

Timing del Curso 18h (3 días - 6h/día)

Día	Contenido	Día	Contenido
1	1.- Introducción a la BBDD Oracle. 2.- Recuperación de Datos (DQL) SELECT 3.- Restricción y Ordenación de Datos 4.- Uso de Funciones de Una Sola Fila para Personalizar la Salida	3	9.- Sentencias DML Insert Update Delete 10.- Sentencias DDL 11.- Otros Objetos de Esquema
2	5.- Funciones de Conversión y Expresiones 6.- Agregación de Datos con Funciones de Grupo 7.- Uso de JOINS 8.- Uso de Subconsultas		

01.- Introducción a la BBDD

- Lista de Funciones de Oracle Database 11g
- Descripción del Diseño Básico, Aspecto Teórico y Físico de una Base de Datos Relacional
- Clasificación de los Distintos Tipos de Sentencias SQL
- Descripción del Juego de Datos Utilizado en el Curso
- Conexión a la Base de Datos mediante el Entorno de SQL Developer
- Consultas Guardadas en Archivos y Uso de Archivos de Comandos en SQL Developer

Entorno y Funciones de un SGBD

SISTEMAS DE INFORMACION

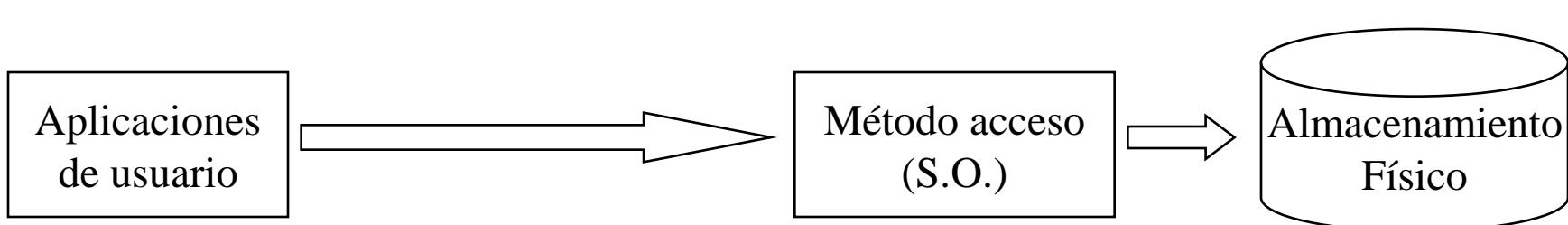
Compuesto de

- datos
- programa de gestión

Objetivos

- no redundancia
- independencia de datos
- interconectividad
- protección (física y lógica)
- accesibilidad en tiempo real

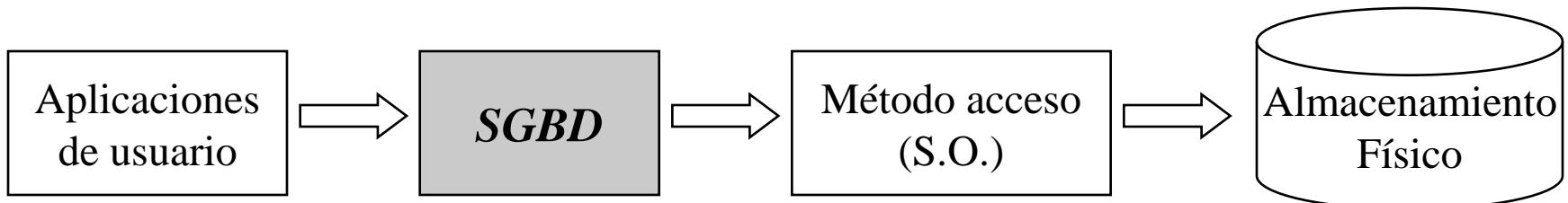
Son responsabilidad del desarrollador



Entorno y Funciones de un SGBD

SGBD compuesto de procesos background que

- Reciben peticiones de acceso a datos físicos desde aplicaciones de usuario
- Realizan operaciones sobre los datos a través de accesos proporcionados por el SO
- Devuelven datos



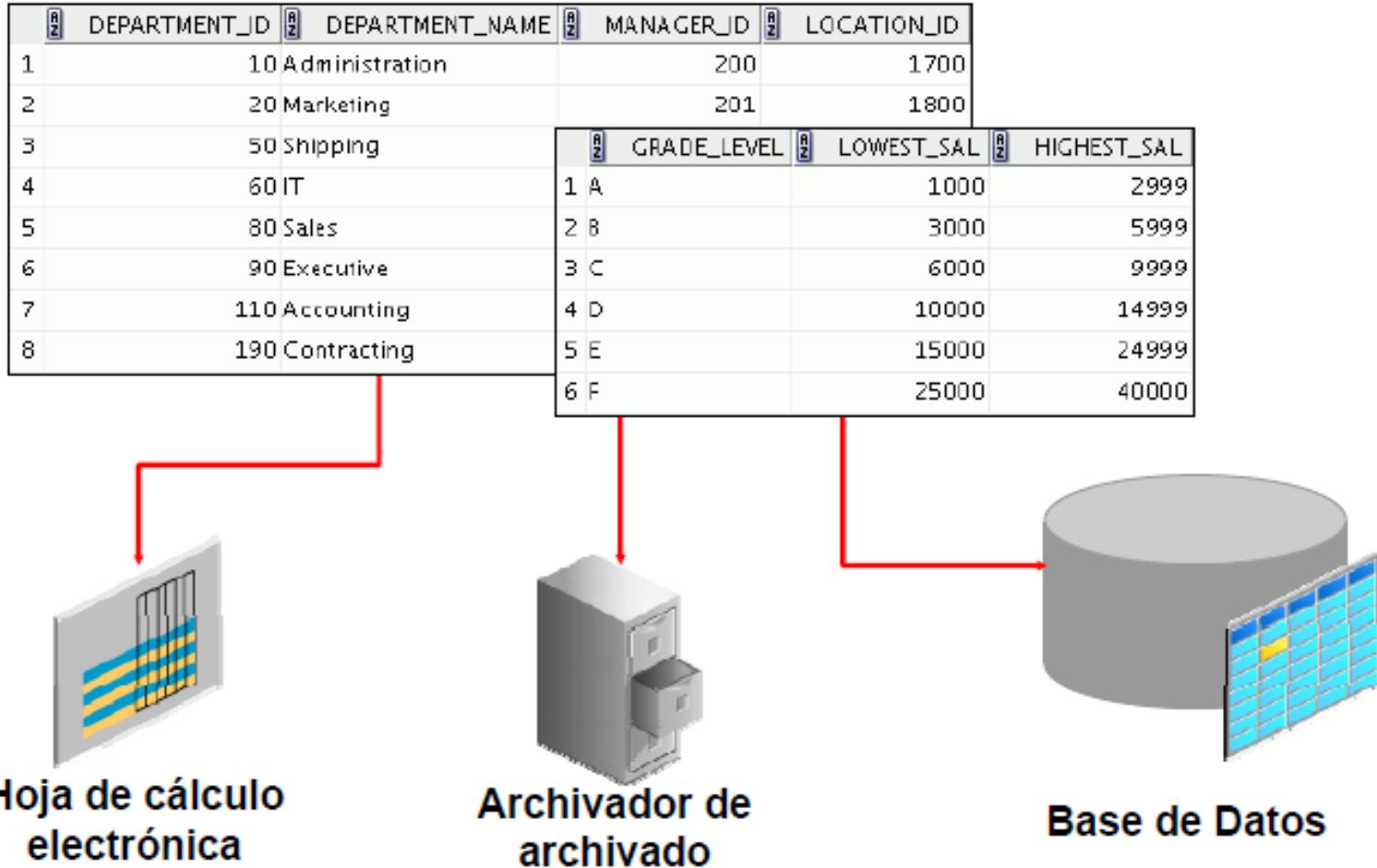
Tipos de Bases de Datos

- Colección de datos organizados e integrados, almacenados en soporte de datos y accesible para el usuario
 - Un SGBD debe proporcionar a los usuarios una visión abstracta de la información

Modelos de Gestores de Bases de Datos

- **Jerárquico**
 - ✓ Los datos se representan mediante registros. Un registro no puede tener más de un parente (IMS/1 de IBM)
- **En red** (1.968)
 - ✓ Como el modelo jerárquico, pero un registro puede tener más de un parente. Grafo dirigido (ADABAS)
- **Relacional** (1.980)
 - ✓ Nace como modelo teórico a principios de los 70. Basado en el álgebra y la teoría de conjuntos, representa los datos y sus relaciones mediante tablas (ORACLE)

Almacenamiento de Datos en Diferentes Medios Físicos

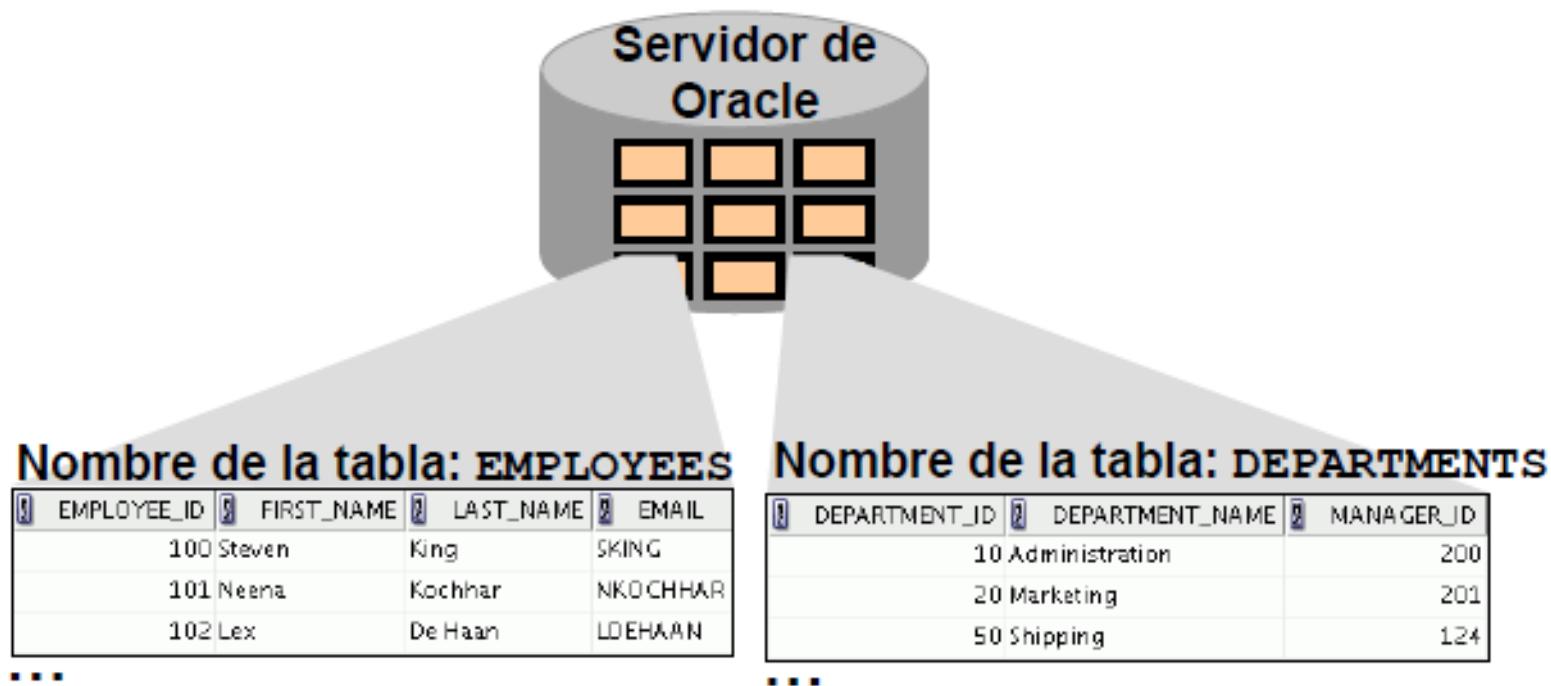


Concepto de Base de Datos Relacional

- El Dr. E. F. Codd propuso el modelo relacional del sistema de bases de datos en 1970.
- Es la base del sistema de gestión de bases de datos relacionales (RDBMS).
- El modelo relacional consta de lo siguiente:
 - Recopilación de objetos o relaciones
 - Juego de operadores que actúan en las relaciones
 - Integridad de datos para su precisión y consistencia

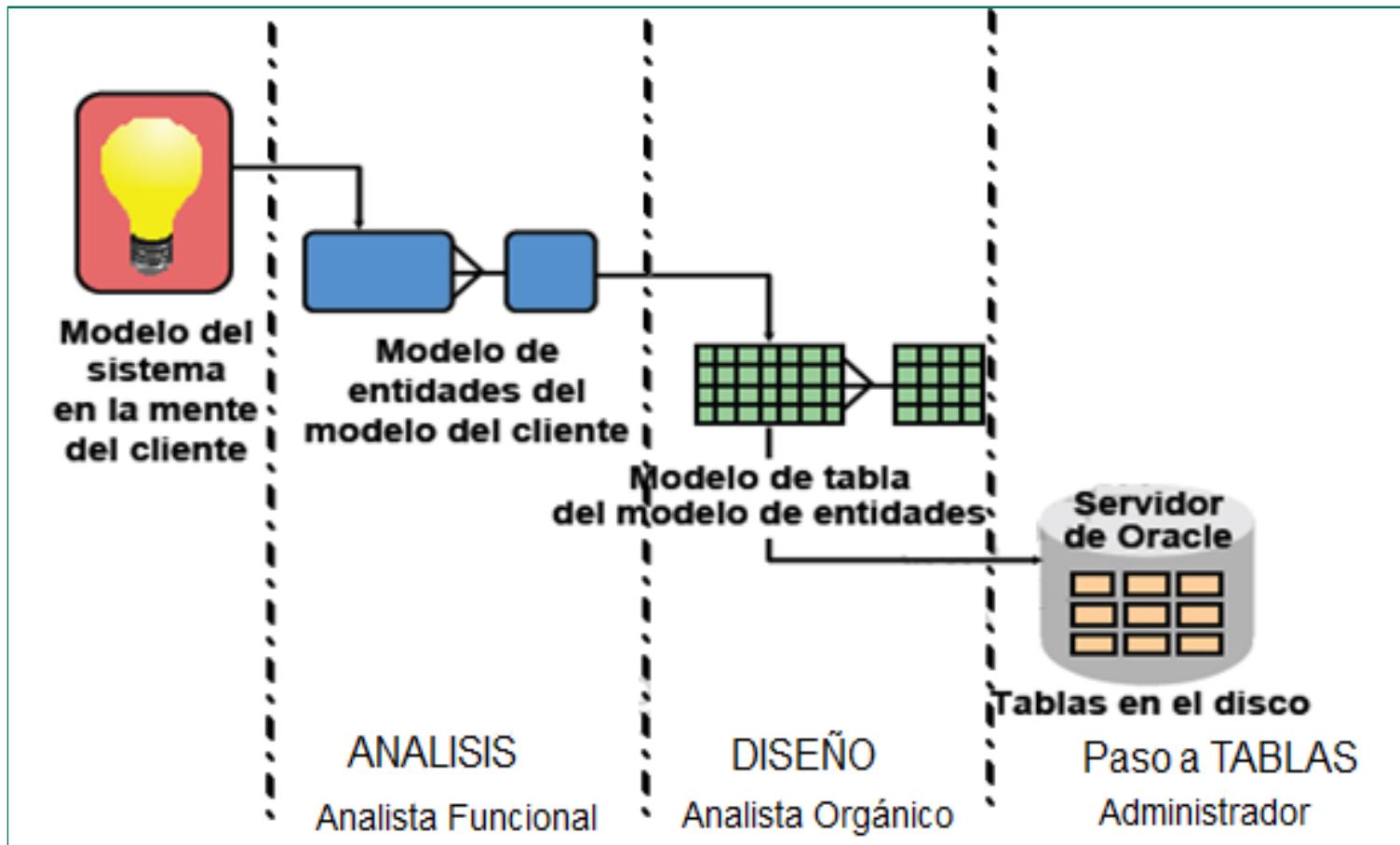
Definición de una Base de Datos Relacional

- Una base de datos relacional es una recopilación de relaciones o tablas bidimensionales.



Tipos de Bases de Datos

- Para poder realizar de forma correcta una Aplicación, se deben de realizar una serie de pasos (MODELIZACION)



Relación de Varias Tablas

- Cada fila de datos de una tabla se identifica como única mediante una clave primaria.
- Puede relacionar de forma lógica desde varias tablas mediante claves ajena.

Nombre de la tabla: DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	(null)	1700

Nombre de la tabla: EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
103	Alexander	Hunold	60
104	Bruce	Ernst	60
107	Diana	Lorentz	60
124	Kevin	Mourgos	50
141	Trenna	Rajs	50
142	Curtis	Davies	50

Clave primaria

Clave ajena

Clave primaria

Terminología de Bases de Datos Relacionales

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	90
101	Neena	Kochhar	17000	(null)	90
102	Lex	De Haan	17000	(null)	90
103	Alexander	Hunold	9000	(null)	60
104	Bruce	Ernst	6000	(null)	60
107	Diana	Lorentz	4200	(null)	60
124	Kevin	Mourgos	5800	(null)	50
141	Trenna	Rajs	3500	(null)	50
142	Curtis	Davies	3100	(null)	50
143	Randall	Matos	2600	(null)	50
144	Peter	Vargas	2500	(null)	50
149	Eleni	Zlotkey	10500	0.2	80
174	Ellen	Abel	11000	0.3	80
176	Jonathon	Taylor	8600	0.1	80
178	Kimberely	Grant	7000	0.15	(null)
200	Jennifer	Whalen	4400	(null)	10
201	Michael	Hartstein	13000	(null)	20
202	Pat	Fay	6000	(null)	20
205	Shelley	Higgins	12000	(null)	110
206	William	Gietz	8300	(null)	110

1.- Fila

2.- Columna Clave

3.- Columna NO Clave

4.- Columna FK

5.- Campo

6.- Campo NULL

Oracle y Las Bases de Datos

- Oracle 11g ofrece una serie de funcionalidades en diferentes áreas:



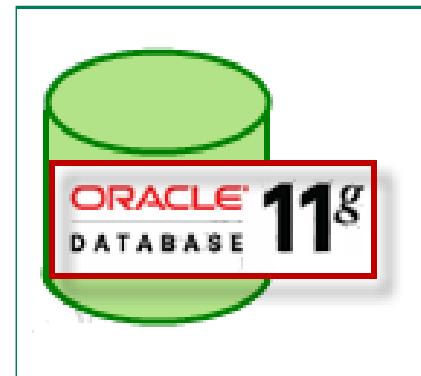
- **Grids de Infraestructura**
 - ✓ Almacenamiento de información, alta disponibilidad y Rendimiento.
- **Gestión de Información**
 - ✓ Gestión de contenidos, integración de información y ciclo de vida de la misma.
- **Desarrollo de aplicaciones**
 - ✓ Capacidad para utilizar y gestionar todos los entornos de desarrollo como PLSQL, Java/JDBC, .NET, PHP, SQL Developer, etc

Oracle y Las Bases de Datos

- Oracle DATABASE 11g es una Base de Datos Relacional, con los últimas funcionalidades del sector.
 - Soporta Teras, Peta, Exa bits de información.
 - Sistema fiable y de recuperación rápida ante cualquier tipo de fallo.
 - Gestión de GRIDS de infraestructura de forma sencilla.
 - Alta disponibilidad y Rendimiento
 - Seguridad en datos, cifrado, enmascaramientos, etc.

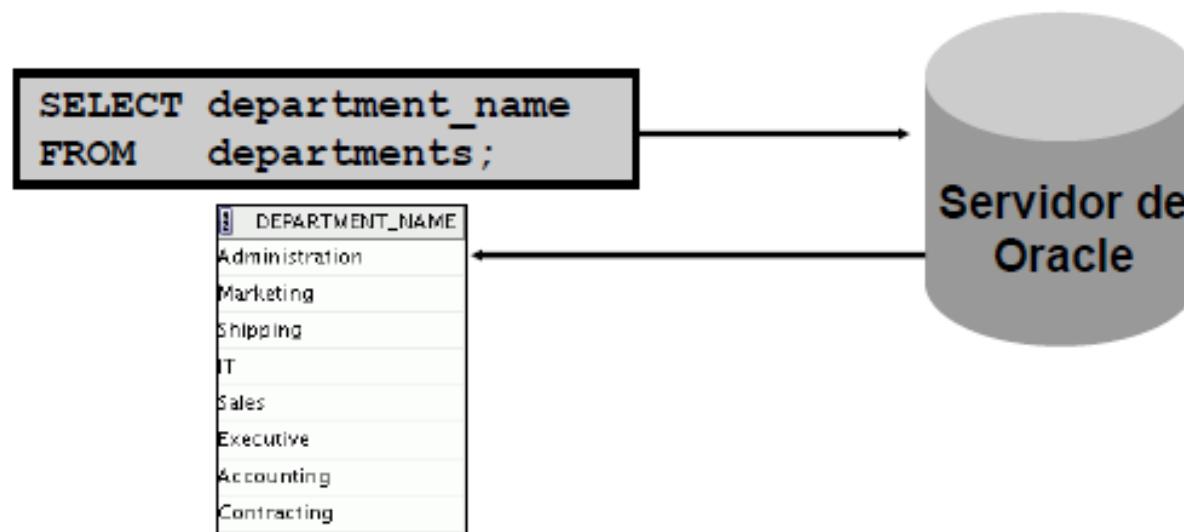
Oracle y Las Bases de Datos

- El SGBD Oracle DATABASE 11g soporta el modelo relacional y el relacional de objeto.
- Dentro del modelo relacional dispone de la posibilidad de creaciones de nuevos tipos de datos complejos
- Soporta objetos grandes y multimedia.
- Incluye funcionalidades para mejorar el rendimiento de Base de datos OLTP y DSS.



Uso de SQL para Consultar Base de Datos

- El lenguaje de consulta estructurado (SQL) es:
 - Lenguaje estándar de ANSI para el funcionamiento de bases de datos relacionales
 - Uso y aprendizaje sencillos y eficaces
 - Funcionalidad completa (con SQL, puede definir, recuperar y manipular datos en las tablas)



Sentencias SQL

SELECT
INSERT
UPDATE
DELETE
MERGE

Lenguaje de Manipulación de Datos (DML)

CREATE
ALTER
DROP
RENAME
TRUNCATE
COMMENT

Lenguaje de Definición de Datos (DDL)

GRANT
REVOKE

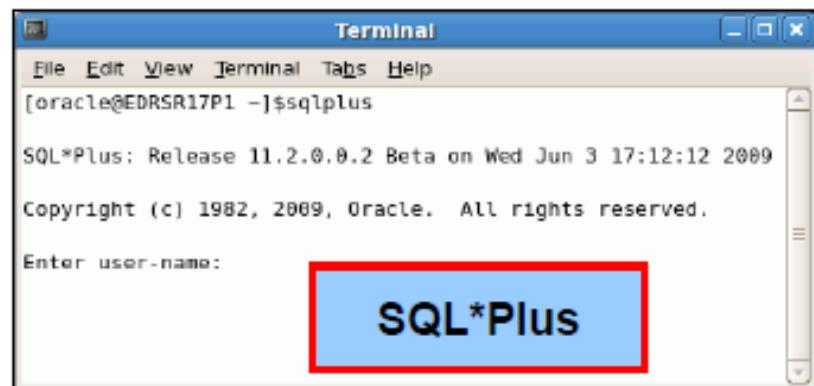
Lenguaje de Control de Datos (DCL)

COMMIT
ROLLBACK
SAVEPOINT

Control de Transacciones

Entornos de Desarrollo para SQL

- Existen dos entornos de desarrollo para este curso:
 - La herramienta principal es Oracle SQL Developer.
 - También se puede utilizar la interfaz de línea de comandos SQL*Plus.



02.- Recuperación de Datos: SELECT

- Enumeración de las Capacidades de las Sentencias SQL SELECT
- Generación de un Informe de Datos a partir de la Salida de una Sentencia SELECT Básica
- Selección de Todas las Columnas
- Selección de Columnas Concretas
- Uso de Valores por Defecto de Cabeceras de Columna
- Uso de Operadores Aritméticos
- Descripción de Prioridad de Operadores

LENGUAJE DE CONSULTA DE DATOS (DQL)

- Se pueden consultar tablas y vistas
- La sentencia utilizada para la elaboración de consultas es la SELECT
- La sentencia SELECT permite realizar las siguientes operaciones:
 - Recuperar toda la información asociada a una tabla
 - Recuperar todas las columnas de una tabla
 - Recuperar sólo las columnas que se especifiquen
 - Recuperar una selección de filas
 - Controlar el orden de salida de las filas en las consultas
 - Obtener en el momento de la consulta columnas productos de operaciones con otras columnas, constantes, etc.
- Las consultas se realizan por asociación de valores, no por la localización de los mismos

SINTAXIS GENERAL DE LA SENTENCIA SELECT

SELECT (columna, ...)	ALIAS	el qué
FROM tabla, ...		de dónde
WHERE condiciones		bajo qué condiciones
GROUP BY valores		condición de agrupamiento
HAVING condiciones		filtra los grupos
ORDER BY columna		ordena la salida

- Solo son obligatorias las cláusulas SELECT y FROM

SINTAXIS GENERAL DE LA SENTENCIA SELECT

SELECT

- ✓ Se indican las columnas a consultar. Si se quiere consultar todas las columnas de una tabla se pone un '*'.

FROM

- ✓ Se indica la tabla o tablas de donde se va a realizar la consulta.

WHERE

- ✓ Se indican las condiciones para la realización de la consulta.

GROUP BY

- ✓ Agrupaciones por determinadas columnas.

HAVING

- ✓ Se indican las condiciones para la realización de las agrupaciones.

ORDER BY

- ✓ Se indican las columnas por las cuales se va a realizar la ordenación de la consulta.

CLÁUSULA SELECT

- SELECT sirve para seleccionar ciertas columnas o valores derivados de ellas, y puede incluir:
 - Constantes numéricas, alfanuméricas y variables
 - Valores de columnas con el formato [nombre de tabla.] nombre de columna
 - Expresiones con nombres de columnas y constantes
 - Todo tipo de funciones aplicadas a columnas de una tabla y/o variables
 - Permite la especificación del título de la columna que aparecerá en la cabecera
- FROM sirve para definir las tablas donde se va a ir a buscar las columnas especificadas en la sentencia SELECT. Se puede asociar un nuevo nombre a las tablas para facilitar la consulta (sinónimo temporal)

DISTINCT / ALL

- La función que tiene esta cláusula es la de eliminar las filas repetidas que se produzcan en una sentencia SELECT

Ejemplo

```
SELECT DISTINCT COD_CAT  
FROM EMP;
```

CLÁUSULA WHERE

- Si la cláusula WHERE está presente en una consulta, sirve para especificar qué filas de las tablas se desean obtener como resultado de la consulta.
- Para determinar las condiciones de selección de filas intervienen tres elementos:
 - Nombre de la columna
 - Operador de comparación
 - Nombre de columna, constante, lista de valores

SINTAXIS

SELECT columnas

FROM tabla,...

WHERE condición (o condiciones) de selección de filas

OPERADORES

DIVISION DE OPERADORES

- Operadores de expresión
 - ✓ Aritméticos
 - ✓ Para fechas
 - ✓ De cadena de caracteres
- Operadores de comparación
 - ✓ Generales
 - ✓ De cadenas de caracteres
 - ✓ Lógicos

OPERADORES EXPRESION

ARITMÉTICOS

- Utilizables para formar expresiones con constantes, valores de columnas y funciones de valores de columnas
 - + Suma
 - * Multiplica
 - Resta
 - / Divide

PARA FECHAS

- Fecha + N Suma un número N de días a una fecha
- Fecha – N Resta un número N de días a una fecha
- Fecha – Fecha Da como resultado un número de días

PARA CADENAS DE CARACTERES

- Existe el operador de concatenación
 - ✓ `cadena_1 || cadena_2`

OPERADORES

GENERALES DE COMPARACIÓN

=	IGUAL	>=	MAYOR ó IGUAL
!=,<>,^=	NO IGUAL	<	MENOR QUE
>	MAYOR QUE	<=	MENOR ó IGUAL

DE CADENAS DE CARACTERES

- LIKE permite los siguientes caracteres especiales en las cadenas de comparación:
 - '%' Cualquier cadena de cero o más caracteres
 - '_' Cualquier carácter
- Las mayúsculas y minúsculas son significativas, esto es, 'm' es distinto de 'M', y las constantes alfanuméricas deben siempre encerrarse entre comillas simples ('XXXXXX')
- Usar ESCAPE para buscar un carácter especial
 - ✓ ... Where nombre LIKE '%AF\$_E%' ESCAPE '\$'

OPERADORES LÓGICOS

[NOT] BETWEEN valor_1 AND valor_2

- Si tiene un valor comprendido entre los valores la fila es seleccionada.

[NOT] IN (valor_1,valor_2,....,valor_n)

- Si la fila tiene un valor igual alguno de los valores la fila es seleccionada

IS [NOT] NULL

- Si la fila tiene valor null la fila es seleccionada

= ANY

- Es equivalente a IN

!= ALL

- Es equivalente a NOT IN

REGLAS DE PRIORIDAD

Operador	Significado
1	Operadores aritméticos
2	Operador de concatenación
3	Condiciones de comparación
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Distinto de
7	Condición lógica NOT
8	Condición lógica AND
9	Condición lógica OR

Puede utilizar los paréntesis para sustituir las reglas de prioridad.

REGLAS DE PRIORIDAD

```
SELECT last_name, job_id, salary  
FROM employees  
WHERE job_id = 'SA_REP'  
OR [ ] job_id = 'AD_PRES'  
AND [ ] salary > 15000;
```

1

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	6600
4	Grant	SA_REP	7000

```
SELECT last_name, job_id, salary  
FROM employees  
WHERE job_id = 'SA_REP'  
OR [ ] job_id = 'AD_PRES'  
AND [ ] salary > 15000;
```

2

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

03.- Restricción y Ordenación de Datos

- Escritura de Consultas que Contienen una Cláusula WHERE para Limitar la Salida Recuperada
- Descripción de las Reglas de Prioridad para Operadores de Comparación y Lógicos
- Uso de Literales de Cadena de Caracteres en la Cláusula WHERE
- Escritura de Consultas que Contienen una Cláusula ORDER BY para Ordenar la Salida de una Sentencia SELECT
- Ordenación de Salida de Forma Descendente y Ascendente

CLÁUSULA ORDER BY

- Sirve para fijar el orden de salida de las filas seleccionadas en una sentencia SELECT
- Permite ordenar las filas utilizando los criterios siguientes:
 - Orden ascendente (valor por defecto)
 - Orden descendente (DESC)
 - Por múltiples columnas (la columna más a la izquierda es por la que primero se clasifica)
 - Con valores nulos
- Debe ser la última cláusula de la sentencia SELECT
- Cuando no se usa, el orden de salida de las filas no está definido

NOTA: Cuando se quiere ordenar por columnas calculadas se indica en la orden ORDER BY el número de columna de la selección.

- ✓ Select nombre, Salario / 12
- ✓ Order by 2.

SUSTITUCION DE VARIABLES

- Hasta ahora las sentencias SQL se han ejecutado con columnas y condiciones predeterminadas y sus valores.
- Puede editar la cláusula WHERE para proporcionar un valor diferente cada vez que ejecute el comando, pero existe también una forma más sencilla.
- Si se utiliza una variable de sustitución en lugar de los valores exactos en la cláusula WHERE, puede ejecutar la misma consulta para diferentes valores.



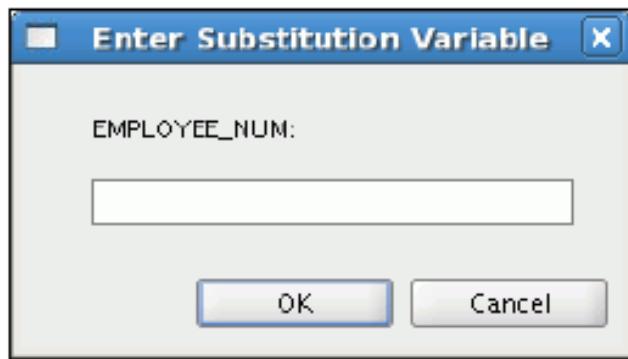
SUSTITUCION DE VARIABLES

- Utilizar variables de sustitución para:
 - Almacenar valores temporalmente con una sustitución de un solo ampersand (&) y de dos ampersands (&&)
- Utilizar las variables de sustitución para complementar:
 - Condiciones WHERE
 - Cláusulas ORDER BY
 - Expresiones de columna
 - Nombres de tabla
 - Sentencias SELECT completas

Uso de la Variable de Sustitución

- Un Solo Ampersand
 - Utilizar una variable prefijada con un ampersand (&) para solicitar al usuario un valor:

```
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num ;
```



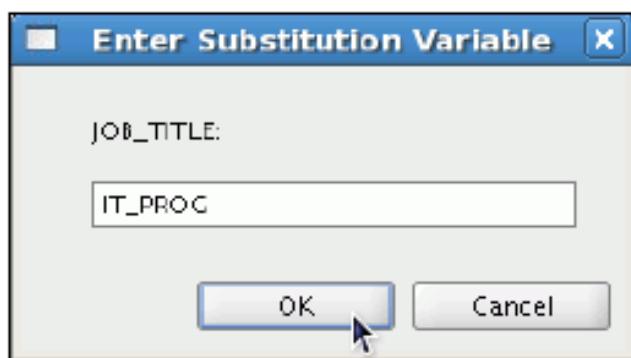
The screenshot shows the same dialog box again, but this time the "EMPLOYEE_NUM:" field contains the value "101". The "OK" button is highlighted with a red border, and a mouse cursor is positioned over it. Below the dialog, a table displays employee information:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

Uso de la Variable de Sustitución

- Valores de Fecha y Carácter
 - Utilizar las comillas simples para los valores de fecha y carácter

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

Uso de la Variable de Sustitución

- Nombres de Columna, Expresiones y Texto

```
SELECT employee_id, last_name, job_id, &column_name  
FROM employees  
WHERE &condition  
ORDER BY &order_column ;
```

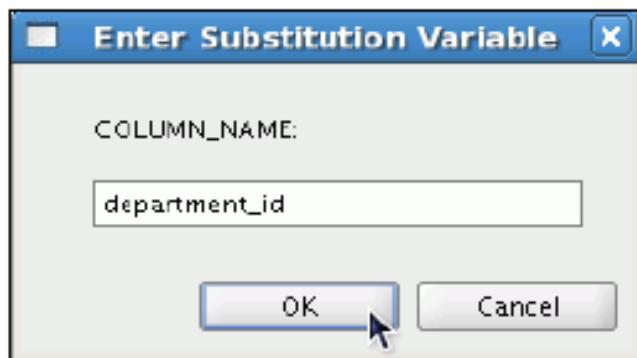
The image shows three overlapping dialog boxes titled "Enter Substitution Variable".

- The first dialog (leftmost) has a "COLUMN_NAME:" label and a text input field containing "salary". An "OK" button is at the bottom.
- The second dialog (middle) has a "CONDITION:" label and a text input field containing "salary >15000". An "OK" button is at the bottom.
- The third dialog (rightmost) has an "ORDER_COLUMN:" label and a text input field containing "last_name". It has "OK" and "Cancel" buttons at the bottom.

Uso de la Variable de Sustitución

- DOS Ampersand
 - Usar dos ampersands (&&) si se desea reutilizar el valor de la variable sin preguntar siempre al usuario:

```
SELECT      employee_id, last_name, job_id, &&column_name  
FROM        employees  
ORDER BY    &column_name ;
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20
...				

Uso del Comando DEFINE

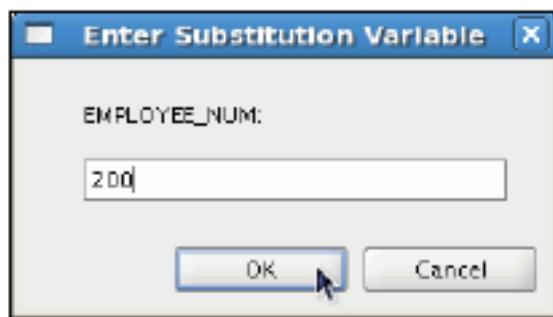
- Usar el comando DEFINE para crear y asignar un valor a una variable.
- Usar el comando UNDEFINE de iSQL*Plus para eliminar una variable.

```
DEFINE employee_num = 200
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num;
UNDEFINE employee_num
```

Uso del Comando VERIFY

- Usar el comando VERIFY para cambiar la visualización de la variable de sustitución, antes y después de que SQL Developer sustituya las variables de sustitución con los valores

```
SET VERIFY ON  
SELECT employee_id, last_name, salary  
FROM employees  
WHERE employee_id = &employee_num;
```



The SQL Developer interface shows the results of the executed query. The 'Script Output' tab is selected. The query is displayed as:

```
SELECT employee_id, last_name, salary  
FROM employees  
WHERE employee_id = 200
```

The results table shows one row selected:

EMPLOYEE_ID	LAST_NAME	SALARY
200	Whalen	4400

Below the table, the message '1 rows selected' is displayed.

Ejercicios

Ejercicios propuestos:

1. Seleccionar todos los empleados cuyo salario esté comprendido entre 1500000 y 4000000.
2. Obtener un listado de todos los sueldos diferentes que hay en la compañía.
3. Obtener una lista con los empleados ordenada por departamento y dentro de él por número de personal.
4. Se pide una lista que recupere el salario de todos los empleados cuyo código de trabajo empiece por 6.

04.- Funciones de Grupo

FUNCIONES DE CONJUNTOS

- **AVG (n)**
 - ✓ Valor medio de n (ignorando los valores nulos)
 - ✓ Sólo columnas numéricas.
- **SUM (n)**
 - ✓ Suma de los valores de n
 - ✓ Sólo columnas numéricas.
- **MAX (expr)**
 - ✓ Máximo valor de "expr"
- **MIN (expr)**
 - ✓ Mínimo valor de "expr"
- **COUNT(expr)**
 - ✓ Número de veces que "expr" evalúa dato con valor no nulo.

PARTICULARIDADES DE LAS FUNCIONES DE CONJUNTOS

- Los valores nulos no participan en el cálculo de las funciones de conjuntos
 - SELECT AVG(SALARIO) y SELECT SUM(SALARIO) / COUNT(*) no siempre tendrán el mismo resultado
- En la cláusula SELECT sólo pueden aparecer funciones de conjuntos y columnas incluidas en la cláusula GROUP BY

CLÁUSULA GROUP BY

- Sirve para crear grupos de filas y muestra una sola fila por cada grupo de la SELECT
- Debe de aparecer el mismo número de columnas en la clausula GROUP BY que en la sentencia SELECT (No tenemos en cuenta las columnas calculadas).

SELECT A, B, AVG(SAL)

.....

GROUP BY A,B

RESTRICCIONES

- No se puede agrupar por columnas calculadas.
- El NULL lo toma como si fuese un valor mas.
- No se puede introducir dentro de una SUBSELECT.

CLÁUSULA HAVING

- La cláusula HAVING se emplea para controlar cuál o cuáles de los conjuntos agrupados por la cláusula GROUP BY se visualizan
- La evaluación de las cláusulas de la sentencia SELECT en tiempo de ejecución se efectúa en el siguiente orden:
 - ✓ WHERE Filtra filas
 - ✓ GROUP BY Crea una tabla de grupos nueva
 - ✓ HAVING Filtra grupos
 - ✓ ORDER BY Clasifica la salida

Ejercicios

Ejercicios propuestos:

1. Calcular el número de empleados que hay en cada uno de los departamentos.
2. Se pide una lista que recupere el salario medio de cada departamento junto con el número de empleados que tiene. El resultado no debe incluir empleados que tengan un código de trabajo mayor que 54, ni departamentos con menos de tres empleados. Se quiere ordenada por número de departamento.
3. Consiga una lista de los empleados masculinos de los departamentos que comiencen por 'D' y 'E', que hayan nacido antes de Junio de 1954. La lista, ordenada por fecha de nacimiento, debe contener el número de personal y apellido, el código de trabajo, el nivel de educación y el salario de cada empleado.

05.- Funciones de Una Sola Fila

- Descripción de las Diferencias entre Funciones de Una y Varias Filas
- Manipulación de Cadenas con una Función de Carácter en las Cláusulas SELECT y WHERE
- Manipulación de Números en las Funciones ROUND, TRUNC y MOD
- Realización de Operaciones Aritméticas con Datos de Fecha
- Manipulación de Fechas con las Funciones DATE

FUNCIONES ARITMÉTICAS

- **ABS(n)**
 - ✓ Valor absoluto de n
- **CEIL (n)**
 - ✓ Entero inmediatamente superior a n (n no entero)
- **FLOOR (n)**
 - ✓ Entero inmediatamente inferior a n (n no entero)
- **MOD (valor,divisor)**
 - ✓ Divide un valor por un divisor obteniendo el resto
- **SIGN (n)**
 - ✓ Si $n < 0$, -1; Si $n = 0$, 0; Si $n > 0$, 1

FUNCIONES ARITMÉTICAS

- **ROUND(valor[,precisión])**
 - ✓ Redondea valor con la precisión indicada
- **TRUNC(valor[,precisión])**
 - ✓ Trunca valor a los decimales indicados por precisión
- **POWER(valor,exponente)**
 - ✓ Eleva valor al exponente indicado
- **SQRT (n)**
 - ✓ Raíz cuadrada de n, si n < 0 NULL

FUNCIONES DE CADENA DE CARACTERES

- **ASCII(s)**
 - ✓ Valor ASCII del primer carácter de la cadena "n"
- **CHR(n)**
 - ✓ Devuelve el carácter cuyo valor ASCII es el numero "n"
- **INITCAP(s)**
 - ✓ Pone en mayúsculas la primera letra de cada palabra de la cadena "n"
- **LOWER(s)**
 - ✓ Cadena "s" con todas sus letras convertidas en minúsculas
- **UPPER (s)**
 - ✓ Cadena "s" con todas sus letras en mayúsculas
- **LENGTH(s)**
 - ✓ El numero de caracteres de "s"

FUNCIONES DE CADENA DE CARACTERES

- LPAD (S1,N[,S2]
 - ✓ S1 visualizado con longitud "n" y justificando a la derecha. "S2" es la cadena con la que se rellena por la izquierda
- RPAD (S1,N[,S2]
 - ✓ Similar al LPAD pero llenando por la derecha
- RTRIM (s)
 - ✓ Suprime los blancos que contenga la cadena "s" a la derecha
- LTRIM (s)
 - ✓ Suprime los blancos que contenga la cadena "s" a la izquierda
- SUBSTR (s,m[,n])
 - ✓ Devuelve la subcadena de "s" que empieza con el carácter número "m" y tiene "n" caracteres de longitud.

06.- Funciones de Conversión

- Descripción de la Conversión de Tipo de Dato Implícito y Explícito
- Uso de las Funciones de Conversión TO_CHAR, TO_NUMBER y TO_DATE
- Anidamiento de Varias Funciones
- Aplicación de las Funciones NVL, NULLIF y COALESCE a Datos
- Uso de la Lógica Condicional IF THEN ELSE en una Sentencia SELECT

FUNCIONES DE CONVERSION

- **TO_NUMBER (s)**
 - ✓ Convierte una cadena alfanumérica que sólo contiene números, a un valor numérico
- **TO_CHAR (d,[,fmt])**
 - ✓ Convierte una fecha "d" a un valor de tipo VARCHAR2 en el formato especificado por fmt. Si se omite fmt, "d" se convierte al formato de fecha estándar
- **TO_DATE (s,[,fmt])**
 - ✓ Convierte una cadena "s" en una fecha con el formato especificado por fmt. Si se omite fmt, "s" debe tener el formato de fecha estándar 'dd-mon-yy'
- TO_CHAR y TO_DATE utilizan máscaras de formato
- Sufijos de máscaras de formato
 - ✓ th Cardinal del número (st,nd,rd,th) ejm:(4th)
 - ✓ sp Escribe el texto correspondiente al valor de un número
 - ✓ sph ó thsp Combina las dos acciones anteriores.

MÁSCARAS DE FORMATO PARA FECHAS

✓ Cc ó scc	Valor del siglo
✓ y,yyy ó sy,yyy	Año con coma, con o sin signo
✓ yyyy	Año sin signo
✓ yyy	Ultimos tres dígitos del año
✓ yy	Ultimos dos dígitos del año
✓ y	Ultimo dígito del año
✓ q	Número de trimestre
✓ ww	Número de semana del año
✓ w	Número de semana del mes
✓ mm	Número del mes
✓ ddd	Número del día del año
✓ dd	Número del día del mes
✓ d	Número del día de la semana

MÁSCARAS DE FORMATO PARA FECHAS

✓ hh ó hh12	Hora (1-12)
✓ hh24	Hora (1-24)
✓ mi	Minutos
✓ ss	Segundos
✓ ssss	Segundos transcurridos desde media noche
✓ j	Juliano
✓ syear o year	Año en inglés (ej: nineteen-eighty-two)
✓ month	Nombre del mes
✓ mon	Abreviatura de tres letras del nombre del mes
✓ day	Nombre del día de la semana
✓ dy	Abreviatura de tres letras del nombre del día
✓ a.m. ó p.m.	Ante-meridiem o post-meridiem
✓ b.c. ó a.d.	Indicador para el año (antes o después de cristo)

FUNCIONES DE FECHAS

- ADD_MONTHS (d,n)
 - ✓ Suma n meses a la fecha d
- LAST_DAY (d)
 - ✓ Fecha del último día del mes contenido en "d"
- MONTHS_BETWEEN (d1,d2)
 - ✓ Diferencia en meses entre la fecha "d1" la "d2". El resultado puede ser negativo y aparecer decimales.
- NEXT_DAY (d,day)
 - ✓ Devuelve la fecha del día de la semana siguiente identificado por day
- SYSDATE
 - ✓ Devuelve el día de hoy

OTRAS FUNCIONES

- DECODE
 - ✓ (var,val1,cod1,val2,cod2, ...,valor_por_defecto)
 - Si var es igual a cualquier valor de la lista devuelve el correspondiente código; en caso contrario se obtiene el valor por defecto
 - val debe ser un dato del mismo tipo que var
- GREATEST (expr,expr,...)
 - ✓ Devuelve el mayor valor de la lista
- LEAST (expr,expr,...)
 - ✓ Devuelve el menor valor de la lista
- NVL (x,expr)
 - ✓ Si x es NULL, devuelve expr; si no, devuelve x. "x" y "expr" puede ser de cualquier tipo.
El tipo de valor resultante es el mismo que el de x
 - ✓ Se utiliza para evitar los valores nulos en expresiones aritméticas y funciones (los valores nulos en las expresiones siempre darán como resultado un valor nulo)
- USER
 - ✓ Devuelve el nombre del usuario.

FUNCIONES SQL99

- **NULLIF(exp1, exp2)**
 - ✓ Devuelve NULL si exp1=exp2 sino devuelve exp1
- **COALESCE(exp1, exp2, exp3,...)**
 - ✓ Devuelve el valor de la primera expresión que sea distinta de NULL
- **(CASE exp ... WHEN Valor.....END)**
 - ✓ Evalúa la exp comprando con cada una de los valores puestos en la cláusula WHEN
 - ✓ Se utiliza como sustituto y ampliación de DECODE
 - ✓ MODO escritura
 - (CASE exp
 - WHEN valor1 THEN salida1
 - WHEN valor2 THEN salida2
 - ELSE salida3
 - END)
 - MODO BUSQUEDA
 - (CASE
 - WHEN exp < valor THEN salida1
 - WHEN exp > valor THEN salida2
 - ELSE salida3
 - END)

Ejercicios

Ejercicios propuestos:

1. Calcular el Valor Absoluto del número entero mayor o igual a la potencia de -17 de 1000000
2. El salario bruto de un empleado es de 3000545. Por Navidad recibirá una paga extraordinaria igual al producto de dividir su salario bruto entre 14 pagas. Si tiene 12 pagas, ¿Cuanto cobrará por Navidad? (nómina + paga extra). El resultado se redondeará a un sólo decimal.
3. Calcular la raíz cuadrada de 284857566. Se desea el resultado despreciando la parte decimal.
4. Utilizando la tabla TEMPLA, obtener el nombre y apellido de los empleados en minúsculas, excepto la primera letra del apellido que se mantendrá en mayúsculas.

07.- Visualización de Datos con JOINS

- Escritura de Sentencias SELECT para Acceder a Datos de Más de Una Tabla
- Visualización de Datos que Normalmente no Cumplen una Condición de Unión mediante Uniones Externas
- Unión de una Tabla consigo Misma mediante Autounión

MULTIPLES TABLAS

COMBINACIONES (JOINS)

- Son utilizadas para recuperar datos de varias tablas

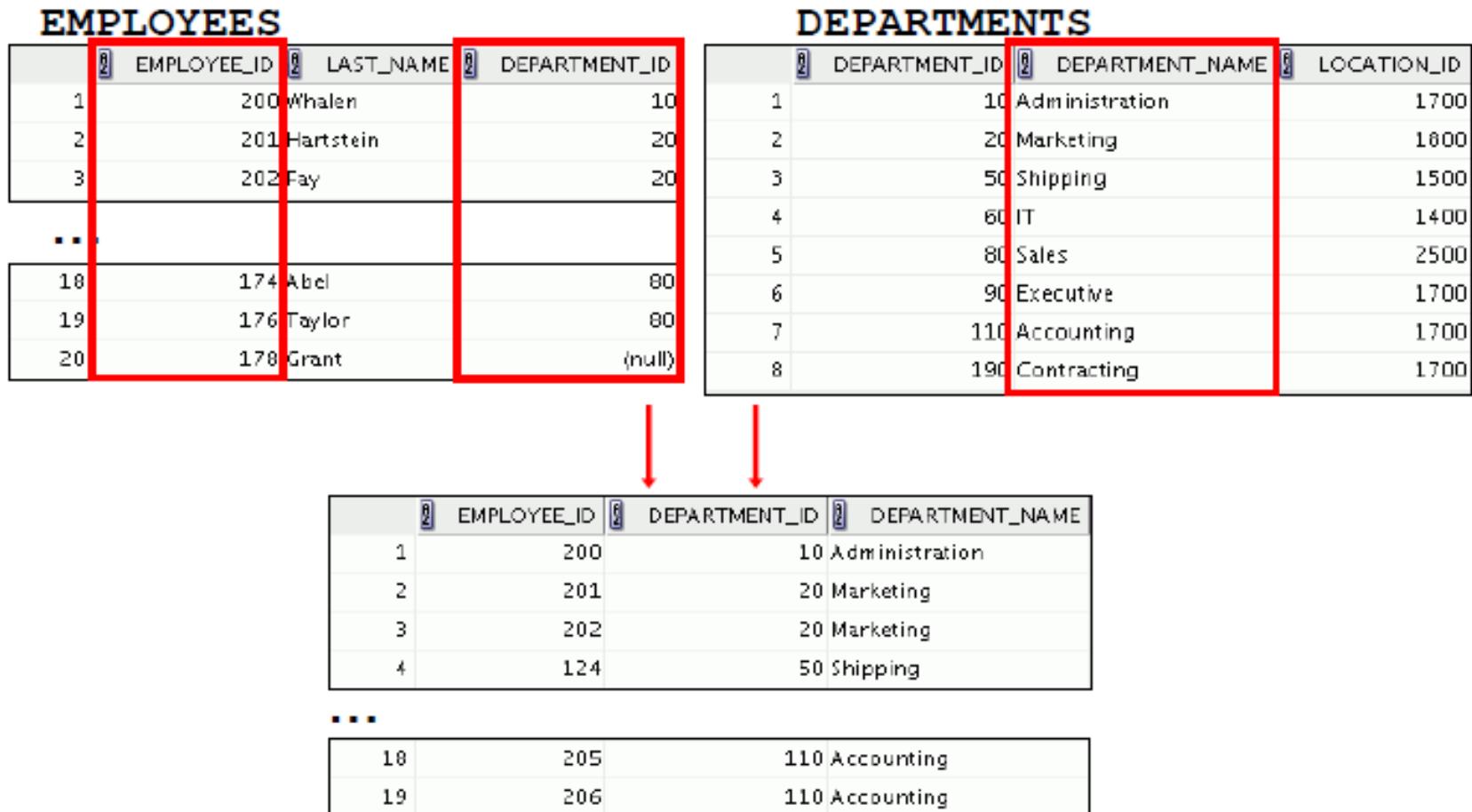
SUBCONSULTAS (SUB-QUERIES)

- Son utilizadas para generar preguntas subsidiarias necesitadas por la consulta principal

EJEMPLO DE COMBINACIÓN DE TABLAS

- Necesitamos unir la tabla de vendedores y la tabla de pedidos para poder obtener la información que deseamos
 - ✓ *SELECT V.COD_VENDE, V.NOMBRE, P.COD_PEDIDO, P.IMPORTE_PED, P.COMISION FROM PEDID P, VEND V;*
 - ✓ Como se puede ver, el resultado obtenido no es lo que se buscaba: se ha conseguido el producto cartesiano de las dos tablas; es decir, que ha combinado todas las filas de una tabla con todas las filas de la otra

Obtención de Datos de Varias Tablas



REGLAS PARA LAS COMBINACIONES (JOINS)

- Se pueden obtener la unión de tantas tablas como se quiera
- Puede usarse más de una pareja de columnas para especificar una condición de combinación entre dos tablas
- Se pueden seleccionar columnas de todas las tablas implicadas en la combinación
- Si tenemos columnas con idéntico nombre en dos tablas de la combinación se deberán calificar las columnas con el nombre de la tabla al que pertenecen
- Como regla general, la combinación tendrá como mínimo tantas condiciones en la cláusula WHERE como número de tablas de la cláusula FROM menos uno

Tipos de JOINS

- Las uniones compatibles con el estándar SQL:1999 incluyen los siguientes elementos:
 - Equi-Join versus NOT Equi-Join
 - Uniones naturales:
 - ✓ Cláusula NATURAL JOIN
 - ✓ Cláusula USING
 - ✓ Cláusula ON
 - Uniones OUTER:
 - ✓ LEFT OUTER JOIN
 - ✓ RIGHT OUTER JOIN
 - ✓ FULL OUTER JOIN
 - Uniones cruzadas

TIPOS DE COMBINACIONES

EQUI-JOIN

- Aquella cuyo criterio de combinación se establece a través del operador igual (=).
- Se utiliza para recuperar filas que tengan valores emparejados en cada una de las columnas citadas en el comando JOIN

NOT EQUI-JOIN

- Es aquella cuyo criterio de combinación se establece a través del operador no igual (!=).
- Se utiliza para recuperar filas que tengan valores no emparejados en cada una de las columnas citadas en el comando JOIN

TIPOS DE COMBINACIONES

NATURAL JOIN

- La cláusula NATURAL JOIN está basada en todas las columnas de las dos tablas que tienen el mismo nombre.
- Selecciona filas de las dos tablas que tienen valores iguales en todas las columnas coincidentes.
- Si las columnas que tienen el mismo nombre tienen tipos de dato diferentes, se devolverá un error.

```
SELECT department_id, department_name,
       location_id, city
  FROM departments
NATURAL JOIN locations ;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1000	Toronto
8	80	Sales	2500	Oxford

Creación de Uniones con la Cláusula USING

- Si varias columnas tienen el mismo nombre pero los tipos de dato no coinciden, utilizar la cláusula USING para especificar las columnas para la unión igualitaria.
- Utilizar USING para que sólo coincida una columna en caso de que coincida más de una.
- Las cláusulas NATURAL JOIN y USING se excluyen mutuamente..

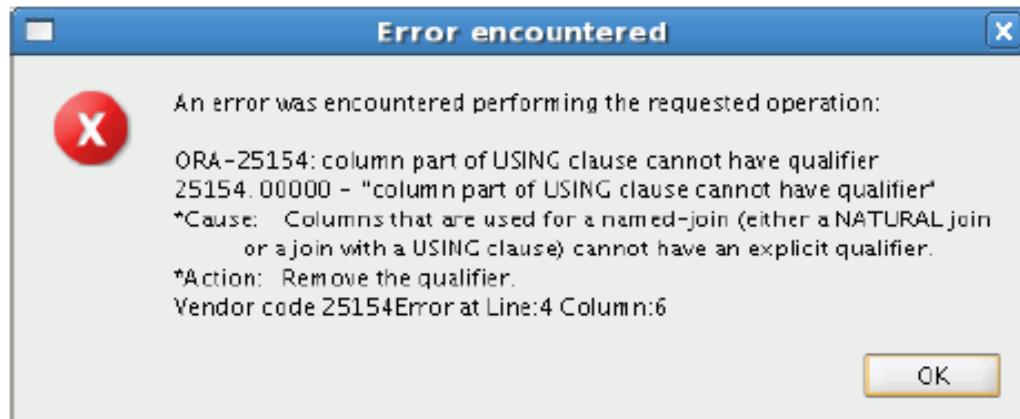
```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
USING (department_id) ;
```

Creación de Uniones con la Cláusula USING

ALIAS

- No cualificar una columna que se utilice en la cláusula USING.
- Si la misma columna se utiliza en otro lugar de la sentencia SQL, no se le puede agregar un alias.

```
SELECT l.city, d.department_name
  FROM locations l JOIN departments d
 USING (location_id)
 WHERE d.location_id = 1400;
```



Creación de Uniones con la Cláusula ON

- La condición de unión de la unión natural es básicamente una unión igualitaria de todas las columnas con el mismo nombre.
- Utilizar la cláusula ON para especificar condiciones arbitrarias o columnas que se van a unir.
- La condición de unión está separada de otras condiciones de búsqueda.
- La cláusula ON facilita la comprensión del código.

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
  FROM employees e JOIN departments d
 WHERE e.department_id = d.department_id;
```

Creación de Uniones en 3 Direcciones con la Cláusula ON

```
SELECT employee_id, city, department_name
FROM employees e
JOIN departments d
ON d.department_id = e.department_id
JOIN locations l
ON d.location_id = l.location_id;
```

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping
9	142	South San Francisco	Shipping

Aplicación de Condiciones Adicionales a una Unión

- Uso de la cláusula AND o la cláusula WHERE para aplicar condiciones adicionales:

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

O bien

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
WHERE  e.manager_id = 149 ;
```

Unión de una Tabla consigo Misma

- Puede que a veces necesite unir una tabla consigo misma.
- Para buscar el nombre de cada gestor del empleado, necesita unir la tabla EMPLOYEES consigo misma o realizar una autounión.
- Durante este proceso, busca dos veces en la tabla.
 - La primera vez, cuando consulta la tabla para buscar a Lorentz en la columna LAST_NAME y el valor de MANAGER_ID de 103.
 - La segunda vez, cuando busca en la columna EMPLOYEE_ID para buscar 103 y en la columna LAST_NAME para buscar a Hunold.

Unión de una Tabla consigo Misma

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...

...

**MANAGER_ID en la tabla WORKER es igual a
EMPLOYEE_ID en la tabla MANAGER.**

Autouniones que Utilizan la Cláusula ON

- También se puede utilizar la cláusula ON

```
SELECT worker.last_name emp, manager.last_name mgr
FROM employees worker JOIN employees manager
ON (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Cietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
...		

TIPOS DE COMBINACIONES

OUTER-JOIN

- Recupera toda la información de dos tablas con coincidencia en la columna indicada en la cláusula where
- Hay que considerar el caso de datos no grabados en una de las tablas
- La información que aún no ha sido grabada no ha sido recuperada
- Para recuperar este tipo de información aunque le falte una parte se utiliza el OUTER-JOIN
- La sintaxis sería:
 - ✓ *SELECT t1.columna,t1.columna2,t2.columna1,t2.columna2
FROM tabla1 t1,tabla2 t2
WHERE t1.columna1 = t2.columna1(+)*

LEFT OUTER JOIN

- Esta consulta recupera todas las filas de la tabla EMPLOYEES que es la tabla de la izquierda, incluso si no hay ninguna coincidencia en la tabla DEPARTMENTS.

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e LEFT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping
...			

16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

RIGHT OUTER JOIN

- Esta consulta recupera todas las filas de la tabla DEPARTMENTS que es la tabla de la izquierda, incluso si no hay ninguna coincidencia en la tabla EMPLOYEES.

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping
...			
18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

FULL OUTER JOIN

- Esta consulta recupera todas las filas de la tabla EMPLOYEES, incluso si no hay ninguna coincidencia en la tabla DEPARTMENTS.
- También recupera todas las filas de la tabla DEPARTMENTS, incluso si no hay ninguna coincidencia en la tabla EMPLOYEES.

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting
...			

17	Zlotkey	80	Sales
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

Productos Cartesianos

- Un producto cartesiano se forma cuando:
 - Se omite una condición de unión
 - Una condición de unión no es válida
 - Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla
- Se incluye siempre una condición de unión válida si desea evitar un producto cartesiano.

EMPLOYEES (20 filas)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
...			
19	176	Taylor	60
20	178	C	

DEPARTMENTS (8 filas)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
...			

Producto cartesiano:
 $20 \times 8 = 160$ filas

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10	1700
2	201	20	1700
...			

21	200	10	1800
22	201	20	1800
...			
159	176	60	1700
160	178	(null)	1700

Ejercicios

Ejercicios propuestos:

1. Obtener un listado de todos los empleados (nombre y apellidos) que ganan más de 200000 pts al mes y que entraron en la compañía después del 1 de Enero de 1975. También se quiere información correspondiente a su código de trabajo.
2. Obtener el nombre, apellido y fecha de ingreso de los directores de departamento ordenados por número de personal.
3. Seleccionar parejas de empleado (de sexo opuesto) que hayan nacido el mismo día (con información acerca de apellidos y fecha de nacimiento).

08.- Subconsultas

- Descripción de los Tipos de Problemas que Pueden Solucionar las Subconsultas
- Definición de Subconsultas
- Lista de Tipos de Subconsultas
- Escritura de Subconsultas de Una o Varias Filas

SUBCONSULTAS

¿Quién tiene un salario mayor que el de Abel?

Consulta principal:



¿Qué empleados tienen un salario mayor que el de Abel?

Subconsulta:



¿Cuál es el salario de Abel?

SUBCONSULTAS

DEFINICION

- Hasta ahora se ha utilizado la cláusula WHERE para definir los subconjuntos de datos que se quieren obtener, emparejando un valor de la columna de una tabla al valor de una constante.
- ¿ Qué hacer cuando se desconoce el valor de la constante? Habrá que adivinar dicho valor.
- El procedimiento sería efectuar una consulta (SELECT) a la misma tabla o a otra tabla, y una vez anotado dicho resultado en un papel, efectuar la consulta primera que queríamos realizar.

SUBCONSULTAS

- ORACLE soluciona este problema mediante las SUBCONSULTAS, y permite que en la cláusula WHERE se realice una consulta en una de las partes de la condición.
- La SELECT de la cláusula WHERE deberá ir encerrada entre paréntesis para que se realice antes esta SELECT, y con el valor obtenido se podrá realizar la primera consulta.
- A la segunda SELECT se la denomina subconsulta anidada.
- Nos encontramos con la posibilidad de que la consulta que efectuamos en la cláusula WHERE no nos devuelva un único valor; dependiendo de los valores devueltos se podrán utilizar unos operadores u otros.

OPERADORES DE LAS SUBCONSULTAS

RESULTADO DE LA SUBCONSULTA

- VALOR SIMPLE
- VALOR MULTIPLE

OPERADORES

- COMPARACION GENERAL
- COMPARACION LOGICA

VALOR DE LOS OPERADORES

- =,!=,<>,>,>=,<,<=
- ANY (SOME), ALL, IN, EXISTS

OPERADORES DE LAS SUBCONSULTAS

VALORES UNICOS

- Si una Subselect devuelve un solo valor, se usarán exclusivamente los operadores de comparación.

= , <> , < , <= , > , >=

✓ *SELECT columnas FROM tabla
WHERE columna > (Subselect.....);*

EJEMPLOS DE SUBCONSULTAS

Ejemplo de subconsulta que devuelve solo un valor

- Obtener los empleados de la tabla de empleados que ganen mas que la media de salarios de toda la compañía:
 - ✓ *SELECT * FROM templas
WHERE salar > (SELECT AVG(salar) from templas);*
- Ver los empleados que ganan más que cualquier empleado del DEP B01:
 - ✓ *SELECT * FROM templas
WHERE salar > (SELECT MAX(salar) FROM templas WHERE dept = 'B01');*

OPERADORES DE LAS SUBCONSULTAS

VALORES MULTIPLES

- Si una Subselect puede devolver mas de un valor, usaremos los operadores de comparación unidos a las siguientes cláusulas:

[oper]ANY , [oper]ALL, IN , EXISTS

[oper]ANY

- ✓ Compara la columna con los valores devueltos por la subselect y si alguno de ellos cumple la condición del operador, el resultado es TRUE.
- ✓ Si la subselect es 0, el resultado es FALSE

*SELECT * FROM tempa*

WHERE salar >ANY (SELECT salar FROM tempa WHERE dept = 'B01');

OPERADORES DE LAS SUBCONSULTAS

[oper]ALL

- ✓ Compara la columna con los valores devueltos por la subselect y si la columna cumple la condición del operador con todos los resultados, el resultado es TRUE.
- ✓ Si la subselect es 0, el resultado es FALSE

```
SELECT * FROM tempa  
WHERE salario >ALL ( SELECT salario FROM tempa WHERE departamento = 'B01');
```

IN (=ANY)

- ✓ Es equivalente a =ANY.
- ✓ Compara la columna con los valores devueltos por la subselect y si la columna es igual a alguno de los resultados devueltos por la subselect, el resultado es TRUE.
- ✓ Si la subselect es 0, el resultado es FALSE

```
SELECT * FROM tempa  
WHERE salario IN ( SELECT salario FROM tempa WHERE departamento = 'B01');
```

OPERADORES DE LAS SUBCONSULTAS

EXISTS

- ✓ Es equivalente a =ANY.
- ✓ Comprueba si la consulta devuelve algún resultado, si lo hace, el resultado es TRUE.
- ✓ Si la subselect es 0, el resultado es FALSE

```
SELECT * FROM tempa
```

```
WHERE EXISTS ( SELECT MAX(salar) FROM tempa WHERE  
dept = 'B01');
```

```
SELECT * FROM tempa t1
```

```
WHERE EXISTS ( SELECT AVG(salar) FROM tempa WHERE  
t1.salar is null);
```

EJEMPLOS DE SUBCONSULTAS

Ejemplo de subconsulta que devuelve una lista de valores

- Obtener los empleados de la empresa 2 cuya categoría sea idéntica a la de cualquier empleado de la empresa cuyo nombre es ALFA S.A.:
 - ✓ *SELECT * FROM tempa WHERE dept='A00' AND cod_cat IN (SELECT cod_cat FROM tempa WHERE cod_empr = (SELECT cod_empr FROM empr WHERE nomb_empr = 'ALFA S. A.'));*
- En el ejemplo anterior, se ve que una sentencia SELECT anidada puede a su vez tener otra SELECT anidada; esta anidación funciona con un número indefinido de niveles

Ejemplo de subconsulta que devuelve más de una columna

- Obtener los empleados cuya categoría y salario sea idéntico que LOPEZ:

✓ *SELECT nombre FROM tempa
WHERE (codtra,salar) = (SELECT codtra,salar FROM tempa
WHERE nombre = 'LOPEZ');*

Ejercicios

Ejercicios propuestos:

1. Obtener un listado de las mujeres de los departamentos que comiencen por D y E cuyo nivel de educación sea superior a la media; en este caso también ordenados por numero de personal.
2. Seleccionar todos los empleados cuyo nombre sea igual al de algunas personas del departamento D21 y cuyo código de trabajo sea diferente de todos los del E21 (la lista debe contener el número de personal, nombre, apellido, departamento y código de trabajo).

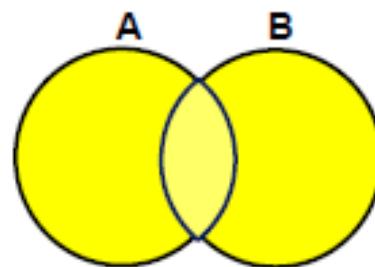
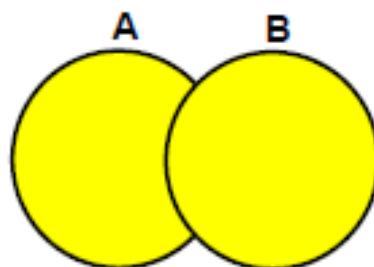
09.- Operadores de Definición

- Descripción de los Operadores SET
- Uso de un Operador SET para Combinar Varias Consultas en una Sola
- Control del Orden de las Filas Devueltas

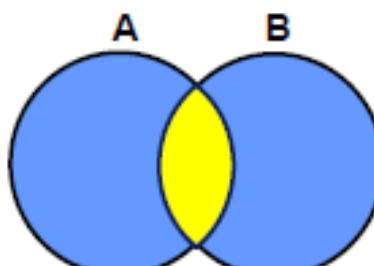
RELACIONES

Existen tres tipos de relaciones

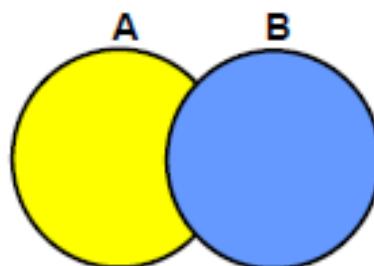
- UNION
- INTERSECT
- MINUS



UNION/UNION ALL



INTERSECT



MINUS

RELACIONES

UNION

- Combinación de todas las filas del primer conjunto con todas las filas del segundo. Cuando se duplique una fila por existencia de la misma en los dos conjuntos solo aparecerá una

INTERSECT

- Combinación de las filas de los dos conjuntos, cuando la fila exista en ambos conjuntos

MINUS

- Combinación de las filas del primer conjunto que no estén en el segundo

Instrucciones de los Operadores de Definición

- La expresiones de las listas SELECT debe coincidir en el número de columnas seleccionadas.
- Los tipos de dato para cada columna de la segunda consulta deben coincidir con los tipos de dato de su columna correspondiente en la primera consulta.
- Los paréntesis se pueden utilizar para modificar la secuencia de ejecución.

Select union select intersect....

Los conjuntos son evaluados de izquierda a derecha

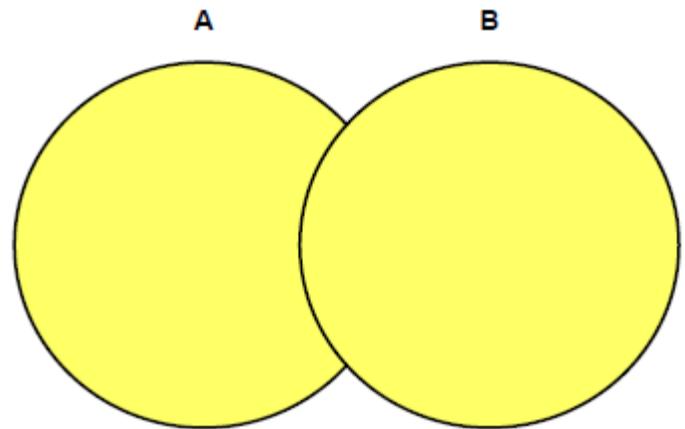
- La sentencia ORDER BY puede aparecer sólo una vez al final de la sentencia.

Servidor de Oracle y Operadores de Definición

- Las filas duplicadas se eliminan automáticamente excepto en UNION ALL.
- Los nombres de columna de la primera consulta aparecen en el resultado.
- Por defecto, la salida se ordena en orden ascendente, excepto en UNION ALL.

Operador UNION

- El operador UNION devuelve todas las filas seleccionadas en cualquier consulta.
- Utilice el operador UNION para devolver todas las filas de varias tablas y eliminar las filas duplicadas.



Instrucciones

- El número de columnas seleccionadas debe ser el mismo.
- Los tipos de dato de las columnas seleccionadas deben pertenecer al mismo grupo de tipo de dato (por ejemplo, numérico o de caracteres).
- El operador UNION funciona en todas las columnas seleccionadas.
- Los valores NULL no se ignoran durante la comprobación de duplicados.
- Por defecto

Operador UNION

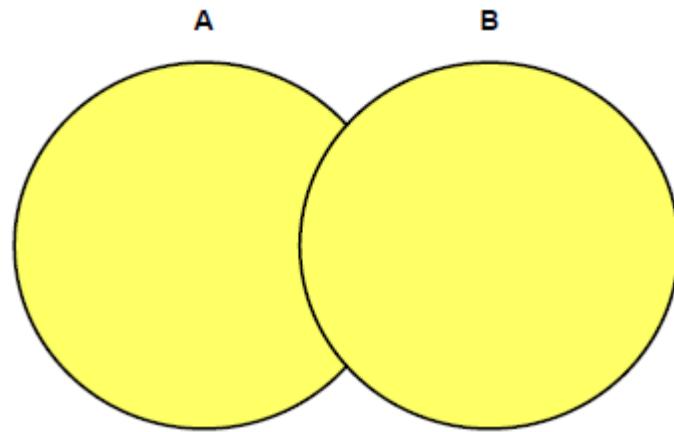
- Mostrar los detalles actuales y anteriores del puesto de todos los empleados. Mostrar cada empleado sólo una vez.

```
SELECT employee_id, job_id  
FROM employees  
UNION  
SELECT employee_id, job_id  
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID
1		100 AD_PRES
2		101 AC_ACCOUNT
...		
22		200 AC_ACCOUNT
23		200 AD_ASST
...		
27		205 AC_MGR
28		206 AC_ACCOUNT

Operador UNION ALL

- Utilice el operador UNION ALL para devolver todas las filas de varias consultas



Instrucciones

- El número de columnas seleccionadas debe ser el mismo.
- Los tipos de dato de las columnas seleccionadas deben pertenecer al mismo grupo de tipo de dato (por ejemplo, numérico o de caracteres).
- El operador UNION funciona en todas las columnas seleccionadas.
- Los valores NULL no se ignoran durante la comprobación de duplicados.
- Devuelve los registros duplicados.

Operador UNION ALL

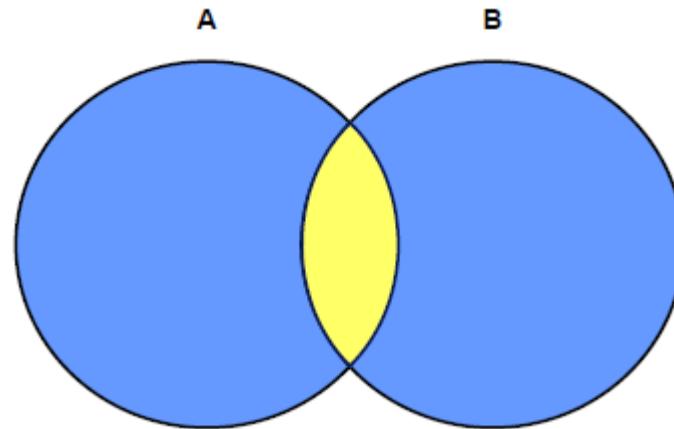
- Mostrar los departamentos actuales y anteriores de todos los empleados.

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
17	149	SA_MAN	80
18	174	SA_REP	80
19	176	SA_REP	60
20	176	SA_MAN	60
21	176	SA_REP	80
22	178	SA_REP	(null)
23	200	AD_ASST	10
30	206	AC_ACCOUNT	110

Operador INTERSECT

- El operador **INTERSECT** devuelve filas comunes a ambas consultas.



Instrucciones

- El número de columnas y los tipos de dato de las columnas seleccionadas por las sentencias SELECT en las consultas deben ser idénticos en todas las sentencias SELECT utilizadas en la consulta.
- No es necesario, sin embargo, que los nombres de las columnas sean idénticos.
- Si se invierte el orden de las tablas intersectadas no se alterará el resultado.
- INTERSECT no ignora los valores NULL.

Operador INTERSECT

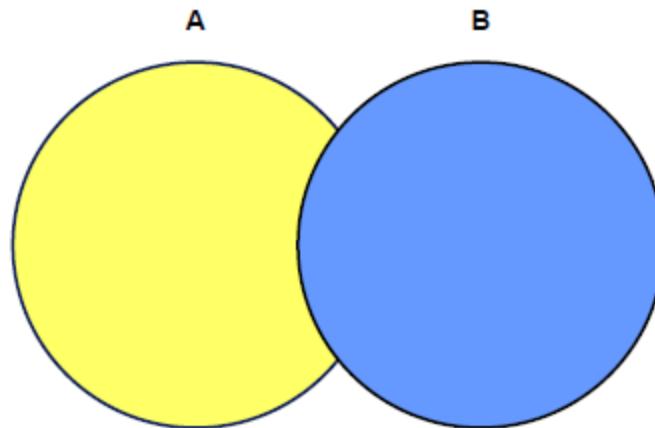
- Mostrar los ID de empleado y de cargo de los empleados que actualmente tienen el mismo puesto que anteriormente (es decir, han cambiado de cargo pero ahora han vuelto a realizar el mismo trabajo que realizaban anteriormente).

```
SELECT employee_id, job_id  
FROM employees  
INTERSECT  
SELECT employee_id, job_id  
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID
1		176 SA_REP
2		200 AD_ASST

Operador MINUS

- El operador MINUS devuelve todas las filas distintas seleccionadas por la primera consulta, pero que no están presentes en el juego de resultados de la segunda consulta.



Instrucciones

- El número de columnas y los tipos de dato de las columnas seleccionadas por las sentencias SELECT de las consultas deben pertenecer al mismo grupo de tipo de dato en todas las sentencias SELECT utilizadas en la consulta.
- No es necesario, sin embargo, que los nombres de las columnas sean idénticos.
- MINUS no ignora los valores NULL.

Operador MINUS

- Mostrar los identificadores de empleado cuyos empleados no han cambiado sus puestos ni una vez.

```
SELECT employee_id  
FROM employees  
MINUS  
SELECT employee_id  
FROM job_history;
```

#	EMPLOYEE_ID
1	100
2	103
3	104

13	202
14	205
15	206

Coincidencia de las Sentencias SELECT

- Debido a que las expresiones de las listas SELECT de las consultas deben coincidir en número, puede utilizar columnas ficticias y funciones de conversión de tipos de dato para cumplir con esta regla.
- Podemos utilizar funciones como TO_CHAR (o cualquier otra función de conversión) cuando las columnas no existan en una tabla o en la otra.

```
SELECT location_id, department_name "Department",
       TO_CHAR(NULL) "Warehouse location"
  FROM departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department",
       state_province
  FROM locations;
```

Coincidencia de las Sentencias SELECT

EJEMPLO

- Utilizar el operador UNION, mostrar el ID de empleado, ID de cargo y salario de todos los empleados.

```
SELECT employee_id, job_id,salary  
FROM employees  
UNION  
SELECT employee_id, job_id,0  
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID	SALARY
1	100	AD_PRES	24000
2	101	AC_ACCOUNT	0
3	101	AC_MGR	0
4	101	AD_VP	17000
5	102	AD_VP	17000
...			
29	205	AC_MGR	12000
30	206	AC_ACCOUNT	8300

Clausula ORDER BY

- La cláusula ORDER BY sólo puede aparecer una vez al final de la consulta compuesta.
- Las consultas de componente no pueden tener cláusulas ORDER BY individuales.
- La cláusula ORDER BY reconoce sólo las columnas de la primera consulta SELECT.
- Por defecto, la primera columna de la primera consulta SELECT se utiliza para ordenar la salida en orden ascendente.

Clausula ORDER BY

EJEMPLO

- Por ejemplo, en la siguiente sentencia, la salida se mostrará en orden ascendente según job_id.

```
SELECT employee_id, job_id, salary
FROM employees
UNION
SELECT employee_id, job_id, 0
FROM job_history
ORDER BY 2;
```

10.- Manipulación de Datos

- Descripción de Cada Sentencia DML
- Inserción de Filas en una Tabla
- Cambio de Filas en una Tabla con la Sentencia UPDATE
- Supresión de Filas de una Tabla con la Sentencia DELETE
- Guardado y Desecho de Cambios con las Sentencias COMMIT y ROLLBACK
- Explicación de la Consistencia de Lectura

LENGUAJE DML: Insert

INSERCIÓN DE FILAS (INSERT)

- Para insertar filas en una tabla se utiliza el comando **INSERT**
- **SINTAXIS**
INSERT INTO Nombre_tabla (columnas a insertar) VALUES (valores)
- Las columnas se identifican por su nombre. La asociación de columna y su valor es posicional. Los valores deben cumplir con el tipo de datos de la columna
- Los valores constantes de tipo carácter o fecha deben ir encerrados entre comillas simples (' ')

LENGUAJE DML: Insert

INSERCIÓN DE FILAS (INSERT)

- También se pueden hacer inserciones múltiples

- **SINTAXIS**

INSERT INTO Nombre_tabla (columnas a insertar)

SELECT Columnas_a_insertar FROM tabla;

- El número y tipo de las columnas recuperadas deben ser iguales a las columnas a insertar
- Restricciones:
 - ✓ Dentro de la SELECT no pueden existir ORDER BY , UNIONES, INTERSECT ni MINUS
 - ✓ La tabla donde se inserta no puede ser la tabla usada para la SELECT

LENGUAJE DML: Insert

Copia de Filas de Otra Tabla

- Para copiar una Tabla en otra, deberemos escribir la sentencia INSERT con una subconsulta:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

4 rows inserted

- No utilizar la cláusula VALUES.
- Hacer coincidir el número de columnas de la cláusula INSERT con el de la subconsulta.
- Inserta todas las filas devueltas por la subconsulta en la tabla, sales_reps.

LENGUAJE DML: Update

MODIFICACIÓN DE FILAS (UPDATE)

- La modificación de los datos ya insertados en una tabla se realiza con el comando UPDATE
- **SINTAXIS**
 - ✓ *UPDATE nombre_tabla
SET nombre_columna = valor,...
WHERE condición;*
- **SET**
 - ✓ Indica la columna a modificar y el valor nuevo que tomará la columna
- **WHERE**
 - ✓ Indica la fila o filas en las que se va a realizar la modificación; si se omite, la modificación se realiza en todas las filas de la tabla

LENGUAJE DML: Update

MODIFICACIÓN DE FILAS (UPDATE)

- Si se especifica la cláusula WHERE, se modifican los valores de una fila o varias filas específicas.

```
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
```

1 rows updated

- Si se omite la cláusula WHERE, se modifican los valores de todas las filas de la tabla:

```
UPDATE copy_emp
SET department_id = 110;
```

22 rows updated

- Especificar SET column_name= NULL para actualizar un valor de columna a NULL.

LENGUAJE DML: Update

Actualización de Dos Columnas con una Subconsulta

- Se puede actualizar varias columnas en la cláusula SET de una sentencia UPDATE mediante la escritura de varias subconsultas.
- EJEMPLO:
 - ✓ Actualizar el cargo y el salario del empleado 113 para que coincida con los del empleado 205.

```
UPDATE employees
SET      job_id  = (SELECT job_id
                     FROM   employees
                     WHERE  employee_id = 205),
        salary   = (SELECT salary
                     FROM   employees
                     WHERE  employee_id = 205)
WHERE employee_id = 113;
1 rows updated
```

LENGUAJE DML: Update

Actualización de Filas Basada en Otra Tabla

- Se puede utilizar subconsultas en las sentencias UPDATE para actualizar los valores de fila de una tabla según los valores de otra tabla:

```
UPDATE copy_emp
SET department_id = (SELECT department_id
                      FROM employees
                      WHERE employee_id = 100)
WHERE job_id          = (SELECT job_id
                          FROM employees
                          WHERE employee_id = 200);
1 rows updated
```

LENGUAJE DML: Delete

BORRADO DE FILAS (DELETE)

- Utilizada para borrar filas de una tabla
- **SINTAXIS**
 - ✓ *DELETE
FROM Nombre de la tabla
WHERE Condición*
- **FROM**
 - ✓ Indica la tabla de donde se pretende borrar
- **WHERE**
 - ✓ Condición de selección de las filas que se pretenden borrar. Si se omite, se borrarán todas las filas de la tabla

LENGUAJE DML: Delete

BORRADO DE FILAS (DELETE)

- Se suprimen filas concretas si se especifica la cláusula WHERE:

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 rows deleted
```

- Se suprimen todas las filas de la tabla si omite la cláusula WHERE:

```
DELETE FROM copy_emp;
22 rows deleted
```

LENGUAJE DML: Delete

- ***Supresión de Filas Basada en Otra Tabla***
 - Utilizar subconsultas en las sentencias DELETE para eliminar filas de una tabla según los valores de otra tabla:

```
DELETE FROM employees
WHERE department_id =
      (SELECT department_id
       FROM departments
       WHERE department_name
             LIKE '%Public%');

1 rows deleted
```

LENGUAJE DML: Truncate

SENTENCIA TRUNCATE

- Elimina todas las filas de una tabla, dejando la tabla vacía y la estructura de la misma intacta.
- Es una sentencia de lenguaje de definición de datos (DDL) en lugar de una sentencia DML; no se puede deshacer fácilmente.
- Sintaxis:

```
TRUNCATE TABLE table_name;
```

- Ejemplo:

```
TRUNCATE TABLE copy_emp;
```

Lenguaje DVL

LENGUAJE DVL

VALIDACION DE DATOS (COMMIT)

- Utilizada para validar la transacción actual
- **SINTAXIS**
COMMIT
- Hace permanentes los cambios producidos en los datos
- Borra los SAVEPOINTS creados
- Desactiva los bloqueos

LENGUAJE DVL

CANCELACION DE DATOS (ROLLBACK)

- Utilizada para la cancelación de la transacción actual
- **SINTAXIS**
ROLLBACK [TO SAVEPOINT Nombre]
- Orden contraria al COMMIT
- Deshace los cambios producidos hasta el principio de la transacción actual o hasta el punto indicado

LENGUAJE DVL

BLOQUEOS EN ORACLE

- Existen diferentes tipos de bloqueos en Oracle:
 - ✓ DML
 - ✓ DDL
 - ✓ Internos (Latch's)
- La mayoría de los bloqueos son gestionados por Oracle (activados, mantenidos y eliminados).
- Los bloqueos obtenidos en las Tablas dependen de las operaciones a realizar DML o DDL.

LENGUAJE DVL

BLOQUEOS DE TABLA (LOCK)

- ORDENES DML
 - ✓ Las sentencias DML, realizan dos tipos de bloqueos:
 - Un bloqueo **ROW EXCLUSIVE** en cada fila o filas que queremos actualizar.
 - Y un bloqueo a nivel tabla en la tabla dependiendo del tipo de operación DML ejecutada.
 - Esto para prevenir que otra sesión bloquee la tabla completa, mientras se realiza el cambio.
- ORDENES DDL
 - ✓ Para ejecutar una sentencia DDL requiere un bloqueo **EXCLUSIVE** en el objeto a modificar.
 - ✓ Si el bloqueo **EXCLUSIVE** no se puede obtener, normalmente porque otra sesión tenga bloqueada la tabla termina inmediatamente con error

LENGUAJE DVL

BLOQUEOS DE TABLA (LOCK)

- Utilizada para bloquear una tabla de diferentes modos.
- El objetivo es garantizar que durante el tiempo que se trabaje con ella, la tabla no es modificada por otros usuarios.
- Los bloqueos se pueden quitar con COMMIT & ROLLBACK
- **SINTAXIS**

LOCK TABLA nombre_tabla IN modo MODE [NOWAIT]

- Modos de bloqueo
 - ✓ Bloqueo de Tabla de Fila Compartida (ROW SHARE) (RS)
 - ✓ Bloqueo de Tabla de Fila Exclusiva (ROW EXCLUSIVE) (RX)
 - ✓ Bloqueo de Tabla Compartida (SHARE) (S)
 - ✓ Bloqueo de Tabla de Fila Compartida Exclusiva (SHARE ROW EXCLUSIVE) (RSX)
 - ✓ Bloqueo de Tabla Exclusiva (EXCLUSIVE) (X)

NOWAIT permite regresar el control inmediatamente, si la tabla está bloqueada por otra sesión

Ejercicios

Ejercicios propuestos:

1. Al departamento D21 se le han asignado dos nuevos empleados, con motivo de la iniciación de un nuevo proyecto. Los únicos datos disponibles de ellos son:

EMPLEADO	NOMBRE	SALARIO	F.NACIMIENTO
-----	-----	-----	-----
000272	LUIS PERNAS	1838500	1964-07-20
000273	ELISA JORDAN	1874000	1969-10-11

Añada los registros de estos empleados a la tabla.

2. De los resultados de ejercicios anteriores, se puede ver que las mujeres del departamento E11 tienen el más bajo factor del código de trabajo. Por lo tanto incremente el salario de estas personas en un 10 %.

Cláusula FOR UPDATE en una Sentencia SELECT

- Cuando se emite una sentencia SELECT en la base de datos para consultar algunos registros, no se coloca ningún bloqueo en las filas seleccionadas.
- Sólo se bloquean aquellos registros que se han cambiado, pero que aún no se han confirmado.
- Aún así, otros usuarios podrán leer dichos registros tal y como aparecían antes del cambio (la “imagen anterior” de los datos).
- Hay ocasiones, sin embargo, en las que puede que desee bloquear un juego de registros incluso antes de cambiarlos en el programa.

Cláusula FOR UPDATE en una Sentencia SELECT

- Oracle ofrece la cláusula **FOR UPDATE** de la sentencia SELECT para realizar este bloqueo.
- Cuando emite una sentencia **SELECT...FOR UPDATE**, Oracle obtiene automáticamente los bloqueos a nivel de fila exclusivos de todas las filas identificadas por la sentencia SELECT.

```
SELECT employee_id, salary, commission_pct, job_id  
FROM employees  
WHERE job_id = 'SA_REP'  
FOR UPDATE  
ORDER BY employee_id;
```

Cláusula FOR UPDATE en una Sentencia SELECT

- Puede adjuntar la palabra clave opcional **NOWAIT** a la cláusula **FOR UPDATE** para indicar al servidor de Oracle que no espere si otro usuario ha bloqueado la tabla.
- En este caso, el control se devolverá inmediatamente al para que pueda realizar otro trabajo o simplemente esperar un período de tiempo antes de volver a intentarlo.
 - Sin la cláusula NOWAIT, el proceso se bloqueará hasta que la tabla esté disponible, cuando otro usuario libere los bloqueos a través de la emisión un comando COMMIT o ROLLBACK.

```
SELECT employee_id, salary, commission_pct, job_id  
FROM employees  
WHERE job_id = 'SA_REP'  
FOR UPDATE WAIT 5  
ORDER BY employee_id;
```

11.- Sentencias DDL

- Clasificación de los Principales Objetos de Base de Datos
- Revisión de la Estructura de la Tabla
- Lista de Tipos de Dato Disponibles para Columnas
- Creación de una Tabla Simple
- Descifrado de Creación de Restricciones al Crear la Tabla
- Descripción del Funcionamiento de los Objetos de Esquema

LENGUAJE DE DEFINICIÓN DE DATOS DDL

- SQL permite crear, modificar y borrar la estructura física de las tablas, vistas o sinónimos a través de los comandos DDL

COMANDOS DDL

- **CREATE** Para crear una tabla, vista o sinónimo
- **DROP** Para borrar una tabla, vista o sinónimo
- **ALTER** Sirve para modificar una tabla o una vista
Los sinónimos no pueden modificarse, sólo se pueden crear o borrar
- **RENAME** Sirve para cambiar de nombre una tabla o una vista
- **TRUNCATE** Para borrar una tabla o un índice
Borra su contenido, no la estructura

Los comandos DDL no permiten rollback de la acción realizada

CREACIÓN DE TABLAS

- Para crear una tabla se debe indicar el nombre de la tabla y las columnas que la forman, así como de qué tipo de dato es cada columna
- Antes de dar de alta una tabla se debe conocer:
 - ✓ Nombre de la tabla
 - ✓ Nombre de cada columna y tipo de dato que va a contener
 - ✓ Tamaño de cada columna
 - ✓ Además se debe conocer
 - Columnas que forman la clave primaria (puede ser sólo una)
 - Columnas, si las hay, que son claves ajenas
 - Columnas que no pueden ser nulas (obligatorias)

RESTRICCIONES EN LA CREACIÓN DE TABLAS

- En el nombre
 - Longitud máxima 30 caracteres
 - Debe comenzar siempre con un carácter alfabético.
 - No hace distinción entre mayúsculas y minúsculas
 - No puede haber nombres de tablas repetidas
- Otras restricciones
 - ORACLE soporta sintácticamente restricciones sobre las tablas y sus columnas
 - Todas las restricciones que se definan quedarán almacenadas en el diccionario de datos. Alguna de estas restricciones son:
 - ✓ Obligatoriedad de columnas (NOT NULL)
 - ✓ Unidad sobre una o varias columnas (PRIMARY KEY)
 - ✓ Claves ajenas (FOREIGN KEY ... REFERENCE)
 - ✓ Verificación de condiciones (CHECK)

TIPOS DE DATOS

- CHAR(tam)
 - ✓ Longitud fija, máximo 2000 caracteres.
 - ✓ Contiene cualquier carácter alfanumérico, en mayúsculas o minúsculas, y caracteres especiales como (+,*,&,\$,@,.....)
- VARCHAR2(tam)
 - ✓ Longitud máxima 4000 caracteres, longitud variable. Contiene cualquier carácter alfanumérico, en mayúsculas o minúsculas, y caracteres especiales como (+,*,&,\$,@,.)
- NCHAR(tam)
 - ✓ Longitud fija en caracteres UNICODE. Máximo 2000 **bytes**
 - ✓ Los caracteres UNICODE AL16UTF16 (2 bytes / carácter)
 - ✓ Los caracteres UNICODE UTF8 (3 bytes / carácter)
- NVARCHAR2(tam)
 - ✓ Longitud variable en caracteres UNICODE. Máximo 4000 **bytes**
 - ✓ Los caracteres UNICODE AL16UTF16 (2 bytes / carácter)
 - ✓ Los caracteres UNICODE UTF8 (3 bytes / carácter)

TIPOS DE DATOS

- **NUMBER(tam, dec)**
 - ✓ Longitud máxima 38 caracteres
 - ✓ Contiene números; el número entero indica la longitud total del campo, y los decimales indican el número de decimales.
- **BINARY_FLOAT**
 - ✓ Almacena un solo número de precisión simple coma flotante hasta 32 bits.
 - ✓ Este tipo de operaciones se realizan más rápidamente que las operaciones con valores numéricos.
 - ✓ BINARY_FLOAT necesita 5 bytes de espacio de almacenamiento.
- **BINARY_DOUBLE**
 - ✓ Almacena un solo número de precisión doble en coma flotante hasta 64 bits.
 - ✓ Este tipo de operaciones se realizan más rápidamente que las operaciones con valores numéricos.
 - ✓ BINARY_DOUBLE necesita 9 bytes de espacio de almacenamiento.

TIPOS DE DATOS

- DEC o DECIMAL
 - ✓ Subtipo de NUMBER
 - ✓ Almacena un número decimal con un máximo de 38 dígitos decimales de precisión.
- REAL, FLOAT
 - ✓ Subtipo de NUMBER
 - ✓ Almacena un número en coma flotante con un máximo de 18 dígitos de precisión.
- INT, INTEGER, SMALL_INT
 - ✓ Subtipo de NUMBER
 - ✓ Almacena un número entero con un máximo de 38 dígitos.

TIPOS DE DATOS

- DATE
 - ✓ Almacena la fecha y hora (Siglo, YYYY, MM, DD, HH24, MI y SS).
 - ✓ El formato por defecto especificado por el parámetro de base de datos NLS_DATE_FORMAT
- LONG
 - ✓ Datos de longitud variable de caracteres de hasta **2 GB**.
 - ✓ Reemplazado por el tipo CLOB y NCLOB, pero compatible con la compatibilidad hacia atrás.
- RAW(tam)
 - ✓ Contiene datos binarios de longitud variable. Máximo de 2.000 bytes.
 - ✓ Reemplazado por el tipo BLOB, pero compatible con la compatibilidad hacia atrás.
- LONGRAW(tam)
 - ✓ Contiene datos binarios de longitud variable hasta **2 Gb**.
 - ✓ Reemplazado por el tipo BLOB, pero compatible con la compatibilidad hacia atrás.

TIPOS DE DATOS

- CLOB / NCLOB
 - ✓ Objetos de gran tamaño para almacenar caracteres normales o UNICODE
 - ✓ Longitud máxima de **128 Tb.**
- BLOB
 - ✓ Objetos de gran tamaño para almacenar datos binarios
 - ✓ Longitud máxima de **128 Tb.**
- BFILE
 - ✓ Objetos de gran tamaño para almacenar datos binarios externos a la BBDD.
 - ✓ Longitud máxima la decide el S.O.

TIPOS DE DATOS

- **TIMESTAM (precision_en_segundos)**
 - ✓ Utilizado para guardar Fecha+Hora+Fracciones_Segundo.
 - ✓ La precisión especifica el número de dígitos para la parte fraccionaria de los segundos, que puede ser un entero del 0 al 9 (por defecto es 6)
- **TIMESTAM (precision_en_segundos) WHIT TIME ZONE**
 - ✓ Ampliación del TIMESTAM con la posibilidad de introducir zona horaria
 - ✓ El tamaño indica la precisión de las fracciones. Defecto (6)
 - ✓ La Zona horaria como: Meridiano Greenwich o Región
- **TIMESTAM (precision_en_segundos) WHIT LOCAL TIME ZONE**
 - ✓ Ampliación del TIMESTAM para convertir una fecha y hora suministrados a la zona horaria local establecido para la base de datos.
 - ✓ El tamaño indica la precisión de las fracciones. Defecto (6)

TIPOS DE DATOS

- ROWID
 - ✓ Dato hexadecimal que almacena la posición física en la base de datos, en el formato bloque.fila.fichero (16/18 bits)
- REF Object_type
 - ✓ Contiene la referencia a un tipo de objeto.
 - ✓ Similar a un puntero en lenguaje de programación C++ .
- XMLType
 - ✓ Almacena valores XML

SINTAXIS DE LA SENTENCIA CREATE TABLE

CREATE TABLE nombre_tabla

(

Nombre_columna1 Tipo_dato(longitud) [Restric_integridad],

Nombre_columna2 Tipo_dato(longitud) [Restric_integridad],

:

:

[Restricciones de Integridad]

)

- Las restricciones de integridad se pueden poner en la definición de cada columna o al final de la definición de la tabla

CONSTRAINT Nombre Restricción (columna,.....)

SINTAXIS DE LA SENTENCIA CREATE TABLE

Restricciones de Integridad:

NOT NULL

CONSTRAINT Nombre PRIMARY KEY (columna,.....)

CONSTRAINT Nombre REFERENCES Tabla (Columna)

CONSTRAINT Nombre CHECK (condición)

CONSTRAINT Nombre DEFAULT (Valor)

:

: Al final de la tabla

CONSTRAINT Nombre PRIMARY KEY (columna,.....)

CONSTRAINT Nombre FOREIGN KEY (columna,.....)

REFERENCES Nombre_tabla (Nombre de referencia)

SINTAXIS DE LA SENTENCIA CREATE TABLE

En las cláusulas

FOREIGN KEY (COLUMNA) REFERENCES TABLA [COLUMNA]
[ON DELETE CASCADE] -- *on update cascade no esta definido en Oracle*

En la opción (ON DELETE CASCADE):

- Con las restricciones de clave foránea podemos eliminar un registro de la tabla cliente y a la vez eliminar un registro de la tabla Primaria usando sólo una sentencia DELETE.
- Todos los registros relacionados son eliminados de acuerdo a las relaciones de clave foránea, tanto en la tabla Primaria como en la Relacionada.
- *No Afecta a operaciones **DROP** solo **DELETE***

Tablas y Vistas

ALL_CONSTRAINTS

- para ver las restricciones de las tablas que dispongamos.

```
SELECT constraint_name, constraint_type, status, deferrable, deferred  
FROM user_constraints  
WHERE table_name = '.....';
```

ALL_CONS_COLUMNS

- para ver las restricciones a nivel de columnas de las tablas que dispongamos

```
COLUMN column_name FORMAT a15  
SELECT constraint_name, column_name  
FROM user_cons_columns  
WHERE table_name = 'ORDER_STATUS2'  
ORDER BY constraint_name;
```

CREACIÓN DE UNA TABLA PARTIENDO DE OTRA

- Se utiliza la cláusula AS

*CREATE TABLE nombre_tabla (columna, ...) AS
SELECT ...*

- No es necesario especificar tipos ni tamaños de las columnas, ya que tomará los tipos y tamaños de la tabla de donde se obtienen los datos
- Además de la creación de la tabla, se crearán las filas que se seleccionaron en la SELECT asociada al CREATE
- Si se pretende dar el mismo nombre a las columnas en las dos tablas no hace falta referenciarlas

Tablas de Sólo Lectura

- Con Oracle Database 11g, puede especificar la opción ***READ ONLY*** para definir una tabla en modo de sólo lectura.
- Cuando la tabla esté en modo ***READ-ONLY***, no puede ejecutar ninguna sentencia DML que afecte a la tabla o a cualquier sentencia ***SELECT ... FOR UPDATE***.
- Puede ejecutar las sentencias DDL siempre y cuando no modifique los datos de la tabla.

```
ALTER TABLE employees READ ONLY;  
  
-- perform table maintenance and then  
-- return table back to read/write mode  
  
ALTER TABLE employees READ WRITE;
```

Tablas de Sólo Lectura

- Especifique READ/WRITE para volver a definir una tabla de sólo lectura en modo de lectura/escritura.
- Se permiten operaciones sobre índices asociados a la tabla cuando la tabla esté en modo READ ONLY.
- **Nota:** si es necesario puede borrar una tabla con modo READ ONLY.
 - El comando DROP se ejecuta sólo en el diccionario de datos, por lo que no es necesario el acceso al contenido de la tabla.
 - El espacio utilizado por la tabla no se reclamará hasta que el tablespace se vuelva a definir en lectura/escritura.

BORRADO DE UNA TABLA

- Mediante el comando **DROP TABLE**

DROP TABLE nombre_tabla [CASCADE CONSTRAINTS];

- El comando DROP borra la información contenida en la tabla y además la definición de dicha tabla contenida en el diccionario
- La opción **CASCADE CONSTRAINTS** elimina la tabla con todas sus filas y elimina también las claves ajena que apunten a cualquier columna de la tabla. (No borra los datos de la tabla ajena)

CAMBIO DE NOMBRE DE UNA TABLA

- Para cambiar de nombre a una tabla se utiliza el comando **RENAME**

RENAME nombre_antiguo TO nombre_nuevo;

Ejemplo

RENAME dept TO departamento;

MODIFICACIÓN DE UNA TABLA

- El comando utilizado para modificar la estructura de una tabla es *ALTER TABLE Nombre_de_tabla*
- El comando **ALTER** permite
 - Modificar el tipo de dato o la precisión de una columna
 - Añadir / Quitar columnas a la tabla
 - Modificar la integridad referencial sobre la tabla

MODIFICACIÓN DE UNA TABLA

SINTAXIS

- Para añadir una columna a la tabla

```
ALTER TABLE Nombre_tabla  
ADD (nueva_columna tipo_dato (longitud) características);  
[ ADD ( CONSTRAINT nombre PK (columna) );
```

- Para modificar una columna de la tabla

```
ALTER TABLE Nombre_tabla  
MODIFY (columna nuevo_tipo_dato (longitud) características);
```

- Para quitar una columna de la tabla

```
ALTER TABLE Nombre_tabla  
DROP COLUMN nombre;
```

MODIFICACIÓN DE UNA TABLA

- Para añadir una restricción a una columna de la tabla

ALTER TABLE Nombre_tabla

ADD (CONSTRAINT nombre_restricción PK (columna));

- Para quitar una restricción de una columna de la tabla

ALTER TABLE Nombre_tabla

DROP CONSTRAINT Nombre_restricción;

- Para modificar una restricción de una columna de la tabla

ALTER TABLE Nombre_tabla

MODIFY CONSTRAINT Nombre_restricción restriccion(Columna);

- Para desactivar/activar una restricción de una columna de la tabla

ALTER TABLE Nombre_tabla

DISABLE/ENABLE CONSTRAINT Nombre_restricción [cascade];

TRUNCADO DE UNA TABLA

- La orden TRUNCATE realiza la eliminación de todos los registros de una tabla completa.

TRUNCATE TABLE nombre_de_tabla;

- Es un método rápido y eficiente para eliminar registros de la tabla.

Ejemplo

TRUNCATE TABLE t1;

Ejercicios

Ejercicios propuestos:

1. Crear una tabla de prueba llamada TEMPLAB y que sea exactamente igual a la tabla TEMPLA.
2. Para futuras referencias deje información en el sistema acerca de la tabla TEMPLAB (en forma de comentario) y de una de sus columnas (Número de empleado)
3. Cree la vista VTEMPLAB sobre su tabla, incluyendo todas las columnas.
4. Inserte los datos en su tabla seleccionándolos desde TEMPLA, lleve cuidado pues sólo nos interesa almacenar los empleados que ganan más de 2000000. Si existen datos en la tabla TEMPLB borrarlos previamente.

12.- Otros Objetos de Esquema

- Creación de una Vista Simple y Compleja
- Recuperación de Datos de las Vistas
- Creación, Mantenimiento y Uso de las Secuencias
- Creación y Mantenimiento de Índices
- Creación de Sinónimos Privados y Públicos

VISTAS

- Una vista es una consulta predefinida en una o más tablas (conocido como tablas de base) que permite recuperar información de una forma mucho mas cómoda.
- Las vistas nunca almacenan registros, su definición se almacena en forma de registro en tablas.
- Cuando son utilizadas, en memoria se crea la misma estructura que una tabla (filas y columnas), y se tratan de forma similar

VISTAS

- BENEFICIOS
 - Se puede definir una consulta completa y mostrarla como una tabla básica a los usuarios finales, ocultando su complejidad.
 - Evita que los usuarios consulten directamente las tablas base mediante la concesión de acceso sólo a la vista.
 - Permite ocultar ciertas filas de una tabla a los usuarios finales, haciendo un filtrado en la creación de una vista mediante condiciones.

VISTAS

SINTAXIS. CREACIÓN DE VISTAS

```
CREATE [ OR REPLACE ] VIEW nombre_vista [ FORCE ]
[(columna,.....)]
AS ( Select .....
[ WITH CHECK OPTION [CONSTRAINT restricción] ]
[WITH READ ONLY];
```

- Para poder crear vistas necesitamos privilegio de **CREATE VIEW**

[FORCE]

- Indica que la vista se tiene que crear aunque no existan alguna de las tablas base que en ella se indica.

VISTAS

[*WITH CHECK OPTION [CONSTRAINT restricción]*]

- Cuando hacemos una inserción o modificación en la Vista, la operación DML debe de cumplir las condiciones de creación de la vista.
- Por defecto esto no se comprueba.

```
SQL> create or replace view v1 as
      select * from tdepta
      where nomdep like '%A%' WITH CHECK OPTION
```

[*WITH READ ONLY*]

- Asegura que ninguna operación DML pueda realizarse sobre esta vista.

OPERACIONES CON VISTAS

OPERACIONES		RESTRICCIONES
CONSULTAS		Sin restricciones
B O R R A D O	A C T U A L I Z A C I Ó N	I N S E R C I ÓN
		<ul style="list-style-type: none">•Vista Monotabla•Vista creada sin cláusulas GROUP BY DISTINCT
		No existen columnas en la vista obtenidas como expresión
		Todas las columnas obligatorias están en la vista

OPERACIONES CON VISTAS

- Los datos se pueden modificar a través de la misma vista o actualizando la tabla a la que pertenece la vista

RESTRICCIONES DE MANIPULACIÓN DE DATOS CON VISTAS

- **BORRADO DE FILAS**
 - ✓ La vista solo puede tener filas de una tabla
 - ✓ No se pueden utilizar en la sentencia SELECT de creación de la vista las cláusulas GROUP BY o DISTINCT
 - ✓ No se pueden utilizar en la sentencia SELECT de creación las funciones de grupo o referencia a pseudocolumnas (ROWID)
- **ACTUALIZACIÓN DE FILAS**
 - ✓ Todas las restricciones de borrado
 - ✓ Ninguna de las columnas a actualizar debe haber sido definida como una expresión
- **INSERCIÓN DE FILAS**
 - ✓ Todas las restricciones anteriores
 - ✓ Deben estar presentes en la vista todas las columnas obligatorias de la tabla asociada

BORRADO DE UNA VISTA

- Mediante el comando **DROP VIEW**
DROP VIEW nombre_vista [CASCADE CONSTRAINTS];
- El comando DROP borra la información de dicha vista en el diccionario
- Las vistas, vistas materializadas, sinónimos que dependan de la vista borrada, NO los borra pero los pone como INVALID.
- La opción **CASCADE CONSTRAINTS** elimina la vista con todas sus filas y elimina también las claves ajena que apunten a cualquier columna de la tabla. (No borra los datos de la tabla/vista ajena)

INFORMACION DE LAS VISTAS

- USER_VIEWS:
 - Muestra información de todas las vistas definidas en el esquema del usuario actual.
 - Para obtener la Select con la que una vista está creada utilizaremos la siguiente Query:

```
SQL> SELECT view_name, text_length, text
      FROM user_views
     ORDER BY view_name;
```

- USER_CONSTRAINTS:
 - Muestra información de todas las restricciones de vistas y tablas del usuario.
 - La columna CONSTRAINT_TYPE →
 - O → Read Only
 - V → Check Option

SINÓNIMOS

DEFINICIÓN

- Consiste en dar otro nombre a una tabla o una vista para poder referenciarla de dos formas distintas.
- Pueden ser de dos tipos: públicos y privados
 - ✓ Privados: Debe de tener privilegios de CREATE SYONYM y Sólo pueden ser usados por el propio usuario que los ha creado.
 - ✓ Públicos: Debe de tener privilegios de CREATE PUBLIC SYONYM y pueden ser usados por todos los usuarios del Sistema

Creación:

- ***CREATE [OR REPLACE] [PUBLIC] SYNONYM emp FOR scott.emp***

SINÓNIMOS

Utilidades

- Permite desarrollar aplicaciones utilizando una tabla o vista de otro usuario llamándole igual que él
- Permite usar el mismo programa contra dos bases de datos sin necesidad de modificarlo
- Cuando hacemos referencia a un nombre de tabla o vista, sin indicar el esquema, Oracle sigue la siguiente búsqueda para localizar el objeto al cual vamos a acceder:
 - ✓ Tablas del propio esquema
 - ✓ Vistas del propio esquema.
 - ✓ Sinónimos privados del propio esquema.
 - ✓ Sinónimos públicos del Sistema.

Borrado de un sinónimo

- **Para eliminar un sinónimo del Gestor usaremos la siguiente sintaxis:**

DROP [PUBLIC] SYNONYM [schema.]synonym

SQL>Drop synonym scott.sinonimo1;

También se pueden crear para un subconjunto de valores

```
CREATE SYNONYM mi_emp  
AS  
SELECT * FROM emp  
WHERE empno > 7500;
```

SECUENCIAS

- Es un elemento de base de datos que genera una secuencia de números enteros.
- Normalmente, se utiliza los números enteros generados para llenar un columna de clave principal.
- Sintaxis

CREATE SEQUENCE sequence_name	
[START WITH start_num]	Comienzo de secuencia (1)
[INCREMENT BY increment_num]	Incremento(1) o (-1)
[{ MAXVALUE maximum_num NOMAXVALUE }]	Valor Maximo de la secuencia
[{ MINVALUE minimum_num NOMINVALUE }]	Valor Mínimo de la secuencia
[{ CYCLE NOCYCLE }]	Secuencia circular o no
[CACHE numero]	Mantiene en cache los 20 siguientes valores

NOCYCLE → Cuando llega al límite superior cualquier intento de generar número da error

SECUENCIAS

Ejemplos:

```
SQL> CREATE SEQUENCE mi_secuencia;
```

El comienzo de la secuencia será 1 y el incremento de 1 en 1.

```
SQL > CREATE SEQUENCE s_test2  
      START WITH 10 INCREMENT BY 5  
      MINVALUE 10 MAXVALUE 20  
      CYCLE;
```

- *Disponemos de la tabla [DBA | ALL | USER]_SEQUENCES para obtener información de las secuencias existentes:*

```
SQL> COLUMN sequence_name FORMAT a13  
SQL> SELECT * FROM user_sequences  
      ORDER BY sequence_name;
```

SECUENCIAS

- Las secuencias contienen 2 pseudo Columnas:

Secuencia.CURRVAL	Valor actual de la Secuencia.
Secuencia.NEXTVAL	Incremento de la Secuencia y muestra valor resultante

NOTA: cuando una secuencia se crea, se inicializa al primer valor definido en la secuencia.

SQL> select test1.currval, test1.nextval from dual;

CURRVAL	NEXTVAL
-----	-----
1	1

SECUENCIAS

NOTAS

- Cuando un número de la secuencia es generado, la secuencia se incrementa, independientemente de si se realiza posteriormente un **commit o rollback**
- Las secuencias no se pueden poner de forma por defecto en la creación de una tabla para que actúe como auto numérico, se debería realizar a través de triggers
- Si dos usuarios al mismo tiempo incrementan la misma secuencia, cada usuario adquiere un número diferente y el autoincremental puede tener lagunas.
- Los números de secuencia son generados de forma independiente a las tablas.

SECUENCIAS

NOTAS

- Las secuencias no se pueden poner de forma por defecto en la creación de una tabla para que actúe como autonumérico.

CREATE SEQUENCE secuencia;

*CREATE TABLE MiTable (ID NUMBER(1) DEFAULT
secuencia.NEXTVAL);*

ORA-00984: column not allowed here

ERROR

- Deberíamos realizarlo a través de TRIGGERS

CREATE TABLE MiTable (ID NUMBER(1));

*CREATE OR REPLACE TRIGGER trig_seq BEFORE INSERT ON MiTable
FOR EACH ROW*

BEGIN

SELECT secuencia.NEXTVAL into :new.ID FROM dual;

END;

SECUENCIAS

- Podemos modificar las secuencias como cualquier otro objeto de Oracle, mediante la orden:

SQL> ALTER SEQUENCE nombre

- Podemos cambiar todo excepto:
 - Valor inicial de la secuencia.
 - El valor mínimo no puede ser mayor del actual.
 - El valor máximo no puede ser menor del actual.

*SQL>ALTER SEQUENCE s_test
INCREMENT BY 2;*

*SQL>ALTER SEQUENCE s_test
MAXVALUE BY 2000;*

SECUENCIAS

BORRADO DE SECUENCIAS

- Para borrar una secuencia utilizamos la orden DROP
- `DROP SEQUENCE Nombre_secuencia`

`SQL>DROP SEQUENCE s_test;`

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 en una base de datos sencilla.
- Los índices se suelen usar sobre aquellos campos/columnas sobre los cuales se hagan frecuentes búsquedas.
- Son usados por el SGBDR y para el usuario son totalmente transparentes.
- El espacio en disco requerido para almacenar el índice es típicamente menor que el espacio de almacenamiento de la tabla

INDICES

- Existen diferentes tipos de índices:
 - B-TREE
 - Índices sobre función
 - BITMAP
 - Índices REVERSE
 - Índices INVISIBLES (11g)
- El 90% de los índices son de tipo B-TREE, un 5% son BITMAP y el otro 5% son del resto de los índices

INDICES

SINTAXIS

- B-TREE

CREATE [UNIQUE] INDEX index_name ON table_name(c1,...)

SQL> Create index i1 on templa (Nombre);

- SOBRE FUNCION

CREATE INDEX index_name ON customers (FUNCION(columna));

SQL> Create index i1 on templa (UPPER (Nombre));

- BITMAP

CREATE BITMAP INDEX index_name ON table_name(c1,...)

SQL> Create BITAMAP index i1 on templa (SEXO);

- REVERSE

CREATE INDEX index_name ON table_name(c1,...) REVERSE

SQL> Create index i1 on templa (Nombre) reverse;

- INVISIBLE

CREATE INDEX index_name ON table_name(c1,...) INVISIBLE

SQL> Create index i1 on templa (Nombre) invisible;

INDICES

CREACION:

***CREATE [UNIQUE] INDEX Nombre ON
Nombre_tabla (Columna [ASC/DESC], Col2...) [TABLESPACE nombre]***

UNIQUE: significa que los datos del índice deben de ser únicos

TABLESPACE: Lugar donde se almacenara el indice.

- NOTAS
 - ✓ El número máximo de columnas dentro de un índice es de 16
 - ✓ No existe número máximo de índices para una tabla

INDICES

MODIFICACION:

ALTER INDEX Nombre

[STORAGE]

[ENABLE / DISABLE]

[REBUILD ONLINE]

[RENAME.....]

[MONITORING USAGE]

BORRADO:

DROP INDEX Nombre ;

INDICES

TABLAS y VISTA

- **USER_INDEXES**

Nos muestra información de los índices que tiene definidos el esquema actual.

- **USER_IND_COLUMNS**

Nos muestra información de los índices y columnas que los contiene dentro del esquema actual.

INDICES

NOTAS

- Generalmente mejoran el resultado de las operaciones de SELECT, pero pueden empeorar el rendimiento en operaciones de DML.
- Oracle los usa de forma automática cuando lo considera necesario, sin intervención del usuario.
- La utilización del índice es optima cuando los datos afectan hasta el 10% del total de los datos.
- Los índices se suelen guardar en sitio diferente a los datos de la tabla para optimizar el rendimiento.