



# BASE DE DATOS II

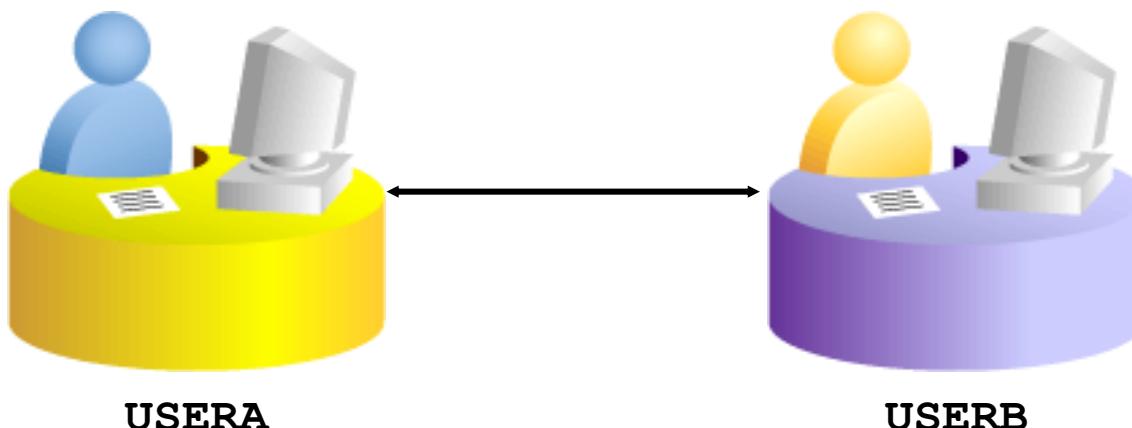
## CREACIÓN DEL ESQUEMA

# APLICACIÓN DE UN “MODELO DE DATOS” EN LA BASE DE DATOS RELACIONAL

- Transformación un modelo “conceptual” de E-R en un MODELO LÓGICO RELACIONAL de la Base de Datos
- Creación de los objetos componentes del “esquema” de la base de datos:
  - TABLAS
  - ÍNDICES
  - CONSTRAINTS
  - Aplicación de las reglas de Integridad Relacional

# Referencia a Tablas de Otro Usuario

- Un esquema (schema) es la colección de objetos de la base de datos que pertenecen a un usuario.
- Por ende, las tablas pertenecientes a otros usuarios no están en el esquema del usuario.
- Debe utilizar el nombre del propietario como prefijo de dichas tablas.



USERA

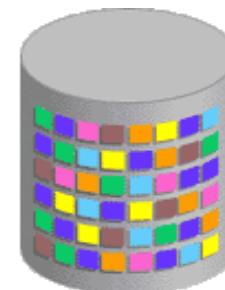
USERB

```
SELECT *  
FROM userB.b_empleados;
```

```
SELECT *  
FROM userA.b_empleados;
```

# La Sentencia CREATE TABLE

- Debe tener:
  - El privilegio CREATE TABLE
  - Un área de almacenamiento



```
CREATE TABLE [schema.]table
    (column datatype [DEFAULT expr] [, . . . ]) ;
```

- Se especifica:
  - Nombre de tabla
  - Nombre de columna, tipo de datos de columna y tamaño de columna

# No olvidar las reglas de Normalización:

Formas Normales	Descripción
Primera Forma Normal	Ningún atributo puede ser multivalorado
Segunda Forma Normal	Todos los atributos deben depender totalmente de la clave primaria completa y no solamente de parte de ella
Tercera Forma Normal	Deben evitarse dependencias transitivas

# Constraints de integridad

- Asegura consistencia de los datos
- El Servidor de Base de Datos se encarga de hacer cumplirlas
- Se aplican para las claves primarias (Primary Key), y las claves ajenas (Foreign Key)

Regla de Integridad	Descripción
De las Entidades	La clave primaria debe ser UNICA, NO NULA y MÍNIMA
Referencial	Los campos de la clave extranjera deben ser TODOS NULOS o coincidir con la clave primaria de la tabla referida
Columna	Los valores de una columna deben coincidir con el tipo de dato definido
Definidas por el usuario	Los valores deben concordar con las reglas del negocio

# Creación Tablas: Sintaxis ampliada

```
CREATE TABLE [esquema.] tabla
  (columna tipo_dato [DEFAULT expr]
  [constraint_columna] ,
  ...
  [constraint_table])
  [TABLESPACE tablespace]
  [PCTFREE entero]
  [INITTRANS entero]
  [MAXTRANS entero]
  [STORAGE (clausula de storage)]
  [LOGGING | NOLOGGING]
  [CACHE | NOCACHE] ;
```

# Parámetros de la cláusula Storage

- **initial:** tamaño, en bytes, de la primera extensión asignada al crearse la tabla (este parámetro no puede asignarse con un ALTER)
- **next:** tamaño, en bytes, de la siguiente extensión a asignar. El segundo extent es igual al valor seteado para el parámetro NEXT. Los subsiguientes tienen el tamaño del previo multiplicado por  $(1+PCTINCREASE/100)$
- **pctincrease:** Porcentaje en el que crecerá la siguiente extensión en relación con la última extensión utilizada. Si pctincrease es 0, entonces todos los extents que se van incrementando son del mismo tamaño.

# Parámetros de la cláusula Storage

- **minextents**: número de extensiones asignadas en el momento de la creación del segmento
- **maxextents**: número máximo de extensiones, incluyendo la primera que puede ser asignada para un segmento
- **pctfree**: porcentaje de espacio libre para actualizaciones de filas que se reserva dentro de cada bloque asignado al segmento .
- **pctused**: porcentaje de utilización del bloque por debajo del cual Oracle considera que un bloque puede ser utilizado para insertar filas nuevas en él.
- **tablespace**: nombre del espacio de tablas donde se creará el segmento para el objeto.

## Reglas para los nombres de tablas y columnas:

- Deben empezar con una letra
- Deben tener 1 a 30 de longitud
- Pueden contener solo los juegos de caracteres A-Z, a-z, 0-9, \_, \$ y #
- No se puede duplicar el nombre de otro objeto que pertenece al mismo usuario
- No debe ser una palabra reservada de ORACLE

# La Opción DEFAULT

- Se puede especificar un valor predefinido por una columna durante una inserción

```
... Fecha_inicio DATE DEFAULT SYSDATE, ...
```

- Los valores legales son generalmente valores literales, expresiones, o funciones SQL como SYSDATE o USER
- No se puede poner por default el nombre de otra columna
- Los valores predefinidos (default) deben emparejar el tipo de dato de la columna

# Creación de Tablas

- Cree la tabla.

```
CREATE TABLE dept
  (deptno      NUMBER(2) ,
   dname       VARCHAR2(14) ,
   loc         VARCHAR2(13) ,
   create_date DATE DEFAULT SYSDATE);
```

Table created.

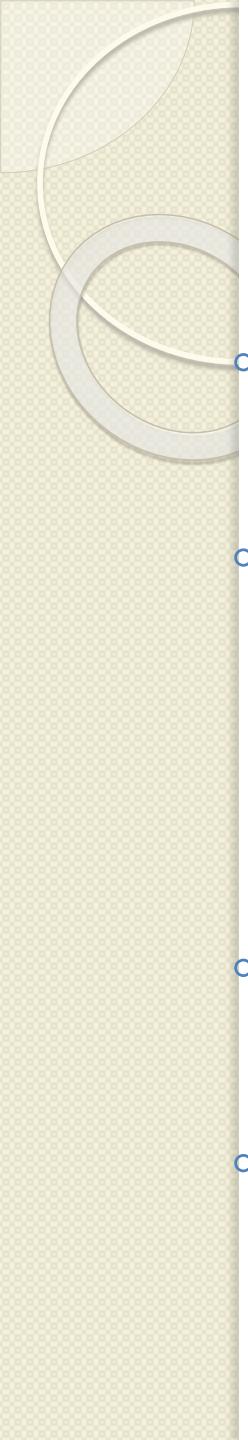
- Confirme la creación de la tabla.

```
DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

# Constraints

- Establecen restricciones a nivel de tabla
- Previene eliminaciones de una tabla si hay dependencias.
- Los tipos de constraints siguientes son válidos en Oracle:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



# Instrucciones de Restricciones

- Puede nombrar una restricción u Oracle Server generará un nombre con el formato SYS\_Cn.
- Cree una restricción en alguno de estos momentos:
  - Al mismo tiempo que se crea la tabla
  - Después de crear la tabla
- Defina una restricción de nivel de columna o de tabla.
- Visualice una restricción en el diccionario de datos.

# Definición de Restricciones (constraints)

- Sintaxis:

```
CREATE TABLE [schema.]table  
    (column datatype [DEFAULT expr]  
     [column_constraint] ,  
      ...  
     [table_constraint] [, . . . ]) ;
```

- Restricción de nivel de columna:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Restricción de nivel de tabla:

```
column, ...  
[CONSTRAINT constraint_name] constraint_type  
(column, . . . ),
```

# Definición de Restricciones

- Restricción de nivel de columna:

```
CREATE TABLE employees(
    employee_id  NUMBER(6)
        CONSTRAINT emp_emp_id_pk PRIMARY KEY,
    first_name    VARCHAR2(20),
    . . . );
```

1

- Restricción de nivel de tabla:

```
CREATE TABLE employees(
    employee_id  NUMBER(6),
    first_name    VARCHAR2(20),
    . .
    job_id        VARCHAR2(10) NOT NULL,
    CONSTRAINT emp_emp_id_pk
        PRIMARY KEY (EMPLOYEE_ID));
```

2

# Restricción NOT NULL

- Garantiza que no se permitan valores nulos en la columna:

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10
...							

20 rows selected.



**Restricción NOT NULL**  
**(Ninguna fila puede**  
**contener un valor**  
**nulo para esta columna.)**



**Restricción**  
**NOT NULL**



**Ausencia de restricción**  
**NOT NULL**  
**(Cualquier fila puede**  
**contener un valor nulo**  
**para esta columna.)**

# El constraint UNIQUE

- Impide que una columna o combinación de columnas tenga valores repetidos. El propósito del constraint UNIQUE es ligeramente diferente al del PRIMARY KEY. Las claves únicas son atributos de la entidad que no admiten duplicación, en tanto que la clave primaria tiene como principal función la de identificar cada fila únicamente
- Permite valores nulos si el constraint está definido sobre columnas que no tengan el constraint NOT NULL
- Se define a nivel de COLUMNA o a nivel de TABLA

```
... telefono VARCHAR2(10)  
      CONSTRAINT s_emp_telefono_uk UNIQUE , ...
```

# Restricción UNIQUE

- Se define en los niveles de tabla o de columna:

```
CREATE TABLE employees (
    employee_id      NUMBER(6) ,
    last_name        VARCHAR2(25) NOT NULL ,
    email            VARCHAR2(25) ,
    salary           NUMBER(8,2) ,
    commission_pct   NUMBER(2,2) ,
    hire_date        DATE NOT NULL ,
    ...
    CONSTRAINT emp_email_uk UNIQUE(email) );
```

# Restricción UNIQUE

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKING
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

└─ Restricción UNIQUE

↑ INSERT INTO

208	Smith	JSMITH
209	Smith	JSMITH

← Permitido

← No permitido: ya existe

# El constraint PRIMARY KEY

- Crea una Clave primaria para la TABLA. Se permite una sola clave primaria para cada tabla
- Automáticamente las columnas se crean como NOT NULL
- Se define a nivel de la tabla o a nivel de columnas
- Automáticamente crea un índice UNIQUE

```
... id NUMBER(7)  
CONSTRAINT p_personal_id_pk PRIMARY KEY,...
```

# Restricción PRIMARY KEY

## DEPARTMENTS

PRIMARY KEY

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

...

INSERT INTO

No permitido  
(valor nulo)

	Public Accounting		1400
50	Finance	124	1500

No permitido  
(50 ya existe)

# El constraint FOREIGN KEY

- Designa una columna o combinación de columnas como una clave extranjera
- Establece una relación entre la clave primaria en la misma tabla o entre las tablas
- Puede definirse a nivel de tabla o columna
- Debe emparejar un valor existiendo en la tabla del padre o debe ser NULO

```
... Id_departamento NUMBER(7)
      CONSTRAINT p_personal_dept_id_fk
      REFERENCES p_departamento(id) ,...
```

# Restricción FOREIGN KEY

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

PRIMARY  
KEY



EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60



↑ INSERT INTO

200	Ford	9
201	Ford	60

No permitido  
(9 no existe)

← Permitido

# Restricción FOREIGN KEY

- Se define en los niveles de tabla o de columna:

```
CREATE TABLE employees (
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY(department_id)
        REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));

```

# Restricción FOREIGN KEY:

## Palabras Clave

- **FOREIGN KEY**: Define la columna de la tabla secundaria en el nivel de restricción de tabla
- **REFERENCES**: Identifica la tabla y la columna de la tabla principal
- **ON DELETE CASCADE**: Suprime las filas dependientes de la tabla secundaria si se suprime una fila de la tabla principal
- **ON DELETE SET NULL**: Convierte valores clave de clave ajena dependientes en valores nulos

# Violación de Restricciones

```
UPDATE employees  
SET     department_id = 55  
WHERE   department_id = 110;
```

```
UPDATE employees  
*  
ERROR at line 1:  
ORA-02291: integrity constraint  
(HR.EMP_DEPT_FK) violated - parent key not  
found
```

- El departamento 55 no existe.

# Violación de Restricciones

- No se puede suprimir una fila que contenga una clave primaria que se utilice como clave ajena en otra tabla.

```
DELETE FROM departments  
WHERE          department_id = 60;
```

```
DELETE FROM departments  
          *  
ERROR at line 1:  
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)  
violated - child record found
```

# El constraint CHECK

- Define una condición que debe satisfacer cada fila
- No se permiten las expresiones siguientes:
  - Referencias a las pseudocolumnas CURRVAL, NEXTVAL, LEVEL y ROWNUM
  - Llamadas a las funciones SYSDATE, UID, USER y USERENV
  - Consultas que hagan referencia a otros valores de otras filas

```
..., salary NUMBER(2)
CONSTRAINT emp_salary_min
    CHECK (salary > 0), ...
```

# Creación Tabla: Ejemplo

```
SQL> CREATE TABLE p_departamento
  2  (id          NUMBER(7)
  3    CONSTRAINT p_departamento_id_pk PRIMARY KEY,
  4  descripcion      VARCHAR2(25)
  5    CONSTRAINT p_departamento_nombre_nn NOT NULL,
  6  id_localidad NUMBER(7)
  7    CONSTRAINT p_departamento_localidad_id_fk REFERENCES
  8        p_localidad (id),
  9    CONSTRAINT s_dept_desc_localidad_id_uk UNIQUE
 10        (descripcion, id_localidad));
```

# Creación Tabla: Ejemplo

```
CREATE TABLE B_CUENTAS
(CODIGO_CTA      NUMBER(7) not null CONSTRAINT
PKCUENTAS PRIMARY KEY,
NOMBRE_CTA      VARCHAR2(40) not null,
ID_TIPO          NUMBER(4) not null,
NIVEL            NUMBER(1) not null,
CTA_SUPERIOR    NUMBER(7),
ORDEN            VARCHAR2(1) not null,
FECHA_APERTURA DATE default SYSDATE not null,
IMPUTABLE        VARCHAR2(1) default 'S' not null)
TABLESPACE BASEDATOS2
STORAGE (INITIAL 50K NEXT 10K
          MAXEXTENTS 100
          PCTINCREASE 0);
```

# Es posible dividir las tablas en diferentes tablespaces según su rango

```
CREATE TABLE emp_adm
  ( empno                      NUMBER(5) PRIMARY KEY,
    nombre                     VARCHAR2(15) NOT NULL,
    titulo                     VARCHAR2(10),
    jefe                       NUMBER(5),
    fecha_ingreso              DATE DEFAULT (sysdate),
    salario                    NUMBER(7,2),
    id_dpto                    NUMBER(3) NOT NULL)
PARTITION BY RANGE(empno)
(PARTITION emp1000 VALUES LESS THAN (1000)
TABLESPACE admin_tbs,
PARTITION emp2000 VALUES LESS THAN (2000)
TABLESPACE admin_tbs2);
```

# También es posible dividir las tablas en diferentes tablespaces según una LISTA

Supongamos que las localidades estuvieran identificadas en aquellas que son de la región Oriental y Occidental:

```
CREATE TABLE localidades
  (id                      NUMBER(5) PRIMARY KEY,
   nombre                  VARCHAR2(15) NOT NULL,
   region                 VARCHAR2(2) NOT NULL)
  PARTITION BY LIST (region)
  (PARTITION occidental VALUES ('OC') TABLESPACE
   basedatos2,
   PARTITION oriental    VALUES      ('OR') TABLESPACE
   based2b);
```

# Creación de una Tabla mediante una Subconsulta

- Cree una tabla e inserte filas combinando la sentencia CREATE TABLE y la opción AS *subquery*.

```
CREATE TABLE table
    [ (column, column...) ]
AS subquery;
```

- Asigne el número de columnas especificadas al número de columnas de subconsulta.
- Defina columnas con nombres de columna y valores por defecto.

# Creación de Tabla usando un Subquery: Ejemplo

- Cree una tabla que contiene a todos los empleados de la categoría salarial 56

```
CREATE TABLE vendedores AS  
SELECT e.cedula, e.nombre, e.apellido,  
e.fecha_ing  
FROM b_empleados e, b_posicion_actual p  
WHERE e.cedula = p.cedula  
AND p.cod_categoria = 56
```

- No se olvide de que sólo se copia el constraint NOT NULL

# Tablas dentro del Servidor de Base de Datos

- Tablas del usuario
  - La colección de tablas creadas y mantenidas por el usuario
  - Contienen información del usuario
- Diccionario de los datos
  - La colección de tablas creadas y mantenidas por el Servidor de Base de Datos
  - Contienen información de la base de datos

# **El Diccionario de Datos**

**Puede consultar las vistas de diccionario que se basen en las tablas de diccionario para buscar información como:**

- Definiciones de todos los objetos de esquema de la base de datos (tablas, vistas, índices, sinónimos, secuencias, procedimientos, funciones, paquetes, disparadores, etc.)**
- Valores por defecto de las columnas**
- Información acerca de las restricciones de integridad**
- Nombres de usuarios Oracle**
- Privilegios y roles que se han otorgado a cada usuario**
- Otra información general de la base de datos**

# Consultas al diccionario de la BD

- Cuatro clases de vistas (prefijos)
  - USER Objetos del usuario
  - ALL Todos los objetos a los que el usuario tiene acceso
  - DBA Todos los objetos. La columna OWNER permite identificar el dueño del objeto
  - V\$ Valores actuales del servidor sobre el rendimiento del sistema
- La tabla DICTIONARY es la que contiene el nombre de todas las tablas y vistas del catálogo

**DICTIONARY**

Descripción del diccionario de tablas y vistas

# Consultas al Diccionario

- Listando todos las vistas del DD a los que el usuario tiene acceso

```
SQL> SELECT *  
2 FROM DICTIONARY;
```

- Desplegando la estructura de la vista de USER\_OBJECTS

```
SQL> DESCRIBE user_objects
```

- Desplegando todos los nombres de objetos que usted posee

```
SQL> SELECT object_name  
2 FROM user_objects  
3 WHERE object_type = 'TABLE';
```

# Consultas al Diccionario de Datos: Ejemplos

- Busque tablas de diccionario de datos en temas específicos en la columna de los COMENTARIOS de la tabla del DICCIONARIO

```
SQL> SELECT      *
  2  FROM dictionary
  3 WHERE LOWER(comments) LIKE '%grant%';
```

# Vista USER\_OBJECTS

- Describe todos los objetos de su propiedad
- Es un modo útil de obtener un listado de todos los nombres y los tipos de objeto del esquema, además de la siguiente información:
  - Fecha de creación
  - Fecha de la última modificación
  - Estado (válido o no válido)
- Consulta ALL\_OBJECTS para ver todos los objetos a los que tiene acceso

# Vista USER\_OBJECTS

Puede consultar la vista USER\_OBJECTS para ver los nombres y los tipos de todos los objetos del esquema. Hay varias columnas en esta vista:

- OBJECT\_NAME: Nombre del objeto
- OBJECT\_ID: Número de objeto
- OBJECT\_TYPE: Tipo de objeto (como TABLE, VIEW, INDEX o SEQUENCE)
- CREATED: Registro de hora de la creación del objeto
- LAST\_DDL\_TIME: Registro de hora de la última modificación del objeto resultante de un comando DDL
- STATUS: Estado del objeto (VALID, INVALID o N/A)
- GENERATED: ¿Ha generado el sistema el nombre del índice? (Y | N)

# Vista USER\_OBJECTS

```
SELECT object_name, object_type,  
created, status  
FROM user_objects  
ORDER BY object_type;
```

OBJECT_NAME	OBJECT_TYPE	CREATED	STATUS
REG_ID_PK	INDEX	10-DEC-03	VALID
...			
DEPARTMENTS_SEQ	SEQUENCE	10-DEC-03	VALID
REGIONS	TABLE	10-DEC-03	VALID
LOCATIONS	TABLE	10-DEC-03	VALID
DEPARTMENTS	TABLE	10-DEC-03	VALID
JOB_HISTORY	TABLE	10-DEC-03	VALID
JOB_GRADES	TABLE	10-DEC-03	VALID
EMPLOYEES	TABLE	10-DEC-03	VALID
JOBS	TABLE	10-DEC-03	VALID
COUNTRIES	TABLE	10-DEC-03	VALID
EMP_DETAILS_VIEW	VIEW	10-DEC-03	VALID

# Vista CAT

- Para ver una consulta y una salida simplificadas, puede consultar la vista CAT. Esta vista sólo contiene dos columnas:
  - TABLE\_NAME
  - TABLE\_TYPE.

Proporciona los nombres de todos los objetos INDEX, TABLE, CLUSTER, VIEW, SYNONYM, SEQUENCE O UNDEFINED.

# Vista USER\_TABLES

- Puede utilizar la vista USER\_TABLES para obtener los nombres de todas las tablas. La vista USER\_TABLES contiene información sobre las tablas. Además de proporcionar el nombre de la tabla, contiene información detallada sobre el almacenamiento.
  - También puede consultar las vistas ALL\_TABLES y TABS para ver un listado de todas las tablas a las que tiene acceso:
    - SELECT table\_name
    - FROM tabs;

# Información de Tabla

USER\_TABLES:

```
DESCRIBE user_tables
```

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLESPACE_NAME		VARCHAR2(30)
CLUSTER_NAME		VARCHAR2(30)
IOT_NAME		VARCHAR2(30)

```
SELECT table_name  
FROM user_tables;
```

TABLE_NAME
JOB_GRADES
REGIONS
COUNTRIES
LOCATIONS
DEPARTMENTS

...

# Vista USER\_TAB\_COLUMNS

- Utilice la vista USER\_TAB\_COLUMNS para buscar información detallada sobre las columnas de las tablas. Mientras que la vista USER\_TABLES proporciona información de los nombres de tabla y el almacenamiento, la información de columna detallada se encuentra en la vista USER\_TAB\_COLUMNS.
- Esta vista contiene información como:
  - Nombres de columnas
  - Tipos de datos de columnas
  - Longitud de tipos de datos
  - Precisión y escala para las columnas NUMBER
  - Si se permiten valores nulos (¿Hay una restricción NOT NULL en la columna?)
  - Valor por defecto

# Información de Columna

- USER\_TAB\_COLUMNS:

```
DESCRIBE user_tab_columns
```

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME	NOT NULL	VARCHAR2(30)
DATA_TYPE		VARCHAR2(106)
DATA_TYPE_MOD		VARCHAR2(3)
DATA_TYPE_OWNER		VARCHAR2(30)
DATA_LENGTH	NOT NULL	NUMBER
DATA_PRECISION		NUMBER
DATA_SCALE		NUMBER
NULLABLE		VARCHAR2(1)
COLUMN_ID		NUMBER
DEFAULT_LENGTH		NUMBER
DATA_DEFAULT		LONG

# Información de Columna

```
SELECT column_name, data_type, data_length,  
       data_precision, data_scale, nullable  
  FROM user_tab_columns  
 WHERE table_name = 'EMPLOYEES';
```

COLUMN_NAME	DATA_TYPE	DATA_LENGTH	DATA_PRECISION	DATA_SCALE	NUL
EMPLOYEE_ID	NUMBER	22	6	0	N
FIRST_NAME	VARCHAR2	20			Y
LAST_NAME	VARCHAR2	25			N
EMAIL	VARCHAR2	25			N
PHONE_NUMBER	VARCHAR2	20			Y
HIRE_DATE	DATE	7			N
JOB_ID	VARCHAR2	10			N
SALARY	NUMBER	22	8	2	Y
COMMISSION_PCT	NUMBER	22	2	2	Y
MANAGER_ID	NUMBER	22	6	0	Y
DEPARTMENT_ID	NUMBER	22	4	0	Y

# Vista de USER\_CONS\_COLUMNS

- Para buscar los nombres de las columnas a las que se aplica una restricción, consulte la vista de diccionario USER\_CONS\_COLUMNS.
- Esta vista le informa del nombre del propietario de una restricción, del nombre de la restricción, de la tabla en la que se encuentra la restricción, del nombre de la columna con la restricción.

# Viendo los Constraints

- Consulte la tabla de USER\_CONSTRAINT para ver todas las definiciones de constraints y nombres.

## Ejemplo

- Verifique los constraints en la tabla de B\_AREAS

```
SELECT constraint_name, constraint_type,  
       search_condition, r_constraint_name  
  FROM user_constraints  
 WHERE table_name = 'B_AREAS';
```

# Información de Restricción (constraint)

- USER\_CONSTRAINTS describe las definiciones de restricción de las tablas.
- USER\_CONS\_COLUMNS describe columnas de su propiedad y especificadas en restricciones.

```
DESCRIBE user_constraints
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2(30)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)

# Información de Restricción (constraint)

```
DESCRIBE user_cons_columns
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME		VARCHAR2(4000)
POSITION		NUMBER

```
SELECT constraint_name, column_name  
FROM user_cons_columns  
WHERE table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_EMAIL_UK	EMAIL
EMP_SALARY_MIN	SALARY
EMP_JOB_NN	JOB_ID
EMP_HIRE_DATE_NN	HIRE_DATE

# Información de Restricción

```
SELECT constraint_name, constraint_type,  
       search_condition, r_constraint_name,  
       delete_rule, status  
FROM   user_constraints  
WHERE  table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	CON	SEARCH_CONDITION	R_CONSTRAINT_NAME	DELETE_RULE	STATUS
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL			ENABLED
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL			ENABLED
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL			ENABLED
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL			ENABLED
EMP_SALARY_MIN	C	salary > 0			ENABLED
EMP_EMAIL_UK	U				ENABLED
EMP_EMP_ID_PK	P				ENABLED
EMP_DEPT_FK	R		DEPT_ID_PK	NO ACTION	ENABLED
EMP_JOB_FK	R		JOB_ID_PK	NO ACTION	ENABLED
EMP_MANAGER_FK	R		EMP_EMP_ID_PK	NO ACTION	ENABLED

# Información de la tabla USER\_CONSTRAINTS

- La columna CONSTRAINT\_TYPE puede ser
  - C (check constraint)
  - P (primary key)
  - U (clave unique)
  - R (integridad referencial)
  - V (with check option, sobre una vista)
  - O (with read-only, sobre una vista)
- El STATUS puede ser:
  - **ENABLED:** El constraint está activo.
  - **DISABLED:** El constraint no está activo.

# Resumen de las principales vistas de consulta frecuente:

TABLA	CONTENIDO
USER_TABLES	Tablas creadas por el usuario
TABS	Sinónimo de USER_TABLES
TAB	Vista de las tablas del usuario (hasta la versión 10g)
USER_TAB_COLUMNS	Columnas de las tablas que se encuentran en USER_TABLES
USER_OBJECTS	Objetos creados por el usuario
USER_CONSTRAINTS	Constraints definidos sobre las tablas creadas por el usuario



# **ALTERACIÓN DE OBJETOS**

# Alteración de Objetos

- El comando ALTER TABLE, le permite:
  - Agregar y modificar columnas
  - Agregar o quitar constraints
  - Habilitar o desactivar constraints
  - Eliminar una columna
- El comando DROP TABLE:
  - Elimina una tabla
- Otras órdenes que afectan una tabla son:
  - RENAME, TRUNCATE, COMMENT.
- Todas estas sentencias tienen un COMMIT automático

# Sentencia ALTER TABLE

- Utilice la sentencia ALTER TABLE para agregar, modificar o borrar columnas.

```
ALTER TABLE table
ADD      (column datatype [DEFAULT expr]
           [, column datatype]...);
```

```
ALTER TABLE table
MODIFY  (column datatype [DEFAULT expr]
           [, column datatype]...);
```

```
ALTER TABLE table
DROP      (column);
```

# Adicionando una Columna: Sintaxis

- La nueva columna se agrega con la cláusula ADD.
- Se puede predefinir un valor para la nueva columna.
- Puede especificar la restricción NOT NULL para obligar a que la columna tenga un valor

```
ALTER TABLE table
ADD  (column datatype [DEFAULT expr] [NOT NULL]
      [, column datatype] ...);
```

**Nota:** Si la tabla ya contiene filas, cuando se agrega una columna, entonces la columna es inicialmente nula, por lo que no se puede establecer un constraint de NOT NULL al inicio salvo que se agregue un valor por defecto con la cláusula DEFAULT (10g en adelante))

# Adicionando una Columna: Ejemplo

- Adicione una columna COMENTARIO a la tabla de B\_LOCALIDAD

```
ALTER TABLE b_localidad  
ADD      (comentario VARCHAR2(255));
```

- La nueva columna se vuelve la última columna.

# Modificando una Columna: Sintaxis

- Se puede cambiar:
  - el tipo de dato de una columna
  - el tamaño
  - el valor predefinido
  - el constraint NOT NULL

```
ALTER TABLE tabla
MODIFY (columna tipo_dato [DEFAULT expr] [NOT NULL]
         [, columna tipo_dato]...);
```

# Modificaciones posibles en una columna

- Aumentar el número de dígitos y la precisión de una columna numérica
- Aumentar el ancho de una columna alfanumérica
- Disminuir el ancho de una columna si la columna contiene valores nulos o si la tabla no tiene ninguna fila o si los valores no sobrepasan el nuevo ancho
- Cambiar el valor predefinido para las posteriores inserciones
- Definir un constraint NOT NULL (sólo si hay valores en la columna)

# Modificando una Columna: Ejemplo

- El tipo de datos puede modificarse si la columna no contiene valores, excepto cuando se cambia de CHAR a VARCHAR2
- El valor por defecto afecta solamente las inserciones posteriores.

## *Ejemplo*

- Extienda la longitud máxima de la columna NOMBRE en la tabla de B\_EMPLEADOS a 50 caracteres.

```
ALTER TABLE b_empleados  
MODIFY (nombre VARCHAR2(50));
```

# Borrando una Columna: Sintaxis

- Se borra una columna a la vez

```
ALTER TABLE tabla  
DROP (columna )
```

- Cuando la columna se borra:
  - Se borran los índices que incluyen a dicha columna
  - Una columna no puede ser borrada si forma parte de un constraint o de una clave, a no ser que se agregue la opción CASCADE

# La opción SET UNUSED

- Borrar una columna puede llevar mucho tiempo si la misma tiene muchos valores. En ese caso, conviene establecer que la columna ya no será utilizada hasta que se pueda borrar en un momento de menor acceso
- Cuando se marca una columna como UNUSED ya no es accesible con el comando SELECT

```
ALTER TABLE <tabla>
SET UNUSED (columna);
o bien
ALTER TABLE <tabla>
SET UNUSED COLUMN <columna>;
```

# Borrar columnas marcadas como UNUSED

- Para borrar las columnas marcadas como UNUSED se utiliza el siguiente comando:

```
ALTER TABLE <tabla>
DROP UNUSED COLUMNS;
```

- Las tablas que tienen columnas definidas como “no utilizadas” pueden verse en **USER\_UNUSED\_COL\_TABS**

# Adición de una Sintaxis de Restricción

- Utilice la sentencia ALTER TABLE para:
  - Agregar o borrar una restricción, pero sin modificar su estructura
  - Activar o desactivar restricciones
  - Agregar una restricción NOT NULL mediante la cláusula MODIFY

```
ALTER TABLE      <table_name>
ADD [CONSTRAINT <constraint_name>]
type (<column_name>);
```

# Adicionado un Constraint: Sintaxis

- Adiciona o borra, pero no modifica un constraint
- Para adicionar un constraint NOT NULL se puede usar la cláusula MODIFY.

```
SQL> ALTER TABLE  table
  2 ADD [CONSTRAINT nombre_constraint] tipo (<columna>);
```

# Adicionando un Constraint: Ejemplo

- Adicione un constraint de tipo FOREIGN KEY a la tabla de B\_EMPLEADOS que indique que un jefe ya debe existir como un empleado válido en la Tabla

```
ALTER TABLE b_empleados  
ADD CONSTRAINT s_emp_cedula_jefe_fk  
FOREIGN KEY (cedula_jefe)  
REFERENCES b_empleados(cedula);
```

# Borrando un Constraint: Ejemplo

- Borrar el constraint de integridad referencial con la tabla B\_EMPLEADOS

```
ALTER TABLE b_empleados  
DROP CONSTRAINT r_jefe_empleado;
```

- Borre el constraint PRIMARY KEY en la tabla de B\_PERSONAS y consecuentemente el FOREIGN KEY asociado en la columna de B\_COMPRAES

```
ALTER TABLE b_personas  
DROP PRIMARY KEY CASCADE;
```

# Desactivar Constraints

- La sentencia ALTER TABLE posee la cláusula DISABLE que permite deshabilitar el funcionamiento una regla de integridad.
- Aplique la opción CASCADE para desactivar constraints dependientes.

```
ALTER TABLE b_empleados  
DISABLE CONSTRAINT r_jefe_empleado;
```

# Activar Constraints

- Active un constraint actualmente desactivado en la definición de la tabla usando la cláusula ENABLE.

```
ALTER TABLE b_empleados  
ENABLE CONSTRAINT r_jefe_empleado;
```

- Si se habilita un constraint UNIQUE o PRIMARY KEY, se crea automáticamente un índice.

# Borrando una Tabla: Sintaxis

```
DROP TABLE nombre_tabla [CASCADE CONSTRAINTS];
```

```
DROP TABLE nombre_tabla PURGE;
```

- Se borra la tabla y todas las filas de la misma
- Se confirman las transacciones pendientes
- Todos los índices se borran
- La opción CASCADE CONSTRAINT elimina los constraints de integridad dependientes
- No se puede realizar un rollback de esta sentencia
- La opción PURGE libera inmediatamente el espacio en el TABLESPACE

# Sentencia FLASHBACK TABLE

- Repara una tabla de modificaciones accidentales
- Restaura una tabla a un punto anterior en el tiempo
- Es fácil de usar y ejecutar

```
FLASHBACK TABLE nombre_tabla [, nombre_tabla]...
TO {TIMESTAMP | SCN } expresión
[ {ENABLE | DISABLE} TRIGGERS ];
```

En donde SCN (System Change Number) es el número que asigna ORACLE a cada registro de REDO al hacer COMMIT de cada transacción

## Sentencia FLASHBACK TABLE – Ejemplo:

```
DROP TABLE vendedores;
```

```
SELECT original_name, operation, droptime  
FROM recyclebin;
```

```
FLASHBACK TABLE vendedores TO BEFORE DROP;
```

## **La sentencia FLASHBACK también permite recuperar transacciones DML luego del COMMIT:**

--Esta sentencia habilita la recuperación

```
ALTER TABLE B_EMPLEADOS ENABLE ROW MOVEMENT;
```

```
UPDATE B_EMPLEADOS
SET DIRECCION = 'Rio de Janeiro 598 c/San Jose'
WHERE CEDULA = 123566;
```

```
SELECT DIRECCION FROM B_EMPLEADOS
VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE CEDULA = 123566;
```

```
FLASHBACK TABLE B_EMPLEADOS TO TIMESTAMP
(SYSTIMESTAMP - INTERVAL '1' minute);
```

## Otra manera de recuperar. Construyendo un punto de restauración:

```
-- Creación de un punto de restauración
CREATE RESTORE POINT DATOS_CORRECTOS;

UPDATE B_EMPLEADOS SET DIRECCION = 'Nuevo
sitio'
WHERE CEDULA = 123566;
COMMIT;

SELECT DIRECCION FROM B_EMPLEADOS WHERE
CEDULA = 123566;

FLASHBACK TABLE B_EMPLEADOS TO RESTORE POINT
DATOS_CORRECTOS;
```

# Cambiar el Nombre de un Objeto

- Ejecute la orden RENAME para cambiar el nombre de una tabla, vista, secuencia, o sinónimo.
- Se debe ser dueño del objeto

```
SQL> RENAME b_empleados TO b_empleados_otro;  
Table renamed.
```

# Truncando una Tabla: sentencia TRUNCATE

- Elimina todas las filas de una tabla.
- Libera el espacio utilizado por esa tabla.
- Es una orden DDL, por lo tanto ejecuta un COMMIT automático

```
SQL> TRUNCATE TABLE b_liquidacion;  
Table truncated.
```

# Comentarios de la tabla

- Usted puede agregar comentarios a una tabla o columna usando la orden COMMENT
- Para limpiar el comentario use el string que simboliza NULL (' ').
- Pueden verse comentarios a través de las siguientes vistas del Diccionario de Datos:

```
SQL> ALL_COL_COMMENTS  
SQL> USER_COL_COMMENTS  
SQL> ALL_TAB_COMMENTS  
SQL> USER_TAB_COMMENTS
```

## Ejemplos:

```
SQL> COMMENT ON TABLE b_empleados IS 'Informacion del  
Empleado';  
SQL> COMMENT ON COLUMN b_empleados.fecha_ing IS 'Fecha de  
ingreso en la empresa';
```



# CREACIÓN DE ÍNDICES

# LOS ÍNDICES

- Son objetos de la Base de Datos, utilizados por la Base de Datos para acelerar la recuperación de filas a través de la utilización de punteros.
- Los índices reducen el tiempo de utilización de los dispositivos de entrada/salida, pues utiliza métodos especiales de acceso
- Los índices son totalmente independientes de las tablas
- Una vez que se definen , el Servidor se encarga de utilizarlos y actualizarlos apropiadamente.

# De qué manera se crean los índices?

- **DE MANERA AUTOMÁTICA**

Cuando se define un PRIMARY KEY o una restricción UNIQUE, el ORACLE genera automáticamente un índice

- **MANUALMENTE**

Los usuarios pueden crear índices sobre una o más columnas para acelerar el acceso a las filas

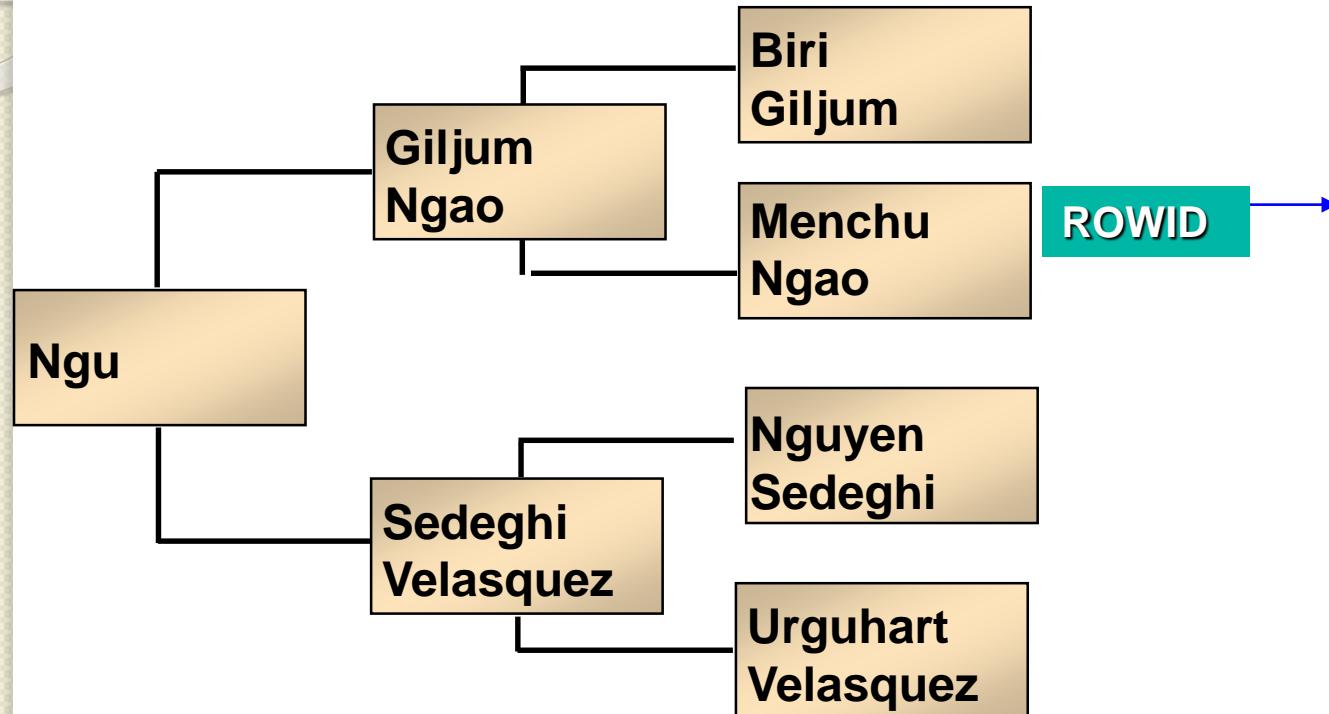
# Utilización de los índices

- Básicamente, no se requiere intervención del usuario para que el Servidor utilice los índices.
- El Servidor utiliza 2 métodos de optimización
  - Basado en reglas
  - Basado en costo

# ESTRUCTURA DE LOS ÍNDICES

- La mayoría de los índices utilizan la estructura de árboles B (B\*tree).
- Un árbol B es un índice de multinivel compuesto de los valores de las columnas y punteros que son el identificador de las filas (ROWIDs) que se organizan en páginas o ramas.
- El algoritmo consiste en buscar a través de las ramas del árbol hasta que se alcanza las hojas con el valor que contiene el puntero.

# Ejemplo de la estructura de árboles B



ID	APELLIDO
1	Velasquez
2	Ngao
3	Nagayama
4	Quick-To-See
5	Ropeburn
6	Urguhart
7	Menchu
8	Biri
9	Catchpole
10	Havel
11	Magee
12	Giljum
13	Sedeghi
14	Nguyen
15	Dumas
16	Maduro
17	Smith
18	Nozaki
19	Patel
20	Newman
21	Markarian
22	Chang
23	Patel
24	Dancs
25	Schwartz

# Tipos de Índice

- ÚNICO: asegura valores únicos en la columna y se generan automáticamente al crear una PK o una restricción UNIQUE.
- NO ÚNICOS: cuyo propósito es acelerar las consultas
- De una sola columna
- De múltiples columnas concatenadas

# Ejemplo

- Sintaxis de la creación de un índice

```
CREATE INDEX index
ON table (column[, column]...);
```

- Si se quisiera mejorar el acceso por la columna APELLIDO en la tabla b\_empleados:

```
SQL> CREATE INDEX    b_emp_apellido_idx
2  ON          b_empleados(apellido);
```

Se puede crear un índice cuando se crea una tabla

```
CREATE TABLE NUEVO_EMP  
  (id_emp NUMBER(6) PRIMARY KEY  
   USING INDEX  
    (CREATE INDEX emp_idx ON  
     NUEVO_EMP(id_emp)) ,  
   nombre VARCHAR2(20) ,  
   apellido VARCHAR2(20)) ;
```

# Índices basados en Funciones

- Una función puede estar basada en expresiones
- La expresión es construida con las columnas de la tabla, constantes, funciones SQL, y funciones definidas por el usuario

```
CREATE INDEX upper_emp_idx  
ON NUEVO_EMP(UPPER(nombre)) ;
```

**Nota:** el parámetro de inicialización  
**QUERY\_REWRITE\_ENABLED** debe estar a **TRUE**

# Cuándo crear un índice:

- La columna es usada frecuentemente en la cláusula WHERE y en las condiciones de JOIN.
- Si la columna tiene un rango de valores bien amplio
- Si se presupone que la columna tiene muchos valores nulos
- Cuando 2 o más columnas son usadas juntas en operaciones de JOIN entonces pueden utilizarse índices compuestos
- Cuando la tabla es grande y se espera que la mayoría de los queries recuperen una mínima cantidad de filas (por debajo del 5%)
- La utilización de muchos índices no siempre permite acelerar las consultas.

# Cuándo NO crear un índice:

- Cuando la tabla es muy pequeña
- Las columnas no son utilizadas frecuentemente en las consultas
- La mayoría de las consultas recuperan la totalidad o la mayor parte de las filas (mucho más del 5%)
- Cuando la tabla es actualizada con mucha frecuencia.
- La columna se referencia como parte de una expresión

# CONSULTA DE LOS ÍNDICES DE UNA TABLA:

- En la vista USER\_INDEXES puede ver los índices
- La vista USER\_IND\_COLUMNS contiene el nombre del índice y que columnas lo conforman

```
SQL> SELECT ic.index_name, ic.column_name,
  2          ic.column_position col_pos, ix.uniqueness
  3  FROM user_indexes ix, user_ind_columns ic
  4 WHERE ic.index_name = ix.index_name
  5 AND    ic.table_name = 'B_EMPLEADOS';
```

# ELIMINACIÓN DEL ÍNDICE

- Se elimina con la sentencia DROP

```
SQL> DROP INDEX s_emp_apellido_idx;  
Index dropped.
```

- Sólo se puede eliminar un índice si se es propietario o si se tiene el privilegio DROP ANY INDEX



# **CREACIÓN DE SECUENCIAS**

# Secuencia

- Genera automáticamente una numeración única y consecutiva
- Como otros objetos, puede ser compartido
- Se usa típicamente para generar los valores de una clave primaria permitiendo evitar programar dicha generación
- Cuando se indica que resida en el caché de memoria, se mejora la eficiencia del acceso

# SINTAXIS

```
CREATE SEQUENCE nombre
    [INCREMENT BY n]
    [START WITH n]
    [ {MAXVALUE n | NOMAXVALUE } ]
    [ {MINVALUE n | NOMINVALUE } ]
    [ {CYCLE | NOCYCLE } ]
    [ {CACHE n | NOCACHE } ]
```

# Creando una Secuencia: Ejemplo

- Cree la secuencia P\_AREA\_ID para ser usado para la clave primaria de la tabla B\_AREAS
- Esta secuencia no será cíclica

```
SQL> CREATE SEQUENCE p_area_id
      2      INCREMENT BY 1
      3      START WITH 51
      4      MAXVALUE 9999999
      5      NOCACHE
      6      NOCYCLE;
Sequence created.
```

# Confirmando Secuencias

- En la tabla USER\_SEQUENCES del Diccionario podrá verificar los parámetros de la secuencia

```
SQL> SELECT sequence_name, min_value,  
2      max_value, increment_by,  
3      last_number  
4 FROM user_sequences;
```

- La columna LAST\_NUMBER despliega el próximo número disponible de la secuencia.

# **Las pseudo columnas**

## **NEXTVAL y CURRVAL**

- NEXTVAL devuelve el próximo valor disponible de la secuencia
- Devuelve un único valor cada vez que es referenciado, no importa si lo usan usuarios diferentes.
- CURRVAL obtiene el valor actual de la secuencia
- CURRVAL no tendrá un valor a no ser que anteriormente se haya ejecutado un NEXTVAL

# Usando la Secuencia: Ejemplos

- Inserte una nueva area nombrada "Giraduria"

```
INSERT INTO b_areas (id, nombre_area, fecha_crea,  
activa, id_area_superior)  
VALUES (s_area_id.NEXTVAL,'Giraduria', sysdate,'S',4);
```

- Vea el valor actual para la sucesión de S\_AREA\_ID

```
SQL> SELECT p_area_id.CURRVAL  
2 FROM dual;
```

# Modificando una Secuencia: Sintaxis

- Se puede cambiar el valor de incremento, valor máximo, valor mínimo, opción del ciclo, y opción del cache

```
ALTER SEQUENCE sequence
    [INCREMENT BY n]
    [ {MAXVALUE n | NOMAXVALUE } ]
    [ {MINVALUE n | NOMINVALUE } ]
    [ {CYCLE | NOCYCLE } ]
    [ {CACHE n | NOCACHE } ]
```

# Eliminación de la secuencia

- Se elimina una secuencia usando la sentencia DROP SEQUENCE.
- Una vez eliminada la secuencia, ella ya no puede ser referenciada.

```
SQL> DROP SEQUENCE p_area_id;  
Sequence dropped.
```

# TENGA EN CUENTA LO SIGUIENTE

- Como se indicó, si se define la secuencia en el caché de memoria, facilita un acceso más rápido (Por defecto ORACLE realiza caché de 20 valores)
- Pueden perderse valores de la secuencia en las siguientes circunstancias:
  - Cuando ocurre un rollback.
  - Se genera algún error de sistema o del hardware.
  - Se utiliza la misma secuencia en alguna otra tabla o aplicación
- Las condiciones con las que se creó la secuencia, así como el siguiente valor pueden consultarse en la tabla USER\_SEQUENCES.

# Información de secuencia

```
DESCRIBE user_sequences
```

Name	Null?	Type
SEQUENCE_NAME	NOT NULL	VARCHAR2(30)
MIN_VALUE		NUMBER
MAX_VALUE		NUMBER
INCREMENT_BY	NOT NULL	NUMBER
CYCLE_FLAG		VARCHAR2(1)
ORDER_FLAG		VARCHAR2(1)
CACHE_SIZE	NOT NULL	NUMBER
LAST_NUMBER	NOT NULL	NUMBER

# Información de secuencia

- Verifique los valores de secuencia en la tabla del diccionario de datos USER\_SEQUENCES.

```
SELECT sequence_name, min_value, max_value,  
       increment_by, last_number  
FROM   user_sequences;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
LOCATIONS_SEQ	1	9900	100	3300
DEPARTMENTS_SEQ	1	9990	10	280
EMPLOYEES_SEQ	1	1.0000E+27	1	207