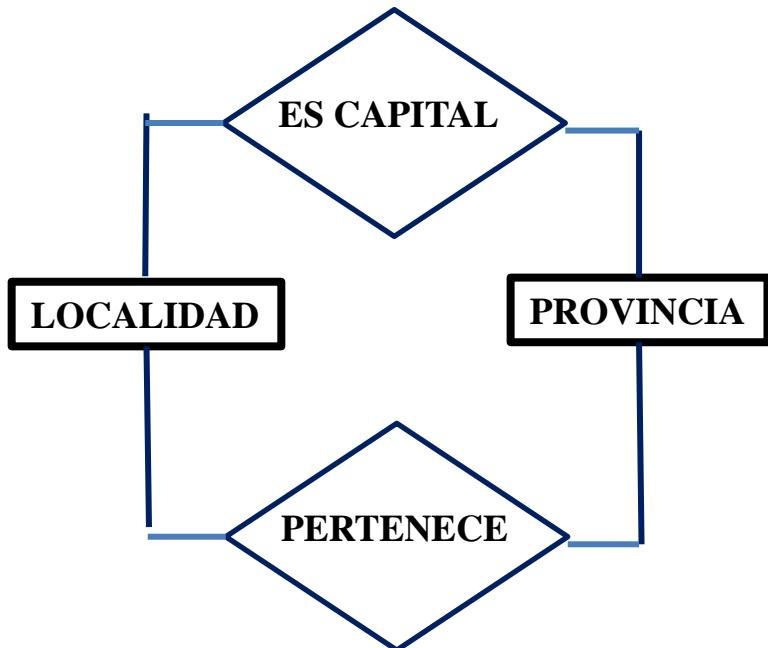


RELACION CUATERNARIA

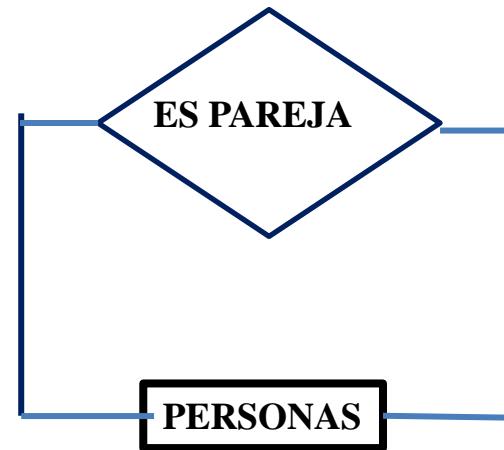


MODELO ENTIDAD RELACION RELACIONES

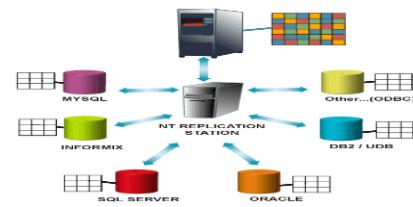
GRADO DE LAS RELACIONES



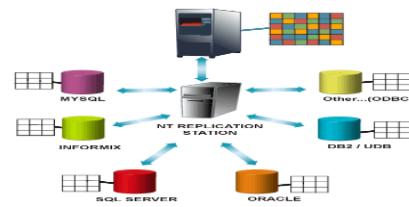
RELACION DOBLE



RELACION REFLEXIVA



MODELO ENTIDAD RELACION RELACIONES



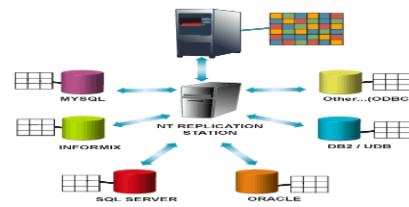
CARDINALIDAD DE LAS RELACIONES

Indica el número de relaciones en las que una entidad puede aparecer. Se anota en términos de:

- Cardinalidad Mínima.** Indica el número mínimo de asociaciones en las que aparecerá cada ejemplar de la entidad (el valor que se anota es de cero o uno)

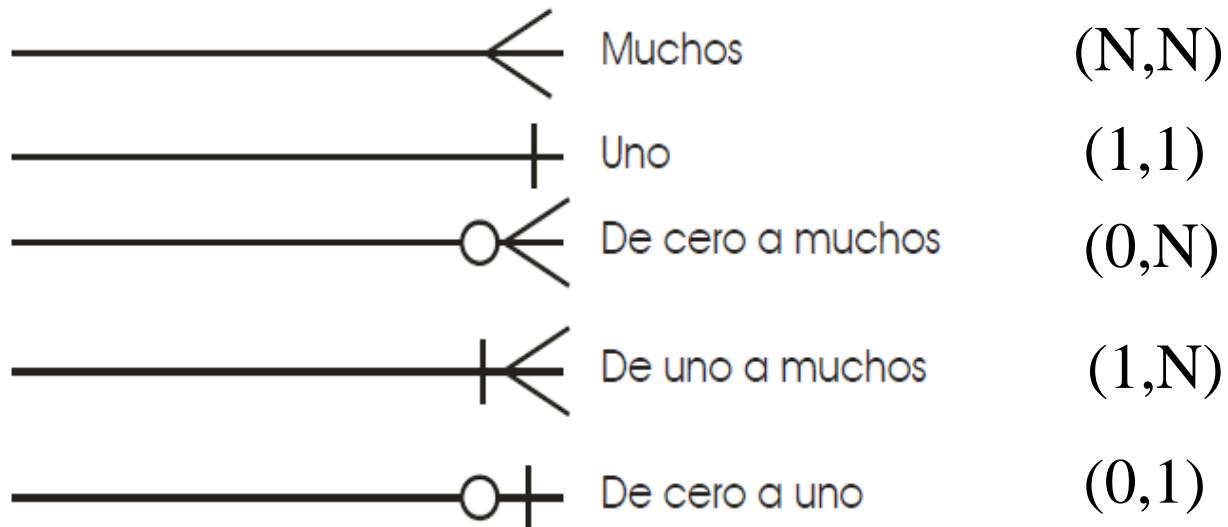
- Cardinalidad Máxima.** Indica el número máximo de relaciones

MODELO ENTIDAD RELACION RELACIONES

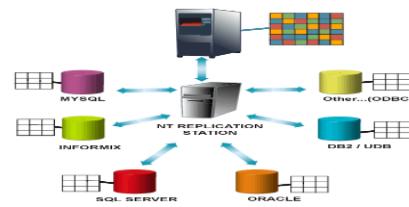


CARDINALIDAD DE LAS RELACIONES

En los esquemas entidad / relación la cardinalidad se puede indicar de muchas formas. Actualmente una de las más usadas es esta:

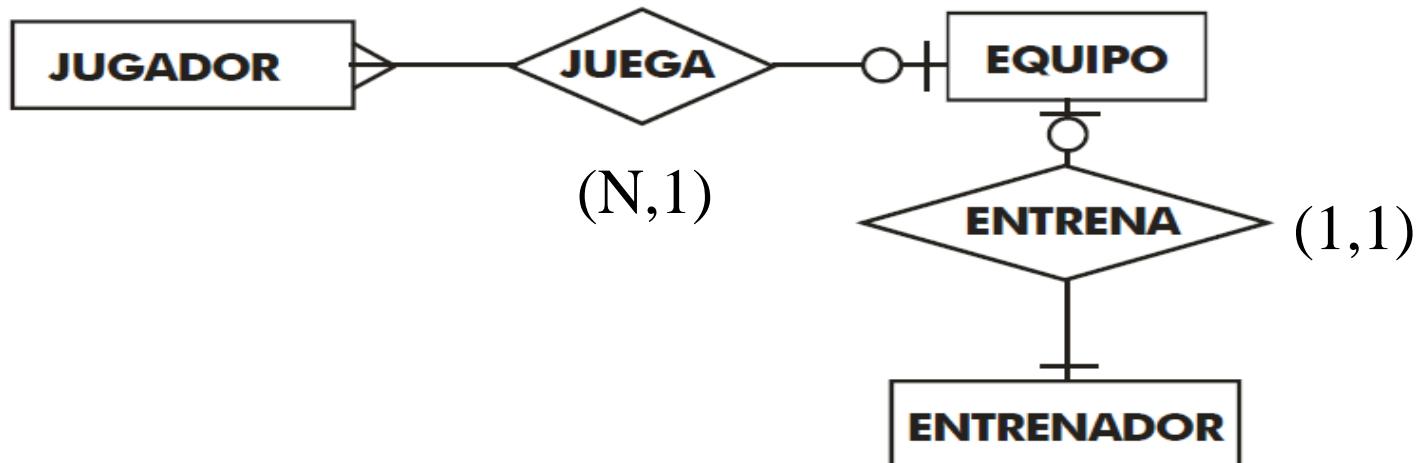


MODELO ENTIDAD RELACION RELACIONES



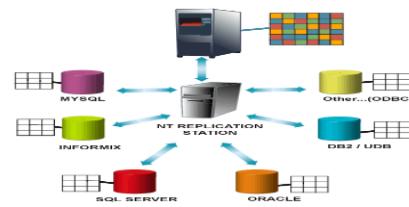
Primer Caso de Estudio: Cardinalidad de las Relaciones

Para un hecho que corresponde a los equipos de football. Cual seria su solución usando el modelo conceptual de base de datos E/R y la cardinalidad. Se sabe que muchos jugadores puede jugar en uno o cero equipos, que un entrenador puede entrenar a uno o cero equipos.



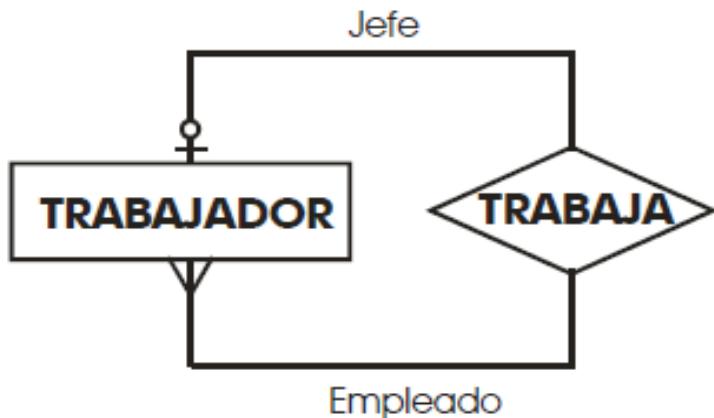
Para el caso, cada equipo cuanta con varios jugadores. un jugador juega como mucho en un equipo y podría no jugar en ninguno. Cada entrenador entrena a un equipo (podría no entrenar a ninguno), el cual tiene un solo entrenador

MODELO ENTIDAD RELACION RELACIONES



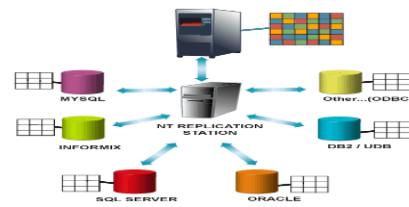
ROLES

En ocasiones en las líneas de la relación se indican **roles**. Los roles representan el papel que juega una entidad en una determinada relación. Ejemplo:



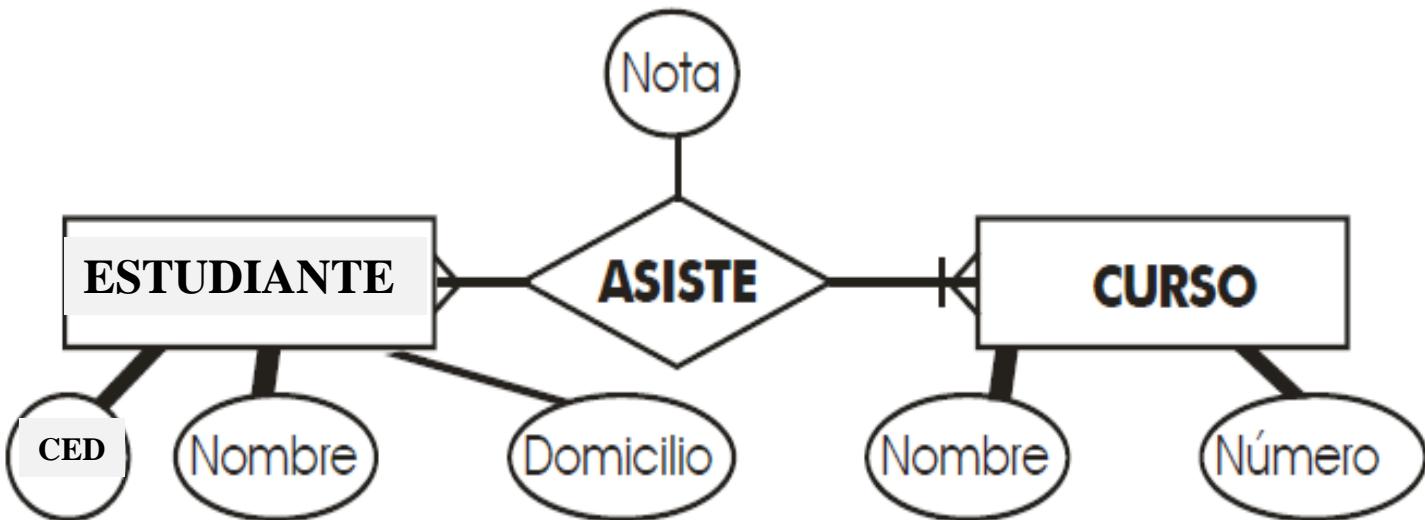
MODELO ENTIDAD RELACION

ENTIDADES Y RELACIONES



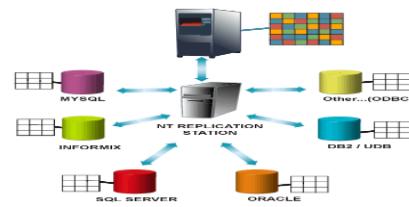
ATRIBUTOS

Los Atributos describen propiedades de las entidades y las relaciones. En el modelo se representan con un círculo, dentro del cual se coloca el nombre del atributo. Ejemplo:



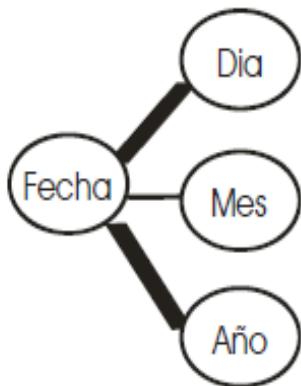
MODELO ENTIDAD RELACION

ENTIDADES Y RELACIONES



TIPOS DE ATRIBUTOS

Compuestos



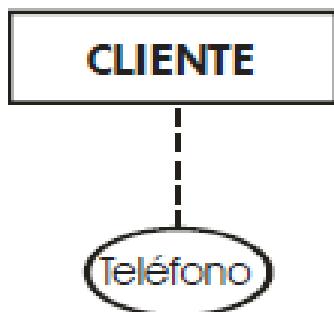
Múltiples

Puedes tomar varios Valores



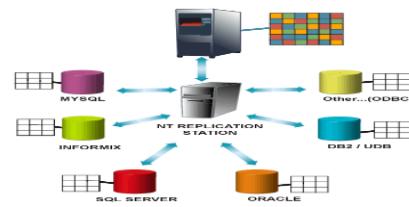
Opcionales

Los son si pueden tomar valores Nulos



MODELO ENTIDAD RELACION

ENTIDADES Y RELACIONES



SEGUNDO CASO DE ESTUDIO

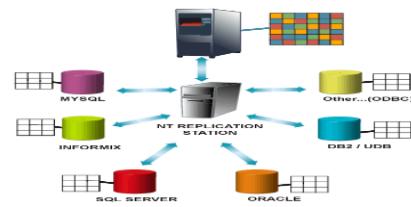
Obtenga el Modelo Conceptual E/R siguientes:

Una escuela cuenta una serie de **ALUMNO** de la cual tiene el registro de su *Núm_Matrícula, Nombre, FechaNacimiento, Teléfono*. De la **ASIGNATURA** que imparte se registra el *Código_asignatura, Nombre* de la misma. De los **PROFESOR** contratados se registra el *Id_Profesor, CIP_P, Nombre, Especialidad, Teléfono*.

Teniendo en cuenta:

- Un alumno puede estar matriculado de una o varias asignaturas.
- Además puede estar matriculado en la misma asignatura más de un curso escolar (si repite).
- Se quiere saber el curso escolar en el que cada alumno está matriculado de cada asignatura.
- En una asignatura habrá como mínimo 10 y como máximo 25 alumnos.
- Una asignatura es impartida por un único profesor.
- Un profesor podrá impartir varias asignaturas.

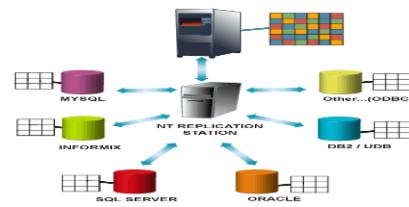
MODELO ENTIDAD RELACION EXTENDIDO



ENTIDADES VS RELACIONES ISA (ES UN)

Las relaciones de tipo is a son aquellas en las que una entidad se descompone en entidades especializadas. Hay dos tipos de entidades isa: **especializaciones** y **generalizaciones**.

MODELO ENTIDAD RELACION EXTENDIDO

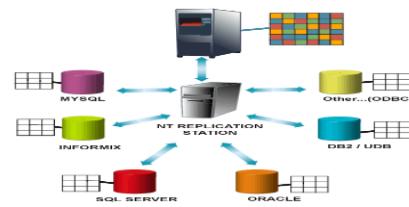


Generalización

Es una relación contenida que existe entre el nivel mas alto(superclase) y uno o mas conjunto de entidades de nivel mas bajo(subclase).

- La generalización permite que las entidades de nivel mas bajo hereden los atributos de entidades generalizadoras de mas alto nivel.
- La entidad general se llama superentidad y las otras subentidades.
- La superentidad normalmente tiene una clave principal distinta de las subentidades. (detalle mas importante para diferenciarlas de la relaciones ISA de especificación)

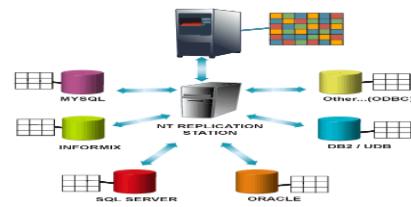
MODELO ENTIDAD RELACION EXTENDIDO



Generalización

- La generalización trata de eliminar la redundancia de atributos, al englobar los atributos semejantes. Las entidades de bajo nivel heredan todos los atributos correspondientes.
- Para representar este tipo de interrelación, se usa un triangulo invertido, con la base paralela al rectángulo que representa el **supertipo (Generalización)** y conectado a este y a los **subtipos (Especialización)** .
- Las cardinalidades siempre son (1,) en el supertipo y (0,1) en los subtipos.

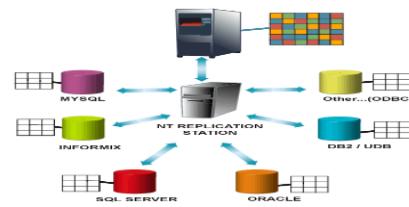
MODELO ENTIDAD RELACION EXTENDIDO



ESPECIALIZACIÓN

- El proceso por el que se definen las diferentes subclases de una superclase se conoce como especialización. Esto ocurre cuando partimos de una entidad que podemos dividir en subentidades para detallar atributos que varían en las mismas.
- Comparten clave con la superentidad y los atributos de la superclase se heredan en las subclases.

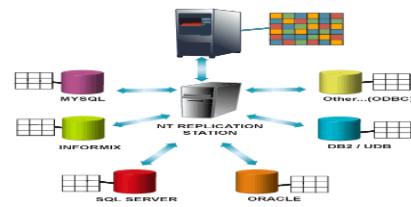
MODELO ENTIDAD RELACION EXTENDIDO



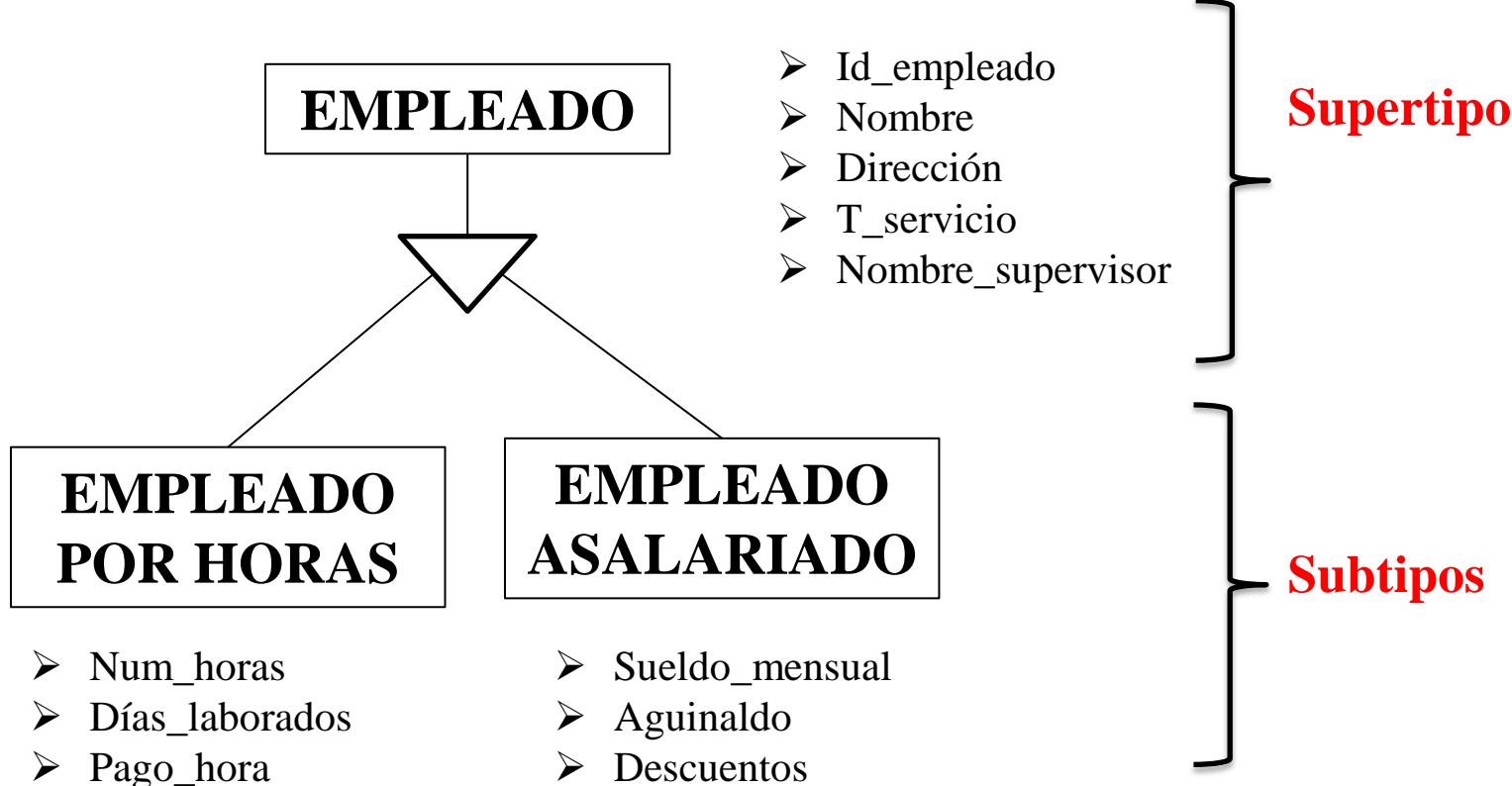
EJEMPLO No.1 Modelo Extendido....

En una compañía de ventas de productos para el hogar cuenta con la entidad empleado que tiene varios atributos como el nombre, dirección, teléfono, fecha de nacimiento, tiempo de servicio, etc. pero un empleado tiene la característica de que puede ser contratado por hora o permanente. Si es por hora necesitamos conocer cuantas horas trabajó en el mes y precio por hora para calcular el salario mensual. En cambio si es asalariado, ya tiene un salario mensual fijo.

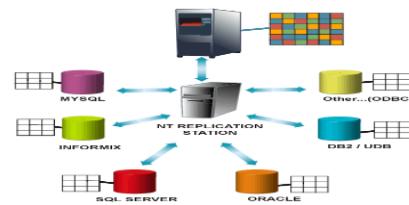
MODELO ENTIDAD RELACION EXTENDIDO



Resolución del Modelo E/R Extendido Ejemplo No. 1



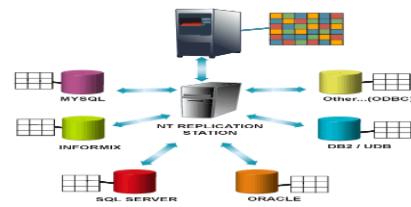
MODELO ENTIDAD RELACION EXTENDIDO



La generalización o la especialización se pueden distinguir por las claves.

- Si se comparte claves entre la superentidad y sus descendientes; se habla de especialización
- Si **no** se comparte claves entre la superentidad y sus descendientes; se habla de generalización.
- En la generalización cada entidad de alto nivel sebe ser también una entidad de bajo nivel. La especialización no tiene esa limitante.

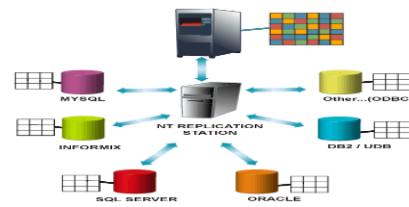
MODELO ENTIDAD RELACION EXTENDIDO



EJEMPLO No.2 Modelo Extendido....

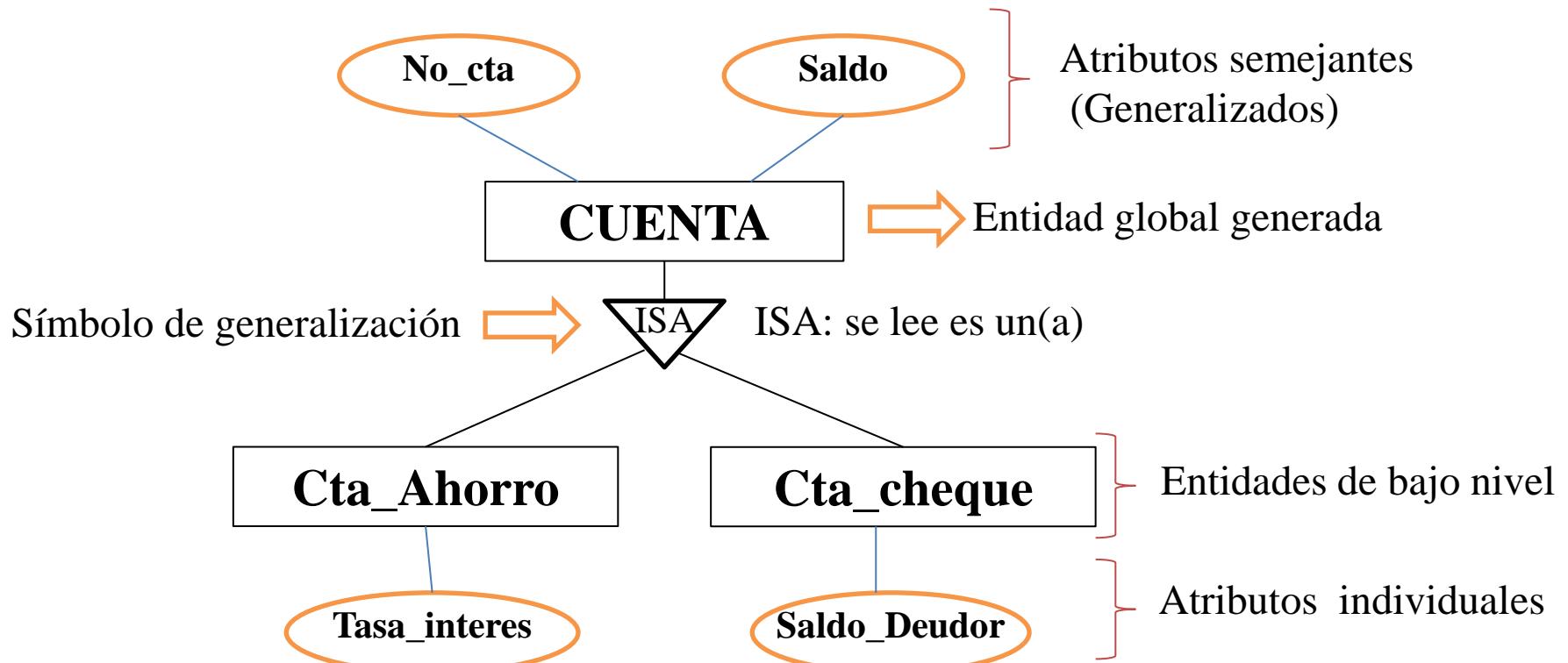
Banco de occidente para el manejo de proceso de negocio financiero de ahorro y prestamos cuenta con las entidades Cta_Ahorro y Cta_Cheques, ambas entidades tiene atributos semejantes de No_cta y Saldo, aunque además de estos dos atributos, Cta_Ahorro tiene el atributo de Tasa_Interes y Cta_Cheques el atributo Saldo_Deudor.

MODELO ENTIDAD RELACION EXTENDIDO

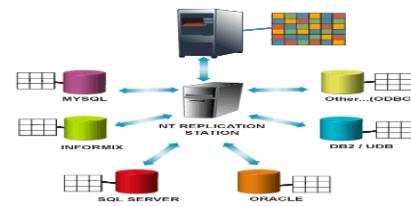


Resolución del Modelo E/R Extendido Ejemplo No. 2

De todos estos atributos podemos juntar(generalizar) No_cta Y Saldo que son iguales en ambas entidades.

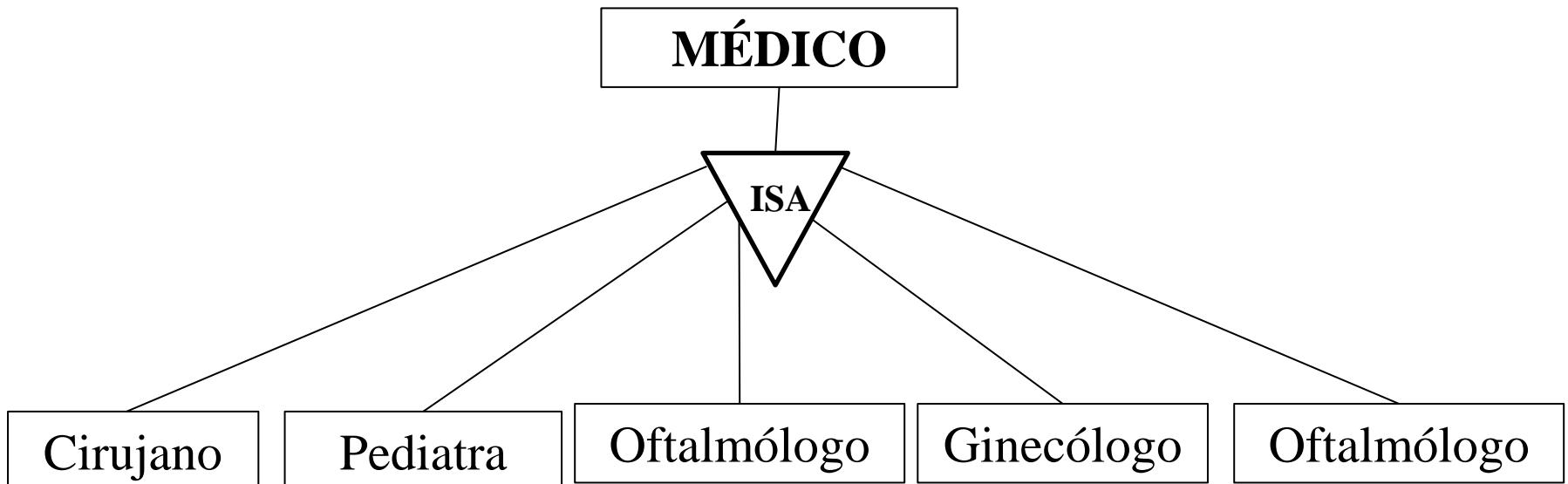


MODELO ENTIDAD RELACION EXTENDIDO

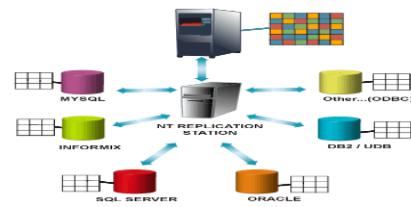


EJEMPLO No. 3

Se tiene la superclase Médico y las subclases Cirujano, Pediatra, Oftalmólogo, Ginecólogo, Dermatólogo. Entonces podemos decir que Cirujano y Pediatra, es un tipo de Médico.

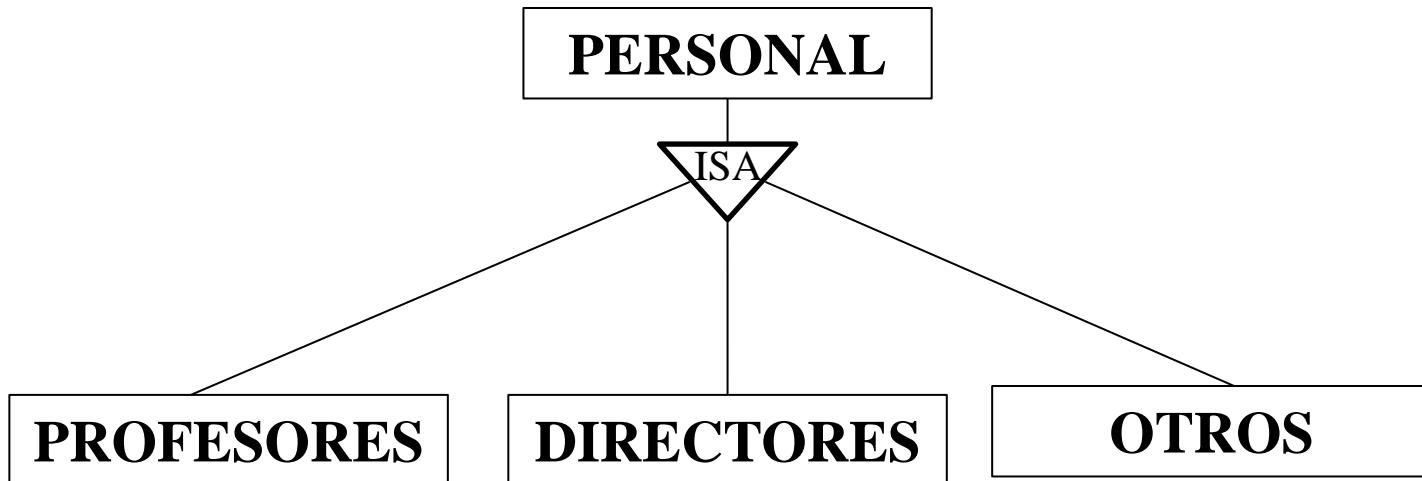


MODELO ENTIDAD RELACION EXTENDIDO

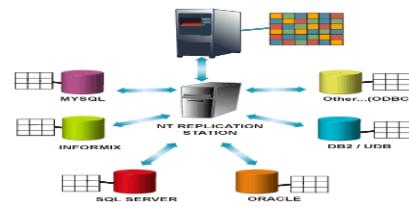


EJEMPLO No. 4

En cualquier caso la representación en el modelo es la misma, se representan con un triángulo que tiene el texto **ISA**. Ejemplo:



MODELO ENTIDAD RELACION EXTENDIDO



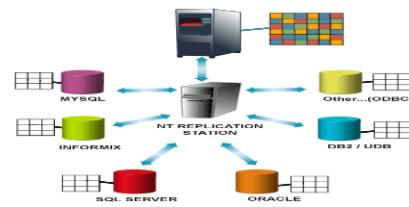
HERENCIAS

En estas relaciones se habla también de herencia, ya que tanto los profesores como los directores como los otros, heredan atributos de la entidad personal (se habla de la superentidad personal y de la subentidad profesores)

Se puede colocar un círculo (como el del número cero) en lado de la superentidad para indicar que es opcional la especialización, de otro modo se tomará como obligatoria (el personal tiene que ser alguna de esas tres cosas)

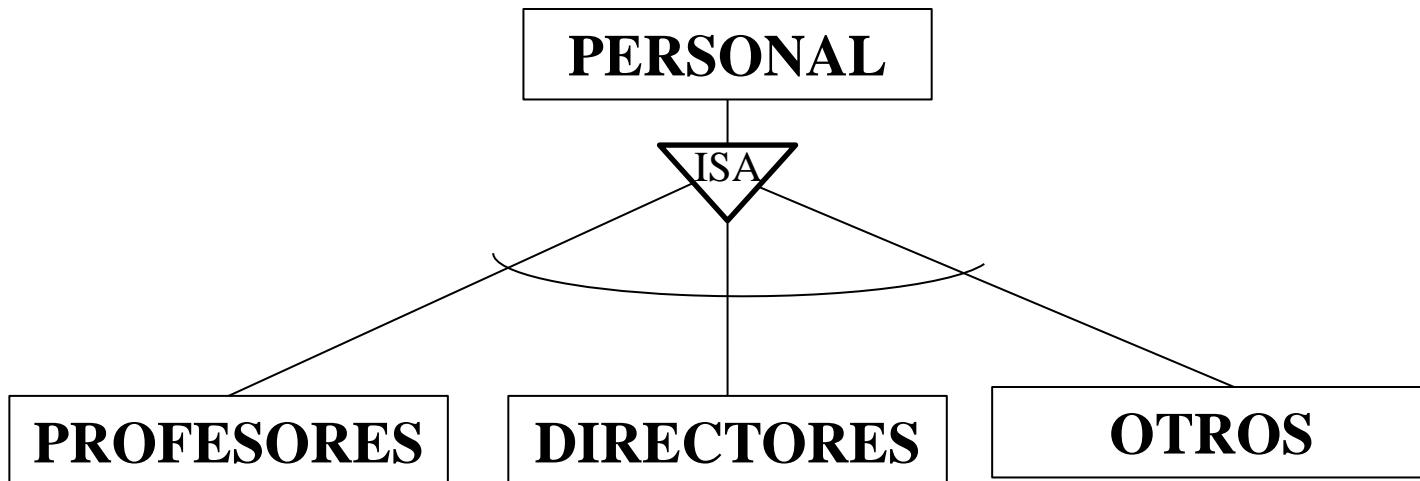
MODELO ENTIDAD RELACION

MODELO EXTENDIDO



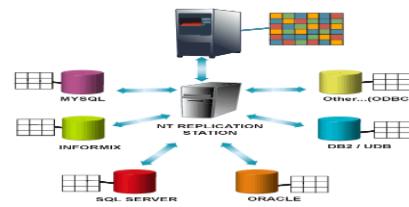
HERENCIAS

Se puede indicar también exclusividad. Esto ocurre cuando entre varias líneas hacia una relación, las entidades sólo pueden tomar una. Se representa con un ángulo en el diagrama:



En el diagrama mostrado el ángulo indica que el personal solo puede ser profesor o director u otros. No puede ser dos cosas a la vez.

MODELO ENTIDAD RELACION EXTENDIDO



CARACTERISTICAS

Generalización:

- Énfasis en las similitudes
- Cada instancia del supertipo es también una instancia de algún de los subtipos.

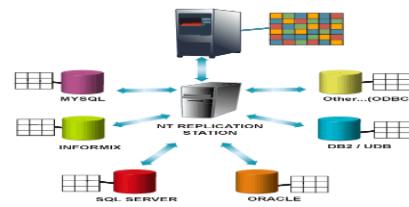
|

Especialización:

- Énfasis en las diferencias
- Alguna instancia del supertipo pueden no ser instancia de ningún subtipo

MODELO ENTIDAD RELACION

ENTIDADES Y RELACIONES



IDENTIFICADORES

Son atributos cuyos datos son únicos por cada fila o registro o tupla de la entidad o relaciones:

CLIENTES

CIP	Nombre	Apellido	Email	Celular	Dirección
8-225-331	Juan	Pérez	correo@gmail.com	6628-6752	Villa Lucia
4-212-333	Patricia	Medina	flia@hotmail.com		Ciudad capital
2-111-112	Patricia	Pérez		671-3333	Parita



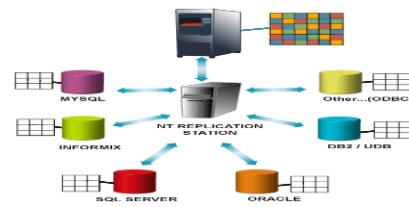
Identificador



Registros, Filas o Tuplas

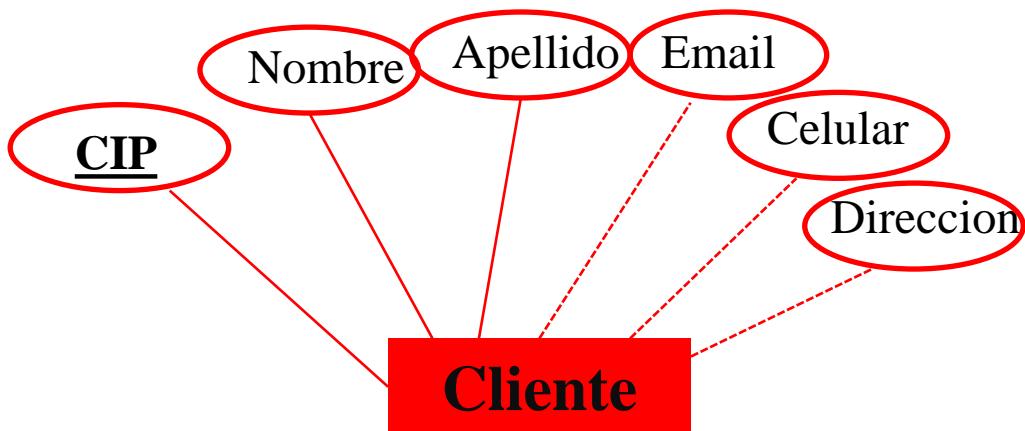
MODELO ENTIDAD RELACION

ENTIDADES Y RELACIONES

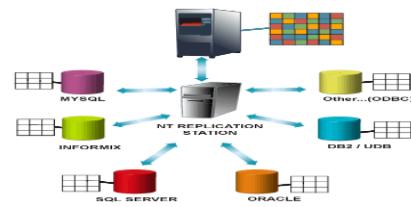


IDENTIFICADORES

Son atributos cuyos datos son únicos por cada fila o registro o tupla de la entidad o relaciones:

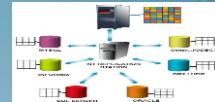


MODELO ENTIDAD RELACION ENTIDADES Y RELACIONES



Caso para desarrollo:

Una tienda vende una serie de **Productos** que contienen los siguientes datos: Marca, Modelo, Descripción, Ficha Técnica, Precio, inventario, Imagen. Cuenta con **Vendedores** los cuales atienden a los clientes y realizan las ventas con los siguientes datos CIP, nombre, edad, cargo, fecha de contrato, director al cual responden, sucursal en la cual trabajan y sus ventas. Los **Clientes** que visitan la tienda a los cuales se les realiza las ventas cuentan con los siguientes datos CIP, Nombre, Nombre del representante de venta asignado, emial, dirección, celular. En las **Sucursales** trabajan los vendedores y las dirige un vendedor con cargo de director, objetivos de ventas y las ventas totales. Se tienen las **Ventas** que contiene el detalle de los productos vendidos con los siguientes datos No de factura, fecha, el cliente al que se le efectuó la venta, los productos vendidos, el total de la venta, y el vendedor que realizó la venta.



SISTEMAS DE BASES DE DATOS II

Ing. Henry J. Lezcano

Departamento de Sistemas de Información y Control

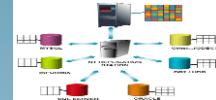
MODELO LOGICO RELACIONAL



✓ Definición de Modelo Relacional

- Este modelo está basado en el concepto de relación(tablas). Una relación(tablas) es un conjunto de n-tuplas o registros. Una tupla(registro), al contrario que un segmento, puede representar tanto entidades como interrelaciones N:M. Los lenguajes matemáticos sobre los que se asienta el modelo relacional, aportan un sistema de acceso y consultas orientado al conjunto.
- El concepto de atomicidad es relevante especialmente en el campo de las bases de datos. Que un elemento sea atómico implica que no puede ser descompuesto en partes más pequeñas.

INTRODUCCIÓN:



- Las dos características más importantes del modelo son:
 - Trabaja con estructuras de datos muy simples: Tablas bidimensionales.
 - Es no navegacional, no hace falta hacer referencia a la forma de acceder a los datos.

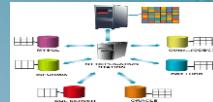
INTRODUCCIÓN:



- En este modelo la base de datos es vista por el usuario como una relación de tablas. Cada fila de la tabla es un registro o tupla y los atributos con columnas o campos.



CONCEPTOS DE BASES DE DATOS RELACIONALES:



- Relación: Película (título, año, duración)

The diagram illustrates the relationship between attributes and tuples in a database table. On the left, the word "Atributos" is connected by a single arrow to the column headers "Titulo", "Año", and "Duración". On the right, the word "Tuplas" is connected by three arrows to the three data rows. Below the table, two arrows point downwards to the labels "Dominio=textos" and "Dominio=enteros", which are positioned under the "Titulo" and "Año" columns respectively.

	Titulo	Año	Duración
	La guerra de las galaxias	1977	123
	El señor de los anillos I	2001	178
	Mar adentro	2004	125

Dominio=textos

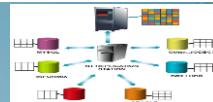
Dominio=enteros

Cardinalidad=3

Grado de la relación=3

SIstema de Bases de Datos II Ing.
Henry Lezcano II Semestre 2020

CONCEPTOS DE BASES DE DATOS RELACIONALES:



- Relación= Conjunto ordenado de n ocurrencias
 - Atributos= Campos de una tabla, propiedades de las entidades
 - Dominio= Conjunto donde los atributos toman valores
 - Tupla= Fila de una tabla
 - Grado de una relación= Numero de atributos o columnas
 - Cardinalidad= Numero de filas o tuplas de una relación

SIstema de Bases de Datos II Ing. Henry Lezcano II Semestre 2020

CONCEPTOS DE BASES DE DATOS RELACIONALES:



- Para dar una definición más adecuada desde el punto de vista de las bases de datos, es preciso distinguir dos conceptos en la definición de la relación:
 - Esquema de relación: es la parte definitoria y estática de la relación (cabecera cuando la relación se percibe como una tabla). Es invariante en el tiempo.
 - Extensión de la relación: conjunto de tuplas que, en un momento determinado, satisface el esquema de la relación y se encuentran almacenadas en la base de datos. Es variante en el tiempo.

CONCEPTOS DE BASES DE DATOS RELACIONALES:

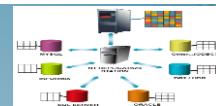


- **Clave primaria**= Es un conjunto de atributos que identifica a cada tupla de una relación y además no hay un subconjunto de ellos que cumplan esa propiedad.
- **Clave foránea**= Es un conjunto de atributos de una tabla que son clave primaria en otra tabla



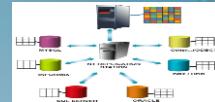
RESTRICCIONES INHERENTES AL MODELO :

- ❖ No puede haber dos tuplas iguales en una misma relación
- ❖ El orden de las tuplas no es significativo
- ❖ El orden de los atributos no es significativo



RESTRICCIONES DE INTEGRIDAD:

- Integridad de la Entidad: Ninguna componente de la clave primaria puede tomar valores nulos o desconocidos, porque entonces no se podrían distinguir dos entidades.
- Integridad Referencial: Cualquier valor que tome un atributo en una relación del que es clave foránea, debe existir en la relación del que es clave primaria.



CONVERSIÓN DEL MODELO CONCEPTUAL AL MODELO RELACIONAL:

- Conversión de Entidades:

Cada entidad de diagrama Entidad/Relación se transforma directamente en una tabla. Los atributos de la entidad pasan a ser automáticamente las columnas de la tabla.

Entidad \longrightarrow Tabla

Atributos \longrightarrow Columnas



CONVERSIÓN DEL MODELO CONCEPTUAL AL MODELO RELACIONAL:

- Conversión de Relaciones:

Cada relación de un diagrama Entidad/Relación se transforma directamente en una tabla. Los campos de esta tabla son las claves primarias de todas las entidades que participen en la relación más todos aquellos atributos que pudiera tener la relación.

Relación \longrightarrow Tabla

MODELO LOGICO RELACIONAL



1

- Toda entidad se convierte en relación o tabla



Entidad en el modelo
Entidad / Relación

C L I E N T E		
PK	CIP	N
	Nombre	N
	Apellido	N
	Email	S
	Celular	S
	Dirección	S

Relación o Tabla en el
Modelo Relación

CONVERSIÓN DEL MODELO CONCEPTUAL AL MODELO RELACIONAL:

- Simplificación del modelo relacional:

Las tablas obtenidas como transformación de relaciones binarias con cardinalidad uno a varios se pueden eliminar.

Los atributos que formaban parte de la tabla pasan a formar parte de la tabla que representa la entidad con cardinalidad “varios”. Asimismo, si la relación tuviera atributos propios, también pasarían a la tabla que representa la entidad con cardinalidad “varios”.

CONVERSIÓN DEL MODELO CONCEPTUAL AL MODELO RELACIONAL:

- Simplificación del modelo relacional:

El número de relaciones que componen la base de datos debe mantenerse en el mínimo posible.

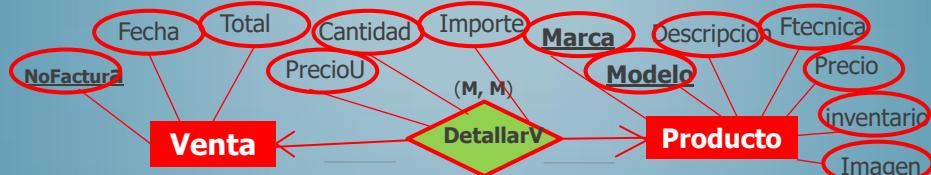
Esto mismo es aplicable a las relaciones binarias con cardinalidad uno a uno, puesto que son un caso particular de las anteriores.

Las tablas con un único atributo se pueden eliminar.

MODELO LOGICO RELACIONAL

1

- En todo relación de muchos a muchos se convierte en una relación o tabla



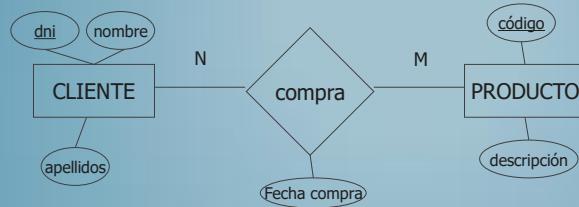
Venta		
PK	NoFactura	N
Fecha	N	
Total	N	

Llaves Compuesta

DetalleV		
PK1	NoFactura	N
PK1	Marca	N
Pk2	Modelo	N
	PrecioU	N
	Cantidad	N
	Importe	N

Producto		
PK1	Marca	N
PK2	Modelo	N
	Descripción	N
	FTécnica	N
	Precio	N
	Inventario	N
	Imagen	N

EJEMPLO 1: RELACIONES N:M



- CLIENTE (**Pk** dni, nombre, apellidos)
- PRODUCTO (**Pk** código, descripción)
- COMPRAS (**Pk** dni_cliente, **Pk** código _ producto, fecha_compra)

MODELO LOGICO RELACIONAL



2

- En todo relación de uno a muchos se debe realizar la propagación de la llave primaria

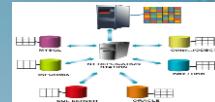


	Sucursal	
PK	Código	N
	Ciudad	N
	Región	N
	Objetivo	S
	Ventas	N

Trabajar

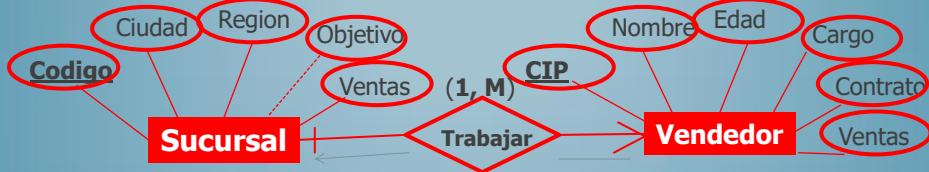
	Vendedor	
PK	CIP	N
	Nombre	N
	Edad	N
	Cargo	N
	Contrato	N
	Ventas	N
FK	Código	S

MODELO LOGICO RELACIONAL



2

- En todo relación de uno a muchos se debe realizar la propagación de la llave primaria



	Sucursal	
PK	Código	N
	Ciudad	N
	Región	N
	Objetivo	S
	Ventas	N

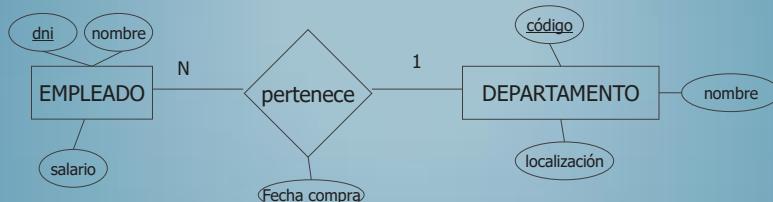
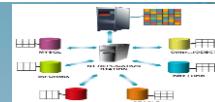
	Vendedor	
PK	CIP	N
	Nombre	N
	Edad	N
	Cargo	N
	Contrato	N
	Ventas	N
FK	Sucursal	S

Se permite el cambio del nombre la FK

Sistema de Bases de Datos II Ing. Henry Lezcano II Semestre 2020

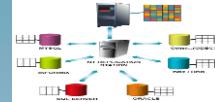
19

EJEMPLO 2: RELACIONES 1:N



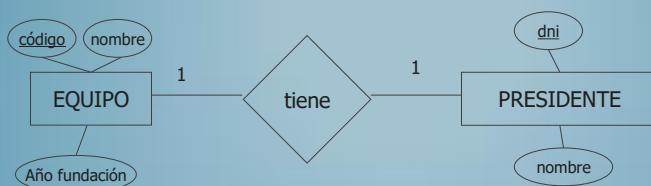
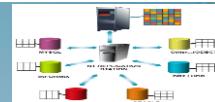
- EMPLEADO (Pk **dni**, nombre, salario, fecha_compra, Fk **código_departamento**)
- DEPARTAMENTO (PK **código**, nombre, localización)

MODELO LOGICO RELACIONAL



- Para una relación de uno a uno 1:1: Con dos tablas es suficiente para representarla y podemos definir la llave foránea en cualquiera de las dos tablas, siguiendo la lógica correspondiente.
 - Si las tablas fueran **PROFESOR** y **GRUPO**, la lógica indica que la llave foránea (FK) debe ser incluida en tabla **GRUPO** siendo lo correcto.

EJEMPLO 3: RELACIONES 1:1



- EQUIPO (**Pk código**, nombre, año_fundación)
- PRESIDENTE (**Pk dni**, nombre, **Fk código_equipo**)
- EQUIPO (**Pk código**, nombre, año_fundación, **Fk dni_presidente**)
- PRESIDENTE (**Pk dni**, nombre)



SISTEMAS DE BASES DE DATOS II

Ing. Henry J. Lezcano

Docente: Departamento de Sistemas de Información y
Control

CAPITULO I NORMALIZACION DE UNA BASE DE DATOS



DEFINICION DE NORMALIZACION

Normalización es el proceso de organizar de manera eficiente los datos dentro de una base de datos. Esto incluye la creación de tablas y el establecimiento de relaciones entre ellas según reglas pre-diseñadas tanto para proteger los datos y la base de datos, como para hacer más flexible al eliminar la redundancia y dependencia incoherente.



OBJETIVOS DE LA NORMALIZACION

- La eliminación de datos redundantes, los cuales ocupan mas espacio en disco y crean problemas de mantenimiento; por ejemplo, cambio de la dirección del cliente es mucho más fácil de implementar si los datos se almacenan sólo en la tabla Clientes y en ninguna otra tabla.
- Evitar problemas de actualización de los datos en las tablas.
- Garantizar que las dependencias que tienen los datos entre ellos, sean lógicas y presenten algún sentido.

Con su aplicación se reducen la cantidad de espacio en la base de datos y aseguran que estos son almacenados de manera lógica (integridad).

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020



DEFINICION DE NORMALIZACION

La normalización también se puede entender como el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020



REGLAS PARA LA NORMALIZACION DE BD

Existen algunas reglas para la normalización de bases de datos. Cada regla se denomina "forma normal".

Si dentro de la base de datos se observa la primera regla se dice que está en "primera forma normal".

Si las tres primeras reglas se observan, la base de datos se considera en "tercera forma normal".

Aunque es posible tener otros niveles de normalización, la tercera forma normal es considerado el más alto nivel necesario para la mayoría de aplicaciones.

Ing. Henry Lezcano Departamento de Sistemas de Información y Control – II Semestre 2020



REGLAS PARA LA NORMALIZACION DE BD

Como ocurre con muchas reglas y especificaciones, en la vida real no siempre es factible el cumplimiento de estas.

En general, la normalización requiere tablas adicionales y para algunos clientes esto no es adecuado.

Si se decide violar una de las tres primeras reglas de normalización, tenga por seguro que su aplicación presentara problemas, como los datos redundantes y dependencias incoherentes.

Ing. Henry Lezcano Departamento de Sistemas de Información y Control – II Semestre 2020

GRADO DE NORMALIZACION: PRIMERA FORMA NORMAL



Los principales objetivos son:

- Eliminar grupos de datos repetidos en tablas individuales.
- Crear una tabla separada para cada conjunto de datos relacionados.
- Identificar cada conjunto de datos relacionados con una clave principal. Ejemplo ID, Primary Key, FK.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

GRADO DE NORMALIZACION: PRIMERA FORMA NORMAL



No utilizar varios campos en una sola tabla para almacenar datos similares.

Por ejemplo, para el seguimiento de un artículo del inventario que proviene de dos fuentes diferentes, el registro puede contener campos para el código de proveedor 1 y un código de proveedor 2.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

GRADO DE NORMALIZACION: PRIMERA FORMA NORMAL



¿Qué sucede cuando se agrega un tercer proveedor? Agregar un campo no es la respuesta, ya que requiere de programación y modificación de tablas y la necesidad de repetirlo cada vez que se agregue a un nuevo proveedor.

En su lugar, se deberá poner toda la información del proveedor en una tabla independiente denominada Proveedores, y vincular el inventario con los proveedores por medio de una clave o de sus claves.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

GRADO DE NORMALIZACION: SEGUNDA FORMA NORMAL



Los principales objetivos son:

- Crear tablas separadas para aquellos conjuntos de valores que se aplican a varios registros. Ejemplo ciudades, profesión.
- Relacionar estas tablas por medio de una clave externa, ejemplo ID, Primary Key, FK.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

GRADO DE NORMALIZACION: SEGUNDA FORMA NORMAL



Los registros no deben depender de nada que no sea la clave primaria de una tabla (una clave compuesta, si es necesario).

Por ejemplo, consideremos la dirección de un cliente en un sistema contable.

- La dirección no solo se necesita en la tabla de clientes, sino también para los pedidos, envío, facturas, cuentas por cobrar, e inclusive en las ordenes.
- En lugar de almacenar la dirección del cliente como una entrada independiente en cada una de estas tablas, guárdela en un lugar, ya sea en la tabla Clientes o en una tabla de direcciones separada.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

GRADO DE NORMALIZACION: TERCERA FORMA NORMAL



Los principales objetivos son:

- Eliminar los campos que no dependan de las claves.

Los valores de un registro que no forman parte de la clave de registro no tienen cabida en la tabla.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

GRADO DE NORMALIZACION: TERCERA FORMA NORMAL



Por ejemplo, en una tabla que contiene los datos de los candidatos a un puesto, el nombre del candidato, nombre de la universidad a la que asistió y la dirección pueden estar incluidos. Pero existen muchas universidades.

Si la información de la universidad se almacena en la tabla de candidatos, no hay manera de listar las universidades que no tengan candidatos.

La mejor opción es crear una tabla separada de Universidades y vincularlo a la tabla Candidatos con una llave de código de la universidad.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



Un dato sin normalizar no cumple con ninguna regla de normalización. Para el caso aplicar cada una de las reglas, consideremos los datos de la siguiente tabla.

ordenes (id_orden, fecha, id_cliente, nom_cliente, estado, num_art, nom_art, cant, precio)

Id_orden	Fecha	Id_cliente	Nom_cliente	Provincia	Num_art	nom_art	cant	Precio
2301	23/02/2011	101	Martin	Chiriquí	3786	Red	3	35,00
2301	23/02/2011	101	Martin	Chiriquí	4011	Raqueta	6	65,00
2301	23/02/2011	101	Martin	Chiriquí	9132	Paq-3	8	4,75
2302	25/02/2011	107	Herman	Colon	5794	Paq-6	4	5,00
2303	27/02/2011	110	Pedro	Herrera	4011	Raqueta	2	65,00
2303	27/02/2011	110	Pedro	Herrera	3141	Funda	2	10,00

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



PRIMERA FORMAL NORMAL (1FN)

Al examinar estos registros, podemos darnos cuenta que contienen un grupo repetido para NUM_ART, NOM_ART, CANT y PRECIO. La 1FN prohíbe los grupos repetidos, por lo tanto tenemos que convertir a la primera forma normal.

Los pasos a seguir son:

- Tenemos que eliminar los grupos repetidos.
- Tenemos que crear una nueva tabla con la PK de la tabla base y el grupo repetido.

Los registros quedan ahora conformados en dos tablas que llamaremos ORDENES y ARTICULOS_ORDENES

ordenes (id_orden, fecha, id_cliente, nom_cliente, estado)

Articulos_ordenes (id_orden, num_art, nom_art, cant, precio)

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



PRIMERA FORMAL NORMAL (1FN)

ordenes (id_orden, fecha, id_cliente, nom_cliente, estado)

Articulos_ordenes (id_orden, num_art, nom_art, cant, precio)

Ordenes				
Id_orden	Fecha	Id_cliente	Nom_cliente	Estado
2301	23/02/2011	101	Martin	Chiriqui
2302	25/02/2011	107	Herman	Colon
2303	27/02/2011	110	Pedro	Herrera

Articulos_ordenes				
Id_orden	Num_art	nom_art	cant	Precio
2301	3786	Red	3	35,00
2301	4011	Raqueta	6	65,00
2301	9132	Paq-3	8	4,75
2302	5794	Paq-6	4	5,00
2303	4011	Raqueta	2	65,00
2303	3141	Funda	2	10,00

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACIÓN DE BD



SEGUNDA FORMAL NORMAL (2FN)

Ahora procederemos a aplicar la segunda formal normal, es decir, tenemos que eliminar cualquier columna no llave que no dependa de la llave primaria de la tabla.

Los pasos a seguir son:

- Determinar cuáles columnas que no son llave no dependen de la llave primaria de la tabla.
- Eliminar esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y la(s) columna(s) de la PK de la cual dependen.

La tabla ORDENES está en 2FN. Cualquier valor único de ID_ORDEN determina un sólo valor para cada columna. Por lo tanto, todas las columnas son dependientes de la llave primaria ID_ORDEN.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACIÓN DE BD



SEGUNDA FORMAL NORMAL (2FN)

Por su parte, la tabla ARTICULOS_ORDENES no se encuentra en 2FN ya que las columnas PRECIO y NOM_ART son dependientes de NUM_ART, pero no son dependientes de ID_ORDEN.

Lo que haremos a continuación es eliminar estas columnas de la tabla ARTICULOS_ORDENES y crear una tabla ARTICULOS con dichas columnas y la llave primaria de la que dependen.

Las tablas quedan ahora de la siguiente manera.

Articulos_ordenes (id_orden, num_art, cant)

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



SEGUNDA FORMAL NORMAL (2FN)

Articulos_ordenes (id_orden, num_art, cant)

Articulos_ordenes		
Id_orden	Num_art	cant
2301	3786	3
2301	4011	6
2301	9132	8
2302	5794	4
2303	4011	2
2303	3141	2

Articulos (num_art, nom_art, precio)

Articulos		
Num_art	nom_art	Precio
3786	Red	35,00
4011	Raqueta	65,00
9132	Paq-3	4,75
5794	Paq-6	5,00
3141	Funda	10,00

Ing. Henry Lezcano Departamento de Sistemas de Información y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



TERCERA FORMAL NORMAL (3FN)

La tercera forma normal nos dice que tenemos que eliminar cualquier columna no llave que sea dependiente de otra columna no llave.

Los pasos a seguir son:

- Determinar las columnas que son dependientes de otra columna no llave.
- Eliminar esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y con la columna no llave de la cual son dependientes.

Ing. Henry Lezcano Departamento de Sistemas de Información y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



TERCERA FORMAL NORMAL (3FN)

Al observar las tablas que hemos creado, nos damos cuenta que tanto la tabla ARTICULOS, como la tabla ARTICULOS_ORDENES se encuentran en 3FN. Sin embargo la tabla ORDENES no lo está, ya que NOM_CLIENTE y ESTADO son dependientes de ID_CLIENTE, y esta columna no es la llave primaria.

Para normalizar esta tabla, moveremos las columnas no llave y la columna llave de la cual dependen dentro de una nueva tabla CLIENTES. Las nuevas tablas CLIENTES y ORDENES se muestran a continuación.

Ing. Henry Lezcano Departamento de Sistemas de Información
y Control – II Semestre 2020

APLICACIÓN DE LAS REGLAS DE NORMALIZACION DE BD



TERCERA FORMAL NORMAL (3FN)

ordenes (id_orden, fecha, id_cliente) *Clientes* (id_cliente, nom_cliente, estado)

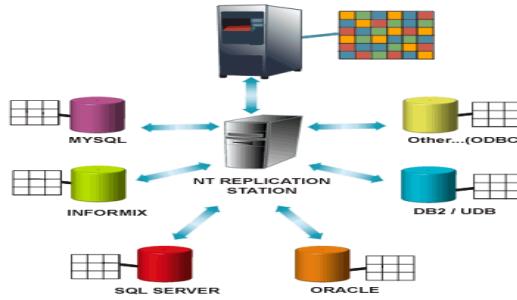
Ordenes		
Id_orden	Fecha	Id_cliente
2301	23/02/2011	101
2302	25/02/2011	107
2303	27/02/2011	110

Cliente		
Id_cliente	Nom_cliente	Estado
101	Martin	Caracas
107	Herman	Coro
110	Pedro	Maracay

Ing. Henry Lezcano Departamento de Sistemas de Información y Control – II Semestre 2020

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.
SISTEMAS BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

Implementacion de un Modelo Base de Datos Relacional
Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



Sistemas de Base de Datos II
 Por. Ing. Henry Lezcano
 II
 Semestre 2020

[1]

OBJETIVOS GENERALES

- *Aplicara el lenguaje de consulta SQL (según el gestor a utilizar) para la definición y manipulación de una base de datos con el objetivo de implementarla con todos los objetos clásicos: tablas, índices, disparadores, vistas, procedimientos almacenados.*
- *Describir el concepto y proceso de transacciones en el entorno de una base de datos.*
- *Reconocer, comprender y utilizar los principales constructores del lenguaje de 4gl a utilizar para desarrollar la programación de los objetos en la base de datos a implementar.*

Sistemas de Base de Datos II
 Por. Ing. Henry Lezcano
 II
 Semestre 2020

[2]

CONTENIDO

Capítulo I Desarrollo del Modelo de Base de Datos (Modelo Conceptual, Logico y Fisico) Implementación de un Modelo de Base de Datos Relacional transaccional 'Comandos de Definición y Manipulación del Lenguaje X-SQL (según el gestor de Base de Datos a utilizar).'



Sistemas de Base de Datos I
Por. Ing. Henry Lezcano
I
Semestre 2020

3

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



INTRODUCCION

PL/SQL significa Procedural Language/SQL 'Lenguaje Procedimental / SQL'

Como su propio nombre lo indica, PL/SQL amplia la funcionalidad de SQL añadiendo estructuras de las que pueden encontrarse en otros lenguajes procedimentales, como:

- **Variables y Tipos (tantos predefinidos como definidos por el usuario)**
- **Estructuras de control, como bucles y órdenes IF-THEN-ELSE**
- **Procedimientos y Funciones**
- **Tipos de Objetos y Métodos.**

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano
II
Semestre 2020

4

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



INTRODUCCION

La construcciones procedimentales están perfectamente integradas con Oracle SQL, lo que da como resultado un lenguaje potente y estructurado.

Por ejemplo supongamos que queremos cambiar la especialidad de un determinado estudiante. Si el estudiante no existe, entonces queremos que se añada el nuevo registro. Esto es una tarea sencilla para PL/SQL:

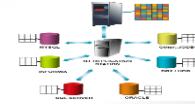
```

DECLARE
    /* Declaración de las variables usadas en la sentencia SQL */
    v_newmajor VARCHAR2(10) := 'History';
    v_firstname VARCHAR2(10) := 'Scott';
    v_lastname VARCHAR2(10) := 'Urman';
BEGIN
    /* Actualización de la tabla estudiante */
    UPDATE estudiante
    SET major = v_newmajor
    WHERE first_name = v_firstname
    And last_name = v_lastname;
    /* Comprobación para encontrarlo, de no ser
    así debe crearlo en la tabla estudiante*/
    IF SQL%NOTFOUND THEN
        INSERT INTO estudiante ( ID, first_name,
                                last_name, major)
        VALUES (student_sequence.NEXTVAL,
                v_firstname, v_lastname,
                v_newmajor);
    END IF;
END;
  
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[5]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



INTRODUCCION

El ejemplo contiene dos ordenes SQL distintas (UPDATE, INSERT), así como diversas declaraciones de variables y la orden IF condicional.

- ❑ Importante destacar que para poder ejecutar el procedimiento primero se deben crear los objetos de base de datos a los que se hacen referencias (Tabla estudiante, secuencia students_sequence)
- ❑ PL/ SQL es único, en el sentido de que combina la flexibilidad de SQL con la potencia y configurabilidad de un 3GL.
- ❑ El lenguaje integra las estructuras procedimentales como el acceso a la base de datos. El resultado es un lenguaje robusto y potente, bien adaptado al diseño de aplicaciones complejas.

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[6]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



CARACTERISTICAS DE PL/SQL

ESTRUCTURA DE BLOQUES

- ❖ La unidad básica de PL/SQL es el bloque. Todos programas PL/SQL están compuesto por bloques, que pueden estar anidados.
- ❖ Por lo general cada bloque realiza una unidad lógica de trabajo en el programa, separando así unas tareas de otras.
- ❖ Un bloque tiene la siguiente estructura:

```

DECLARE
    /* Sección declarativa – Aquí se incluyen las variables PL/SQL, tipos, cursos,
    y subprogramas locales */
BEGIN
    /* Sección ejecutable – Aquí se incluyen las órdenes SQL y procedimentales.
    Esta es la sección principal del bloque y la única que es obligatoria*/
EXCEPTION
    /* Sección de manejo de excepciones- Aquí se incluye las órdenes para el manejo de las
    excepciones.*/
END;
  
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[7]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



CARACTERISTICAS DE PL/SQL

MANEJO DE ERRORES

- ❖ La sección de manejo de errores de un bloque se usa para responder a los errores de ejecución con los que se encuentra el programa.
- ❖ Al separar el código de gestión de errores del cuerpo principal del programa, se consigue que la estructura de esta sea clara.
- ❖ Un bloque con manejo de excepciones :

```

DECLARE
    v_Errorcode  NUMBER; -- Código del error
    v_Errormsg  VARCHAR2(200); -- Texto del mensaje de error
    v_Currentuser VARCHAR2(8); -- Usuario actual de la Base de Datos
    v_Information VARCHAR2(100); -- Información sobre el error
BEGIN
    /* Código que procesa algunos datos*/
    NULL;
EXCEPTION
    WHEN OTHER THEN
        -- Asignación de Valores a las Variables de Registro,
        -- utilizando funciones predefinidas.
        v_Errorcode := SQLCODE;
        v_Errormsg := SQLERRM;
        v_Currentuser := USER;
        v_Information := 'Error encountered on ' || TO_CHAR(SYSDATE) || ' by database user ' || v_Currentuser;
        INSERT INTO log_table (code, message, info)
        VALUES (v_Errorcode, v_Errormsg, v_Information);
END;
  
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[8]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



CARACTERISTICAS DE PL/SQL

VARIABLES Y TIPOS

- ❖ La información se transmite entre PL/SQL y la base de datos mediante **VARIABLES**. Una variables es una zona de almacenamiento que puede ser leída o escrita por programa.
- ❖ Las variables se declaran en la sección declarativa del bloque. Cada variable tiene un **tipo** específico asociado. El tipo define la clase de información que se puede almacenar en la variable.
- ❖ Las variables PL/SQL del mismo tipo de las columnas de una tabla en la base de datos:

```
DECLARE
    v_studentname      VARCHAR2 (20);
    v_currentdate      DATE;
    v_numbercredits    NUMBER (3);
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[9]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



CARACTERISTICAS DE PL/SQL

VARIABLES Y TIPOS

- ❖ O pueden ser de tipos adicionales:

```
DECLARE
    v_loopercounter  BINARY_INTEGER;
    v_currentlyregistered BOOLEAN;
```

- ❖ PL/SQL también admite tipos definidos por el usuario; tablas y registros. Los tipos definidos por el usuario permiten personalizar la estructura de los datos manipulados por un programa:

```
DECLARE
    TYPE t_studentrecord IS RECORD (
        Firstname VARCHAR2(20),
        Lastname VARCHAR2 (20),
        Currentcredits NUMBER (3 ) );
    V_student t_studentrecord;
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[10]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



CARACTERISTICAS DE PL/SQL

ESTRUCTURA DE BUCLE

- ❖ PL/SQL admite diferentes tipos de bucles. Un bucle permite ejecutar repetidamente una cierta secuencia de ordenes.
- ❖ Ejemplo

```
DECLARE
    v_loopcounter  BINARY_INTEGER := 1;
BEGIN
    LOOP
        INSERT INTO temp_table (num_col) VALUES (v_loopcounter);
        v_loopcounter := v_loopcounter + 1;
        EXIT WHEN v_loopcounter > 50;
    END LOOP;
END;
```

- ❖ También se podría usar otro tipo de bucle, e bucle numero FOR, que tiene una sintaxis mas simple. Y quedaría definido como:

```
BEGIN
    FOR v_loopcounter IN 1..50 LOOP
        INSERT INTO temp_table (num_col) VALUES (v_loopcounter);
    END LOOP;
END;
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[11]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



CARACTERISTICAS DE PL/SQL

CURSORES

- ❖ Un CURSOR se emplea para procesar múltiples filas extraídas de la base de datos (con una orden SELECT). Utilizando el cursor, programa puede recorrer, una por una, las filas devueltas y procesarlas.
- ❖ Ejemplo

```
DECLARE
    firstname VARCHAR2(20);
    lastname VARCHAR2(20);
    -- Declaración del cursor. Esto define la orden SQL para devolver las filas
    CURSOR c_students IS
        SELECT first_name, last_name
        FROM estudiante;

    BEGIN
        -- Inicia del procedimiento del cursor
        OPEN c_students;
        LOOP
            --Recupera una Fila
            FETCH c_students INTO v_firstname, v_lastname;
            -- Salida del bucle después de haber recuperado todas las filas
            EXIT WHEN c_students%NOTFOUND;
            /* Aquí se procesarían los datos */
        END LOOP;
        -- Fin del Procesamiento
        CLOSE c_students;
    END;
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

[12]

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

TIPOS DE COMANDOS PL/SQL



- Los comandos SQL son instrucciones que se utilizan para comunicarse con la base de datos para realizar tareas específicas que funcionan con datos.
- Los comandos SQL se puede utilizar no sólo para buscar en la base de datos, sino también para realizar otras funciones como, por ejemplo, puede crear tablas, agregar datos a las tablas, o modificar los datos, eliminar la tabla, establecer permisos para los usuarios.
- Los comandos SQL se agrupan en cuatro grandes categorías según su funcionalidad:
 - **Data Definition Language (DDL)** - Estos comandos SQL se utilizan para crear, modificar y quitar la estructura de los objetos de base de datos. Los comandos son CREATE, ALTER, DROP, RENAME, y TRUNCATE.

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

TIPOS DE COMANDOS PL/SQL



Continuación.....

- Los comandos SQL se agrupan en cuatro grandes categorías según su funcionalidad:
 - **Lenguaje de manipulación de datos (DML)** - Estos comandos SQL se utilizan para almacenar, recuperar, modificar y eliminar datos. Estos comandos son SELECT, INSERT, UPDATE y DELETE.
 - **Transaction Control Language (TCL)** - Estos comandos SQL se utilizan para gestionar los cambios que afectan a los datos. Estos comandos son ROLLBACK, COMMIT, y SAVEPOINT.
 - **Data Control Language (DCL)** - Estos comandos SQL se utilizan para proporcionar seguridad a los objetos de base de datos. Estos comandos se GRANT y REVOKE.

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

- ❑ **CREATE TABLE:** La sintaxis para esta sentencia de creación de tablas se define:

```
CREATE TABLE nombre_tabla (columna1 tipodato, columna2 tipodato);
```

```
CREATE [GLOBAL TEMPORARY] TABLE [esquema.]tabla
    columna datatype [DEFAULT expr] [column_constraint(s)]
    [,columna datatype [...] ]
    table_constraint
    table_ref_constraint
    [ON COMMIT {DELETE|PRESERVE} ROWS]
    storage_options [COMPRESS int|NOCOMPRESS]
    [LOB_storage_clause][varray_clause][nested_storage_clause] [XML_type_clause]
    Partitioning_clause
    [[NO]CACHE] [[NO]ROWDEPENDENCIES] [[NO]MONITORING] [PARALLEL parallel_clause]
    [ENABLE enable_clause | DISABLE disable_clause]
    {ENABLE|DISABLE} ROW MOVEMENT
    [AS subquery]
```

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

CREATE TABLE:

- ✓ **Ejemplo de la creación de una tabla**

```
CREATE TABLE PRODUCTOS (
    numeroproducto number,
    descriproducto varchar2(10)
);
```

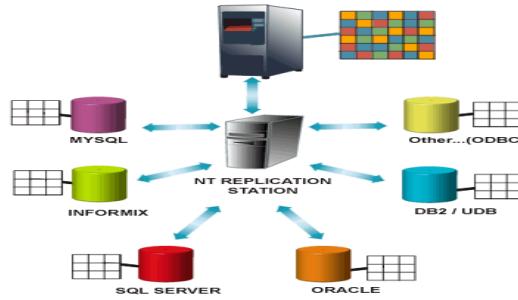
```
CREATE TABLE CLIENTES (
    id_cliente number PRIMARY KEY,
    nombre varchar2(20),
    apellido varchar2(20)
);
```

- ✓ **Ejemplo de la creación de una tabla con PRIMARY KEY incluidas y restricciones.**

```
CREATE TABLE PEDIDOS (
    numeropedido number PRIMARY KEY,
    fechapedido date,
    id_cliente number,
    CONSTRAINT no_pedido FOREIGN KEY (id_cliente) REFERENCES CLIENTES (id_cliente));
```

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.
SISTEMAS BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

Implementacion de un Modelo Base de Datos Relacional
Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



Sistemas de Base de Datos II
 Por. Ing. Henry Lezcano
 II
 Semestre 2020

[1]

OBJETIVOS GENERALES

- *Aplicara el lenguaje de consulta SQL (según el gestor a utilizar) para la definición y manipulación de una base de datos con el objetivo de implementarla con todos los objetos clásicos: tablas, índices, disparadores, vistas, procedimientos almacenados.*
- *Describir el concepto y proceso de transacciones en el entorno de una base de datos.*
- *Reconocer, comprender y utilizar los principales constructores del lenguaje de 4gl a utilizar para desarrollar la programación de los objetos en la base de datos a implementar.*

Sistemas de Base de Datos II
 Por. Ing. Henry Lezcano
 II
 Semestre 2020

[2]

CONTENIDO

Capítulo I Desarrollo del Modelo de Base de Datos (Modelo Conceptual, Logico y Fisico) Implementación de un Modelo de Base de Datos Relacional transaccional 'Comandos de Definición y Manipulación del Lenguaje X-SQL (según el gestor de Base de Datos a utilizar).'



Sistemas de Base de Datos II
Por. Ing. Henry Lezcano
II
Semestre 2020

3

1.4 Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

CREATE TABLE:

✓ Ejemplo de la creación de una tabla

```
CREATE TABLE PRODUCTOS (
    numeroproducto number,
    descriproducto varchar2(10)
);
```

```
CREATE TABLE CLIENTES (
    id_cliente number PRIMARY KEY,
    nombre varchar2(20),
    apellido varchar2(20)
);
```

✓ Ejemplo de la creación de una tabla con PRIMARY KEY incluidas y restricciones.

```
CREATE TABLE PEDIDOS (
    numeropedido number PRIMARY KEY,
    fechapedido date,
    id_cliente number,
    CONSTRAINT no_pedido FOREIGN KEY (id_cliente) REFERENCES CLIENTES (id_cliente));
```

Sistemas de Base de Datos II
Por. Ing. Henry Lezcano
II
Semestre 2020

4



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

CREATE TABLE:

- ✓ Ejemplo de la creación de una tabla con PRIMARY KEY donde es necesario incluir un índice unico (unique index) y es posible identificar el tablespace donde queremos crear el indice.

```
CREATE TABLE PEDIDOS (
    numeropedido number PRIMARY KEY,
    fechapedido date,
    id_cliente number
    CONSTRAINT fk_cliente FOREIGN KEY (id_cliente) REFERENCES CLIENTES (id_cliente)
    CONSTRAINT pk_pedido (numeropedido) USING INDEX tablespace ts_index
);
```



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

ALTER TABLE:

Con esta instrucción podemos cambiar columnas y restricciones definidas sobre las tablas.

- ✓ La sintaxis para esta sentencia es la siguiente

```
ALTER TABLE [esquema.]tabla {ADD |MODIFY| DROP}...
```

- ✓ Si queremos añadir una columna a la tabla la sentencia seria:

```
ALTER TABLE PEDIDOS ADD TEXTOPEDIDO VARCHAR2(35);
```

- ✓ Si queremos cambiar el tamaño de columna a la tabla la sentencia seria:

```
ALTER TABLE PEDIDOS MODIFY TEXTOPEDIDO VARCHAR2(135);
```



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

ALTER TABLE:

Con esta instrucción podemos cambiar columnas y restricciones definidas sobre las tablas.

- ✓ Si queremos asignar el valor NOT NULL una columna de la tabla la sentencia seria:

ALTER TABLE PEDIDOS MODIFY (TEXTOPEDIDO NOT NULL);

- ✓ Si queremos eliminar una columna de la tabla la sentencia seria:

ALTER TABLE PEDIDOS DROP COLUMN TEXTOPEDIDO;

- ✓ Si queremos asignar un valor por defecto una columna de la tabla la sentencia seria:

ALTER TABLE PEDIDOS MODIFY TEXTOPEDIDO VARCHAR2(135) DEFAULT 'NUECES';



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

ALTER TABLE:

Con esta instrucción podemos cambiar columnas y restricciones definidas sobre las tablas.

- ✓ Si queremos añadir dos columnas a la tabla la sentencia seria:

ALTER TABLE PEDIDOS ADD (PEDIDO_ID INT, TEXTOPEDIDO VARCHAR2(35));

- ✓ **LA SINTAXIS ALTER TABLE PARA LAS RESTRICCIONES:**

```
ALTER TABLE [esquema..] tabla
  Constraint_clause, ...
  [ENABLE enable_clause | DISABLE disable_clause
  [ {ENABLE | DISABLE} TABLE LOCK]
  [ {ENABLE | DISABLE} ALL TRIGGERS]
```



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

ALTER TABLE:

- ✓ LA SINTAXIS **ALTER TABLE** PARA LAS RESTRICCIONES:

- ✓ DONDE EL CONSTRAINT (**constraint_clause**) PUEDEN SER ALGUNA DE LAS ENTRADAS:

- ❖ ADD **out_of_line_constraint(s)**
- ❖ ADD **out_of_line_referential_constraint**
- ❖ **DROP PRIMARY KEY [CASCADE] [{KEEP|DROP} INDEX]**
- ❖ **DROP UNIQUE (column, ...)[{KEEP|DROP} INDEX]**
- ❖ **DROP CONSTRAINT constraint [CASCADE]**
- ❖ **MODIFY CONSTRAINT constraint constraint_state**
- ❖ **MODIFY PRIMARY KEY constraint constraint_state**
- ❖ **MODIFY UNIQUE (column, ...) constraint constraint_state**



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

ALTER TABLE:

- ✓ LA SINTAXIS **ALTER TABLE** PARA LAS RESTRICCIONES:

- ✓ DONDE A SU VEZ **constraint_state** PUEDE SER:

```
[[NOT] DEFERRABLE] [INITIALLY {IMMEDIATE | DEFERRED}]
[RELY | NORELY] [USING INDEX using_index_clause]
[ENABLE | DESABLE] [VALIDATE | NOVALIDATE]
[EXCEPTIONS INTO [schema.] table]
```



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

PARA CAMBIAR LAS RESTRICCIONES Y LA LLAVE PRIMARIA EN LAS TABLAS DEBEMOS USAR
ALTER TABLE:

- ✓ Si queremos crear llaves primarias a la tabla la sentencia seria:
`ALTER TABLE PEDIDOS ADD CONSTRAINT pk_pedido PRIMARY KEY (numeropedido, lineapedido);`
- ✓ Si queremos crear llaves foránea a la tabla para la integridad referencial la sentencia seria:
`ALTER TABLE PEDIDOS ADD CONSTRAINT FK_PEDIDOS_CLIENTES FOREIGN KEY (id_cliente)
REFERENCES CLIENTES (id_cliente);`
- ✓ Si queremos establecer un control de valores en tabla la sentencia seria:
`ALTER TABLE PEDIDOS ADD CONSTRAINT CK_ESTADO CHECK (estado IN (1,2,3));`



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

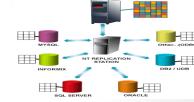
COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

PARA CAMBIAR LAS RESTRICCIONES Y LA LLAVE PRIMARIA EN LAS TABLAS DEBEMOS USAR
ALTER TABLE:

- ✓ Si queremos crear un restricción UNIQUE en la tabla la sentencia seria:
`ALTER TABLE PEDIDOS ADD CONSTRAINT uk_estado UNIQUE (id_correo);`
- ✓ Si queremos borrar un restricción en la tabla la sentencia seria:
`ALTER TABLE PEDIDOS DROP CONSTRAINT con_pedidos_clientes;`
- ✓ Si queremos deshabilitar un restricción en la tabla la sentencia seria:
`ALTER TABLE PEDIDOS DISABLE CONSTRAINT con_pedidos_clientes;`



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

CREACION DE UN BASE DE DATOS

DEFINICION DE TABLAS

PARA CAMBIAR LAS RESTRICCIONES Y LA LLAVE PRIMARIA EN LAS TABLAS DEBEMOS USAR
ALTER TABLE:

- ✓ Si queremos habilitar un restricción en la tabla la sentencia seria:

ALTER TABLE PEDIDOS ENABLE CONSTRAINT con_pedidos_clientes;



Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

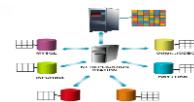
[13]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

MANIPULACION DE OBJETOS DE TABLAS

En la manipulación de objetos de tablas en la Base de Datos es importante que para que los cambios sobre estos, se hagan efectivos debemos ejecutar la sentencia COMMIT y para cancelar la operación ejecutada, se ejecuta la sentencia ROLLBACK



Sistemas de Base de Datos II
Por. Ing. Henry Lezcano II
Semestre 2020

- ✓ SENTENCIA INSERCIÓN: SINTAXIS

INSERT INTO nombre-tabla
VALUES (serie de valores);

- ✓ La forma en que se asignan los valores en la cláusula VALUES tiene que coincidir con el orden en que se definieron las columnas en la creación de la tablas, dado que los valores se asignan por posicionamiento relativo.

INSERT INTO PEDIDOS
VALUES (125, 3, 'PEDRO');

[14]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

MANIPULACION DE OBJETOS DE TABLAS

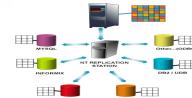
En la manipulación de objetos de tablas en la Base de Datos es importante que para que los cambios sobre estos, se hagan efectivos debemos ejecutar la sentencia COMMIT y para cancelar la operación ejecutada, se ejecuta la sentencia ROLLBACK

✓ **SENTENCIA INSERCIÓN**: SINTAXIS DE OTRA FORMA

```
INSERT INTO nombre-tabla (columna1, columna2,...)
VALUES (valor1, valor3, .....);
```

- ✓ En este caso los valores se asignarán a cada una de las columnas mencionadas por posicionamiento relativo. Es necesario que por lo menos se asigne valores a todos aquellas columnas que no admiten valores nulos en las tablas(NOT NULL).

```
INSERT INTO PEDIDOS (COD_PEDIDO, ESTADO)
VALUES (125, 3);
```



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

MANIPULACION DE OBJETOS DE TABLAS

En la manipulación de objetos de tablas en la Base de Datos es importante que para que los cambios sobre estos, se hagan efectivos debemos ejecutar la sentencia COMMIT y para cancelar la operación ejecutada, se ejecuta la sentencia ROLLBACK

✓ **SENTENCIA UPDATE**: SINTAXIS

```
UPDATE nombre-tabla
SET columna1 = valor 1 [ columna2 = valor2,...]
[WHERE condición]
```

- ✓ Se actualizaran los campos correspondientes con los valores que se le asignen, en el subconjunto de filas que cumplan con la condición.
- ✓ Si no se pone condición de selección la actualización se dara en todos las filas de las tablas.
- ✓ Si se desea actualizar a campos en nulos, se asignara el valor ce NULL.



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

MANIPULACION DE OBJETOS DE TABLAS

En la manipulación de objetos de tablas en la Base de Datos es importante que para que los cambios sobre estos, se hagan efectivos debemos ejecutar la sentencia COMMIT y para cancelar la operación ejecutada, se ejecuta la sentencia ROLLBACK

✓ **SENTENCIA UPDATE :**

✓ Se modifica el nombre y estado de un pedido:

```
UPDATE PEDIDOS
SET NOMBRE = 'JUAN', ESTADO = 1
WHERE COD_PEDIDO = 125;
```

✓ Se modifica el estado de todos los pedidos:

```
UPDATE PEDIDOS
SET ESTADO = 1;
```

✓ Se modifica el nombre de un pedido a nulo:

```
UPDATE PEDIDOS
SET NOMBRE = NULL
WHERE COD_PEDIDO = 125;
```



II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

MANIPULACION DE OBJETOS DE TABLAS

En la manipulación de objetos de tablas en la Base de Datos es importante que para que los cambios sobre estos, se hagan efectivos debemos ejecutar la sentencia COMMIT y para cancelar la operación ejecutada, se ejecuta la sentencia ROLLBACK

✓ **SENTENCIA DELETE :** SINTAXIS

```
DELETE nombre-tabla
[WHERE condición];
```

✓ Se borra la tabla de pedido:

```
DELETE FROM PEDIDOS ;
```

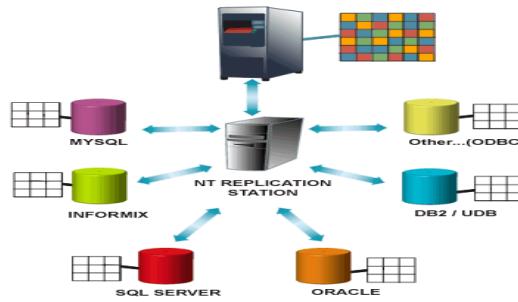
✓ Se borra un registro de la tabla:

```
DELETE FROM PEDIDOS
WHERE COD_PEDIDO = 15;
```



UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.
SISTEMAS BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

Implementacion de un Modelo Base de Datos Relacional
Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE



Sistemas de Base de Datos II
 Por. Ing. Henry Lezcano
 II
 Semestre 2020

[1]

OBJETIVOS GENERALES

- *Aplicara el lenguaje de consulta SQL (según el gestor a utilizar) para la definición y manipulación de una base de datos con el objetivo de implementarla con todos los objetos clásicos: tablas, índices, disparadores, vistas, procedimientos almacenados.*
- *Describir el concepto y proceso de transacciones en el entorno de una base de datos.*
- *Reconocer, comprender y utilizar los principales constructores del lenguaje de 4gl a utilizar para desarrollar la programación de los objetos en la base de datos a implementar.*

Sistemas de Base de Datos II
 Por. Ing. Henry Lezcano
 II
 Semestre 2020

[2]

CONTENIDO

Capítulo.I Desarrollo del Modelo de Base de Datos (Modelo Conceptual, Logico y Fisico) Implementación de un Modelo de Base de Datos Relacional transaccional 'Comandos de Definición y Manipulación del Lenguaje X-SQL (según el gestor de Base de Datos a utilizar).'



Sistemas de Base de Datos I
Por. Ing. Henry Lezcano
I Semestre 2020

[3]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

IMPLEMENTACION DE INDICE Y VISTAS

DEFINICION DE INDICES

El índice de una base de datos es una estructura de datos que mejora la velocidad de las operaciones, permitiendo un rápido acceso a los registros de una tabla. Al aumentar drásticamente la velocidad de acceso, se suelen usar sobre aquellos campos sobre los cuales se vayan a realizar búsquedas frecuentes.

El índice tiene un funcionamiento similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos.

Para buscar un elemento que esté indexado, sólo hay que buscar en el índice de dicho elemento para, una vez encontrado, devolver el registro que se encuentre en la posición marcada por el índice.



Implementación de Base de Datos
Por. Ing. Henry Lezcano
I Semestre 2020

[4]

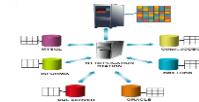
II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS IMPLEMENTACION DE INDICE Y VISTAS

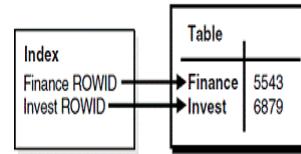
DEFINICION DE INDICES

Los índices pueden ser creados usando una o más columnas, preparando la base de datos tanto para búsquedas rápidas al azar como para ordenaciones eficientes de los registros.

Los índices son construidos sobre árboles B, B+, B* o sobre una mezcla de ellos, funciones de cálculo u otros métodos.



Regular Table and Index



Implementación de Base de Datos
Por. Ing. Henry Lezcano
II Semestre 2020

[5]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS IMPLEMENTACION DE INDICE Y VISTAS

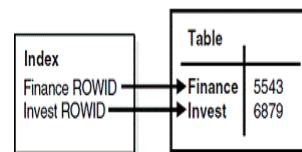
DEFINICION DE INDICES

El espacio en disco requerido para almacenar el índice es típicamente menor que el espacio de almacenamiento de la tabla (puesto que los índices generalmente contienen solamente los campos clave de acuerdo con los que la tabla será ordenada, y excluyen el resto de los detalles de la tabla), lo que da la posibilidad de almacenar en memoria los índices de tablas que no cabrían en ella.

En una base de datos relacional un índice es una copia de parte de una tabla.



Regular Table and Index



Implementación de Base de Datos
Por. Ing. Henry Lezcano
II Semestre 2020

[6]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL –ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS
IMPLEMENTACION DE INDICE Y VISTAS



Cómo crear índices en Oracle

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[7]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL –ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS
IMPLEMENTACION DE INDICE Y VISTAS



Cómo crear índices en Oracle

Creación de un índice al crear una tabla de Oracle

Crearemos una tabla para el ejemplo de creación de índices ejecutando los siguiente comandos del DDL.

```
CREATE TABLE facturacion (
  codigo number(10) not null,
  fecha date default sysdate,
  codigocliente number(10),
  nombrecliente varchar(100),
  observacion varchar(2000),
  constraint pk_facturacion_codigo
  primary key (codigo)
);
```

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[8]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL –ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

Creación de un índice al crear una tabla de Oracle

```
CREATE TABLE facturacion (
  codigo number(10) not null,
  fecha date default sysdate,
  codigocliente number(10),
  nombrecliente varchar(100),
  observacion varchar(2000),
  constraint pk_facturacion_codigo
  primary key (codigo)
)
```

Como se puede observar en la secuencia de comandos del DDL la línea ...

```
constraint      pk_facturacion_codigo
primary key (codigo)
```

Estamos indicando a Oracle que cree la tabla "facturacion", con el campo "codigo" y que éste sea clave primaria, por lo que creará un índice automáticamente para este campo. Esta es una forma de crear índices, en la creación de la tabla

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[9]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL –ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

Creación de un índice al en una tabla existente de Oracle

La creación de un índice en Oracle se realiza mediante el comando ***create index***.

Cuando se define una clave primaria o una columna única (UNIQUE) durante la creación de una tabla o su mantenimiento, Oracle creará automáticamente un índice de tipo UNIQUE que gestione dicha restricción, como hemos indicado anteriormente

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[10]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL

ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

Creación de un índice al en una tabla existente de Oracle

Siguiendo con el ejemplo, añadiremos un índice normal para la columna "nombreciente" de la tabla "facturacion". Para ello ejecutaremos la siguiente secuencia de comandos del DDL:

```
create index IN_FACTURACION_NOMBRECLIENTE
on FACTURACION (NOMBRECLIENTE);
```

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[11]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL

ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

Creación de un índice al en una tabla existente de Oracle

Se puede agregar un índice de tipo UNIQUE, obligando a que los valores del campo indexado sean únicos, que no se puedan repetir en el campo de la tabla, ejecutaremos secuencia de comandos ...

```
create unique index
IN_FACTURACION_COD_CODCLI_FE
on FACTURACION (CODIGOCLIENTE,
FECHA)
```

De esta forma Oracle **no** permitirá que haya dos registros en la tabla "facturacion" con el mismo valor en los campos "codigocliente" y "fecha", es decir, sólo podrá añadirse una factura por cliente y por día, un cliente no podrá tener dos facturas en un mismo día.

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[12]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL –ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS
IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

Creación de un índice al en una tabla existente de Oracle

Secuencia de comandos del DML:

```
insert into facturacion
(codigo, codigocliente, fecha)
values (6900, 500,
to_date('31/12/2009', 'DD-MM-YYYY'))
```

Secuencia de comandos del DML:

```
insert into facturacion
(codigo, codigocliente, fecha)
values (6910, 500,
to_date('31/12/2009', 'DD-MM-
YYYY'))
```

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

[13]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL –ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS
IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear vistas en Oracle

Implementación de Base de Datos
II
Por. Ing. Henry Lezcano
Semestre 2020

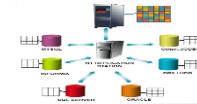
[14]

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

IMPLEMENTACION DE INDICE Y VISTAS

DEFINICION DE VISTAS



Una vista es un objeto. Una vista es una alternativa para mostrar datos de varias tablas; es como una tabla virtual que almacena una consulta. Los datos accesibles a través de la vista no están almacenados en la base de datos, en la base de datos se guarda la definición de la vista y no el resultado de ella.

Entonces, una vista almacena una consulta como un objeto para utilizarse posteriormente. Las tablas consultadas en una vista se llaman tablas base. En general, se puede dar un nombre a cualquier consulta y almacenarla como una vista.

Una vista suele llamarse también tabla virtual porque los resultados que retorna y la manera de referenciarlas es la misma que para una tabla.

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

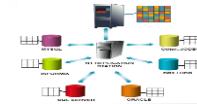
COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

IMPLEMENTACION DE INDICE Y VISTAS

DEFINICION DE VISTAS

Las vistas permiten:

- **simplificar la administración de los permisos de usuario:** se pueden dar al usuario permisos para que solamente pueda acceder a los datos a través de vistas, en lugar de concederle permisos para acceder a ciertos campos, así se protegen las tablas base de cambios en su estructura.
- **mejorar el rendimiento:** se puede evitar tipear instrucciones repetidamente almacenando en una vista el resultado de una consulta compleja que incluya información de varias tablas.

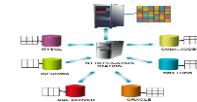


II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS

IMPLEMENTACION DE INDICE Y VISTAS

DEFINICION DE VISTAS



Las vistas permiten:

- **Podemos crear vistas con:** un subconjunto de registros y campos de una tabla; una unión de varias tablas; una combinación de varias tablas; un subconjunto de otra vista, combinación de vistas y tablas.
- Una vista se define usando un "select".

La sintaxis básica para crear una vista es la siguiente:

create view NOMBREVISTA as SUBCONSULTA;

El contenido de una vista se muestra con un "select":

select *from NOMBREVISTA;

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL -ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS
IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

Ejemplos de creación de Vistas en Oracle

En el siguiente ejemplo creamos la vista "vista_empleados", que es resultado de una combinación en la cual se muestran 4 campos:

```
create view vista_empleados as
select (e.apellido||' '||e.nombre) as
nombre, sexo,
s.nombre as seccion, e.cantidadhijos
from empleados e
join secciones s
on codigo=seccion;
```

Para ver la información contenida en la vista creada anteriormente escribimos los comandos DML:

select *from vista_empleados;

Podemos realizar consultas a una vista como si se tratara de una tabla:

**select seccion,count(*) as cantidad
from vista_empleados;**

II. Comandos de Definición y Manipulación del Lenguaje PL-SQL ORACLE

COMANDOS DEL LENGUAJE DE DEFINICION DE DATOS IMPLEMENTACION DE INDICE Y VISTAS

Cómo crear índices en Oracle

PRIVILEGIOS ASIGNAR A LOS USUARIOS

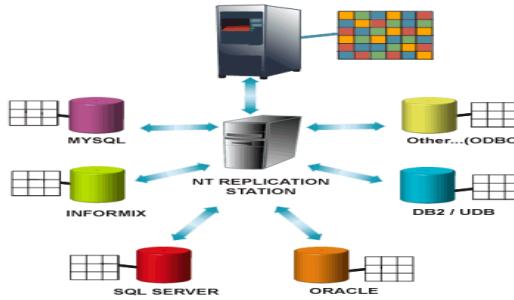
Grant create view to usuario;

Grant create view to javier;

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION

SISTEMAS DE BASE DE DATOS II
PL/SQL

II. Tipos de Datos y Funciones-ORACLE



Sistemas de Base de Datos II Por:
 Ing. Henry Lezcano II Semestre del
 2020

[1]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN LENGUAJE

- Existen cuatro categorías de tipos de datos: Escalares, Compuestos, Referencias, y LOB (Large Objects)
- Los tipos Escalares no tienen componentes, mientras que los tipos Compuestos si, por que las referencias son punteros a otros tipos.
- Los tipos de datos PL/SQL se definen un paquete llamado STANDARD, cuyos contenidos son accesibles desde cualquier bloque PL/SQL.
- El paquete STANDARD también define las funciones predefinidas SQL y de conversión disponibles en PL/SQL.

Sistemas de Base de Datos II Por:
 Ing. Henry Lezcano II Semestre del
 2020

[2]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

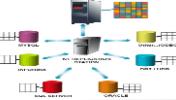
1. TIPOS ESCALARES

- Los tipos escalares validos son los mismos tipos que puede ser usado en una columna de la base de datos, con algunos adicionales.
- Existen siete familias de tipos escalares:
 - Numéricos
 - De Carácter
 - Raw
 - De Fecha
 - Rowid
 - Booleano
 - trusted

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[3]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipos Numericos:

Los tipos numéricos pueden almacenar valores enteros y reales

Existen tres tipo básico:

- NUMBER** pueden contener valores enteros y reales
- PLS_NUMBER** solo puede contener valores enteros
- BINARY_INTEGER** solo puede contener valores enteros

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[4]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipos Numericos:

- **NUMBER** : Este tipo puede contener un valor numérico entero o de punto flotante. Es igual al tipo **NUMBER** de la base datos.
 - La sintaxis para la declaración de una variable de tipo NUMBER es: **NUMBER(P, S)** ; donde **P** es la precisión es el numero y **S** la escala. La precisión es el numero de digito en el valor y la escala el numero de digito a la derecha del punto decimal. Ambas son opcionales, pero si se usan deben ser las dos. La precisión máximas es 38 y la escala puede variar entre -84 y 127.
 - Pueden usarse subtipos equivalentes a NUMBER como lo son: **DEC, DECIMAL, DOUBLE PRECISION, INTERGER, INT, NUMERIC, REAL, SMALLINT**)

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[5]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipos Numericos:

- **BINARY_INTEGER** : Para valores que solo se usen durante los cálculos, y no vayan ser almacenados en la base de datos, se puede utilizar este tipo.
 - Permite almacenar valores enteros con signos en un rango de **-2147483647 a +2147483647**.
 - Los valores se almacenan en formato binario con complemento a 2 lo que permite usarlo en los calculos sin necesidad de hacer ningún tipo de conversión.
 - **BINARY_INTEGER** también tiene subtipos definidos, pero estos cuentan con restricciones para su uso. Estos serían (**NATURAL, NATURALN, POSITIVE, POSITIVEN, SIGNTYPE**)

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[6]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipos Numericos:

- **PLS_INTEGER** : el rango es el mismo que **BINARY_INTEGER** de **-2147483647** a **+2147483647**.
- Los valores se almacenan en formato binario con complemento a 2 nativo. Sin embargo cuando se produce desbordamiento en un calculo en el que este involucrado este tipo de dato, se genera un error.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[7]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo Caracter:

- Las variables de tipo carácter permiten almacenar cadenas o datos tipos carácter. La familia esta compuesta por:

- **VARCHAR2**
- **CHAR**
- **LONG**
- **NCHAR**
- **NVARCHAR2**

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[8]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo Caracter:

VARCHAR2

- Es equivalente su comportamiento al **VARCHAR2** usado en la base datos. Pueden contener cadena de caracteres de longitud variable, con una longitud máxima especificada.
 - Su sintaxis es: **VARCHAR2(L)**; L es la longitud máxima de la cadena, no existiendo un valor predeterminado
 - La longitud máxima para **VARCHAR2** es de 32767 bytes.
 - Para una columna en la base de datos solo puede contener 4000 bytes. Si una variable requiere un mayor espacio debe almacenarse en un tipo **LONG**.
 - **VARCHAR** es equivalente a **VARCHAR2**

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[9]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo Caracter:

CHAR

- Las variables de este tipo son cadenas de caracteres de longitud fija.
 - Su sintaxis es: **CHAR(L)**; L es la longitud máxima de la cadena en bytes, si no se especifica la longitud tomara por defecto 1
 - La longitud máxima para **CHAR** es de 32767 bytes.
 - Para una columna en la base de datos solo puede contener 2000 bytes. Si una variable requiere un mayor espacio debe almacenarse en un tipo **VARCHAR2** o **LONG**.
 - **CHARACTER** es equivalente a **CHAR**

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[10]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo Caracter:

NCHAR y NVARCHAR2

- Tipos de datos del Oracle se emplean para almacenar cadenas de caracteres en un conjunto de caracteres distintos del propio lenguaje PL/SQL
 - A este conjunto de caracteres se conoce como conjunto nacional de caracteres.
 - Se especifican de la misma forma que CHAR Y VARCHAR2, lo longitud puede variar dependiendo del conjunto de caracteres nacionales
 - AL estudiante le corresponde indentificar cuando son usadas.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[11]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo Raw:

- Tipos de datos raw se emplean para almacenar datos binarios.

RAW : Son similares a las variables tipo CHAR excepto que no se realizan conversiones entre conjunto de caracteres.

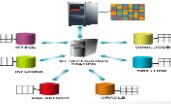
La Sintaxis es : **RAW(L)** donde L es la longitud en byte de la variable. Tiene las misma restricciones que la variable tipo CHAR.

LONG RAW : Son similares a las variables tipo LONG excepto que no se realizan conversiones entre conjunto de caracteres.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[12]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo DATE:

- Esta familia tiene un único miembro **DATE** y se comparte de la misma manera que el tipo equivalente de la base de datos.
- Se emplea para almacenar información sobre la fecha y la hora (año, mes, día, hora minuto y segundo).
- Una variable de este tipo tiene 7 bytes con un byte para cada componente.
- Para asignar valores a las variables de tipo DATE se emplea normalmente la función predefinida **TO_DATE**.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[13]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

1. TIPOS ESCALARES- Tipo DATE:

- La función **TO_DATE** permite convertir de manera sencilla las variables de caracteres a variables de tipo DATE.
- La función **TO_CHAR** realiza la conversión de tipo DATE a carácter.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[14]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



TIPOS DE DATOS EN PL/SQL

2. TIPOS COMPUESTOS

Hay disponibles 3 tipos de compuestos: registros, tablas y arreglos.

- Un tipo compuesto es aquel que consta de una serie de componentes.
- Una variable de tipo compuesto contendrá una o mas variables escalares.

3. TIPOS DE REFERENCIA

Se emplea para dar nombre a un espacio de memoria y hacer referencia a el en un programa

- Los tipos de referencia que son equivalentes a las variables tipo puntero definidas en C.

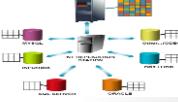
4. TIPOS LOB

Se emplean para almacenar objetos de gran tamaño. Este objeto puede ser hasta de 4 gigabytes. Estos pueden contener datos nos estructurados a los que se les puede acceder de mas eficientes y con menos restricciones.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[15]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



CONVERSIONES ENTRE TIPOS DE DATOS

PL/SQL puede manejar conversiones entre tipos de datos escalares de las distintas familias. Para los tipos de una familia se puede realizar conversiones sin ninguna restricción, excepto las impuestas a las variables.

Por ejemplo un CHAR(10) no puede convertirse en un VARCHAR2(1), un NUMBER(3,2) no puede convertirse en un NUMBER(3) porque el cuando se producen violaciones a las restricciones el compilador PL/SQL no dará un error pero se puede producir errores en tiempo de ejecución dependiendo de los valores de las variables a convertir.

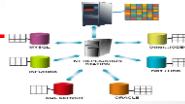
Los tipos compuesto no pueden convertirse entre si, porque son demasiados distintos. Pero se puede escribir un función que realice la conversión, basándose en tipo de dato que tenga en el programa.

Existen dos tipos de conversión: implícita y explícita

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[16]

II. Tipos de Datos y Funciones en Lenguaje PL-SQL - ORACLE



CONVERSIONES ENTRE TIPOS DE DATOS

Conversión explícita de tipos de datos

Las función de conversión de SQL también están disponibles en PL/SQL. Pueden ser empleadas cuando se requiera, para realizar conversiones explícitas entre variables de diferentes familia de tipos.

Función	Descripción	Familias que se Puede Convertir
TO_CHAR	Convierte su argumento en tipo VARCHAR2, dependiendo del especificador de formato opcional	Numéricos, Fechas
TO_DATE	Convierte su argumento en tipo DATE, dependiendo del especificador de formato opcional	Carácter
TO_NUMBER	Convierte su argumento en tipo NUMBER, dependiendo del especificador de formato opcional	Carácter
RAWTOHEX	Convierte un valor RAW en un representación hexadecimal de la cantidad en binario	Raw
HEXTORAW	Convierte una representación hexadecimal en el equivalente binario	Carácter(en representación hexadecimal)
CHARTOROWID	Convierte una representación de caracteres de un ROWID el formato interno binario	Carácter(en formato rawid d 18 caracteres)
ROWIDTOCHAR	Conviene una variable interna ROWID al formato externo de 18 caracteres	Rowid

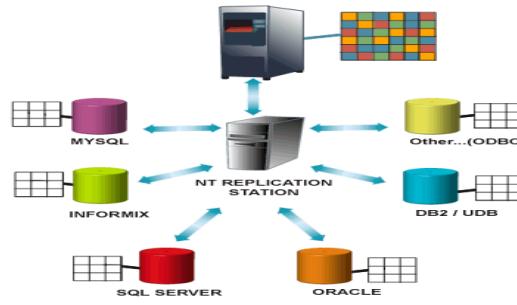
Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[17]

UNIVERSIDAD TECNOLOGICA DE PANAMA
FACULTAD DE INGENIERIA DE SISTEMAS
LICENCIATURA EN INGENIERIA DE SISTEMA DE INFORMACION

SISTEMAS DE BASE DE DATOS II

IMPLEMENTACION DE REGLAS DE INTEGRIDAD DE LA BASE DE DATOS



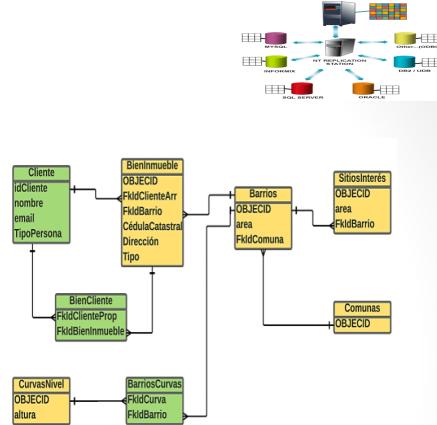
Por. Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

1

CONTENIDO

a. Diseño de Relaciones en el Modelo Relacional....

b. Diseño de Restricciones del Modelo Relacional....



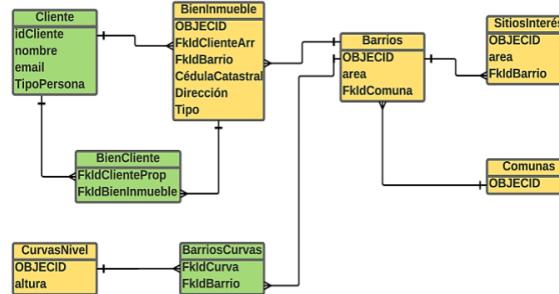
Por. Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

2

III. IMPLEMENTACION DE REGLAS DE INTEGRIDAD DE LA BASE DE DATOS



a. Diseño de Relaciones(tablas) del Modelo Relacional....



Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

[3]

b. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

- **Relación:** tabla bidimensional, a nivel lógico
- **Registro o tuplas:** fila de la tabla
- **Campo:** columna de la tabla

Ejemplo: relación ESCRITOR (2 registros, 4 campos)

DNI	Nombre	Dirección	Fecha
44345789	Ana Pérez	Sol, 17	9/5/1960
56123009	Luis Gómez	Feria,2	5/5/1961

Las relaciones se *enlazan mediante campos con contenido común*.

Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

[4]

b. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

Una relación de grado m consta de dos partes:

Cabecera: conjunto fijo de m campos.

Cada campo está definido por su **Nombre** y su **Dominio**

(que indica el tipo de valores que contendrá dicho campo).

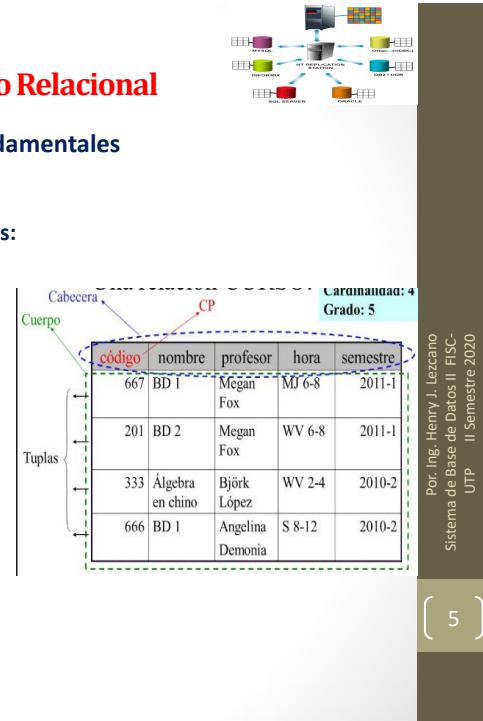
$\{(Nombre_1 : Dominio_1), \dots, (Nombre_m : Dominio_m)\}$

Cuerpo: conjunto variable de registros (también denominados tuplas).

Cada registro es un conjunto de m valores:

$Reg_1 \{(Nombre_1 : Valor_{1,1}), \dots, (Nombre_m : Valor_{1,m})\}$

...



a. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

Una relación de grado m consta de dos partes:

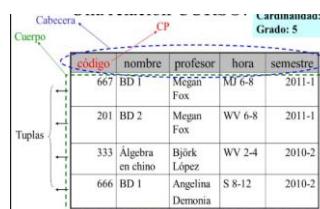
➤ Cada relación tiene asociado un **Nombre** que la identifica.

➤ Una relación de **grado m** puede representarse mediante una tabla bidimensional de m columnas y tantas filas como registros aparezcan en la relación.

➤ Cada **valor** de un registro **debe pertenecer** al correspondiente **dominio** especificado en la **cabecera**.

Ejemplo: relación ESCRITOR (2 registros, 4 campos)

DNI	Nombre	Dirección	Fecha
44345789	Ana Pérez	Sol, 17	9/5/1960
56123009	Luis Gómez	Feria,2	5/5/1961



a. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

Ejemplo: relación ESCRITOR (2 registros, 4 campos)

DNI	Nombre	Dirección	Fecha
44345789	Ana Pérez	Sol, 17	9/5/1960
56123009	Luis Gómez	Feria,2	5/5/1961



La cabecera de la relación ESCRITOR es:

- (DNI:Numérico), (Nombre:Texto), (Direccion:Texto), (Fecha:Fecha/Hora)

El cuerpo de la misma está formado por 2 registros:

- { (DNI:56123009), (Nombre:'Luis Gomez'), (Direccion:'Feria,2'), (Fecha:5/5/1961), }
- { (DNI:44345789), (Nombre:'Ana Perez', (Direccion:'Sol,17'), (Fecha:9/5/1960) }

Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

[7]

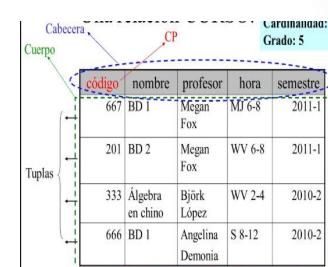
a. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

CABECERA

Cada relación tiene asociada, como vimos, una cabecera formada por un número fijo de campos.

- Notación: **NOMBRE1.Nombre2** denota el campo *Nombre2* de la cabecera de la relación **NOMBRE1**.
- Dos campos pertenecientes a la cabecera de la misma relación no pueden tener el mismo nombre.
- El orden de los campos en la cabecera de una relación no importa.



Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

[8]

Campos de relaciones distintas sí pueden tener el mismo nombre:

- **ESCRITOR.DNI** denota el campo DNI de la relación ESCRITOR.
- **CLIENTE.DNI** denota el campo DNI de la relación CLIENTE.

a. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

CUERPO

- Todos los registros del cuerpo en una relación deben tener el *mismo número de campos*, aunque *alguno* este *vacío*. En este caso, dicho campo vacío toma el valor **NULL**.
- Los **valores de los campos son atómicos**: fijado un registro, cada campo toma un único valor (no se admiten campos multivaluados).
- No se admiten registros duplicados. Dos registros de una relación deben diferir, al menos, en el valor de un campo.
- El orden de los registros en el cuerpo de una relación no importa.



Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

[9]

a. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

CAMPOS DE UNA RELACION

Cada campo debe poseer un **Nombre** (relacionado con los datos que contendrá) y debe tener asociado un **Tipo de dato**. Algunos tipos posibles (no los únicos) serían:

- **Texto:** cadenas de caracteres, ya sean letras, números con los que no realizar operaciones o símbolos.
- **Numérico:** números sobre los que tiene sentido realizar operaciones.
- **Fecha/hora:** almacena fechas, horas o ambas.
- **Sí/No:** datos que solo tengan dos posibilidades (verdadero-falso).
- **Autonumérico:** valor numérico (1,2,...) que el SGBD incrementa de modo automático cuando se añade un registro.



Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

[10]

a. Diseño de Relaciones del Modelo Relacional

Conceptos Fundamentales

CAMPOS DE UNA RELACION

Un campo puede poseer **opcionalmente** las siguientes propiedades:

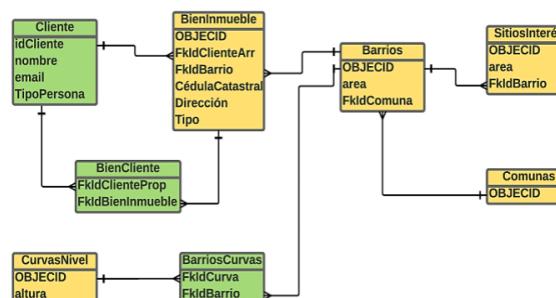
- **Descripción:** texto breve que aclara el contenido o la finalidad del campo.
- **Tamaño:** indica el tamaño máximo permitido (aplicable a campos de texto o numéricos).
- **Rango** de valores posibles, dentro de una lista de valores permitidos.
- **Requerido o NOT NULL:** no se permiten valores nulos para dicho campo.
- **Predeterminado:** se fija un valor por defecto para el campo.

Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISCS-
UTP II Semestre 2020

[11]

III. IMPLEMENTACION DE REGLAS DE INTEGRIDAD DE LA BASE DE DATOS

b. Diseño de Restricciones del Modelo Relacional....



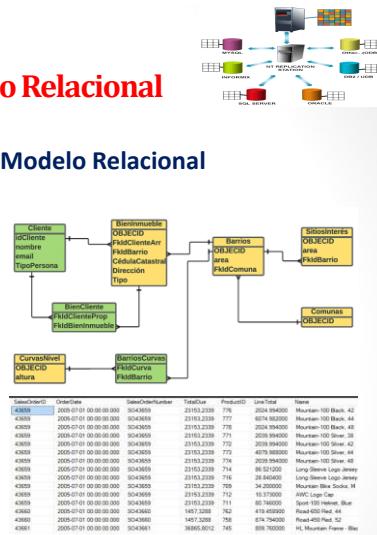
Por: Ing. Henry J. Lezcano
Sistema de Base de Datos II FISCS-
UTP II Semestre 2020

[12]

b. Diseño de Restricciones del Modelo Relacional

Restricciones Inherentes del Modelo Relacional

- ❑ No existen registros o filas repetidas (obligatoriedad de clave primaria).



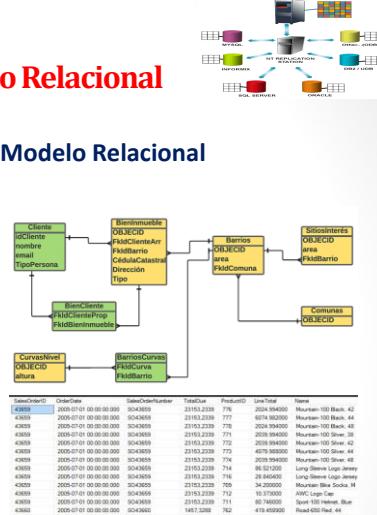
Por. Ing. Henry J. Lezzano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

13

b. Diseño de Restricciones del Modelo Relacional

Restricciones Inherentes del Modelo Relacional

- Cada atributo de cada fila o registro solo puede tomar un único valor sobre el dominio sobre el que está definido.



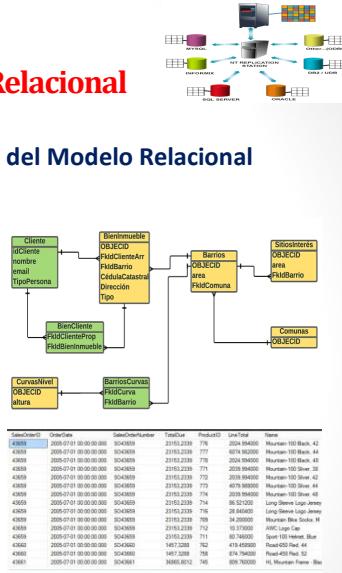
Por. Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

14

b. Diseño de Restricciones del Modelo Relacional

Restricciones Semánticas o del Usuario del Modelo Relacional

- **Restricción de Clave Primaria (PRIMARY KEY)**, permite declarar un atributo o conjunto de atributos como la clave primaria de una relación.



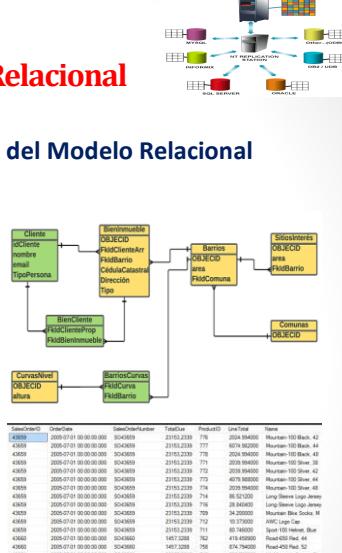
Por. Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

15

b. Diseño de Restricciones del Modelo Relacional

Restricciones Semánticas o del Usuario del Modelo Relacional

- ❑ **Restricción de Obligatoriedad (NOT NULL)**, permite declarar si uno o varios atributos de una relación debe tomar siempre un valor.



Por. Ing. Henry J. Lezcano
Sistema de Base de Datos II FISC-
UTP II Semestre 2020

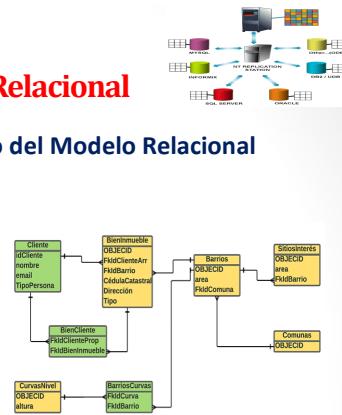
16

b. Diseño de Restricciones del Modelo Relacional

Restricciones Semánticas o del Usuario del Modelo Relacional

□ Restricción de Valor por Defecto (DEFAULT), permite que cuando se

inserte una fila o registro en una tabla, para aquellos atributos para los cuales no se indique un valor exacto se les asigne un valor por defecto.



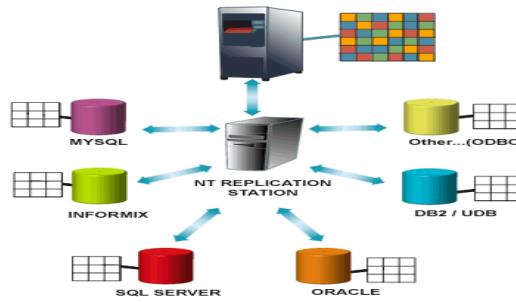
□ Restricción de Verificación o Chequeo (CHECK), en ocasiones puede ocurrir

que sea necesario especificar una condición que deben cumplir los valores de determinados atributos de una relación de la Base de Datos.

OrderID	OrderDate	Shipper	ShipVia	ShipName	TotalDue	ProductID	LineTotal	Name
10243	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	776	2024 994000	Muebles-100 Black, 42
10244	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	778	6074 994000	Muebles-100 Black, 44
10245	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	779	2024 994000	Muebles-100 Black, 46
10246	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	780	2024 994000	Muebles-100 Black, 48
10247	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	781	2024 994000	Muebles-100 Black, 50
10248	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	782	2024 994000	Muebles-100 Black, 52
10249	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	775	4074 993000	Muebles-100 Silver, 42
10250	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	776	2024 994000	Muebles-100 Silver, 44
10251	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	777	2024 994000	Muebles-100 Silver, 46
10252	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	778	2024 994000	Muebles-100 Silver, 48
10253	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	779	2024 994000	Muebles-100 Silver, 50
10254	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	780	2024 994000	Muebles-100 Silver, 52
10255	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	781	2024 994000	Muebles-100 Silver, 54
10256	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	782	2024 994000	Muebles-100 Silver, 56
10257	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	783	2024 994000	Muebles-100 Silver, 58
10258	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	784	2024 994000	Muebles-100 Silver, 60
10259	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	785	10 377000	Long Service Logo, Jersey
10260	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	786	34 200000	Long Service Logo, Jersey
10261	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	787	80 740000	Spit 100 Helvet, Blue
10262	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	788	433 920000	Spit 100 Helvet, White
10263	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	789	874 794000	Read 450 Net, 52
10264	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	790	874 794000	Read 450 Net, 54
10265	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	791	874 794000	Read 450 Net, 56
10266	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	792	874 794000	Read 450 Net, 58
10267	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	793	874 794000	Read 450 Net, 60
10268	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	794	874 794000	Read 450 Net, 62
10269	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	795	874 794000	Read 450 Net, 64
10270	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	796	874 794000	Read 450 Net, 66
10271	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	797	874 794000	Read 450 Net, 68
10272	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	798	874 794000	Read 450 Net, 70
10273	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	799	874 794000	Read 450 Net, 72
10274	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	800	874 794000	Read 450 Net, 74
10275	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	801	874 794000	Read 450 Net, 76
10276	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	802	874 794000	Read 450 Net, 78
10277	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	803	874 794000	Read 450 Net, 80
10278	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	804	874 794000	Read 450 Net, 82
10279	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	805	874 794000	Read 450 Net, 84
10280	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	806	874 794000	Read 450 Net, 86
10281	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	807	874 794000	Read 450 Net, 88
10282	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	808	874 794000	Read 450 Net, 90
10283	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	809	874 794000	Read 450 Net, 92
10284	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	810	874 794000	Read 450 Net, 94
10285	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	811	874 794000	Read 450 Net, 96
10286	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	812	874 794000	Read 450 Net, 98
10287	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	813	874 794000	Read 450 Net, 100
10288	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	814	874 794000	Read 450 Net, 102
10289	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	815	874 794000	Read 450 Net, 104
10290	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	816	874 794000	Read 450 Net, 106
10291	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	817	874 794000	Read 450 Net, 108
10292	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	818	874 794000	Read 450 Net, 110
10293	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	819	874 794000	Read 450 Net, 112
10294	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	820	874 794000	Read 450 Net, 114
10295	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	821	874 794000	Read 450 Net, 116
10296	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	822	874 794000	Read 450 Net, 118
10297	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	823	874 794000	Read 450 Net, 120
10298	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	824	874 794000	Read 450 Net, 122
10299	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	825	874 794000	Read 450 Net, 124
10300	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	826	874 794000	Read 450 Net, 126
10301	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	827	874 794000	Read 450 Net, 128
10302	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	828	874 794000	Read 450 Net, 130
10303	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	829	874 794000	Read 450 Net, 132
10304	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	830	874 794000	Read 450 Net, 134
10305	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	831	874 794000	Read 450 Net, 136
10306	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	832	874 794000	Read 450 Net, 138
10307	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	833	874 794000	Read 450 Net, 140
10308	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	834	874 794000	Read 450 Net, 142
10309	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	835	874 794000	Read 450 Net, 144
10310	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	836	874 794000	Read 450 Net, 146
10311	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	837	874 794000	Read 450 Net, 148
10312	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	838	874 794000	Read 450 Net, 150
10313	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	839	874 794000	Read 450 Net, 152
10314	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	840	874 794000	Read 450 Net, 154
10315	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	841	874 794000	Read 450 Net, 156
10316	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	842	874 794000	Read 450 Net, 158
10317	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	843	874 794000	Read 450 Net, 160
10318	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	844	874 794000	Read 450 Net, 162
10319	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	845	874 794000	Read 450 Net, 164
10320	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	846	874 794000	Read 450 Net, 166
10321	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	847	874 794000	Read 450 Net, 168
10322	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	848	874 794000	Read 450 Net, 170
10323	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	849	874 794000	Read 450 Net, 172
10324	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	850	874 794000	Read 450 Net, 174
10325	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	851	874 794000	Read 450 Net, 176
10326	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	852	874 794000	Read 450 Net, 178
10327	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	853	874 794000	Read 450 Net, 180
10328	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	854	874 794000	Read 450 Net, 182
10329	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	855	874 794000	Read 450 Net, 184
10330	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	856	874 794000	Read 450 Net, 186
10331	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	857	874 794000	Read 450 Net, 188
10332	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	858	874 794000	Read 450 Net, 190
10333	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	859	874 794000	Read 450 Net, 192
10334	2005-07-01 00:00:00	SOA-001	2	2024 Muebles	23101,2339	860	874 794000</td	

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMA DE INFORMACION
SISTEMAS DE BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

CURSORES -PL/SQL ORACLE



Sistemas de Base de Datos II Por:
 Ing. Henry Lezcano II Semestre del
 2020

[1]

CONTENIDO

Capítulo III. Procedimientos

- ***Fundamentos de Lenguaje PL/SQL***
- ***Cursos***
- ***Procedimientos***

Sistemas de Base de Datos II Por:
 Ing. Henry Lezcano II Semestre del
 2020

[2]

4.2. CURSORES

Que es un cursor?

Para poder procesar una orden SQL, Oracle asigna un área de memoria que se conoce como *área de contexto*. Esta área contiene la información necesaria para el procesamiento, incluyendo el numero de filas procesadas por la orden, un puntero a la versión analizada de la orden y en el caso de las consultas, el *conjunto activo*, que es el conjunto de filas resultado de la consulta.

Un *cursor* es un puntero al área de contexto. Mediante un cursor, un programa PL/SQL puede controlar el área de contexto y lo que en ella suceda a medida que se procesa la orden.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[3]



4.2. CURSORES

```

DECLARE
  /* Variable de salida para almacenar los resultados de la Consulta */
  v_StudentID      students.id%TYPE;
  v_FirstName       students.first_name%TYPE;
  v_LastName        students.last_name%TYPE;

  /* Valores de acoplamiento utilizado en la consulta */
  v_Major           students.major%TYPE := 'Computer Science';

  /* Declaración del Cursor */
  CURSOR c_Students IS
    SELECT id, first_name, last_name
    FROM students
    WHERE major = v_Major;

BEGIN
  /* Identificar las filas en el conjunto activo y preparar el procesamiento ulterior de los datos */
  OPEN c_Students;
  LOOP
    /* Recuperar cada fila del conjunto activo y almacenarlos en las variables PL/SQL */
    FETCH c_Students INTO v_StudentID, v_FirstName, v_LastName;

    /* Si no hay más filas que recuperar, salir del bucle */
    EXIT WHEN c_Students%NOTFOUND;
  END LOOP;
  /* Liberar los recursos usados para la consulta */
  CLOSE c_Students;
END;

```

El bloque mostrado ilustra el concepto de cursor explícito, donde se asigna explícitamente el nombre del cursor a una orden **SELECT**, mediante la orden **CURSOR.. IS**.

El resto de las ordenes SQL se utiliza un cursor implícito. Estos cursos implícitos, por su parte son procesados automáticamente por PL/SQL.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[4]



4.2. CURSORES

Procesamiento de Curosres Explicito

Los cuatro pasos PL/SQL necesario para el procesamiento de un cursor explicito son:

- *Declaración de Cursor*
- *Apertura del cursor para una consulta.*
- *Recogida de los resultados en variables PL/SQL*
- *Cierre del Cursor*

Declaración del cursor:

La declaración de un cursor define su nombre y asocia el cursor con una orden SELECT. La sintaxis es:

```
CURSOR nombre_cursor IS orden_SELECT;
```

Donde *nombre_cursor* es el nombre del cursor y *orden_SELECT* es la consulta que el cursor procesará. Los nombres de curosres tienen las mismas reglas de ámbito y visibilidad a los identificadores PL/SQL

4.2. CURSORES

Procesamiento de Curosres Explicito

Declaración del cursor:

La declaración de un cursor define su nombre y asocia el cursor con una orden SELECT. La sintaxis es:

```
/* Declaración correcta de un curso */
DECLARE
    v_Departement      classes.department%TYPE;
    v_Course           classes.course%TYPE;
CURSOR c_Classes IS
    SELECT * from classes
    WHERE department = v_Department
    AND course = v_Course;
```

```
/* Declaración incorrecta de un curso */
DECLARE
    CURSOR c_Classes IS
        SELECT * from classes
        WHERE department = v_Department
        AND course = v_Course;
    v_Departement      classes.department%TYPE;
    v_Course           classes.course%TYPE;
```

Una declaración de cursor puede hacer referencia a variables PL/SQL en la clausula WHERE.

Estas variables se consideran variables acopladas, deben ser visibles en el punto donde se declara el cursor, como la forma presentada.

Si embargo una declaración con la planteada en el segundo ejemplo seria ilegal.

Es recomendable que las variables de referencias en una declaración de cursor sean declaradas antes de la referencia y los curosres al final.



4.2. CURSORES

Procesamiento de Curosres Explicito

Apertura del cursor:

La sintaxis para abrir un cursor es:

OPEN nombre_cursor;

Donde *nombre_cursor* representa el nombre del cursor previamente declarado.

Cuando se abre el cursor suelen ocurrir tres cosas:

- Se examinan los valores de las variables acopladas
- Se determina el conjunto activo, basándose en los valores de dichas variables
- Se hace apuntar el puntero del conjunto activo a la primera fila.

Las variables acopladas se examinan al momento de abrir el cursor y solo en el ese momento. Cualquier cambio que se de a estas variables en el proceso no tendrá efecto alguno sobre la consulta.



Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[7]

4.2. CURSORES

Procesamiento de Curosres Explicito

Apertura del cursor:

Las variables acopladas se examinan al momento de abrir el cursor y solo en el ese momento. Cualquier cambio que se de a estas variables en el proceso, no tendrá efecto alguno sobre la consulta.

```

DECLARE
  v_RoomID      classes.room_id%TYPE;
  v_Building    rooms.building%TYPE;
  v_Department  classes.department%TYPE;
  v_Course      classes.course%TYPE;
  CURSOR c_Building IS
    SELECT building
    FROM rooms,classes
    WHERE rooms.room_id = classes.room_id
    AND department = v_Department
    AND course = v_Course;
BEGIN
  -- Asignar las variables de Acoplamiento antes de abrir el cursor
  v_Department := 'HIS';
  v_Course := 101;
  -- Abrir el Cursor
  OPEN c_Building;
  -- Reasignar las variables de acoplamiento – No tienen efecto alguno, ya que el cursor esta abierto
  v_Department := 'XXX';
  v_Course := -1;
END;
  
```

Aunque las variables de acoplamiento cambien después de la orden **OPEN**, el conjunto activo de la consulta no cambia. A este hecho se le conoce con el nombre de **consistencia de lectura** y se diseño así para asegurar la integridad de los datos.

Es legal abrir un cursor que ya esta abierto. PL/SQL ejecutara implícitamente una orden **CLOSE** antes de re-abrir el cursor con la segundo orden **OPEN**. También puede haber mas de un cursor abierto al mismo tiempo



Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[8]

4.2. CURSORES

Procesamiento de Curosres Explicito

Extracción de los datos del cursor:

La cláusula INTO de la consulta es parte de la orden **FETCH**. Dicha orden tiene dos formas posibles,

```
FETCH nombre_cursor INTO lista_variables;
Y
FETCH nombre_cursor INTO registro_PL/SQL;
```

Donde *nombre_cursor* representa el nombre del cursor previamente declarado y abierto, *lista_variables* es una lista de variables PL/SQL previamente declaradas y separadas por comas. *registro_PL/SQL* es un registro PL/SQL previamente declarado.

En ambos casos la variable o variables de la cláusula INTO debe ser compatible en cuanto tipo con la lista de selección de la consulta.

Dada la declaración precedente del cursor *c_Building*, la siguiente orden **FETCH** sería válida:

```
FETCH c_Building INTO v_Building;
```

4.2. CURSORES

Procesamiento de Curosres Explicito

Extracción de los datos del cursor:

En el siguiente bloque se proporcionan ejemplos de órdenes **FETCH** legales e ilegales:

```
DECLARE
  v_Department      classes.department%TYPE;
  v_Course          classes.course%TYPE;
  CURSOR  c_AllClasses IS
  SELECT *
  FROM  classes;
  v_ClassesRecord  c_AllClasses%ROWTYPE;
BEGIN
  OPEN  c_AllClasses;
  /* Esta es una orden FECH correcta, que almacena la primera fila en el registro PL/SQL con una estructura igual a
  la lista la selección de la consulta */
  FETCH c_AllClasses INTO v_ClassesRecord;
  /* Esta orden FETCH es incorrecta, ya que la lista de la selección de la consulta devuelve 7 columnas de la tabla
  classes, y solo estamos almacenando en dos 2 variables: Estos dará un error de asignación de valores E-PLS-394 */
  FETCH c_AllClasses INTO v_Department, v_Course;
END;
```

Después de cada **FETCH**, se incrementa el puntero del conjunto activo, para que apunte a la siguiente fila.

De esta forma, cada **FETCH** devolverá filas sucesivas del conjunto activo, hasta que devuelva el conjunto completo.

El atributo **%NOTFOUND** se utiliza para determinar cuando se ha terminado de extraer todo el conjunto activo. Es uno de los atributos de los curosres

4.2. CURSORES



Procesamiento de Cursos Explicito

Cierre de un cursor:

Cuando se ha terminado de extraer el conjunto activo, debe cerrarse el cursor. Esta acción informa a PL/SQL de que el programa ha terminado de usar el cursor y que se pueden liberar los recursos con él asociados. Estos recursos incluyen el área de almacenamiento empleada para contener el conjunto activo, así como cualquier espacio temporal usado en la determinación de dicho conjunto.

```
CLOSE nombre_cursor ;
```

Donde *nombre_cursor* representa el nombre del cursor previamente declarado y abierto.

Una vez se cierra el cursor, es ilegal realizar extracciones de él. Si se intentara hacerlo, se produce el error oracle: ORA-1001 or ORA-1002, cursor invalido, Fetch fuera de secuencia.

Dada la declaración precedente del cursor *c_Building*, la siguiente orden CLOSE sería valida:

```
CLOSE c_Building;
```

4.2. CURSORES



Procesamiento de Cursos Explicito

Atributos de los cursos:

Existen cuatro atributos que pueden ser aplicados a los cursos. Estos atributos se añaden, en un bloque PL/SQL al nombre del curso, de forma similar a %TYPE y %ROWTYPE, sin embargo la diferencia es que los atributos del cursor no devuelven un tipo, sino un valor que puede emplearse como parte de una expresión. Estos atributos son:

- **%FOUND:** es un atributo booleano. Devuelve TRUE si la ultima orden *FETCH* devolvió una fila y FALSE en caso contrario. Si el cursor no esta abierto y tratamos de comprobar el valor de %FOUND mandara un error ORA-1001.
- **%NOTFOUND:** se comporta de forma opuesta a %FOUND; si la extracción anterior devuelve una fila , entonces %NOTFOUND tiene el valor FALSE. %NOTFOUND devuelve el valor TRUE solo si la extracción anterior no devuelve una fila. Este atributo se utiliza a menudo como condición de salida para un bucle de extracción. Si el cursor no esta abierto y tratamos de comprobar el valor de %NOTFOUND mandara un error ORA-1001

4.2. CURSORES

Procesamiento de Cursos Explicito

Atributos de los cursos:

Existen cuatro atributos que pueden ser aplicados a los cursos. Estos atributos se añaden, en un bloque PL/SQL al nombre del curso, de forma similar a %TYPE y %ROWTYPE, sin embargo la diferencia es que los atributos del cursor no devuelven un tipo, sino un valor que puede emplearse como parte de una expresión. Estos atributos son: [continuación....](#)

- **%ISOPEN:** este atributo booleano se utiliza para determinar si el cursor asociado esta o no abierto. Si la esta, %ISOPEN devuelve TRUE; si no devuelve FALSE.
- **%ROWCOUNT:** este atributo numérico devuelve el numero de filas extraídas por el cursor hasta el momento. Si el cursor no esta abierto y tratamos de comprobar el valor de %ROWCOUNT mandara un error ORA-1001

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[13]



4.2. CURSORES

Procesamiento de Cursos Explicito

Cursos Parametrizados:

Existen otras formas de emplear las variables de acoplamiento en un cursor. Existe un tipo de cursor, el cursor parametrizado, admite argumentos de la misma forma que los procedimientos.

Veamos la siguiente declaración de cursor:

```
DECLARE
  v_Department      classes.department%TYPE;
  v_Course          classes.course%TYPE;
  CURSOR  c_Classes IS
    SELECT * FROM classes
    WHERE department = v_Department
    AND    course   = v_Course;

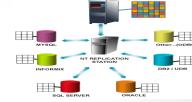
  DECLARE
  CURSOR  c_Classes (v_Department classes.department%TYPE,
                      v_Course   classes.course%TYPE) IS
    SELECT * FROM classes
    WHERE department = v_Department
    AND    course   = v_Course;
```

- **c_Classes** tienen dos variables de acoplamiento, **v_Department** y **v_Course**.
- El cursor **c_Classes** podría ser modificado para obtener un cursor parametrizado.
- En el caso de los cursos parametrizados, se utiliza la orden **OPEN** para pasar los valores reales al cursor. Y se podría abrir con:

```
OPEN  c_Classes ('HIS', 101);
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[14]



4.2. CURSORES

Procesamiento de Curosres Implícito

Los curosres implícitos sirven para procesar ordenes **SELECT** que devuelven mas de una fila. Todas las ordenes SQL se ejecutan dentro del área de contexto por lo tanto, tienen un cursor que apunta a dicha área. Este cursor se conoce con el nombre de **cursor SQL**.

A diferencia de los curosres explícitos, el programa no abre ni cierre el cursor SQL, sino que PL/SQL lo abre de forma implícita, procesa la orden SQL en el contenido y cierra el cursor después.

Los curosres implícitos sirven para procesar la ordenes **INSERT, UPDATE, DELETE**, y las ordenes **SELECT .. INTO** de una sola fila. A este tipo de curosres se le pueden aplicar los atributos del cursor explícito.

El siguiente bloque ejecutara una orden INSERT si la orden UPDATE no encuentra ninguna fila coincidente:

```

BEGIN
    UPDATE rooms
    SET number_seats = 100
    WHERE room_id = 99980;
    -- Si la anterior orden UPDATE no se aplica a ninguna fila, inserta una nueva fila en la tabla rooms.
    IF SQL%NOTFOUND THEN
        INSERT INTO rooms ( room_id, number_seats)
        VALUES (99980, 100);
    END IF;
END;

```

4.2. CURSORES

Procesamiento de Curosres Implícito

*El siguiente bloque ejecutara una orden INSERT si la orden UPDATE no encuentra ninguna fila coincidente y podríamos hacer el mismo proceso anterior si usamos el atributo **SQL%ROWCOUNT***

```

BEGIN
    UPDATE rooms
    SET number_seats = 100
    WHERE room_id = 99980;
    -- Si la anterior orden UPDATE no se aplica a ninguna fila, inserta una nueva fila en la tabla rooms.
    IF SQL%ROWCOUNT = 0 THEN
        INSERT INTO rooms ( room_id, number_seats)
        VALUES (99980, 100);
    END IF;
END;

```

Aunque se puede emplear **SQL%NOTFOUND** con las ordenes **SELECT .. INTO** no resulta útil, por la orden **SELECT .. INTO** produce el error ORA-1403 : no data found y cuando esto ocurre la orden se pasa de forma inmediata al manejo de excepciones.

4.2. CURSORES

Procesamiento de Curosres Implícito

Aunque se puede emplear **SQL%NOTFOUND** con las ordenes **SELECT .. INTO** no resulta útil, por la orden **SELECT .. INTO** produce el error ORA-1403 : no data found y cuando esto ocurre la orden se pasa de forma inmediata al manejo de excepciones.

Veamos el siguiente ejemplo:

```

DECLARE
    -- Registro para almacenar la información acerca de una clase.
    V_RoomData    rooms%ROWTYPE;
BEGIN
    -- Extraer la información sobre la clase ID -1
    SELECT * INTO V_RoomData
    FROM rooms
    WHERE room_id = -1;
    /* La siguiente orden no se ejecutará nunca, ya que el control pasa inmediatamente al gestor de excepciones */
    IF SQL%NOTFOUND THEN
        INSERT INTO temp_table ( char_col) VALUES ( 'Not Found');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        INSERT INTO temp_table ( char_col) VALUES ( 'Not Found, Exception Handler');
    END;

```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[17]



4.2. CURSORES

Bucles de Extracción Mediante Cursor

La operación mas común que se puede realizar con un cursor consiste en extraer todos las filas de un conjunto activo. Para esto usamos un **bucle de extracción**, que no es mas que un bucle que procesa una a una las filas del conjunto activo. Existen diferentes tipos de bucles de extracción mediante curosres.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[18]

Bucle Simple

Se usa la sintaxis de bucles simples conocidos (**LOOP .. END LOOP**) para el procesamiento, controlándose el numero de veces que se ejecuta el bucle mediante atributos explícitos del cursor.



4.2. CURSOS

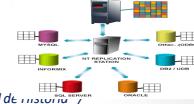
Bucle Simple: Ejemplo1 de extracción usando este medio

```

DECLARE
  /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  v_StudentID  students.id%TYPE;
  v_FirstName  students.first_name%TYPE;
  v_LastName   students.last_name%TYPE;
  -- Cursor para recuperar la información sobre los estudiantes de Historia
  CURSOR c_HistoryStudents IS
    SELECT id, first_name, last_name
    FROM students
    WHERE major = 'History';
  BEGIN
    -- Abre el cursor e inicializa el conjunto activo
    OPEN c_HistoryStudents;
    LOOP
      -- Recupera la información del siguiente estudiante
      FETCH c_HistoryStudents INTO v_StudentID, v_FirstName, v_LastName ;
      -- Salida del bucle cuando no hay más filas por recuperar
      EXIT WHEN c_HistoryStudents%NOTFOUND;
      /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla registered_students.
      Registra también el nombre y el apellido en la tabla temp_table */
      INSERT INTO registered_students ( students_id, department, course)
      VALUES ( v_StudentID, 'HIS', 301);

      INSERT INTO temp_table ( numcol, char_col)
      VALUES ( v_StudentID, v_FirstName || ' ' || v_LastName);
    END LOOP;
    -- Libera los recursos utilizados por el curso
    CLOSE c_HistoryStudents;
    -- Confirmamos el trabajo
    COMMIT;
  END;

```



Para este bloque la orden **EXIT WHEN**, esta inmediatamente después de la orden **FETCH**. Después de extraer la ultima fila, **c_HistoryStudents%NOTFOUND** toma el valor de **TRUE** y sale del bucle.

Se estructura de esa forma la orden **EXIT WHEN** antes del procesamiento de los datos, con el objeto de asegurar que no se procesen filas duplicadas

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

19

4.2. CURSOS

Bucle Simple: Ejemplo2 de extracción usando este medio

```

DECLARE
  /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  v_StudentID  students.id%TYPE;
  v_FirstName  students.first_name%TYPE;
  v_LastName   students.last_name%TYPE;
  -- Cursor para recuperar la información sobre los estudiantes de Historia
  CURSOR c_HistoryStudents IS
    SELECT id, first_name, last_name
    FROM students
    WHERE major = 'History';
  BEGIN
    -- Abre el cursor e inicializa el conjunto activo
    OPEN c_HistoryStudents;
    LOOP
      -- Recupera la información del siguiente estudiante
      FETCH c_HistoryStudents INTO v_StudentID, v_FirstName, v_LastName ;
      -- Salida del bucle cuando no hay más filas por recuperar
      EXIT WHEN c_HistoryStudents%NOTFOUND;
      /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla registered_students.
      Registra también el nombre y el apellido en la tabla temp_table */
      INSERT INTO registered_students ( students_id, department, course)
      VALUES ( v_StudentID, 'HIS', 301);

      INSERT INTO temp_table ( numcol, char_col)
      VALUES ( v_StudentID, v_FirstName || ' ' || v_LastName);
    END LOOP;
    -- Libera los recursos utilizados por el curso
    CLOSE c_HistoryStudents;
    -- Confirmamos el trabajo
    COMMIT;
  END;

```



La ultima orden **FETCH** no modificará **v_StudentID**, **v_FirstName**, ni **v_LastName**, puesto que ya no hay fila en el conjunto activo.

Las variables de salida mantendrán los valores correspondientes a la ultima fila extraída. Pero como la comprobación se realiza después el procesamiento de estos valores duplicados se insertan en las tablas **registered_students** y **temp_table**, lo que no deberá ser correcto.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

20

4.2. CURSORES

Bucle WHILE También se puede construir un bucle de extracción mediante un cursor usando la sintaxis WHILE .. LOOP

```

DECLARE
  -- Cursor para recuperar la información sobre los estudiantes de Historia
  CURSOR c_HistoryStudents IS
    SELECT id, first_name, last_name
    FROM students
    WHERE major = 'History';
  -- Declaración el registro para almacenar información extraída
  v_StudentData c_HistoryStudents%ROWTYPE;
BEGIN
  -- Abre el cursor e inicializa el conjunto activo
  OPEN c_HistoryStudents;
  -- Recupera la información del siguiente estudiante
  FETCH c_HistoryStudents INTO v_StudentData;
  -- El bucle continua mientras haya mas filas que extraer
  WHILE c_HistoryStudents%FOUND LOOP
    /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
    registered_students. Registro también el nombre y el apellido en la tabla temp_table */
    INSERT INTO registered_students ( students_id, department, course)
    VALUES ( v_StudentData.ID, 'HIS', 301);

    INSERT INTO temp_table ( numcol, char_col)
    VALUES ( v_StudentData.ID, v_StudentData.first_name || ' ' || v_StudentData.last_name);
    -- Recuperar la fila siguiente. La condición %FOUND se comprobara antes de que el bucle continúe
    FETCH c_HistoryStudents INTO v_StudentData;
  END LOOP;
  -- Libera los recursos utilizados por el curso
  CLOSE c_HistoryStudents;
  -- Confirmamos el trabajo
  COMMIT;
END;

```

Este ejemplo de extracción se comporta de la misma forma que el ejemplo **LOOP .. END LOOP** de programas anteriores. En este caso del **FETCH** aparece dos veces en el bloque, una antes del bucle y la otra después del procesamiento incluido en este.

Es necesario para que la condición del bucle (**c_HistoryStudents%FOUND**) sea evaluada en cada iteración del bucle.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[21]



4.2. CURSORES

Bucle de cursor FOR

Los dos tipos de extracción descritos requieren un procesamiento explícito del cursor, mediante órdenes OPEN, FETCH y CLOSE. PL/SQL proporciona un tipo de bucle más simple, que realiza de modo implícito el procesamiento del cursor. El bucle del cursor FOR

```

DECLARE
  -- Cursor para recuperar la información sobre los estudiantes de Historia
  CURSOR c_HistoryStudents IS
    SELECT id, first_name, last_name
    FROM students
    WHERE major = 'History';
BEGIN
  /* Inicio del bucle. Aquí se ejecuta una orden OPEN
  implícita sobre c_HistoryStudents */
  FOR v_StudentData IN c_HistoryStudents LOOP
    -- Aquí se ejecuta una orden FETCH implícita

    /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
    registered_students. Registro también el nombre y el apellido en la tabla temp_table */
    INSERT INTO registered_students ( students_id, department, course)
    VALUES ( v_StudentData.ID, 'HIS', 301);

    INSERT INTO temp_table ( numcol, char_col)
    VALUES ( v_StudentData.ID, v_StudentData.first_name || ' ' || v_StudentData.last_name);
    -- Antes de continuar con el bucle, aquí se hace una comprobación implícita de c_HistoryStudents%NOTFOUND.

  END LOOP;
  -- Ahora el bucle ha terminado, se hace cierre implícito del cursor c_HistoryStudents
  -- Confirmamos el trabajo
  COMMIT;
END;

```

En primer lugar, el registro **v_StudentData** no se declara en la sección declarativa del bloque, el compilador PL/SQL la declara de forma implícita.

En segundo lugar, el bucle abre, extrae los datos y cierra **c_HistoryStudents%FOUND** de manera implícita.

Los bucles de cursor FOR tienen la ventaja de proporcionar la funcionalidad de un bucle de extracción mediante cursor de una forma simple y limpia, con una sintaxis mínima.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

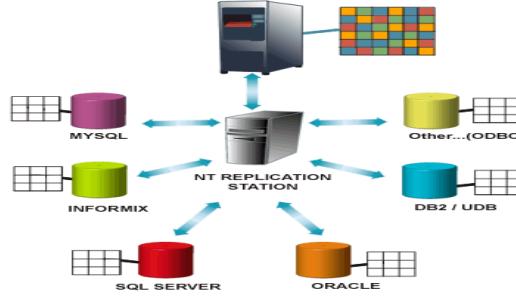
[22]



UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA EN SISTEMAS DE INFORMACION.

SISTEMAS DE BASE DE DATOS II ORACLE PROGRAMACION PL/SQL

Procedimiento –PL/SQL ORACLE



Por: Ing. Henry Lezcano Sistemas
 de Base de Datos II II Semestre del
 2020

[1]

CONTENIDO

Capítulo IV. Procedimientos

- *Fundamentos de Lenguaje PL/SQL*
- *Cursos*
- *Procedimientos*

Por: Ing. Henry Lezcano Sistemas
 de Base de Datos II II Semestre del
 2020

[2]

4.3 PROCEDIMIENTOS

- Los procedimientos y funciones de PL/SQL se comportan de manera similar a los procedimiento y funciones de otros lenguajes, comparten muchas de sus propiedades.
- También son conocidos como subprogramas.

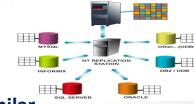
```
CREATE OR REPLACE PROCEDURE AddNewStudent (
    p_StudentID      students.id%TYPE,
    p_FirstName       students.first_name%TYPE,
    p_LastName        students.last_name%TYPE
    p_Major           students.major%TYPE ) AS
BEGIN
    -- Inserta una nueva fila en la tabla students. Usa
    -- student_sequence para generar el nuevo ID del estudiante y
    -- asigna el valor 0 a current_credits.

    INSERT INTO students (ID, first_name, last_name , major, current_credits)
        VALUES ( student_sequence.next, p_FirstName, p_LastName, p_Major, 0 );

    COMMIT;
END AddNewStudent;
```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[3]



4.3 PROCEDIMIENTOS

Una vez creado el procedimiento, puede ser invocado desde otro bloque PL/SQL, por ejemplo

```
BEGIN
    AddNewStudent ( 'David', 'Dinsmore', 'Music' );
END;
```

Este ejemplo ilustra varios puntos de importancia:

- En primer lugar se crea el procedimiento **AddNewStudent**, con la orden **CREATE OR REPLACE PROCEDURE**. Al crear el procedimiento, este se compila y se almacena en la Base de Datos en forma compilada.
- Cuando es invocado, puede pasársele parámetros. Para el ejemplo, se pasa en tiempo de ejecución el nombre, apellido y especialidad del nuevo estudiante. Dentro del procedimiento el parámetro **p_FirstName** tendrá el valor de 'David', **p_LastName** tendrá el valor de 'Dinsmore' y **p_Major**, tendrá el valor de 'Music', ya que se pasan estas literales al procedimiento en el momento de invocarlo.
- Una llamada a un procedimiento es una orden PL/SQL por si misma. La llamada no se produce como parte de la expresión. Cuando se llama a un procedimiento, el control pasa a la primera orden ejecutable dentro de él. Cuando procedimiento termina, se devuelve el control a la orden que sigue a la llamada al procedimiento.
- Un procedimiento es un bloque PL/SQL, con una sección declarativa ejecutable y una sección de manejo de excepciones. Al igual que los bloques anónimos, la única sección obligatoria es la sección ejecutable. **AddNewStudent** solo consta de una sección ejecutable.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[4]



4.3 PROCEDIMIENTOS

Creación de un Procedimiento

La sintaxis para la orden CREATE OR REPLACE PROCEDURE es

```
CREATE [OR REPLACE] PROCEDURE nombre_procedimiento (
  [ ( argumento [{IN | OUT | IN OUT}] tipo,
  ...
  argumento [{IN | OUT | IN OUT}] tipo ) { IS | AS }
  cuadro_procedimiento
```

Donde *nombre_procedimiento* es el nombre del procedimiento que se quiere crear, *argumento* es el nombre de un parámetro, tipo es el tipo del parámetro asociado y el *cuadro_procedimiento* es un bloque PL/SQL que contiene el código del procedimiento.

- Para poder cambiar el código de un procedimiento, es necesario eliminarlo y volverlo a crear.
- La palabra clave OR REPLACE permite hacer esto, en una única operación.
- Si queremos eliminar un procedimiento usamos la orden DROP PROCEDURE *nombre_procedimiento*.
- Si el procedimiento existe y no se le incluye la palabra clave OR REPLACE la orden CREATE mandara un error de Oracle. Ora-009955 name is ready used by an existing object.
- Al igual que sucede con la orden CREATE, la creación de un procedimiento es una orden DDL, así que el COMMIT es implícito. Se puede usar IS o AS ambas son equivalentes.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[5]



4.3 PROCEDIMIENTOS

Parámetros y Modos

Dado el procedimiento AddNewStudent , podemos invocarlo desde el siguiente bloque PL/SQL anónimo:

```
DECLARE
  -- Variables que describen al nuevo estudiante
  v_NewFirstName      students.first_name%TYPE := 'Margaret';
  v_NewLastName       students.last_name%TYPE := 'Mason';
  v_NewMajor          students.major%TYPE := 'History';
BEGIN
  -- Añade Margaret Mason a la Base de Datos
  AddNewStudent(v_NewFirstName, v_NewLastName, v_NewMajor );
END;
```

- Las variables declaradas en el bloque precedente (v_NewFirstName, v_NewLastName, v_NewMajor) se pasan como argumentos a **AddNewStudent** .
- Bajo esta perspectiva dichas variables reciben el nombre de *parametros reales*, mientras que los parametros en la declaración del procedimiento (p_FirstName, p_LastName, p_Major) se denominan *parametros formales*.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[6]



4.3 PROCEDIMIENTOS

Parámetros y Modos



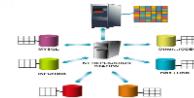
- Los parámetros reales contienen los valores que se pasan al procedimiento cuando este es invocado y reciben el resultado del procedimiento cuando este termina.
- Los valores de los parámetros reales son los que usan dentro del procedimiento.
- Los valores de los parámetros formales son meros contenedores para los valores de los parámetros reales.
- Cuando se llama al procedimiento se asigna el valor de los parámetros reales a los parámetros formales.
- Dentro del procedimiento, se hace referencia a dichos valores mediante los parámetros formales.
- Cuando el procedimiento termina, se asigna el valor de los parámetros formales a los parámetros reales siguiendo las reglas de asignación y conversión del PL/SQL.
- Los parámetros formales pueden tener 3 modos: IN, OUT o IN OUT.
- Sino se especifica el modo de un parámetro formal se adopta por defecto IN.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[7]

4.3 PROCEDIMIENTOS

Parámetros y Modos



Modo	Descripción
IN	El valor del parámetro real se pasa al procedimiento cuando este es invocado. Dentro del procedimiento, el parámetro formal se considera como de sólo lectura y no puede ser cambiado. Cuando se termina el procedimiento, y se devuelve el control al entorno que realizó la invocación, el parámetro real no sufre cambio.
OUT	Se ignora cualquier valor que tenga el parámetro real cuando se llama el procedimiento. Dentro del procedimiento, el parámetro formal se considera como de sólo escritura no puede ser leído, sino que tan solo puede asignársele valores. Cuando termina el procedimiento y se devuelve al entorno que realizó la llamada, los contenidos del parámetro formal se asignará al parámetro real.
IN OUT	Este modo es una combinación de IN y OUT. El valor del parámetro real se pasa al procedimiento cuando este es invocado. Dentro del procedimiento, el parámetro formal puede ser tanto leído como escrito . Cuando termina el procedimiento y se devuelve el control al entorno que realizó la llamada, los contenidos del parámetro formal se asignan al parámetro real.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[8]

4.3 PROCEDIMIENTOS

Parámetros y Modos

Cuales de las asignaciones de los variables es correcta?

```
CREATE OR REPLACE PROCEDURE ModeTest (
    p_InParameter    IN    NUMBER,
    p_OutParameter  OUT    NUMBER,
    p_InOutParamter IN OUT NUMBER) IS
    v_LocalVariable Number ;
BEGIN
    v_LocalVariable := p_InParameter;
    p_InParameter  := 7;
    p_OutParameter := 7;
    v_LocalVariable := p_OutParameter;
    v_LocalVariable := p_InOutParamter;
    p_InOutParamter := 7;
END ModeTest;
```



Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[9]

4.3 PROCEDIMIENTOS

Parámetros y Modos

Cual de la siguientes invocaciones es correcta?

```
CREATE OR REPLACE PROCEDURE ModeTest (
    p_InParameter    IN    NUMBER,
    p_OutParameter  OUT    NUMBER,
    p_InOutParamter IN OUT NUMBER) IS
    v_LocalVariable Number ;
BEGIN
    v_LocalVariable := p_InParameter;
    p_InParameter  := 7;
    p_OutParameter := 7;
    v_LocalVariable := p_OutParameter;
    v_LocalVariable := p_InOutParamter;
    p_InOutParamter := 7;
END ModeTest;
```



-- Invocación No.1

```
DECLARE
    v_Variable1 NUMBER,
    v_Variable2 NUMBER,
BEGIN
    ModeTest(12, v_Variable1, v_Variable2);
END;
```

Invocación No. 2

```
DECLARE
    v_Variable1 NUMBER,
    v_Variable2 NUMBER,
BEGIN
    ModeTest(12, v_Variable1, 11);
END;
```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[10]

4.3 PROCEDIMIENTOS

El Cuerpo del Procedimiento

- El cuerpo del procedimiento es un bloque PL/SQL, con sus acciones declarativa, ejecutable y de manejo de excepciones.
- La sección declarativa se sitúa entre la palabra clave IS o AS y la palabra BEGIN.
- La ejecutable (la única obligatoria) esta comprendida entre las palabras clave BEGIN y EXCEPTION.
- La sección de excepciones, por su parte esta delimitada por las palabras clave EXCEPTION y END.
- La estructura de un procedimiento tendrá la forma siguiente:

```
CREATE OR REPLACE PROCEDURE nombre_procedimiento AS
    /* Section declarativa */
    BEGIN
        /* Sección Ejecutable */
    EXCEPTION
        /* Sección de Excepciones */
    END [nombre_procedimiento];
```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[11]



4.3 PROCEDIMIENTOS

Restricciones sobre los Parámetros Formales

- En una declaración de procedimiento, es **illegal** restringir un parámetro **CHAR** o **VARCHAR2** con determinada longitud, o un parámetro **NUMBER** con un valor de precisión y/o escala
- *La declaración del procedimiento es illegal:*

```
CREATE OR REPLACE PROCEDURE ParameterLength(
    p_Parameter1 IN OUT VARCHAR2(10),
    p_Parameter2 IN OUT NUMBER(3, 2) AS
BEGIN
    p_Parameter1 := 'abcdefghijklm';
    p_Parameter2 := 12.3;
END ParameterLength;
```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[12]



```
CREATE OR REPLACE PROCEDURE ParameterLength(
    p_Parameter1 IN OUT VARCHAR2,
    p_Parameter2 IN OUT NUMBER) AS
BEGIN
    p_Parameter1 := 'abcdefghijklm';
    p_Parameter2 := 12.3;
END ParameterLength;
```

4.3 PROCEDIMIENTOS

Restricciones sobre los Parametros Formales

- En una declaración de procedimiento, es **Illegal** restringir un parámetro **CHAR** o **VARCHAR2** con determinada longitud, o un parámetro **NUMBER** con un valor de precisión y/o escala
- La declaración del procedimiento es legal:*

```
CREATE OR REPLACE PROCEDURE ParameterLength (
    p_Parameter1 IN OUT VARCHAR2,
    p_Parameter2 IN OUT NUMBER) AS
BEGIN
    p_Parameter1 := 'abcdefghijklm';
    p_Parameter2 := 12.3;
END ParameterLength;
```

-- Invocación No.1

```
DECLARE
    v_Variable1 VARCHAR2(40);
    v_Variable2 NUMBER(3,4);
BEGIN
    ParameterLength(v_Variable1, v_Variable2);
END;
```

-- Invocación No.2

```
DECLARE
    v_Variable1 VARCHAR2(10);
    v_Variable2 NUMBER(3,4);
BEGIN
    ParameterLength(v_Variable1, v_Variable2);
END;
```

Cual seria la situación presentada aquí?

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

13



4.3 PROCEDIMIENTOS

Valores predeterminados de los parametros

- Al igual que con las declaraciones de las variables, los parámetros formales de un procedimiento o función pueden tener valores predeterminados.
- Si un parámetro tiene un valor predeterminado, no tiene por que ser pasado desde el entorno que realizo la llamada. Si es pasado, se usara el valor real en lugar del valor predeterminado.
- La sintaxis es:

nombre_parámetro [modo] tipo_parámetro { := | DEFAULT } valor_inicial

Ejemplo:

```
CREATE OR REPLACE PROCEDURE AddNewStudent (
    p_FirstName          students.first_name%TYPE ,
    p_LastName           students.last_name%TYPE ,
    p_Major              students.major%TYPE DEFAULT 'Economic') AS
BEGIN
    -- Inserta una nueva fila en la tabla students. Usa student_sequence
    -- para generar el nuevo valor ID del estudiante y asigna el valor 0
    -- a current_credits
    INSERT INTO students VALUES (student_sequence.nextval, p_FirstName, p_LastName, 0);
END AddNewStudent ;
```

-- Invocación No.1 – Notación Posicional

```
BEGIN
    AddNewStudent('Barbara', 'Blues');
END;
```

-- Invocación No.2 – Con Notación nominal

```
BEGIN
    AddNewStudent( p_Firstname => 'Barbara',
                    p_LastName => 'Blues');
END;
```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

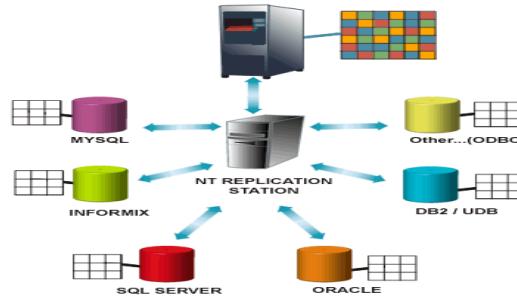
14



UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION

SISTEMAS DE BASE DE DATOS II ORACLE PROGRAMACION PL/SQL

Fundamentos del Lenguaje PL-SQL-ORACLE



Sistemas de Base de Datos II
 Ing. Henry Lezcano II Semestre del
 2020

[1]

CONTENIDO

Capítulo IV. Procedimientos

- ***Fundamentos de Lenguaje PL/SQL***
- ***Cursos***
- ***Procedimientos***



Sistemas de Base de Datos II
 Ing. Henry Lezcano II Semestre del
 2020

[2]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

INTRODUCCION

Para el desarrollador es necesario conocer en forma previa la sintaxis básica del lenguaje PL_SQL:

- Las reglas sintácticas son los bloques, componentes de cualquier lenguaje de programación.
- Se presentan los componentes de un bloque PL-SQL.
- Las declaraciones de variables y tipos de datos
- Las estructuras procedimentales básicas.
- Los cursor y subprogramas
- También se trata el tema de estilo de programación de PL-SQL y se presentan técnicas que ayudan a escribir código bien elegantes y de fácil comprensión.
- Las ordenes de PL-SQL son procedimentales, al igual que las ordenes de SQL. Incluyen declaraciones de variables, llamadas a procedimientos y estructuras de bucles. Con referencia a las ordenes de SQL estas nos permiten acceder a las base de datos.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

3



4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

El Bloque PL-SQL

- Los bloques representan la unidad básica de cualquier programa PL-SQL.
- Todos los PL-SQL están compuestos por bloques, que pueden estar situados de forma secuencial (uno detrás de otro) o pueden estar anidados (uno dentro de otro).
- Hay diferentes tipos de bloques:
 - Los bloques anónimos ('anonymous Blocks') se construyen, por regla general, de manera dinámica y se ejecuta una sola vez.
 - Los bloques nominados ('named Blocks') son bloques dinámicos con una etiqueta que le da al bloque un nombre. Se construyen, por regla general, de manera dinámica y se ejecuta una sola vez.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

4



4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

El Bloque PL-SQL

- Hay diferentes tipos de bloques:

- Los *subprogramas* son procedimientos, paquetes y funciones almacenados en la base de datos. Estos bloques no cambian, por regla general, después de su construcción y se ejecutan múltiples veces.
 - Los subprogramas se ejecutan explícitamente, mediante una llamada al procedimiento, paquetes o funciones.
- Los disparadores ('triggers') son bloques nominados que también se almacenan en la base de datos. Tampoco cambian, generalmente, después de su construcción se ejecutan múltiples veces.
 - Los disparadores se ejecutan de manera implícita cada vez que tiene lugar un suceso de disparo. El suceso de disparo es una orden del lenguaje DML que se ejecuta sobre una tabla de la base de datos, entre estas ordenes están INSERT, UPDATE y DELETE.



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[5]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

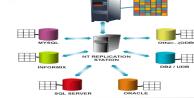
El Bloque PL-SQL

1. Ejemplo de un bloque anónimo:

```

DECLARE
  /* Declaración de las variables que se usan en este bloque */
  v_Num1 NUMBER := 1;
  v_Num2 NUMBER := 2;
  v_String1 VARCHAR(50) := 'Hello World!';
  v_String2 VARCHAR(50) := 'This message brought to you by PL/SQL!';
  v_OutputStr VARCHAR2(50);
BEGIN
  /* Primero inserta dos filas en temp_table, utilizando los valores de las variables */
  INSERT INTO temp_table (num_col, char_col) VALUES (v_Num1, v_String1);
  INSERT INTO temp_table (num_col, char_col) VALUES (v_Num2, v_String2);
  /* Ahora consulta temp_table para las dos filas que se acaban de insertar y las presenta en pantalla utilizando el paquete
  DBMS_OUTPUT */
  SELECT char_col INTO v_OutputStar
  From temp_table
  WHERE num_col = v_Num1;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
  /*SELECT char_col INTO v_OutputStar
  From temp_table
  WHERE num_col = v_Num2;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);*/
END;

```



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[6]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

El Bloque PL-SQL

2. Ejemplo de un bloque nominado, esta etiqueta se coloca antes de la palabra clave:

```
<<InsertIntoTemp>>
DECLARE
  /* Declaración de las variables que se usan en este bloque */
  v_Num1 NUMBER := 1;
  v_Num2 NUMBER := 2;
  v_String1 VARCHAR(50) := 'Hello World!';
  v_String2 VARCHAR(50) := 'This message brought to you by PL/SQL!';
  v_OutputStr VARCHAR2(50);
BEGIN
  /* Primero inserta dos filas en temp_table, utilizando los valores de las variable */
  INSERT INTO temp_table (num_col, char_col) VALUES (v_num1, v_String1);
  INSERT INTO temp_table (num_col, char_col) VALUES (v_num2, v_String2);
  /*Ahora consulta temp_table para las dos filas que se acaban de insertar y las presenta en pantalla utilizando
  el paquete DBMS_OUTPUT */
  SELECT char_col INTO v_OutputStar
  From temp_table
  WHERE num_col = v_num1;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
  SELECT char_col INTO v_OutputStar
  From temp_table
  WHERE num_col = v_num2;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
END InsertIntoTemp;
```



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[7]

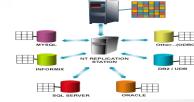
4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

El Bloque PL-SQL

3. Podemos transformar un bloque en un procedimiento almacenado reemplazando la palabra clave **DECLARE** con las palabras claves **CREATE OR REPLACE PROCEDURE**

```
CREATE OR REPLACE PROCEDURE InsertIntoTemp AS
  /* Declaración de las variables que se usan en este bloque */
  v_Num1 NUMBER := 1;
  v_Num2 NUMBER := 2;
  v_String1 VARCHAR(50) := 'Hello World!';
  v_String2 VARCHAR(50) := 'This message brought to you by PL/SQL!';
  v_OutputStr VARCHAR2(50);
BEGIN
  /* Primero inserta dos filas en temp_table, utilizando los valores de las variable */
  INSERT INTO temp_table (num_col, char_col) VALUES (v_num1, v_String1);
  INSERT INTO temp_table (num_col, char_col) VALUES (v_num2, v_String2);
  /*Ahora consulta temp_table para las dos filas que se acaban de insertar y las presenta en pantalla utilizando
  el paquete DBMS_OUTPUT */
  SELECT char_col INTO v_OutputStar
  WHERE num_col = v_num1;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
  SELECT char_col INTO v_OutputStar
  WHERE num_col = v_num2;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
END ;
```



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[8]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

El Bloque PL-SQL

4. Podemos construir un disparador sobre la tabla `temp_table` para que solo introduzca valores positivos en `num_col`. Este disparador se activara cada vez que se inserte una nueva columna en `temp_table` o actualice una fila ya existente

```
CREATE OR REPLACE TRIGGER OnlyPositive
  BEFORE INSERT OR UPDATE OF num_col
  ON temp_table
  FOR EACH ROW
BEGIN
  IF :new.num_col < 0 THEN
    RAISE_APPLICATION_ERROR (-2000, 'Please insert a positive value');
  END IF;
END OnlyPositive ;
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[9]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURA BASICA DE UN BLOQUE

Todos los bloques tienen tres secciones diferenciadas:

```
DECLARE
  /*Sección declarativa */
BEGIN
  /*Sección ejecutable */
EXCEPTION
  /* Sección Excepciones */
END;
```

- **La sección declarativa:** donde se localizan todas la variables, cursos y tipos usados por los bloques. También podemos declarar en esta sección las funciones y procedimientos locales, que solo están disponibles para este bloque.
- **La sección ejecutable:** donde se lleva a cabo el trabajo del bloque. En esta sección pueden aparecer tantos ordenes de SQL como ordenes procedimentales.
- **La sección de excepciones:** aquí se lleva a cabo el tratamiento de errores. El código incluido que se incluya en esta sección no se ejecuta a menos que ocurra un error.
- Las palabras claves del bloque `DECLARE`, `BEGIN`, `EXCEPTION` y `END` delimitan cada una de las secciones. El punto y coma también es obligatorio, forma parte de la sintaxis del bloque.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[10]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURA BASICA DE UN BLOQUE

Ejemplo de un bloque anónimo con todas la sección:

```

DECLARE
    /* Inicio de la sección declarativa */
    v_StudentID NUMBER(5) := 10000;    -- Variable numérica inicializada con 10,000
    v_FirstName VARCHAR2;           -- Cadena de caracteres de longitud variable con longitud máxima de 20
BEGIN
    /* Inicio de la sección de ejecutable */
    -- Recupera el nombre del estudiante con ID igual a 10,000
    SELECT first_name INTO v_FirstName
    FROM student
    WHERE id = v_StudentID;
EXCEPTION
    /* Inicio de la sección de excepciones */
    WHEN NO_DATA_FOUND THEN
        -- Manejo de la codificación de error
        INSERT INTO log_table (info)
        VALUES ('Student 10,000 does not exists!');
END;

```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[11]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Cualquier programa PL/SQL esta compuesto por unidades léxicas: los bloques que son los componentes del lenguaje. Esencialmente, **una unidad léxica es una secuencia de caracteres**, donde los caracteres pertenecen a un conjunto de caracteres permitidos en el lenguaje PL/SQL.

Las letras mayúsculas y minúsculas	A-Z y a-z
Los dígitos	0 - 9
Espacios en blanco	Tabuladores, caracteres de espaciados y retornos de carro
Símbolos matemáticos	+ - * / < > =
Símbolos de puntuación	() { } [] ¿ i ~ ; : . ' " @ # % \$ ^ & _

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[12]

Cualquier elemento de estos conjuntos puede ser usado como parte de un programa PL/SQL. No se diferencia entre la mayúscula y minúscula, excepto en el interior de una cadena delimitada por comillas.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LÉXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

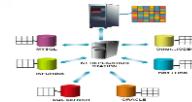
IDENTIFICADORES

- Son usados para dar nombre a los objetos PL/SQL, como variables, cursores, tipos, programas.
- Constan de una letra, seguida por una secuencia opcional de caracteres, que pueden incluir letras, números, signos de dólar(\$), caracteres de subrayado y símbolos de almohadilla(#). Los demás caracteres no son permitidos.
- La longitud máxima de un identificador es de 30 caracteres, y todos los caracteres son significativos.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[13]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LÉXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

IDENTIFICADORES LEGALES E ILEGALES

Identificadores Legales	Identificadores Ilegales
x	x+y
v_StudentID	_temp_
pempvari	First Name
v1	Este_es_un_identificador_muy_largo
v2	1_variable
social_security_#	

No hay diferencia entre mayúsculas y minúsculas por lo que estos identificadores son equivalentes

Room_Description
Room_description
ROOM_DESCRIPTION
rOOm_DescriPTION

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[14]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

DELIMITADORES

Los delimitadores son símbolos formados por uno o mas caracteres que tienen un significado especial para PL/SQL. Son usadas para separar unos identificadores de otros.

Símbolo	Descripción	Símbolo	Descripción
+	Operador de Suma	-	Operador de resta
*	Operador de Multiplicación	/	Operador de división
=	Operador de Igualdad	<	Operador menor que
>	Operador Mayor que	(Delimitador inicial de expresión
)	Delimitador Final de Expresión	;	Terminador de orden
.	Selector de Componente	,	Separador de elemento
'	Delimitador de Cadena de Caracteres	@	Indicador de enlace a base de datos
:	Indicador de variable de asignacion	''	Delimitador de cadena entrecomillada
<>	Operador distinto de	**	Operador de Exponenciación

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

15

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

DELIMITADORES-Continuación

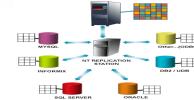
Los delimitadores son símbolos formados por uno o mas caracteres que tienen un significado especial para PL/SQL. Son usadas para separar unos identificadores de otros.

Símbolo	Descripción	Símbolo	Descripción
~=	Operador distinto de , !=	!=	Operador distinto que , <>
<=	Operador menor igual que	^=	Operador distinto que , ~=
:=	Operador de asignación	>=	Operador mayor igual a
..	Operador de rango	=>	Operador de asociación
<<	Delimitador de comienzo de etiqueta		Operador de concatenación
--	Indicador de comentario una sola línea	>>	Delimitador de fin de etiqueta
/	Indicador de cierre de comentario multilíneas	/	Indicador de cierre de comentario multilíneas
<tab>	Carácter de tabulación	<space>	Espacio
%	Indicador de atributo	<cr>	Retorno de carro

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

16

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serían **-23.456** y **NULL**

LITERALES DE CARÁCTER

- Las literales de carácter, también conocidos como literales de cadena, constan de uno o más caracteres delimitadas por comillas simples.
- Los literales de carácter pueden asignarse a variables de tipo CHAR o VARCHAR2, sin necesidad de hacer ningún tipo de conversión. Ejemplos válidos:
 - '12345'
 - 'Four score and seven years ago...'
 - '100%'
 - ''''
 - '' que es una cadena de longitud cero el literal de la cadena de longitud cero se considera idéntico a NULL.

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[17]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serían **-23.456** y **NULL**

LITERALES NUMERICAS

Un literal numérica representa un valor entero o real, y puede ser asignado a una variable de tipo NUMBER sin necesidad de efectuar conversión alguna.

Algunos ejemplos válidos son:

Literales Enteras	Literales reales
123	-17.1
-7	23.0
+12	3.
0	

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[18]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serían **-23.456** y **NULL**

LITERALES BOOLEANOS

- Solo hay tres posibles literales booleanos: **TRUE** (verdadero), **FALSE** (falso) y **NULL** (nulo).
- Estos valores solo pueden ser asignados a una variable booleana.
- Las literales representan la verdad o falsedad de una condición y se utilizan en las ordenes **IF** y **LOOP**

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[19]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serían **-23.456** y **NULL**

COMENTARIOS

- Los comentarios facilitan la lectura y ayudan a comprender mejor el código fuente de los programas
- Están clasificados como **comentarios monolíneas** y **comentarios multilíneas**

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[20]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

COMENTARIOS

```

DECLARE
  -- Declaración de las variables que se usan en este bloque COMENTARIO MONOLINEA
  v_Num1 NUMBER := 1;
  v_Num2 NUMBER := 2;
  v_String1 VARCHAR(50) := 'Hello World!';
  v_String2 VARCHAR(50) := 'This message brought to you by PL/SQL!';
  v_OutputStr VARCHAR2(50);
BEGIN
  -- Primero inserta dos filas en temp_table, utilizando los valores de las variable COMENTARIO MONOLINEA
  INSERT INTO temp_table (num_col, char_col) VALUES (v_num1, v_String1);
  INSERT INTO temp_table (num_col, char_col) VALUES (v_num2, v_String2);
  /*Ahora consulta temp_table para las dos filas que se acaban de insertar y las presenta en pantalla utilizando el paquete
  DBMS_OUTPUT */
  SELECT char_col INTO v_OutputStar
  WHERE num_col = v_num1;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
  SELECT char_col INTO v_OutputStar
  WHERE num_col = v_num2;
  DBMS_OUTPUT.PUT_LINE(v_OutputStar);
END;

```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[21]



4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

La comunicación con la base de datos tiene lugar mediante el uso de variables incluidas en los bloques PL/SQL.

- Las variables son espacios de memoria que pueden contener valores de datos
- A medida que se ejecuta el programa, el contenido de la variable puede cambiar y suele hacerlo.
- Se puede asignar información de la base de datos a las variables y también puede insertar el contenido de una variable en la base de datos.
- Las variables son declaradas en la sección declarativa de un bloque y cada una de ellas tiene un tipo específico, que describe el tipo de información que podemos almacenar en ella.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[22]



4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

SINTAXIS DE LAS DECLARACIONES

Las declaraciones de variables tienen lugar en la sección declarativa del bloque. Y la sintaxis general sería

```
nombre_variable tipo [CONSTANT][NOT NULL] [:= valor]
```

Algunos ejemplos de declaración Legal:

```
DECLARE
    v_Description      VARCHAR(50);
    v_NumberSeats     NUMBER := 45;
    v_Counter          BINARY_INTEGER := 0;
```

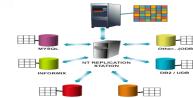
Ejemplo de Declaración Ilegal

```
V_tempVar      NUMBER NOT NULL;
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[23]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

SINTAXIS DE LAS DECLARACIONES

Ejemplo de Declaración Ilegal y corrigiendo el error:

```
DECLARE
    v_tempVar      NUMBER NOT NULL := 0;
```

Si se utiliza la restricción NOT NULL, se debe asignar un valor cuando se declara la variable

Si queremos declarar un constante:

```
DECLARE
    v_MinimumStudentID CONSTANT NUMBER(5) := 10000;
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[24]

Podemos usar la palabra clave DEFAULT:

```
v_NumberSeats      NUMBER DEFAULT 45;
v_Counter          BINARY_INTEGER DEFAULT 0;
v_FirstName        VARCHAR2 (20) DEFAULT 'Scott';
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

SINTAXIS DE LAS DECLARACIONES

Solo puede haber una declaración de variable por línea en la sección declarativa del bloque.

Declaración Ilegal:

```
DECLARE
  V_FirstName, v_LastName      VARCHAR2 (20);
```

Declaración Legal:

```
v_FirstName  VARCHAR2 (20);
v_LastName   VARCHAR2 (20);
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[25]

INICIALIZACION DE VARIABLES:

- EN PL/SQL se define el contenido de las variables inicializadas, al contenido de estas variables se le asigna el valor de NULL.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

TIPOS DE DATOS EN PL/SQL (hacer referencia al Capítulo Anterior)

- Existen cuatro categorías de tipos de datos: Escalares, Compuestos, Referencias, y LOB (Large Objects)
- Los tipos Escalares no tienen componentes, mientras que los tipos Compuestos si, por que las referencias son punteros a otros tipos.
- Los tipos de datos PL/SQL se definen un paquete llamado STANDARD, cuyos contenidos son accesibles desde cualquier bloque PL/SQL.
- El paquete STANDARD también define las funciones predefinidas SQL y de conversión disponibles en PL/SQL.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[26]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UTILIZACION DE %TYPE

Las variables PL/SQL se emplean para manipular datos almacenados en la base de datos. La variable debe tener el mismo tipo que la columna de la tabla.

Por ejemplo, columna `first_names` de la tabla `students` que es de tipo `VARCHAR2(20)`. Tomando como referencia esta información podemos declarar:

```
Declare
    v_FirstName      VARCHAR2(20);
```

Pero que ocurre si cambia la dimensión a mayor de la columna y se esta usando en un programa, habría cambiar variable que haga referencia a la columna.

Para corregir esto usamos en `%TYPE` para definir la variable:

```
Declare
    v_FirstName      students.first_names%TYPE;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UTILIZACION DE %TYPE

El siguiente ejemplo ilustra varias aplicación del atributo `%TYPE`

```
DECLARE
    v_RoomID      classes.room_id%TYPE;          -- devuelve NUMBER (5)
    v_RoomID2     v_RoomID %TYPE;                  -- devuelve NUMBER (5)
    v_TempVar     NUMBER(7,3) NOT NULL := 12.3;
    v_AnotherVar  v_TempVar %TYPE;                -- devuelve NUMBER (7,3)
```

Si se aplica `%TYPE` a una variable o a una columna que haya sido definida con la restricción `NOT NULL` como `classes.room_id` y `v_TempVar`, el tipo devuelto no tiene esta restricción.

La utilización de `%TYPE` constituye una buena práctica de programación, porque hace que los programas PL/SQL sean mas flexibles y capaces de adaptarse a las definiciones cambiantes de la base datos.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

CONVERSIONES ENTRE TIPOS DE DATOS

PL/SQL puede manejar conversiones entre tipos de datos escalares de las distintas familias. Para los tipos de una familia se puede realizar conversiones sin ninguna restricción, excepto las impuestas a las variables.

Por ejemplo un CHAR(10) no puede convertirse en un VARCHAR2(1), un NUMBER(3,2) no puede convertirse en un NUMBER(3) porque el cuando se producen violaciones a las restricciones el compilador PL/SQL no dará un error pero se puede producir errores en tiempo de ejecución dependiendo de los valores de las variables a convertir.

Los tipos compuesto no pueden convertirse entre si, porque son demasiados distintos. Pero se puede escribir un función que realice la conversión, basándose en tipo de dato que tenga en el programa.

Existen dos tipos de conversión: implícita y explícita

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[29]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

CONVERSIONES ENTRE TIPOS DE DATOS

Conversión explícita de tipos de datos

Las función de conversión de SQL también están disponibles en PL/SQL. Pueden ser empleadas cuando se requiera, para realizar conversiones explícitas entre variables de diferentes familia de tipos.

Función	Descripción	Familias que se Puede Convertir
TO_CHAR	Convierte su argumento en tipo VARCHAR2, dependiendo del especificador de formato opcional	Numéricos, Fechas
TO_DATE	Convierte su argumento en tipo DATE, dependiendo del especificador de formato opcional	Carácter
TO_NUMBER	Convierte su argumento en tipo NUMBER, dependiendo del especificador de formato opcional	Carácter
RAWTOHEX	Convierte un valor RAW en un representación hexadecimal de la cantidad en binario	Raw
HEXTORAW	Convierte una representación hexadecimal en el equivalente binario	Carácter(en representación hexadecimal)
CHARTOROWID	Convierte una representación de caracteres de un ROWID el formato interno binario	Carácter(en formato rawid d 18 caracteres)
ROWIDTOCHAR	Conviene una variable interna ROWID al formato externo de 18 caracteres	Rowid

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[30]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

CONVERSIONES ENTRE TIPOS DE DATOS

Conversión implícita de tipos de datos

PS/SQL realizará conversiones automáticas entre familias de tipos, siempre que sea posible.

Por ejemplo, el siguiente bloque extrae el número actual de crédito del estudiante 10002.

```
DECLARE
    v_CurrentCredits    VARCHAR2(5);
BEGIN
    SELECT current_credits
    INTO v_CurrentCredits
    FROM students
    WHERE id = 10002;
END;
```

En la base de datos, `current_credits` es un campo tipo `NUMBER(3)`, mientras que `v_CurrentCredits` es una variable `VARCHAR2(5)`. PLSQL convertirá automáticamente el dato numérico en una cadena de caracteres, y asignará esta a la variable de tipo carácter.

PL/SQL puede realizar conversiones entre

- Caracteres y números
- Caracteres y fechas

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Las expresiones y los operadores son el pegamento que permite unir las variables PS/SQL.

- *Los operadores definen como se asignan valores a la variables y como se manipulan dichos valores.*
- *Una expresión es una secuencia de valores y literales, separados por operadores.*
- *El valor de una expresión se determina a partir de los valores de las variables y literales que la componen y de la definición de los operadores.*

Asignación

Es el operador más básico. Su sintaxis es

`variable := expresión;`

Donde `variable` y `expresión` son variables y expresión de PL/SQL

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Asignación

Ejemplo de una orden de asignación

```

DECLARE
    v_String1    VARCHAR2(10);
    v_String2    VARCHAR2(15);
    v_Numeric    NUMBER;
BEGIN
    v_String1 := 'Hello';
    v_String2 := v_String1;
    v_Numeric := -12.4;
END;

```

En una orden dada solo puede haber una asignación. A diferencia de otros lenguajes. Esto es una asignación ilegal

```

DECLARE
    v_Val1  NUMBER;
    v_Val2  NUMBER;
    v_Val3  NUMBER;
BEGIN
    v_Val1 := v_Val2 := v_Val3 := 0;
END;

```

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[33]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones

Las expresiones PL/SQL son valores. Por si solo una expresión no es valida como orden independiente, sino que tiene ser parte de otra orden. Los operadores que componen una expresión determinan, junto con el tipo de los operando, cual es el tipo de la expresión.

Expresiones Numérica

Estas expresión dependen de la procedencia de los operadores

Estas dos expresiones no son equivalentes $3 + 5 * 7 = (3 + 5) * 7$ ¿ Porque?

Expresiones de Caracteres

El único operador de caracteres es la concatenación (||).

Ejemplo de una expresión: 'Hello' || 'World' || '!' el resultado 'Hello World!'

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[34]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

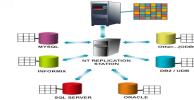
Expresiones de Caracteres

Si todos los operando de un expresión de concatenación son tipo CHAR, entonces la expresión también lo es. Si uno de los operando es tipo VARCHAR2, entonces la expresión también lo es. Las literales de cadena se consideran de tipo CHAR de forma que la expresión resultante también lo es.

```

DECLARE
    v_TempVar  VARCHAR2(10) := 'PL';
    v_Result    VARCHAR2(10) ;
BEGIN
    v_Result := v_TempVar || '/SQL';
END;

```



Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[35]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

Todas las expresiones de control PL/SQL (excepto GOTO) incluyen expresiones booleanas, también denominadas condiciones. Una expresión booleana es una expresión que tiene como resultado un valor booleano (TRUE, FALSE o NULL)

Ejemplos de expresiones booleanas:

X > Y
NULL
(4 > 5) OR (-1 != Z)



Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[36]

Hay tres operadores (AND , OR y NOT) que admiten argumentos booleanos y devuelven valores booleanos. Su comportamiento se describen en la tabla de la verdad.

Estos operadores implementan la lógica trivaluada estándar.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

Estos operadores implementan la lógica trivaluada estándar.

- Por ejemplo **AND** devuelve **TRUE** si sus dos operando toman el valor de **TRUE**. **Y OR** devuelve **FALSE** si su dos operando toman el valor de **FALSE**
- Los valores **NULL** añaden complejidad a las expresiones booleanas 'NULL significa valor desconocido o no definido'. La expresion
 - **TRUE AND NULL** da como resultado **NULL** porque no sabemos si el segundo operando tiene el valor de **TRUE** o no.
- El operador **IS NULL** devuelve **TRUE** solo si su operador es **NULL**.



Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[37]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

- El Operador **LIKE** se usa para comparaciones con patrones en cadena de caracteres. El carácter de subrayado (_) se corresponde con exactamente un carácter, mientras que el carácter de porcentaje(%) se corresponde con cero o mas caracteres.
- Ejemplo de expresiones que devuelven valor **TRUE**
 - 'Scott' LIKE 'Sc%tt'
 - 'Scott' LIKE 'SC_tt'
 - 'Scott' LIKE '%'
- El operador **BETWEEN** combina **<= y >=** en una única expresión. La siguiente expresion, por ejemplo :
 - 100 BETWEEN 110 AND 120 Resultado es **FALSE**
 - 100 BETWEEN 90 AND 110 resultado es **TRUE**



Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[38]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

- El operador IN devuelve TRUE si su primer operando esta contenido en un conjunto identificado por un segundo operando. La siguiente expresion, por ejemplo :

`'Scott' IN ('Mike', 'Pamela', 'Fred')` Resultado es FALSE

Operadores Validos

Operador	Definición
=	Igual a
!=	Distinto de
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

TABLA DE LA VERDAD

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL
AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	NULL	NULL
OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

39

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- El PL/SQL tiene diversas estructuras de control que permiten controlas el comportamiento de los bloques a medida que estos se ejecutan. Pueden contener ordenes condicionales y los bucles. Son estas estructuras que combinadas con la variables dotan a PL/SQL de su poder y flexibilidad.

- IF-THEN-ELSE

La sintaxis para esta orden seria

```
IF expresion_booleana1 THEN
    secuencia_de_ordenes1;
[ELSIF expresion_booleana2 THEN
    Secuencia_de_ordenes2;]
[ELSE
    secuencia_de_ordenes3;]
END IF;
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

40

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

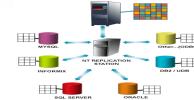
- Ejemplo de la orden IF-THEN-ELSE

```

DECLARE
    v_NumberSeats rooms.number_seats%TYPE;
    v_Comment VARCHAR(35);
BEGIN
    /* Extrae el numero de asiento de la habitación, cuyo identificador es 99999 y almacena el
    resultado en v_NumberSeats. */
    SELECT number_seats
    INTO v_NumberSeats
    FROM rooms
    WHERE room_id = 99999;
    IF v_NumberSeats < 50 THEN
        v_comment := 'Fairly small';
    ELSIF v_NumberSeats < 100 THEN
        v_comment := 'A little bigger';
    ELSE
        v_comment := 'Lots of room';
    END IF;
END;

```

El comportamiento del bloque resulta bastante evidente.



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[41]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- En el ejemplo anterior cada secuencia de orden consta de una sola orden procedural, sin embargo pueden haber tantas orden (procedimientos o SQL) como sea necesario.

```

DECLARE
    v_NumberSeats rooms.number_seats%TYPE;
    v_Comment VARCHAR(35);
BEGIN
    /* Extrae el numero de asiento de la habitación, cuyo identificador es 99999 y almacena el resultado en
    v_NumberSeats. */
    SELECT number_seats
    INTO v_NumberSeats
    FROM rooms
    WHERE room_id = 99999;
    IF v_NumberSeats < 50 THEN
        v_comment := 'Fairly small';
        INSERT INTO temp_table (char_col) VALUES ('Nice and cozy');
    ELSIF v_NumberSeats < 100 THEN
        v_comment := 'A little bigger';
        INSERT INTO temp_table (char_col) VALUES ('Some breathing room');
    ELSE
        v_comment := 'Lots of room';
    END IF;
END;

```



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[42]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Condiciones Nulas

Una secuencia de ordenes dentro de un orden IF-THEN-ELSE se ejecuta solo si su condición asociada es verdadera(toma el valor TRUE). Si la condición toma el valor FALSE o NULL, la secuencia de ordenes no se ejecuta. Ejemplos:

```
/* Bloque 1 */
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 < v_Number2 THEN
    v_Result := 'Yes';
  ELSE
    v_Result := 'No';
  END IF;
END;
```

```
/* Bloque 2 */
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 >= v_Number2 THEN
    v_Result := 'No';
  ELSE
    v_Result := 'Yes';
  END IF;
END;
```

Si `v_Number1 = 3` y `v_Number2 = 7`. Se comportan ambos bloques de la misma forma?

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

43

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Condiciones Nulas

A los bloques mostrados se le puede añadir una comprobación adicional de nulidad:

```
/* Bloque 1 */
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 IS NULL OR
  v_Number2 IS NULL THEN
    v_Result := 'Unknown';
  ELSIF v_Number1 < v_Number2 THEN
    v_Result := 'Yes';
  ELSE
    v_Result := 'No';
  END IF;
END;
```

```
/* Bloque 2 */
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 IS NULL OR
  v_Number2 IS NULL THEN
    v_Result := 'Unknown';
  ELSIF v_Number1 >= v_Number2 THEN
    v_Result := 'No';
  ELSE
    v_Result := 'Yes';
  END IF;
END;
```

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

44

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES

PL/SQL permite ejecutar ordenes en forma repetida, utilizando los bucles. Existen 4 tipos de bucles: *bucles simples*, *bucles WHILE*, *bucles FOR numéricos*, y *bucles FOR de cursor*.

Bucles Simples

Son el tipo de bucle más básico. Su sintaxis es:

```
LOOP
    secuencia_de_órdenes;
END LOOP;
```

La *secuencia_de_órdenes* se ejecutara indefinidamente, puesto que el bucle no tiene ninguna condición de parada. Si embargo, podemos añadir una condición mediante la orden **EXIT** cuya sintaxis es:

```
EXIT [WHEN condición];
```

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[45]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCELES SIMPLE-LOOP

El siguiente bloque, del ejemplo inserta 50 filas en la tabla **temp_table**.

Bloque1

```
DECLARE
    v_counter BINARY_NUMBER := 1;
BEGIN
LOOP
    -- Insertar una fila en temp__table con el valor actual del
    -- contador del bucle.
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Index');
    v_counter := v_counter + 1;
    -- Condición de salida – Cuando el contador
    -- del bucle sea > 50 se saldrá del bucle
    IF v_counter > 50 THEN
        EXIT;
    END IF;
END LOOP;
END;
```

Bloque2-Se comporta igual al anterior

```
DECLARE
    v_counter BINARY_NUMBER := 1;
BEGIN
LOOP
    -- Insertar una fila en temp__table con el valor actual del
    -- contador del bucle.
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Index');
    v_counter := v_counter + 1;
    -- Condición de salida – Cuando el contador
    -- del bucle sea > 50 se saldrá del bucle
    EXIT WHEN v_counter > 50;
END LOOP;
END;
```

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[46]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -WHILE

La sintaxis de un bucle WHILE es.

```
WHILE condición LOOP
    secuencia_de_órdenes;
END LOOP;
```

La condición se evalúa antes de cada iteración del bucle. Si es verdadera se ejecuta la secuencia_de_órdenes. Si la condición es falsa o nula, el bucle termina y el control se transfiere a lo que está a continuación de la orden END LOOP. Ejemplo:

```
DECLARE
    v_counter BINARY_NUMBER := 1;
BEGIN
    -- comprueba el contador del bucle antes de cada iteración
    -- para asegurarse que todavía es menor a 50.
    WHILE v_counter <= 50 LOOP
        INSERT INTO temp_table
        VALUES (v_counter, 'Loop Index');
        v_counter := v_counter + 1;
    END LOOP;
END;
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[47]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -WHILE

Se puede usar las ordenes EXIT o EXIT WHEN dentro un bucle WHILE para salir de forma prematura. Tenga presente que si la condición del bucle no toma el valor TRUE la primera vez que se comprueba, el bucle no llega a ejecutarse.

Ejemplo :

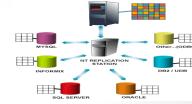
```
DECLARE
    v_counter BINARY_NUMBER;
BEGIN
    -- Esta condición se evaluará como NULL, ya que
    v_counter
    -- se inicializa con el valor predeterminado NULL.
    WHILE v_counter <= 50 LOOP
        INSERT INTO temp_table
        VALUES (v_counter, 'Loop Index');
        v_counter := v_counter + 1;
    END LOOP;
END;
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

[48]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA



ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR numéricos

El numero de interacciones de los **bucles simples** y de los bucles **WHILE** no se conoce de antemano, sino que depende de la condición del bucle. Los bucles **FOR numéricos** por lo contrario, tienen el numero de iteración definido. Su sintaxis es:

```
FOR contador_bucle IN [REVERSE] limite_inferior... limite_superior LOOP
    secuencia_de_órdenes;
END LOOP;
```

Donde *contador_bucle* es la variable de indice declarada de modo explícito, *limite_inferior* y *limite_superior* especifican el numero de iteraciones y *secuencia_de_órdenes* es el contenido del bucle.

Los límites del bucle solo se evalúan una sola vez. Estos valores determinan el numero total de interacciones, en las que el *contador_bucle* varia entre los valores del *limite_inferior* y *limite_superior* incrementándose en una unidad a la vez, hasta que el bucle se completa.

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[49]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA



ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR numericos

En el ejemplo del bucle utilizando el bucle FOR:

```
DECLARE
    v_counter BINARY_NUMBER;
BEGIN
    -- Esta condicion se evaluara como NULL, ya que
    v_Counter
    -- se inicializa con el valor predeterminado NULL.
    WHILE v_counter <= 50 LOOP
        INSERT INTO temp_table
        VALUES (v_counter, 'Loop Index');
        v_counter := v_counter + 1;
    END LOOP;
END;
```

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[50]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR *numericos*

En el ejemplo del bucle utilizando el bucle FOR:

```

BEGIN
  FOR v_counter IN 1..50 LOOP
    INSERT INTO temp_table
      VALUES (v_counter, 'Loop Index');
  END LOOP;
END;

```

Reglas de ámbitos. El índice de un bucle **FOR** se declara implícitamente como un **BINARY_INTEGER**, no siendo necesario declararlo antes del bucle. Si lo declara, el índice del bucle ocultara la declaración exterior al bucle, de la misma forma que en una declaración de variable en un bloque puede ocultar una declaración en un bloque externo ese.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[51]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR *numericos*

En el ejemplo del bucle utilizando el bucle FOR:

```

DECLARE
  v_counter BINARY_NUMBER := 7;
BEGIN
  -- Inserta el valor 7 en la tabla temp_table.
  INSERT INTO temp_table (num_col) VALUES (v_counter);
  -- Este bucle declara de nuevo v_counter como BINARY_INTEGER, lo
  -- que anula la declaración de NUMBER de v_counter
  FOR v_counter IN 20..30 LOOP
    --Dentro del bucle, el rango de v_counter es de 20 a 30
    INSERT INTO temp_table (num_col) VALUES (v_counter);
  END LOOP;
  -- Inserta otro 7 en la tabla temp_table
  INSERT INTO temp_table (num_col) VALUES (v_counter);
END;

```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[52]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR *numericos*

UTILIZACION de REVERSE Si se incluye la palabra clave REVERSE en el bucle FOR, el indice del bucle realizará las iteraciones desde el límite superior al límite inferior. Su sintaxis es la siguiente, con el ejemplo correspondiente

```
BEGIN
  FOR v_counter IN REVERSE 10..50 LOOP
    -- v_counter se iniciara en 50 y se excrementara en una
    -- unidad cada vez que se ejecute el bucle
    NULL;
  END LOOP;
END;
```

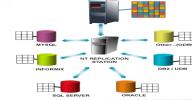
RANGO DE LOS BUCLES los límites inferiores o superiores no tienen porque ser literales numéricas, sino que puede ser cualquier expresión que puede ser convertida en un valor numérico. Ejemplo

```
DECLARE
  v_lowValue NUMBER := 10;
  v_highValue NUMBER := 40;
BEGIN
  FOR v_counter IN v_lowValue .. v_highValue LOOP
    INSERT INTO temp_table
    VALUES (v_counter, 'Dynamically specified loop range');
  END LOOP;
END;
```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

53

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- ORDENES GOTO y etiquetas

PL/SQL también disponen de una orden de salto GOTO. La sintaxis es la siguiente

```
GOTO etiqueta
```

Donde **etiqueta** es una etiqueta definida en el bloque PL/SQL. Las etiquetas se encierran entre corchetes angulares dobles. Cuando se evalúa una orden **GOTO**, el control pasa inmediatamente a la identificadas por la etiqueta. Ejemplo

```
DECLARE
  v_counter BINARY_INTEGER := 1;
BEGIN
  LOOP
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Count');
    v_counter := v_counter + 1;
    IF v_counter > 50 THEN
      GOTO 1_EndOfLoop;
    END IF;
  END LOOP;

  <<1_EndOfLoop>>
  INSERT INTO temp_table (char_col) VALUES ('Done!');
END;
```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

54

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Restriccion en el uso de la orden GOTO

PL/SQL impone una serie de restricciones para el uso de la orden GOTO. Es ilegal realizar un salto al interior de un bloque interno, de un bucle o de una orden IF, e incluso es prohibitivo hacerlo dentro del bloque de las excepciones. Ejemplo:

```

BEGIN
  GOTO 1_InnerBlock; -- Illegal, no se puede saltar a un bloque interno
  BEGIN
    .....
    << 1_InnerBlock>>
  END;
  GOTO 1_InsideIf; -- Illegal, no se puede saltar al interior de una orden IF
  IF X > 3 THEN
    << 1_InsideIf>>
    INSERT INTO.....;
  END IF;
END;

```

Si este tipo de salto fuera legal, entonces las ordenes situadas dentro de la orden IF podría ser ejecutada incluso si la condición de la orden IF no fuera cierta.

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

55

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Restriccion en el uso de la orden GOTO

Es ilegal saltar desde la rutina de manejo de excepciones de vuelta al bloque actual

```

DECLARE
  v_Room    room%ROWTYPE;
BEGIN
  -- Recupera una fila de la tabla room
  SELECT *
  INTO v_Room
  WHERE rowid = 1;
  <<1_Insert>>
  INSERT INTO temp_table (char_col)
  VALUES ('Found a row');
EXCEPTION
  WHERE NO_DATA_FOUND THEN
    GOTO 1_insert; -- Illegal no se puede saltar al interior del bloque
                     -- actual
END;

```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

56

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Etiqueta de los Bucles

También los propios bucles podrían ser etiquetado. Si lo hacemos así, puede emplearse la etiqueta en la orden EXIT para indicar el bucle del que hay que salir.

```

BEGIN
  <<1_Outer>>
  FOR v_OuterIndex IN 1..50 LOOP
    ... <<1_Inner>>
    FOR v_InnerIndex IN 2.. 20 LOOP
      ...
      IF v_OuterIndex > 40 THEN
        EXIT 1_Outer; -- Salida de ambos bucles
      END IF;
    END LOOP 1_Inner;
  END LOOP 1_Outer;
END;
  
```

Si se etiquetan los bucles, el nombre de las etiquetas puede ser incluido opcionalmente después de la orden END LOOP.

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

57

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Etiqueta de los Bucles

También los propios bucles podrían ser etiquetado. Si lo hacemos así, puede emplearse la etiqueta en la orden EXIT para indicar el bucle del que hay que salir.

```

BEGIN
  <<1_Outer>>
  FOR v_OuterIndex IN 1..50 LOOP
    ... <<1_Inner>>
    FOR v_InnerIndex IN 2.. 20 LOOP
      ...
      IF v_OuterIndex > 40 THEN
        EXIT 1_Outer; -- Salida de ambos bucles
      END IF;
    END LOOP 1_Inner;
  END LOOP 1_Outer;
END;
  
```

Si se etiquetan los bucles, el nombre de las etiquetas puede ser incluido opcionalmente después de la orden END LOOP.

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

58

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- La Orden NULL

En algunos casos explícitamente se quiere indicar que no se realice ninguna acción. Esto se puede realizar mediante la orden NULL, que es un orden que no tiene efecto alguno.

```

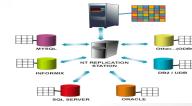
DECLARE
    v_TempVar NUMBER := 7;

BEGIN
    IF v_TempVar < 5 THEN
        INSERT INTO temp_table (char_col)
        VALUES ('Too Small');
    ELSIF v_TempVar < 10 THEN
        INSERT INTO temp_table (char_col)
        VALUES ('Just Right');
    ELSE
        NULL; -- No hace nada
    END IF;
END;

```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

59



4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

- No existen reglas absolutas referente al estilo de escritura de un programa. El estilo de programación incluye conceptos como los *nombres de las variables, utilización de las letras mayúsculas, espacios en blanco, y el uso de los comentarios*.
- Estos no son aspectos que afecten necesariamente a la ejecución de un programa; si reescribimos un programa con estilo distinto, el programa continuara haciendo lo mismo.
- Un programa escrito con un buen estilo será mucho mas fácil de entender y de mantener que un programa con un estilo pobre.
- Un buen estilo de programación que se tardara menos en entender lo que hace el programa, cuando se le ve por primera vez. Asimismo, ayudara al propio desarrollador a entender lo que el programa hace, tanto a medida que lo escribe como cuando lo vuelve a ver un mes mas tarde.

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

60



4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Observe los dos bloques siguientes. Cual de ellos es mas fácil de entender?

```

declare
  x number;
  y number;
begin if x < 10 then y := 7; else y := 13; end if; end;

DECLARE
  v_Test NUMBER;    -- Variable que se examinará
  v_Result NUMBER; -- Variable para almacenar resultado
BEGIN
  -- Examina v_Test y asigna 7 a v_Result si v_Test < 10
  IF v_Test < 10 THEN
    v_Result := 7;
  ELSE
    v_Result := 3;
  END IF;
END;

```

Ambos programas realizan la misma función. Sin embargo, el flujo del programa resulta mas fácil de comprender en el segundo caso.



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

61

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Comentarios

Se usan como mecanismo para informar al lector sobre cual es el propósito y como funciona el programa. Se pueden incluyen comentarios en:

- Al inicio de cada bloque y/o procedimientos. Estos comentarios deberían explicar que es o que hace el bloque o el procedimiento. De forma especial para los procedimientos, es importante enumerar las variables o parámetros que el procedimiento leerá (entrada) y escribirá (salida). También es una idea importante enumerar las tablas de la base de datos a las que accede.
- Junta a cada declaración de variable, para describir el uso que se le va a dar. A menudo es suficiente con usar comentarios de una sola línea.

```
v_SSN CHAR(11); -- Numero de seguro social
```

- Antes de cada una de las secciones principales del bloque. No es necesario colocar comentarios a cada orden, pero un comentario que explique el propósito del siguiente conjunto de ordenes resultara útil.



Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

62

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Comentarios continuacion...

Se usan como mecanismo para informar al lector sobre cual es el propósito y como funciona el programa. Se pueden incluyen comentarios en:

- En los algoritmos utilizados pueden ser obvio a partir del propio código así que es mejor describir el propósito del algoritmo y para que usara los resultado, en lugar de los detalles del método.
- Los comentarios deben ser significativos y no volver a expresar lo que el propio código PL/SQL ya expresa. Ejemplo

```
DECLARE
    v_Temp  NUMBER := 0; --Asigna 0 a v_Temp
```

- Este otro comentario seria mejor porque nos dice cual es el propósito de la variable **v_Temp**:

```
DECLARE
    v_Temp  NUMBER := 0; --Variable temporal usada en el bucle principal
```

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

63

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de los nombres de variables

La clave de los nombres de variables consiste en hacerlos descriptivos. La declaracion:

```
x number;
```

No nos dice nada sobre el propósito de x. Sin embargo,

```
v_StudentID  NUMBER(5);
```

Nos dice que esta variable será probablemente usada para almacenar el numero de identificación de un estudiante, incluso aunque no pongamos un comentario explicativo al lado de la declaración. Siempre debemos tomar en cuenta el tamaño del identificador PL/SQL.

El nombre de una variable también nos puede informar acerca de su uso. Se puede usar para esto un código de una letra, separado por un carácter de subrayado del resto de las variables Ejemplos:

v_NombreVariable	Variable del programa
e_NombreExcepcion	Excepción definida por el usuario
t_NombreTipo	Tipo definido por el usuario
p_NombreParametro	Parámetro de un procedimiento o función
c_ValorConstante	Variable restringida mediante la clausula CONSTANT

Sistemas de Base de Datos II Por:
Ing. Henry Lezcano II Semestre del
2020

64

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA



GUIA DE ESTILO DE PL/SQL

Estilo de uso de las mayúsculas

PL/SQL no diferencia entre mayúsculas y minúsculas. Si embargo, el uso apropiado de las mayúsculas y minúsculas incrementa sobremanera la legibilidad de una programa. Algunas reglas de uso:

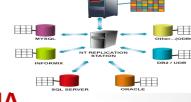
- Las palabras reservadas se escriben en mayúsculas (por ejemplo, BEGIN, DECLARE o ELSEIF)
- Las funciones predefinidas se escriben en mayúsculas (SUBSTR, COUNT, TO_CHAR)
- Los tipos predefinidos se escriben en mayúsculas (NUMBER(7,2), BOOLEAN, DATE)
- Las palabras claves de SQL se escriben en mayúsculas (SELECT, INTO, UP, DATE, WHERE)
- Los objetos de la base de datos se escriben en minúsculas (log_table, classes, students)
- Los nombres de variables se escriben con una mezcla de mayúsculas y minúsculas, poniendo en mayúsculas la primera letra de cada palabra componente (v_HireDate, e_TooManyStudents, t_StudentsRecordType)

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[65]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA



GUIA DE ESTILO DE PL/SQL

Estilo de Indentacion

Una de las cosas que menos cuesta es usar los *espacios en blanco* (retorno de carro, caracteres de espacio y tabulaciones) que pueden tener un gran efecto sobre la legibilidad del programa.

Compare las dos estructuras IF-THEN-ELSE indentadas siguientes:

```
IF x < y THEN IF z IS NULL THEN x:= 3; ELSE x := 2 END IF;
ELSE x := 4; END IF;

IF x < y THEN
  IF z IS NULL THEN
    x := 3;
  ELSE
    x := 2;
  END IF;
ELSE
  x := 4;
END IF;
```

Sistemas de Base de Datos II
Por:
Ing. Henry Lezcano II Semestre del
2020

[66]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Indentación

Una de las cosas que menos cuesta es usar los *espacios en blanco* (retorno de carro, caracteres de espacio y tabulaciones) que pueden tener un gran efecto sobre la legibilidad del programa.

Compare las dos estructuras IF-THEN-ELSE indentadas siguientes:

```
IF x < y THEN IF z IS NULL THEN x:= 3; ELSE x := 2 END IF;
ELSE x := 4; END IF;
```

```
IF x < y THEN
  IF z IS NULL THEN
    x := 3;
  ELSE
    x := 2;
  END IF;
ELSE
  x := 4;
END IF;
```

Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[67]

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Indentación continuación

Generalmente para lograr este estilo se suele indentar cada línea dentro de un bloque con dos espacios. Se acostumbra a indentar el contenido de los bloques con respecto a la palabra clave **DECLARE.....END**, y también los bucles y las ordenes **IF-THEN-ELSE**. También dentro de las ordenes SQL que ocupan mas de una línea por ejemplo:

```
SELECT id, first_name, last_name
  INTO v_StudentID, v_FirstName, v_LastName
  FROM students
 WHERE id = 10002;
```

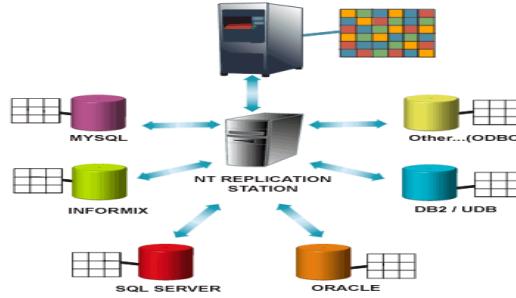
Sistemas de Base de Datos II
Ing. Henry Lezcano II Semestre del
2020

[68]

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.

SISTEMAS DE BASE DE DATOS II ORACLE PROGRAMACION PL/SQL

Funciones-Disparadores-Vistas-PL/SQL ORACLE



Por: Ing. Henry Lezcano Sistemas
 de Base de Datos II Semestre del
 2020

[1]

CONTENIDO

Capítulo V. Funciones disparadores y Vistas

- *Funciones*
- *Disparadores*
- *Vistas*

Por: Ing. Henry Lezcano Sistemas
 de Base de Datos II Semestre del
 2020

[2]

5.1 Funciones

Creacion de Funciones



- Una función es similar a un procedimiento. Ambos aceptan argumentos y estos pueden ser de cualquiera de los modos presentados.
- Ambos son formas diferentes de bloques PL/SQL, con sus secciones declarativas, ejecutable y de excepciones.
- Ambos pueden ser almacenados en la Base de Datos o ser declarados dentro de un bloque(procedimientos y funciones que no son almacenados en la base de datos)
- Sin embargo, una llamada a un procedimiento es una orden PL/SQL en si misma, mientras que una llamada a una función se realiza como parte de una expresión.
- Una llamada a una función es un valor.

Para el ejemplo la siguiente función devuelve un valor TRUE si la clase especifica tiene ocupación mayor del 90% y FALSE en caso contrario.

Por Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[3]

5.1 Funciones

Creacion de Funciones

Para el ejemplo la siguiente función devuelve un valor TRUE si la clase especifica tiene ocupación mayor del 90% y FALSE en caso contrario.

```
CREATE OR REPLACE FUNCTION AlmostFull (
    p_Department    classes.department%TYPE,
    p_Course        classes.course%TYPE)
    RETURN BOOLEAN IS
    v_CurrentStudents      NUMBER;
    v_MaxStudents          NUMBER;
    v_ReturnValue          BOOLEAN;
    v_FullPercent          CONSTANT NUMBER := 90;
BEGIN
    -- Obtiene el valor actual y máximo de estudiantes para el cuyo solicitado
    SELECT current_students, max_students
    INTO v_CurrentStudents, v_MaxStudents
    FROM classes
    WHERE department = p_Department
    AND course = p_Course;
    -- Si la clase está mas llena que el porcentaje dado por v_FullPercent , devuelve TRUE. En caso Contrario,
    FALSE;
    IF (v_CurrentStudents / v_MaxStudent * 100) > v_FullPercent THEN
        v_ReturnValue := TRUE;
    ELSE
        v_ReturnValue := FALSE;
    END IF;
    RETURN v_ReturnValue;
END AlmostFull;
```

Por Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[4]

5.1 Funciones

Creacion de Funciones

Se puede llamar a la función **AlmostFull** desde el siguiente bloque PL/SQL , en el que podemos observar que la función no es una orden en si misma, sino que usa como parte de la orden IF situada dentro del bucle.

```

DECLARE
    CURSOR c_Classes IS
        SELECT department, course
        FROM classes;
    BEGIN
        FOR v_ClassRecord IN c_Classes LOOP
            -- Registra todos los cursos que no tienen mucho espacio vacio en temp_table
            IF AlmostFull(v_ClassRecord.department, v_ClassRecord.course) THEN
                INSERT INTO temp_table (char_col) VALUES
                (v_ClassRecord.department || ' ' || v_ClassRecord.course || ' is almost full! ');
            END IF;
        END LOOP;
    END AlmostFull;

```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[5]

5.1 Funciones

Sintaxis de la Funciones

Las sintaxis para crear una función almacenada es muy similar a la de un procedimiento:

```

CREATE [OR REPLACE] FUNCTION nombre_función
[(argumento [ {IN | OUT | IN OUT } ] tipo,
...
argumento [ {IN | OUT | IN OUT } ] tipo)]
RETURN tipo_retorno { IS | AS}
cuerpo_función

```

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

Donde **nombre_función** es el nombre de la función, **argumento y tipo** son iguales que un procedimiento, **tipo_retorno** es el tipo del valor que devuelve la función y **cuerpo_función** es un bloque PL/SQL que contiene el código de la función.

Al igual que con los procedimiento, la lista de argumento es opcional. Si no hay argumento no hay paréntesis ni en la declaración de la función ni en la llamada a ella. Sin embargo el *tipo de retorno de la función es obligatorio*, dado que la llamada a la función parte de una expresión. El tipo de la función se usa para determinar el tipo de la expresión que contiene la llamada a la función.

[6]

5.1 Funciones

La Orden RETURN



Dentro del cuerpo de la función la orden RETURN se emplea para devolver el control y un valor, al entorno que hizo llamada. Las sintaxis general de la orden RETURN es:

RETURN expresión;

Donde **expresión** es el valor que la función devuelve, el cual se convierte en el tipo especificado en la cláusula RETURN de la definición de la función, si es que no es ya de ese tipo. Cuando se ejecuta la orden RETURN , se devuelve el control inmediatamente al entorno que hizo la invocación.

Puede haber de una orden RETURN e una función, aunque solo se ejecutara una de ellas. Es un error que una función concluya sin ejecutar una orden RETURN.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[7]

5.1 Funciones

La Orden RETURN



El siguiente ejemplo ilustra el caso de múltiples ordenes RETURN dentro de una función. Aunque se presentan 5 ordenes diferentes de RETURN en la función , solo una de ellas se ejecutara.

```

CREATE OR REPLACE FUNCTION ClassInfo (
    p_Department classes.department%TYPE,
    p_Course      classes.course%TYPE)
RETURN VARCHAR2 IS
    v_CurrentStudents      NUMBER;
    v_MaxStudents          NUMBER;
    v_PercentFull          NUMBER;
BEGIN
    -- Obtiene la cantidad actual y máxima de estudiantes para el curso solicitado
    SELECT current_students,      max_students
    INTO v_CurrentStudents, v_MaxStudents
    FROM classes
    WHERE department = p_Department
    AND course = p_Course;
    -- Calcula el porcentaje actual
    v_PercentFull := v_CurrentStudents / v_MaxStudents * 100;
    IF v_PercentFull = 100 THEN
        RETURN 'Full';
    ELSIF v_PercentFull > 80 THEN
        RETURN 'Some Room';
    ELSIF v_PercentFull > 60 THEN
        RETURN 'More Room';
    ELSIF v_PercentFull > 0 THEN
        RETURN 'Lost of Room';
    ELSE
        RETURN 'Empty';
    END IF;
END ClassInfo;
    
```

- Cuando se emplea una función, la orden RETURN debe tener una expresión asociada.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[8]

5.1 Funciones

Cuando utiliza una Función.

La función comparten muchas de las características de los procedimientos:

- Las funciones pueden devolver mas de un valor, mediante parámetros OUT
- El código de la función tiene secciones declarativas, ejecutables y de manejo de excepciones.
- Las funciones pueden aceptar valores predeterminados.
- Puede llamarse a las funciones utilizando notación posicional o nominal

Cuando debemos utilizar una función y cuando un procedimiento?

- Generalmente esto depende de cuantos valores deba devolver el programa y de como vaya a usarse dichos valores.
- Una regla práctica es que se use un procedimiento siempre que haya mas de un valor de retorno. Si el valor de retorno es único, entonces se puede emplear una función.
- Aunque es legal que las funciones incluyan parámetros OUT (y por lo tanto devolverán mas de un valor), no resulta recomendable desde el punto de vista del estilo de programación.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

9



5.1.1. PROCEDIMIENTO Y FUNCIONES

Eliminacion de Procedimiento y Funciones

Al que las tablas, los procedimientos y funciones también pueden ser eliminados, lo que los borra de diccionario de datos.

La sintaxis para eliminar un procedimiento es la siguiente

DROP PROCEDURE *nombre_procedimiento;*

La sintaxis para eliminar una función es la siguiente

DROP FUNCTION *nombre_función;*

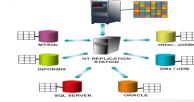
Donde *nombre_procedimiento* y *nombre_función* son el nombre de un procedimiento o función existentes, respectivamente. Por ejemplo la eliminación de AddNewStudent;

DROP PROCEDURE AddNewStudent;

DROP es una orden DDL, así que se ejecuta una orden COMMIT implícita tanto antes como después de la orden DROP.

Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

10



5. SITUACIONES DE LOS SUBPROGRAMAS

Los programas almacenados y el diccionario de los Datos

- Los subprogramas pueden ser almacenados en el diccionario de datos, como todos ejemplos mostrados. Primero se crea con la orden CREATE OR REPLACE y luego puede ser llamado desde otro bloque PL/SQL.
- También podemos definir un subprograma en sección declarativa de un bloque, en cuyo caso se denomina subprograma local.

Intentamos crear este procedimiento. Que pasara cuando se almacena en el diccionario de la Base de Datos?

```
SQL> create or replace procedure simple as
2  v_counter number;
3  begin
4    v_counter := 7;
5  end simple;
6 /
Advertencia: Procedimiento creado con errores de compilación.

SQL> show error
Errores para PROCEDURE SIMPLE:
LINE/COL  ERROR
5/1    PL/SQL: Encountered the symbol "END" when expecting one of the
          following:
          * & - + ; < / > at in is mod remainder not rem
          <an exponent (<+>) > < - > < = > < > and < > like like2
          like2 like2 between || multistatement construct
          The symbol ":" was substituted for "END" to continue.

SQL> |
```



Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[11]

5.1.2 SITUACIONES DE LOS SUBPROGRAMAS

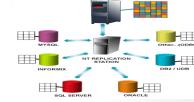
Los programas almacenados y el diccionario de los Datos

Podemos observar la situación de las situación de objetos que estamos creado accesando el contenedor de estos en la base de datos '`user_objects`' .

Si desde un bloque intentamos invocar este procedimiento , se producirá el error:
PLS-905: object is invalid.

```
SQL> select object_name, object_type, status
2  from user_objects
3  where object_name = 'SIMPLE';

OBJECT_NAME          OBJECT_TYPE      STATUS
SIMPLE              PROCEDURE        INVALID
```



Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[12]

5.1.2 SITUACIONES DE LOS SUBPROGRAMAS

Subprogramas Locales

Ejemplo de un subprograma local, declarado en la sección declarativa del PL/SQL

```

DECLARE
    CURSOR c_AllStudents IS
        SELECT first_name, last_name
        FROM students;
    v_FormattedName VARCHAR2(50);
    --Función que devolverá el nombre y apellido concatenado y separado por un espacio
    FUNCTION FormatName (p_FirstName IN VARCHAR2,
                          p_LastName IN VARCHAR2)
    RETURN VARCHAR2 IS
    BEGIN
        RETURN p_FirstName || ' ' || p_LastName;
    END FormatName;
    -- Inicia el programa principal
    BEGIN
        FOR v_StudentRecord IN c_AllStudents LOOP
            v_FormattedName := FormatName (v_StudentRecord.first_name,
                                            v_StudentRecord.last_name);
            INSERT INTO temp_table (char_col) VALUES (v_FormattedName);
        END LOOP;
    END;

```

La función **FormatName** se declara en la sección declarativa del bloque anónimo. El nombre de la función es un identificador PL/SQL y sigue por lo tanto las misma reglas de ámbito y visibilidad que cualquier otro identificador. Para ser mas exacto, la función solo es visible en el bloque que ha sido declarada y su ámbito se extiende desde el punto de la declaración hasta el final del bloque. Ningún otro bloque puede hacer llamado a **FormatName**, dado que no es visible fuera del bloque.



Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[13]

5.1.2 SITUACIONES DE LOS SUBPROGRAMAS

Subprogramas Locales

Los subprogramas locales deben ser declarados al final de la sección declarativa. Si situamos **FormatName** por encima de la declaración de **C_AllStudents**, como se muestra en el siguiente ejemplo, obtendríamos un error de compilación.

```

DECLARE
    /* Declara en primer lugar FormatName. Esto generará un error de compilación, ya que todas
    las declaraciones tienen que estar antes de cualquier subprograma local. */
    FUNCTION FormatName (p_FirstName IN VARCHAR2,
                          p_LastName IN VARCHAR2)
    RETURN VARCHAR2 IS
    BEGIN
        RETURN p_FirstName || ' ' || p_LastName;
    END FormatName;
    CURSOR c_AllStudents IS
        SELECT first_name, last_name
        FROM students;
    v_FormattedName VARCHAR2(50);
    -- Inicio del bloque principal
    BEGIN
        NULL;
    END;

```



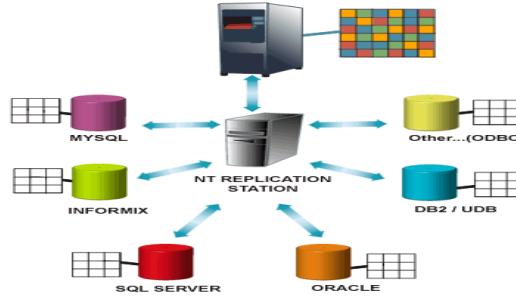
Por: Ing. Henry Lezcano Sistemas
de Base de Datos II Semestre del
2020

[14]

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD DE INGENIERIA DE SISTEMAS
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.

SISTEMA DE BASE DE DATOS II ORACLE PROGRAMACION PL/SQL

TRIGGERS -PL/SQL.ORACLE

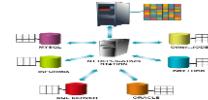


Sistemas de Base de Datos II Ing.
 Henry Lezcano II Semestre 2020

[1]

CONTENIDO

*Capítulo V. Implementación de Otros Objetos de servidor.
 (Procedimientos, almacenados, disparadores, cursares).*



Sistemas de Base de Datos II Ing.
 Henry Lezcano II Semestre 2020

[2]

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Los disparadores se asemejan a los procedimientos y funciones en que son bloques PL/SQL nominados con secciones declarativas, ejecutable y de manejo de excepciones.

Al igual que los objetos de base de datos, los disparadores deben ser almacenados en la base de datos y no pueden ser locales a un bloque. Sin embargo, un procedimiento se ejecuta de manera explícita desde otro bloque mediante una llamada de procedimiento, que puede también usar argumentos.

Un disparador, por el contrario, se ejecuta de manera implícita cada vez que tiene lugar el suceso de disparo, y el disparador no admite argumentos. El acto de ejecutar disparador se conoce como disparo o trigger.

El suceso de disparo es una operación DML (INSERT, UPDATE o DELETE) sobre una tabla de la base de datos.



[3]

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Los disparadores pueden ser usados para muchas cosas diferentes, incluyendo:

- El mantenimiento de restricciones de integridad compleja, que no sea posible con las restricciones declarativas definidas en el momento de crear la tabla.
- La auditoria de la información contenida en la tabla, registrando los cambios realizados y la identidad del que los llevo a cabo.
- El aviso automático a otros programas de que hay que llevar a cabo una determinada acción, cuando se realiza un cambio en la tabla.

Por ejemplo, supongamos que queremos mantener una serie de estadísticas acerca de las diferentes especialidades, incluyendo el numero de estudiantes matriculados y la cantidad total de créditos seleccionados. Se deberá almacenar estos resultados en la tabla *major_stats*:

```
CREATE TABLE major_stats (
    major      VARCHAR2(30),
    total_credits  NUMBER,
    total_students NUMBER);
```



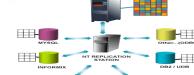
[4]

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Para poder mantener actualizada la información estadística actualizada en la tabla `major_stats` crearemos un disparador para la tabla `students` que actualice `major_stats` cada vez que se modifique la tabla `students`. Esto es lo que hace el disparador `UpdateMajorStats`, que mostramos a continuación:

```
CREATE OR REPLACE TRIGGER UpdateMajorStats
/* mantiene la tabla de especialidades major_stats actualizada, de acuerdo con los cambios realizados en la
tabla estudiante students */
AFTER INSERT OR DELETE OR UPDATE ON students
DECLARE c_Statistics IS
    SELECT major, count(*) total_students, SUM(current_credits) total_credits
    FROM students
    GROUP BY major;
BEGIN
    /* Recorrer mediante un bucle cada especialidad. Intenta actualizar las estadísticas en major_stats que
correspondan a dicha especialidad. Si la fila no existe , se creara. */
    FOR v_StatsRecord IN c_Statistics LOOP
        UPDATE major_stats
        SET total_credits = v_StatsRecord.total_credits,
            total_students = v_StatsRecord.total_students
        WHERE major = v_StatsRecord.major;
        -- Comprueba la existencia de la fila
        IF SQL%NOTFOUND THEN
            INSERT INTO major_stats (major, total_credits, total_students)
            VALUES (v_StatsRecord.major, v_StatsRecord.total_credits, v_StatsRecord.total_student);
        END IF;
    END LOOP;
END UpdateMajorStats;
```



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La sintaxis general para crear un disparador es:

```
CREATE [OR REPLACE ] TRIGGER nombre_disparador
{BEFORE| AFTER} suceso_disparo ON referencia_tabla
[FOR EACH ROW [ condicion_disparo]]
cuadro_disparador;
```

Donde *nombre_disparador* corresponde al nombre del disparador, *suceso_disparo* especifica cuando se activa el disparador (en el caso del trigger anterior `UpdateMajorStats`, después de cualquier operación DML), *referencia_tabla* es la tabla para la cual se define el disparador y *cuadro_disparador* es código principal del disparador. Antes se evalúa la condición disparo en la cláusula WHEN, si es que esta presente. El cuadro del disparador se ejecuta solo cuando dicha condición se evalúa como verdadera.



Componentes de un disparador

Los componentes requeridos en un disparador son *el nombre*, el *suceso de disparo* y el *cuerpo*. La cláusula **WHEN** es opcional.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Nombre de los disparadores

El espacio de nombres para los disparadores es diferente del de los otros subprogramas. El espacio de nombre es un conjunto de identificadores validos que pueden emplearse como nombre de un objeto.

Los procedimientos, las funciones, paquetes y tablas tienen el mismo espacio de nombre, lo que quiere decir, que dentro de un esquema de base de datos, todos los objetos de cualquiera de dichos tipos deben tener nombre diferentes. Es ilegal, por ejemplo dar el mismo nombre a un procedimiento y a un paquete.

Los disparadores, sin embargo tienen un espacio de nombres separado, por lo que un disparador puede tener el mismo nombre que una tabla o que un procedimiento. Dentro de un esquema de base de datos todos los disparadores deben tener nombres diferentes entre si. Estos nombres son identificadores de base de datos y por lo tanto deben seguir las misma reglas que resto de los identificadores.



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Nombre de los disparadores

Ejemplo, podemos crear un disparador llamado **major_stats** sobre la tabla **major_stats**, pero es ilegal que un procedimiento también se llame **major_stats**.

```
SQL> CREATE OR REPLACE TRIGGER major_stats
      BEFORE INSERT ON major_stats
      BEGIN
        INSERT INTO temp_table( char_col)
        VALUES ( 'Trigger called!');
      END major_stats;
      /
Trigger Created.
```

```
SQL> CREATE OR REPLACE PROCEDURE major_stats AS
      BEGIN
        INSERT INTO temp_table( char_col)
        VALUES ( 'Trigger called!');
      END major_stats;
      /
CREATE OR REPLACE PROCEDURE major_stats AS
*
Error at line 1:
ORA-00955 name is already used by an existing object
```

Aunque es posible usar el mismo nombre para un disparador que para una tabla, no es recomendable. Es mejor dar a cada trigger un nombre diferente que identifique su función como la tabla sobre la cual se define.

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Tipos de Disparadores

Los sucesos de disparos determina el tipo de disparador. Los disparadores pueden definirse para las operaciones de INSERT, UPDATE o DELETE y pueden dispararse antes o después de la operación. El nivel de los disparadores puede ser la fila o la orden.

Los valores de la orden, de la temporización y de nivel determinan el tipo de disparador. Hay un total de 12 tipos posibles: 3 ordenes por opciones de de temporización, por 2 niveles. Todos los siguientes tipos de disparadores por ejemplo, son validos:

- Previo a la actualización y con nivel de orden*
- Posterior a la inserción y con nivel de fila*
- Previo al borrado con el nivel de fila.*

Puede definirse hasta 12 disparadores en una tabla, una de cada tipo, sin embargo, una tabla puede tener mas de un disparador de cada tipo. Esta capacidad permite cuantos disparadores se quiera para una tabla.



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Tipos de Disparadores

Un disparador también puede activarse para mas de un tipo de orden. El disparador **UpdateMajorStats**, por ejemplo se activa con las ordenes **INSERT, UPDATE y DELETE**. El suceso de disparo especifica uno o varias operaciones DML que den activar el disparador.

Categoría	Valores	Comentario
Orden	INSERT, UPDATE, DELETE	Define que tipo orden DML provoca la activación del disparador.
Temporización	BEFORE o AFTER	Define si el disparador se activa antes o después de que se ejecute la orden (disparador previo o posterior)
Nivel	Fila u Orden	Los disparadores con nivel de fila se activan una vez por cada fila afectada por la orden que provocó el disparo. Los disparadores con nivel de orden se activan solo una vez, antes o después de la orden. Los disparadores con nivel de fila se identifican por la clausula FOR EACH ROW en la definición del disparador.

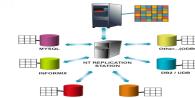
5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Restricciones de los Disparadores

El cuerpo de un disparador es bloque PL/SQL. Cualquier orden que sea legal en un bloque PL/SQL, es legal en el cuerpo de un disparador, con las siguientes restricciones:

- ❑ Un disparador no puede emitir ninguna orden de control de transacciones: COMMIT, ROLLBACK o SAVEPOINT. El disparador se activa como parte de la ejecución de la orden que provoca el disparo, y forma parte de la misma transacción que dicha orden. Cuando la orden que provoca el disparo es confirmada o cancelada, se confirma o cancela también el trabajo realizado por el disparador.
- ❑ Por razones idénticas, ningún procedimiento o función llamado por el disparador puede emitir ordenes de control de transacciones.
- ❑ El cuerpo del disparador no puede contener ninguna declaración de variables LONG o LONG RAW. Asimismo a columnas de tipo LONG o LONG RAW de la tabla sobre la que se define el disparador.
- ❑ Existen restricciones acerca de a que tablas puede acceder el cuerpo de un disparador. Dependiendo del tipo de disparador y de las restricciones que afecten a las tablas, dichas tablas pueden ser mutantes.



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Los Disparadores Y el Diccionario de Datos.

De forma similar a lo que sucede con los subprogramas almacenados, algunas vistas del diccionario de datos contienen información acerca de los disparadores y de su estado. Estas vistas se actualizan cada vez que se crea o elimina un disparador.



Vista del diccionario de datos

Cuando se crea un disparador, su código fuente se almacena en la vista ***user_triggers*** del diccionario de datos. Esta vista incluye el cuerpo de disparador, la cláusula WHEN, la tabla de disparo y el tipo de disparador. La consulta siguiente, por ejemplo, devuelve información acerca de ***UPDATEMAJORSTATS***:

```
SQL> SELECT trigger_type, table_name, triggering_event
  FROM user_triggers
 WHERE triggers_name = 'UPDATEMAJORSTATS';
 /
```

trigger_type	table_name	triggering_event
AFTER STATEMENT	STUDENTS	INSERT OR UPDATE OR DELETE

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Eliminación y deshabilitación de los Disparadores.

Al igual que los procedimientos y paquetes, también los disparadores pueden eliminarse. La sintaxis de la orden que realiza esta operación es:

```
DROP TRIGGER nombre_disparador;
```

Donde nombre_disparador es el nombre del disparador a eliminar. Esta orden elimina el trigger del diccionario datos de forma permanente. Al igual que los subprogramas, puede incluirse la cláusula OR REPLACE en la orden CREATE de creación del disparador.

A diferencia de los procedimientos y paquetes, sin embargo se puede deshabilitar un disparador sin necesidad de eliminarlo. Cuando se deshabilita el trigger, continua existiendo en el diccionario de datos, pero no puede llegar a dispararse.

Para desactivar un disparador, se utiliza la orden **ALTER TRIGGER**,

```
ALTER TRIGGER nombre_disparador {DISABLE | ENABLE};
```



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Eliminación y deshabilitación de los Disparadores.

```
ALTER TRIGGER nombre_disparador {DISABLE | ENABLE};
```

Donde nombre_disparador es el nombre del disparador. Todos los disparadores están en principio, habilitados en el momento de la creación. ALTER TRIGGER puede deshabilitar y luego habilitar, cualquier disparador.

Ejemplos:

```
SQL> ALTER TRIGGER UpdateMajorStats DISABLE;
```

Trigger altered.

```
SQL> ALTER TRIGGER UpdateMajorStats ENABLE;
```

Trigger altered.



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

Eliminación y deshabilitación de los Disparadores.

También se puede habilitar o deshabilitar todos los disparadores de una tabla determinada, utilizando la orden ALTER TABLE con la cláusula **ENABLE ALL TRIGGERS** o **DISABLE ALL TRIGGERS**.

Ejemplos:

```
SQL> ALTER TABLE students
  ENABLE ALL TRIGGERS;
Table altered.
```

```
SQL> ALTER TABLE students
  DISABLE ALL TRIGGERS;
Trigger altered.
```

La columna status del user_triggers contiene el valor 'ENABLE' o "DISABLE", indicando el estado actual de cada disparador. Al deshabilitar el trigger no lo elimina del diccionario de datos, como si lo haría la operación de borrado

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

Los disparadores se activan al ejecutar la orden DML. Algoritmo de ejecución de una orden DML es el siguiente:

1. Ejecutar, si existe, el disparador de tipo BEFORE (disparador previo) con nivel de orden.
2. Para cada fila a la que afecte la orden:
 - a) Ejecutar, si existe, el disparador BEFORE con nivel de fila.
 - b) Ejecutar la propia orden.
 - c) Ejecutar, si existe, el disparador de tipo AFTER (disparador posterior) con nivel de fila.
3. Ejecutar, si existe, el disparador de tipo AFTER con nivel de orden.

El ejemplo a continuación.....

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

Los disparadores se activan al ejecutar la orden DML. Algoritmo de ejecución de una orden DML es el siguiente:

1. Ejecutar, si existe, el disparador de tipo BEFORE (disparador previo) con nivel de orden.
2. Para cada fila a la que afecte la orden:
 - a) Ejecutar, si existe, el disparador BEFORE con nivel de fila.
 - b) Ejecutar la propia orden.
 - c) Ejecutar, si existe, el disparador de tipo AFTER (disparador posterior) con nivel de fila.
3. Ejecutar, si existe, el disparador de tipo AFTER con nivel de orden.

El ejemplo a continuación.....



[17]

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

El ejemplo a continuación.....

```

CREATE SEQUENCE trigger_seg
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER classes_BStatement
BEFORE UPDATE ON classes
BEGIN
  INSERT INTO temp_table (num_col, char_col)
  VALUES (trigger_seg.NEXTVAL, 'Before Statement trigger');
END classes_BStatement;

CREATE OR REPLACE TRIGGER classes_AStatement
AFTER UPDATE ON classes
BEGIN
  INSERT INTO temp_table (num_col, char_col)
  VALUES (trigger_seg.NEXTVAL, 'After Statement trigger');
END classes_AStatement;
  
```



[18]

5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

El ejemplo a continuación.....

```
CREATE OR REPLACE TRIGGER classes_BRow
  BEFORE UPDATE ON classes
  FOR EACH ROW
BEGIN
  INSERT INTO temp_table (num_col, char_col)
  VALUES (trigger_seg.NEXTVAL, 'Before Row trigger');
END classes_BRow;

CREATE OR REPLACE TRIGGER classes_ARow
  AFTER UPDATE ON classes
  FOR EACH ROW
BEGIN
  INSERT INTO temp_table (num_col, char_col)
  VALUES (trigger_seg.NEXTVAL, 'After Row trigger');
END classes_ARow;
```



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:

La orden de activación de los Disparadores.

Suponga que, a continuación, ejecutamos la siguiente orden UPDATE:

```
UPDATE classes
  SET num_credits = 4
 WHERE department IN ('HIS', 'CS', 'CA');
```

Esta orden afecta a cuatro filas. Los disparador previo y posterior con nivel de orden se ejecutan una sola vez, y los disparadores previo y posterior con nivel de fila se ejecutan cuatro veces cada uno. Si efectuamos entonces una selección sobre temp_table, obtendremos:

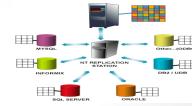
```
SQL> SELECT * FROM temp_table
  ORDER BY cum_col;
```

NUM_COL	CHAR_COL
1	Before Statement trigger
2	Before Row trigger
3	After Row trigger
4	Before Row trigger
..
10	After Statement trigger



5.2. DISPARADORES -TRIGGER

Creacion de Disparadores:



La Cláusula WHEN

La cláusula **WHEN** solo es valida para disparadores con nivel de fila. Si esta presente, el cuerpo del disparador solo se ejecutara para las filas que cumplan con la condicion especificada en la clausula. La clausula **WHEN** tiene la forma :

WHEN *condición*

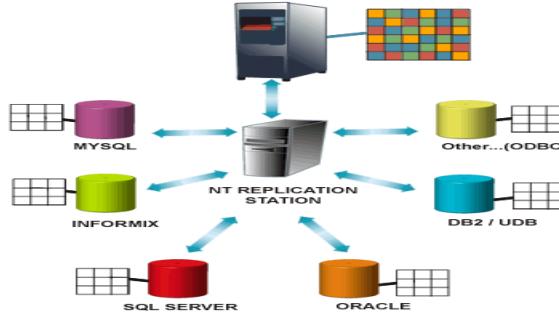
Donde *condición* es la expresión booleana que será evaluada para cada fila.

```
CREATE OR REPLACE TRIGGER CheckCredits
  BEFORE INSERT OR UPDATE OF current_credits ON students
  FOR EACH ROW
  WHEN (new.current_credits > 20)
  BEGIN
    /* Aquí estaría el cuerpo el disparador */
  END;
```

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD EN INGENIERIA DE SISTEMAS COMPUTACIONALES
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION

SISTEMAS DE BASE DE DATOS II

Definición y Conceptos de Transacciones y Boqueo Control de Concurrencia



SISTEMAS DE BASE DE DATOS II
 Por: Ing. Henry J. Lezcano

[1]

CONTENIDO

Capítulo 6. Conceptos Básicos de transacciones y Control de Concurrencia

1. Definición y conceptos de Bases de Datos transaccionales.

6.1 Concepto de transacciones y sus propiedades

Principios de control de concurrencia t manejo de transacciones

Serialización y métodos de Serialización

Manejo de transacciones y bloqueos

Políticas de bloqueos

SISTEMAS DE BASE DE DATOS II
 Por: Ing. Henry J. Lezcano

[2]

CONTENIDO

Capítulo 6. Conceptos Básicos sobre transacciones y control de concurrencia

1. Que es una Transacción?
2. Que significa que una transacción sea atómica?
3. Que significa que una transacción sea consistente?
4. Que significa que una transacción este en aislamiento?
5. Que significa que una transacción sea durable?
6. Cual podría ser el estado de una transacción?
7. Que significa Bloqueo de transacciones?
8. Que significa seriabilidad de transacciones?

[3]



VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

6.1 Definición y Conceptos de BD transaccionales

QUE ES UNA TRANSACCION?

- Una transacción es una secuencia de operaciones llevadas a cabo como una unidad lógica de trabajo simple.
- Una acción o serie de acciones llevada a cabo por único usuario o por un programa de aplicación y que lee y actualiza el contenido de la base de datos.



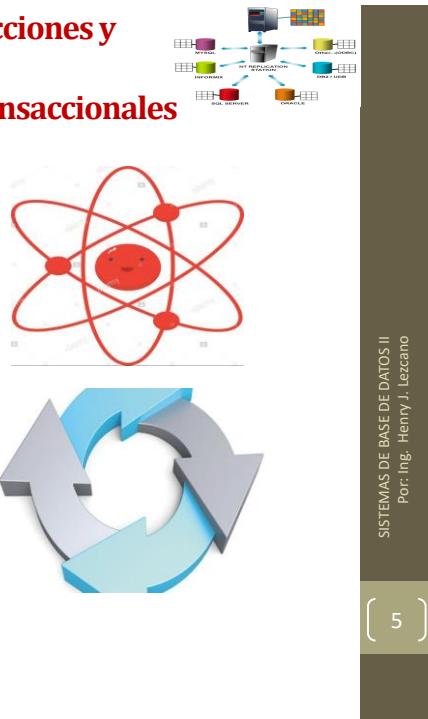
[4]

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales

PROPIEDADES DE LAS TRANSACCIONES

- **ATOMICIDAD:** Una transacción debe ser una unidad atómica de trabajo: o todas sus operaciones se llevan a cabo o no se realiza ninguna de ellas.
- **CONSISTENCIA:** una transacción debe llevar a la base de datos de un estado consistente a otro.



[5]

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales

PROPIEDADES DE LAS TRANSACCIONES

- **AISLAMIENTO:** Las modificaciones realizada por una transacción deben aislarse de las modificaciones llevadas a cabo por otras posibles transacciones concurrentes.



[6]

- **DURABILIDAD:** Una vez la transacción ha terminado con éxito sus efectos deben hacerse permanentes en la base de datos.



VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales



TRANSACCIONES

- Las transacciones aseguran que varias modificaciones a los datos se procesados como unidad 'atomicidad'.
 - Por ejemplo una transacción de una cuenta de bancaria podría abonar en una cuenta y cargar en otra. Los pasos se deben completar al mismo tiempo.

BLOQUEOS

- Las transacciones utilizan los bloqueos para impedir que los usuarios cambien o lean los datos de una transacción que no se ha completado.
- El bloqueo es necesario en el Proceso de transacciones en línea (OLTP, Online Transaction Processing) en los sistemas multiusuarios.

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales



BLOQUEOS continuación...

- Los bloqueos impiden los conflictos de actualización. Los usuarios no pueden leer o modificar los datos que están en proceso de modificación por parte de otros usuarios.
 - Por ejemplo, si deseamos calcular una función de agregado y asegurarse que otra transacción no modifique el conjunto de datos que se utiliza para calcular la función de agregado, puede solicitar que el sistema establezca bloqueos en los datos.
 - Investigar el concepto de 'función agregado'
- Los bloqueos hacen posible la serialización de transacciones de forma que solo una persona a la vez pueda modificar un elemento de datos.
 - Por ejemplo en el sistema de reserva de una línea área los bloqueos aseguran que solo se asigne el asiento concreto a una sola persona.

VI. Conceptos Básicos sobre Transacciones y

Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales

BLOQUEOS continuación...

- En Oracle los bloqueos se hacen de manera automática, el SGBD establece y ajusta dinámicamente el nivel de bloqueo que tendría que realizar, el usuario no interviene en la toma de decisión referente a este proceso.
- Los bloqueos son necesarios para que las transacciones concurrentes o simultáneas permitan que los usuarios tengan acceso y actualicen los datos al mismo tiempo.
 - La alta concurrencia o simultaneidad significa que hay varios usuarios que consiguen un buen tiempo de respuesta con pocos conflictos.



SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

[9]

VI. Conceptos Básicos sobre Transacciones y

Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales

BLOQUEOS continuación...

- Desde la perspectiva del Administrador de Sistemas los problemas principales son el numero de usuarios, el numero de transacciones y el rendimiento.



SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano



[10]

VI. Conceptos Básicos sobre Transacciones y

Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales



ESTADO DE UNA TRANSACCION

- En ausencias de fallos todas las transacciones se completan con éxito. Sin embargo, una transacción puede que no siempre acabe su ejecución con éxito.
- A una transacción que no acaba con éxito se denomina transacción abortada.
- Una vez, deshecho todos los cambios efectuados por una transacción abortada, se dice que transacción esta retrocedida o fallida.
- Una transacción que termina su ejecución con éxito se dice que esta comprometida o confirmada.

VI. Conceptos Básicos sobre Transacciones y

Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales



ESTADO DE UNA TRANSACCION

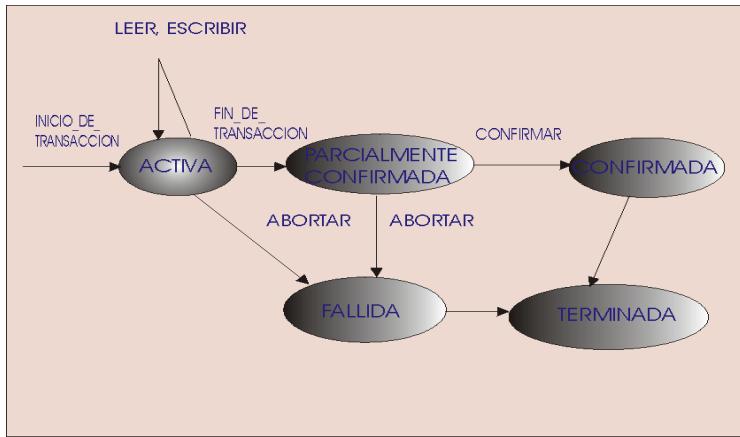
Una transacción debe estar en uno de los siguientes estados:

- Activa: estado inicial, permanece en ese estado durante la ejecución.
- Parcialmente confirmada: después de ejecutarse la ultima instrucción.
- Fallida: tras descubrir que no puede terminar la ejecución normal.
- Abortada: después del retroceso de la transacción y de haber restablecido la base de datos a su estado anterior al comienzo de la transacción.
- Confirmada: tras completarse con éxito.

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia

6.1 Definición y Conceptos de BD transaccionales

ESTADO DE UNA TRANSACCION



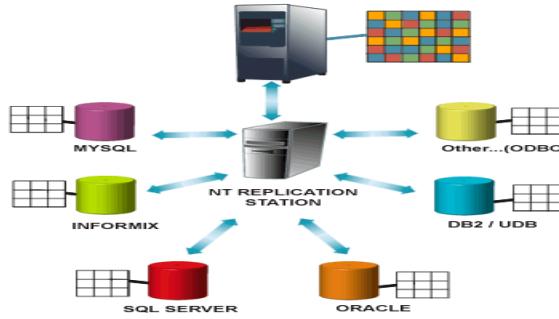
SISTEMAS DE BASE DE DATOS I
Por: Ing. Henry J. Lezcano

[13]

UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD EN INGENIERIA DE SISTEMAS COMPUTACIONALES
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION

SISTEMAS DE BASE DE DATOS II

Conceptos Básicos sobre Transacciones y Control de Concurrencia



SISTEMAS DE BASE DE DATOS II
 Por: Ing. Henry J. Lezcano

[1]

CONTENIDO

Capítulo 6. Conceptos Básicos de transacciones y Control de Concurrencia

1. Definición y conceptos de Bases de Datos transaccionales.

6.1 Concepto de transacciones y sus propiedades

Principios de control de concurrencia t manejo de transacciones

Serialización y métodos de Serialización

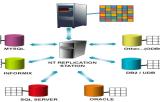
Manejo de transacciones y bloqueos

Políticas de bloqueos

SISTEMAS DE BASE DE DATOS II
 Por: Ing. Henry J. Lezcano

[2]

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia



CONCURRENCIA

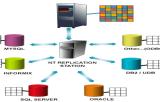
- Se refiere a la capacidad de que los Sistemas de Gestión de Base de Datos, de permitir que múltiples procesos o transacciones sean ejecutados al mismo tiempo o accedan la Base de Datos a la vez y también puedan interactuar entre sí.
 - Los procesos concurrentes pueden ser ejecutados realmente de forma simultánea, sólo cuando cada uno es ejecutado en diferentes procesadores.



SISTEMAS DE BASE DE DATOS II

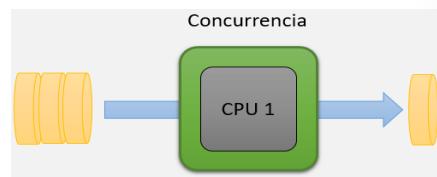
3

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia



CONCURRENCIA

- La concurrencia es simulada si solo existe un procesador encargado de ejecutar todas los procesos, simulando la concurrencia, ocupándose de forma alternada de uno y otro proceso a muy pequeños intervalos de tiempo.



ISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

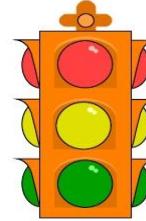
4

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia



CONTROL DE CONCURRENCIA

- Cuando existen varios usuarios intentando modificar los datos al mismo tiempo, se necesita establecer algún tipo de control para que dichas modificaciones de un usuario no interfieran en las de los otros. A este sistema se le llama control de concurrencia.
- El objetivo del control de concurrencia y recuperación es asegurar que cada transacción se ejecuta atómicamente es decir:
 - Cada transacción accede a la información compartida sin interferir con otras transacciones y si una transacción termina normalmente, todos los efectos son permanentes, en caso contrario no tiene efecto alguno en la Base de Datos.



VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia



CONTROL DE CONCURRENCIA

Actualmente se conocen dos tipos de control de concurrencia:

- **El Control de Concurrencia Pesimista:** este control bloquea los datos cuando se leen para preparar una actualización. Los demás usuarios no pueden realizar acciones que alteren los datos subyacentes hasta que el usuario que ha aplicado el bloqueo termine con los datos.
 - El control concurrencia pesimista se puede utilizar donde haya una alta contención de los datos y el costo de proteger los datos con bloqueos sea menor que el costo de deshacer transacciones si se producen conflictos de concurrencia.



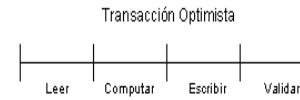
VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia



CONTROL DE CONCURRENCIA

Actualmente se conocen dos tipos de control de concurrencia:

El Control de Concurrencia Optimista: no bloquea los datos cuando se leen inicialmente. Cuando se ejecuta el proceso de actualización se deben realizar comprobaciones para determinar si los datos subyacentes no han cambiado desde que se leyeron al inicio. De ser así, la transacción se deshace y los usuarios deberán volver a empezar.



SISTEMAS DE BASE DE DATOS I
Por: Ing. Henry J. Lezcano

[7]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

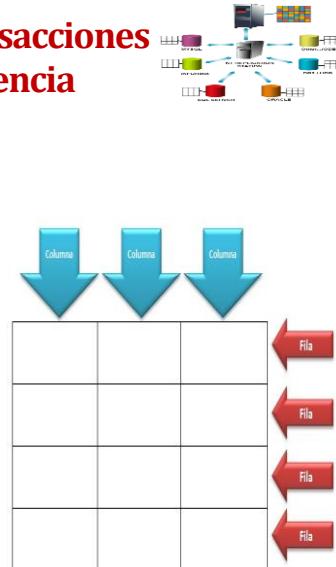
CONTROL DE CONCURRENCIA

GRANULARIDAD

- Al referirnos a lo que es bloqueo en bases de datos en realidad utilizamos lo que se conoce como granularidad del bloqueo.
- La granularidad se refiere a que tan bueno o efectivo se quiere que sea un bloqueo.

Por ejemplo

- ¿desea bloquear la tabla completa (un bloqueo de granularidad pesada)?
- o solo desea bloquear una fila específica (un bloqueo de granularidad buena)?.



SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

[8]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

GRANULARIDAD

Deben tenerse en cuenta algunos elementos relacionados con la profundidad de los bloqueos.

- *Con un bloqueo de granularidad buena se adquieren muchos recursos para administrar el bloqueo, pero se asegura la consistencia de los datos.*
- *Con un bloqueo de granularidad pesada se utilizan menos recursos pero se aumenta el riesgo de inconsistencia de los datos, y tal vez hasta se evite que otros usuarios realicen sus tareas.*



SISTEMAS DE BASE DE DATOS I
Por: Ing. Henry J. Lezcano

[9]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

GRANULARIDAD continuación....

- Pero ¿que pasa si se desea establecer específicamente un nivel de bloqueo?
 - Por lo general esto no es necesario en aplicaciones que tienen solo unos cuantos usuarios, pero tal vez algunas veces encontraremos que los bloqueos no se liberan tan rápido como uno quisiera, si esto ocurre se puede especificar la granularidad del bloqueo que se requiera.
 - **En cuanto a la Base de Datos la Oracle, esta se encarga del bloqueo por nosotros.**



ORACLE
DATABASE



SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

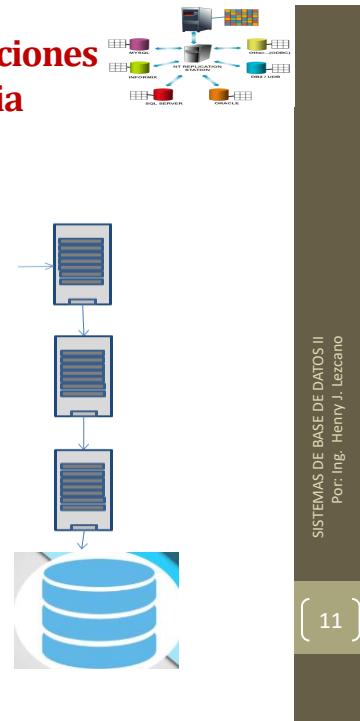
[10]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

SERIABILIDAD

- Una forma de evitar los problemas de interferencia es no permitir que las transacciones se intercalen. Una ejecución en la cual ninguna de dos transacciones se intercalan se conoce como transacción serial.
- Una ejecución se dice que es serial si para todo par de transacciones, todas las operaciones de una transacción, se ejecutan antes de cualquier operación de la otra transacción.

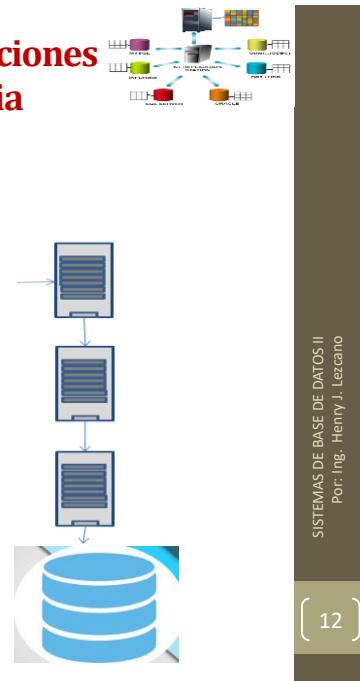


VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

SERIABILIDAD

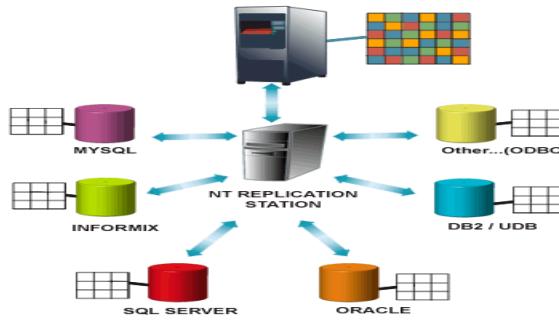
- Desde el punto de vista del usuario, en una ejecución serial se observa como si las transacciones fueran operaciones que el DBMS procesa atómicamente.
- Las transacciones seriales son correctas dado que cada transacciones es correcta individualmente y las transacciones que se ejecutan serialmente no pueden interferir con otras. Cumpliendo con las propiedades de las transacciones.



UNIVERSIDAD TECNOLOGICA DE PANAMA
 FACULTAD EN INGENIERIA DE SISTEMAS COMPUTACIONALES
 LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION

SISTEMAS DE BASE DE DATOS II

Conceptos Básicos sobre Transacciones y Control de Conurrencia



SISTEMAS DE BASE DE DATOS II
 Por: Ing. Henry J. Lezcano

[1]

CONTENIDO

Capítulo 6. Conceptos Básicos de transacciones y Control de Conurrencia

1. Definición y conceptos de Bases de Datos transaccionales.

6.1 Concepto de transacciones y sus propiedades

Principios de control de concurrencia t manejo de transacciones

Serialización y métodos de Serialización

Manejo de transacciones y bloqueos

Políticas de bloqueos

SISTEMAS DE BASE DE DATOS II
 Por: Ing. Henry J. Lezcano

[2]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

INTERBLOQUEO –BLOQUEO MORTAL

Los interbloqueos se producen cuando dos o más subprocesos de la aplicación en una base de datos intentan bloquear el acceso de la misma información al mismo tiempo.

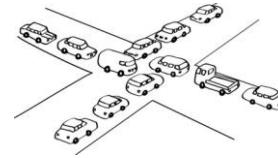
Los subprocesos esperan regularmente por un bloqueo en una información específica, y tienen un bloqueo en una variedad de cuadros y vistas.

Esto significa que un programa de diagnóstico debe buscar cuidadosamente los signos de un interbloqueo.



SISTEMAS DE BASE DE DATOS I
Por: Ing. Henry J. Lezcano

[3]



VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

INTERBLOQUEO –BLOQUEO MORTAL

Cuando un interbloqueo existe, es porque hay un conjunto de transacciones, de manera que toda transacción del conjunto esta esperando un elemento de datos bloqueados por otra transacción del conjunto



El estudiante debe validar estos métodos

Existen métodos para tratar este problema:

- Temporizaciones **Milagros Campos**
- Prevención de interbloqueo **Reynaldo Rojas**
- Detección y recuperación del interbloqueos **Emanol Gonzalez**

Hay varios programas disponibles para detectar y eliminar los interbloqueos, o usted puede crear su propio programa. Dentro de las estrategias podrían usar eliminar el interbloqueo, apropiación temporal, Puntos de conformidad, sincronismo o checkpoints.

SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

[4]

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia



CONTROL DE CONCURRENCIA

INTERBLOQUEO –BLOQUEO MORTAL

Sincronización

Los subprocesos de la aplicación acceden a la información en cuestión de segundos, la mayoría de las transacciones tienen lugar en menos de un segundo. Parte de los criterios utilizados para determinar cuándo se produce un interbloqueo es el tiempo de espera.

Al configurar la aplicación, o una aplicación de detección de interbloqueos, debe decidir en un período de tiempo de espera adecuado. Este es el tiempo que la aplicación va a esperar antes de decidir que los subprocesos se encuentran estancado en la base de datos. En general 30 a 60 segundos es más que suficiente.



SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

(5)

VI. Conceptos Básicos sobre Transacciones y Control de Conurrencia



CONTROL DE CONCURRENCIA

INTERBLOQUEO –BLOQUEO MORTAL

Múltiples aplicaciones que no responden

Puede diagnosticar los síntomas de un interbloqueo fácilmente si más de una aplicación está accediendo a la base de datos.

Una aplicación puede contener un error que no le permita acceder a una parte específica de información. Sin embargo, es menos probable que múltiples aplicaciones contenga el mismo error.

Cuando las solicitudes de acceso múltiples de la misma base de datos dejan de responder, lo más probable es que se trate de un interbloque.



ISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

6

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONTROL DE CONCURRENCIA

INTERBLOQUEO –BLOQUEO MORTAL

Registros

La base de datos Oracle mantiene un registro de actividad constante para cada cuadro y vista. Estos registros se pueden utilizar para diagnosticar un bloqueo de base de datos..

Las respuesta a aplicaciones por causa de los bloqueos serán en menos de 30 segundos. Los bloqueos más largos son signos de un interbloqueo.

- ¿Qué ES UN BLOQUEO MORTAL EN BASES DE DATOS ? **investigador Elionays Rosas**



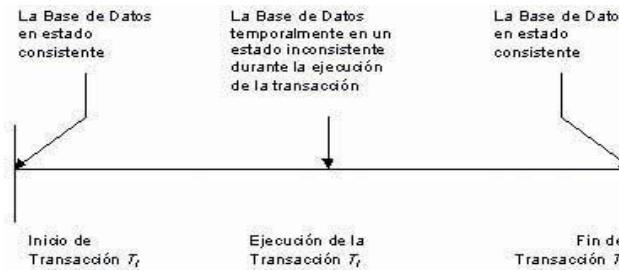
SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

[7]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia

CONSISTENCIA EN LA BASE DE DATOS

Durante la ejecución de una transacción, la base de datos puede estar temporalmente en un estado inconsistente. El punto importante aquí es asegurar que la base de datos regrese a un estado consistente al finalizar la ejecución de una transacción.



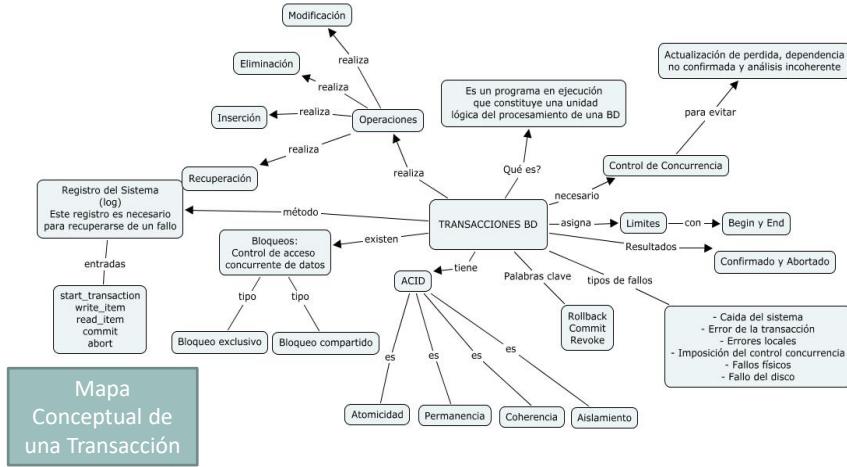
SISTEMAS DE BASE DE DATOS II
Por: Ing. Henry J. Lezcano

[8]

VI. Conceptos Básicos sobre Transacciones y Control de Concurrencia



MAPA CONCEPTUAL DE UNA TRANSACCION

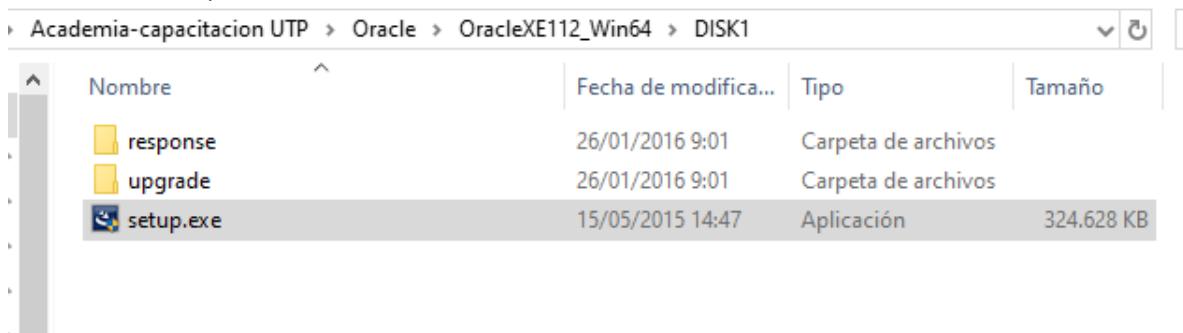


UNIVERSIDAD TECNOLOGICA DE PANAMA
FACULTAD DE INGENIERIA DE SISTEMAS COMPUTACIONALES
LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION

SISTEMAS DE BASE DE DATOS II

GUIA DE INTALACION DE LA HERRAMIENTA OracleXE112_Win64

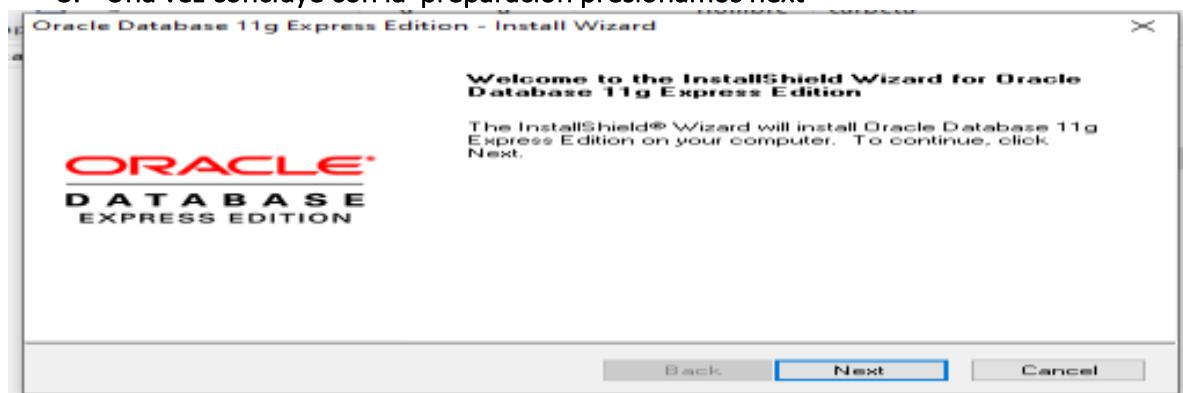
1. Descomprimir el archivo



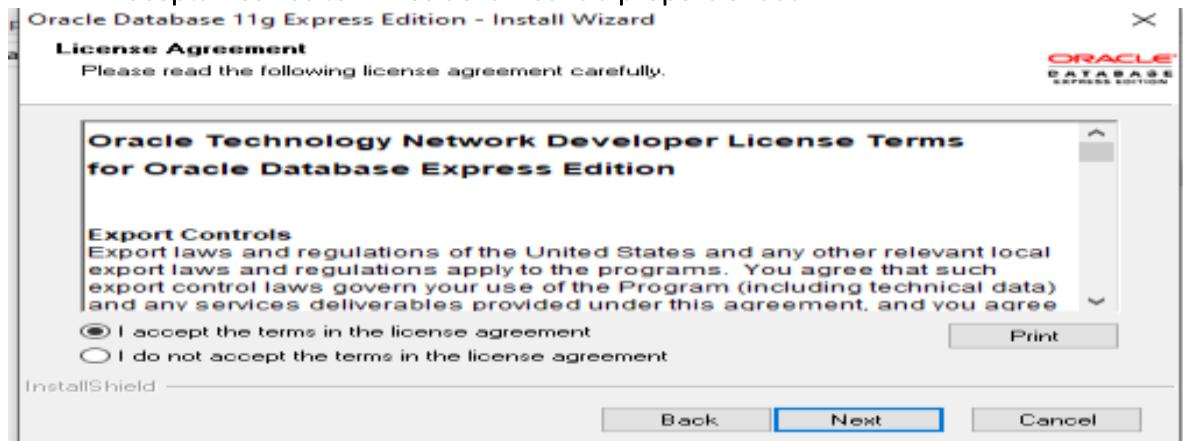
2. Iniciamos la instalación desde el archivo setup.exe



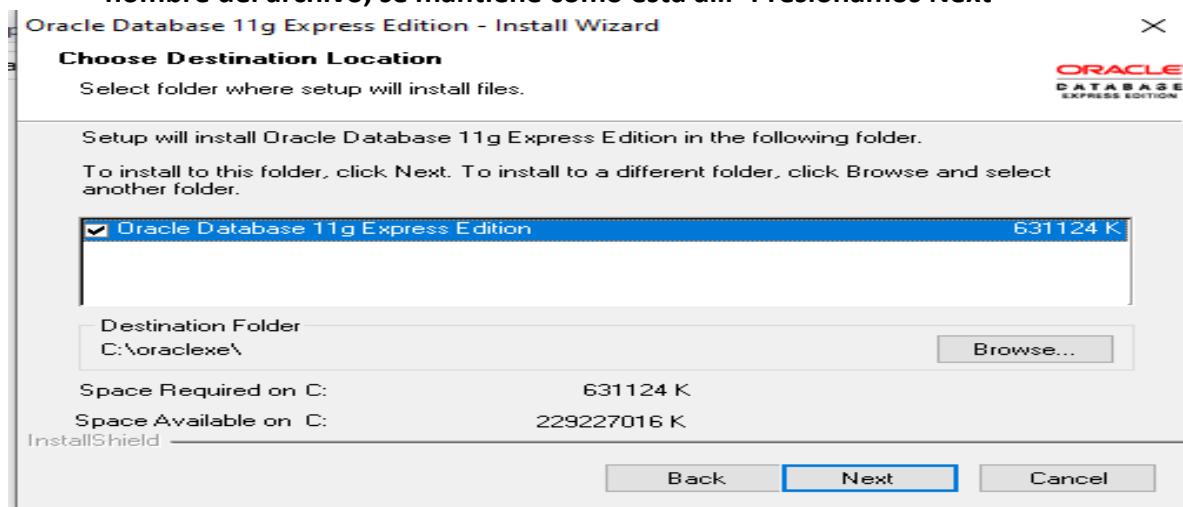
3. Una vez concluye con la preparación presionamos next



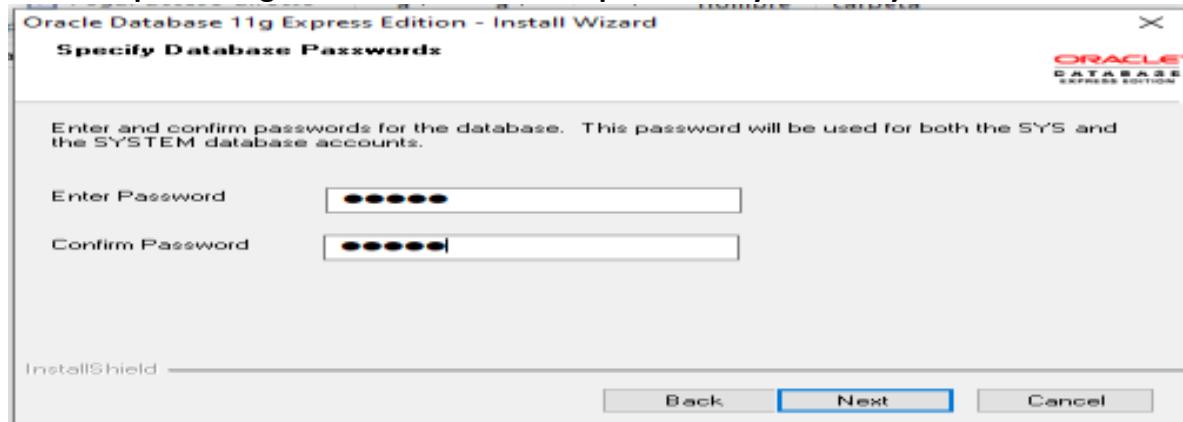
4. Y aceptamos los términos de la Licencia proporcionada



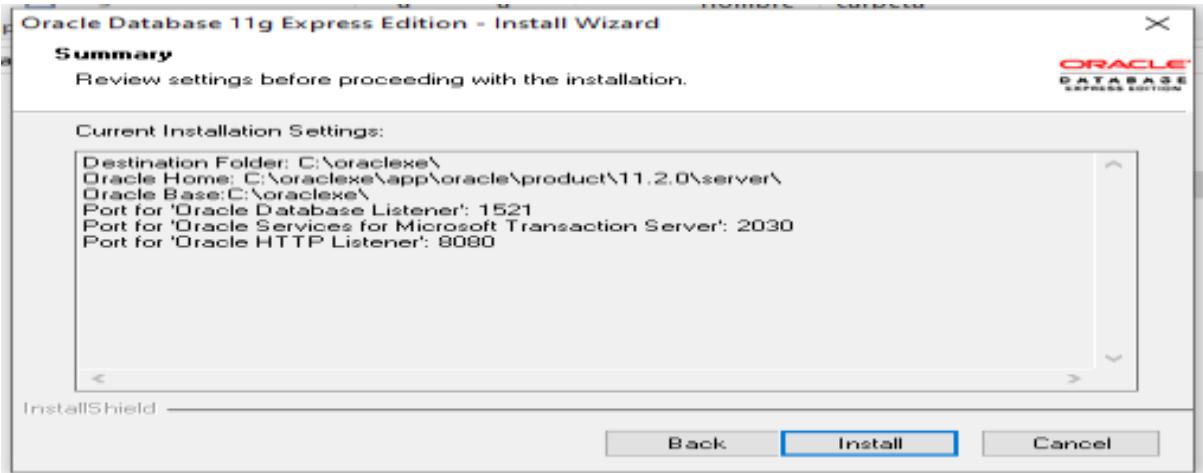
5. Se establece la herramienta a instalar mas el directorio donde será instalada y el nombre del archivo, se mantiene como esta allí- Presionamos Next



6. Aquí es obligatorio colocar una clave para el SYS y SYSTEM y damos Next



7. Aquí se procede a realizar la instalación de la herramienta. Se puede realizar configuraciones sobre los puertos próximamente de ser necesario. Presionamos **Install**



8. Se procede a la instalación de la herramienta

