3

PARTICIONAMIENTO EN ORACLE

Contenido:

CAPITULO III

1. PARTICIONAMIENTO EN ORACLE

1.1. INTRODUCCIÓN

Oracle brinda una colección de comandos SQL para administrar las tablas particionadas.

El particionado es una técnica de optimización para el mejoramiento de los tiempos de respuesta de las consultas, donde las tablas son muy grandes.

Al crear una tabla se puede elegir qué rangos de datos van almacenados en un tablespace (segmento lógico) u otro, se puede elegir dónde estarán ubicados, es decir en qué discos se guardaran los datafiles (segmentos físicos) de esos tablespace. Se tiene algunas ventajas como:

- **Segmentos de datos más pequeños**: Oracle sabe en qué partición debe buscar cuando hace referencia a la tabla particionada. Esto influye directamente en el rendimiento de las búsquedas porque cada partición es tratada como si fuera una tabla diferente.
- **Índices más pequeños**: Es posible crear índices individuales para cada partición mediante la partición por rangos.

- **Respaldo más rápido**: El respaldo puede hacerse en paralelo, ya que los datos se encuentran en segmentos separados.

BENEFICIOS DEL PARTICIONAMIENTO

El particionamiento puede brindar grandes beneficios, como es una amplia variedad de aplicaciones al mejorar la capacidad de administración, el desempeño y la disponibilidad.

FUNDAMENTOS DEL PARTICIONAMIENTO

Una tabla particionada es idéntica a una tabla no particionada, no se necesitan modificaciones cuando se accede a una tabla particionada utilizando comandos SQL. El particionamiento permite subdividir una tabla, un índice o una tabla organizada por índices en partes más pequeñas. Cada partición tiene su propio nombre, y sus propias características de almacenamiento.

TIPOS DE ÍNDICES PARTICIONADOS.

- **Índices locales:**

Cada partición de un índice local corresponde a una y solo una partición de la tabla subyacente. Un índice local es un índice en una tabla particionada de la misma forma de la tabla

particionada subyacente de tal manera que hereda la partición de la tabla.

- **Índices Particionados Globales:**

Los índices globales solo pueden particionarce utilizando la partición por rango. Un índice particionado global es un índice en una tabla particionada o no particionada que se particiona utilizando una clave de particionamiento de la tabla.

- **Índices Globales No Particionados:**

Un índice global no particionado es esencialmente idéntico a un índice en una tabla no particionada. La estructura del índice no está particionada y no se acopla con la tabla subyacente.

1.2. ESTRUCTURA LÓGICA Y FÍSICA DE LA BASE DE DATOS ORACLE

NIVELES DE ALMACENAMIENTO

La Base de Datos Oracle está constituida a por los siguientes niveles que son:

- Físico: De ficheros
- Lógico: De tablespaces

ESTRUCTURA LÓGICA. Indica la composición y distribución teórica de la base de datos. La estructura lógica sirve para que las aplicaciones puedan utilizar los elementos de la base de datos sin saber realmente cómo se están almacenando. Se divide en unidades de almacenamiento lógicas como son los Tablespaces. Cada Base de Datos estará formada por uno o más tablespaces. Cada tablespace se corresponde con uno o más ficheros de datos.

El Tablespace SYSTEM se crea automáticamente al hacer la instalación de Oracle, o al crear una Base de Datos, este Tablespace contiene el diccionario de datos.

ESTRUCTURA FÍSICA. Es la estructura de los datos tan cual se almacenan en las unidades de disco. La correspondencia entre la estructura lógica y la física se almacena en la base de datos es decir en los metadatos.

Una B.D. tiene uno o más ficheros de datos. Estos ficheros son de tamaño fijo y se establecen en el momento en que se crea la base de datos o en el momento en el que se crean tablespaces.

Los datos del fichero de datos son leídos cuando se necesitan y situados en una caché de memoria compartida para que el próximo acceso a los mismos sea más rápido.

CREACIÓN DEL TABLESPACES

Una base de datos está formada por una o varias unidades lógicas llamadas tablespaces, y cada tablespaces está formado por uno o varios ficheros físicos que son los datafiles. Un datafile solamente puede pertenecer a un tablespace.

Cuando se crea una base de datos, hay que crear al menos un tablespace, por lo que durante el proceso de creación de la base de datos siempre se indica el tablespace principal de ésta, que se llama SYSTEM.

Es recomendable crear otro tablespace distinto al SYSTEM de modo que todos los nuevos usuarios, tablas e índices se almacenarán en un tablespace diferente a SYSTEM, para evitar que se bloquee toda la base de datos si ocurre algo en el tablespace SYSTEM.

COMO CREAR UN TABLESPACE.

Debemos iniciar la sesión en la base de datos con un usuario con permisos de administración, y este es el usuario SYSTEM Esto lo podemos hacer desde el SQLPLUS o desde el Toad. Recordemos que la contraseña del system fue cambiada a oracle al momento de crear la base de datos oracle.

Vamos a crear un tablespace llamado TACADEMICO para nuestra base de datos, la sentencia es la siguiente:

```
CREATE TABLESPACE TACADEMICO DATAFILE 'C:\ORACLE\ACADEMICO\TACADEMICO.DBF' SIZE 300M;
```

Con esta sentencia estamos creando en nuestra base de datos un tablespace nuevo llamado "TACADEMICO" y que está formado físicamente por un fichero (datafile) llamado TACADEMICO.dbf de 300 Mbytes y que está en el directorio "'C:\oracle\academico\", la carpeta academico fue creada previamente. Esta sentencia crea físicamente el fichero.

Ahora vamos a crear los diferentes tablespace que utilizaremos para nuestras particiones:

```
CREATE TABLESPACE TEISIC DATAFILE 'H:\ORACLE\EISIC\TEISIC.DBF' SIZE 300M;
CREATE TABLESPACE TEITEX DATAFILE 'H:\ORACLE\EITEX\TEITEX.DBF' SIZE 300M;
CREATE TABLESPACE TCIME DATAFILE 'H:\ORACLE\CIME\TCIME.DBF' SIZE 300M;
CREATE TABLESPACE TCIERCOM DATAFILE 'H:\ORACLE\CIERCOM\TCIERCOM.DBF' SIZE 300M;
CREATE TABLESPACE TESDYM DATAFILE 'H:\ORACLE\ESDYM\TESDYM.DBF' SIZE 300M;
CREATE TABLESPACE TESIIN DATAFILE 'H:\ORACLE\ESIIN\TESIIN.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE1 DATAFILE 'H:\ORACLE\DOCENTE1\TDOCENTE1.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE2 DATAFILE 'H:\ORACLE\DOCENTE2\TDOCENTE2.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE3 DATAFILE 'H:\ORACLE\DOCENTE3\TDOCENTE3.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE4 DATAFILE 'H:\ORACLE\DOCENTE4\TDOCENTE4.DBF' SIZE 300M;
```

La creación de estos tablespaces no es obligatoria, pero sí recomendable, así cada usuario de la Base de Datos tendrá su propio espacio de datos.

BORRANDO UN TABLESPACE.

Para eliminar un tablespace de la base de datos se debe utilizar la sentencia:

```
Drop tablespace TEISIC;
```

CREACIÓN DE UNA BASE DE DATOS Y TABLAS

Es sencillo crear un nuevo esquema-usuario de Oracle. Para poder realizar estos pasos es necesario iniciar la sesión en la base de datos con un usuario con permisos de administración, y este es el usuario SYSTEM:

Ahora vamos a crear el usuario que va a trabajar sobre el tablespaces creado previamente el cual contendrá a toda la base de datos.

```
CREATE USER academico IDENTIFIED BY academico DEFAULT TABLESPACE TACADEMICO;
```

PERMISOS

```
GRANT dba, connect, resource TO academico;
GRANT CREATE ANY VIEW TO academico WITH ADMIN OPTION;
```

Si no se especifica un tablespace, la Base de Datos le asignará el tablespace USERS, que es el tablespace que se utiliza por defecto para los nuevos usuarios.

ELIMINAR UN USUARIO DE LA BASE DE DATOS

Para eliminar un usuario de la Base de Datos se hace uso de la clausula DROP USER y opcionalmente se puede utilizar CASCADE, para decirle que también elimine todos los objetos creados por ese usuario.

```
DROP USER academico CASCADE;
```

PERMISOS

Se debe asignar permisos necesarios para poder trabajar con nuestra Base de datos.

Oracle incluye tres roles de sistema: CONNECT, RESOURCE y DBA, cuyos privilegios son:

Rol	Privilegios
CONNECT	alter session, create session, create cluster, create table, create view, create synonym, create sequence, create database link
RESOURCE	create cluster, create table, create procedure, create sequence, create trigger
DBA	todos los privilegios de sistema con la opcion with admin option

Privilegios de Oracle

Ahora el usuario ya puede conectarse y comenzar a trabajar sobre su esquema.

Recordemos que para ingresar a nuestro esquema debemos acordarnos del nombre y su contraseña.

Nombre: academico

Contraseña: academico

Una vez ingresado al esquema podemos crear las tabas necesarias, la sentencia es la siguiente:

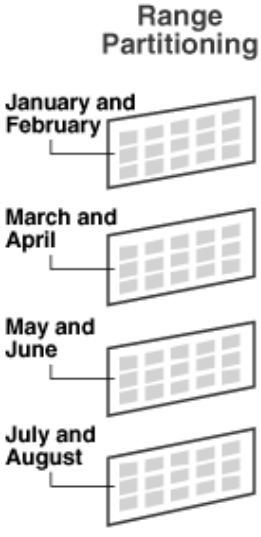
```
create table ESTUDIANTES  (
   CEDULA              VARCHAR2(10)                not null,
   NOMBRES              VARCHAR2(40),
   APELLIDOS            VARCHAR2(40),
   SEXO               CHAR(1),
    ID_ESCUELA           INTEGER,
   constraint PK_ESTUDIANTES primary key (CEDULA)
)
```

1.3. ESTRATEGIAS BÁSICAS DE PARTICIONAMIENTO

TIPOS DE PARTICIONADO EN ORACLE

- **Oracle 8.0**: Particionamiento por Rango.
- **Oracle 8i**: Particionamiento por Rango, Hash y Composite.
- **Oracle 9iR2/10g**: Particionamiento por Rango, Hash, Composite y el tipo List
- **Oracle 11g**: Columnas virtuales para particionar (que no existen fisicamente en la tabla), particionado de Sistema (donde podemos gestionar directamente en que partición de la tabla se insertan los registros) y el particionado por Intervalos.

1.3.1. PARTICIONAMIENTO POR RANGO



1. Particionamiento por Rango [IMAG.04][1]

[1] http://www.dataprix.com/blogs/respinosamilla/particionado-tablas-oracle

Los datos se distribuyen de acuerdo con el rango de valores de la clave de particionamiento. La distribución de datos es continua.

Se requiere que los registros estén identificado por un "partition key" relacionado por un predefinido rango de valores, el valor de las columnas "partition key" determina la partición a la cual pertenecerá el registro.

Se deben considerar las siguientes reglas:

- Cada partición se define con la clausula VALUES LESS THAN, la que indica el límite superior no inclusive para las particiones, cualquier valor de la clave de la partición igual o superior, es añadida a la próxima partición.
- Todas las particiones, excepto la primera, tienen un límite inferior, especificado en la clausula VALUES LESS THAN de la partición previa.
- Un literal MAXVALUE puede ser definido para la última partición; representa un valor virtual de infinito.

Ejemplo

```
create table ESTUDIANTES  (
  CEDULA            VARCHAR2(10)              not null,
  NOMBRES           VARCHAR2(40),
  APELLIDOS         VARCHAR2(40),
  SEXO              CHAR(1),
   ID_ESCUELA         INTEGER,
  constraint PK_ESTUDIANTES primary key (CEDULA)
)
 PARTITION BY RANGE (id_escuela)

  (PARTITION EISIC VALUES LESS THAN (2) TABLESPACE TEISIC,
   PARTITION EITEX VALUES LESS THAN (3) TABLESPACE TEITEX,
           PARTITION CIME VALUES LESS THAN (4) TABLESPACE TCIME,
           PARTITION CIERCOM VALUES LESS THAN (5) TABLESPACE TCIERCOM;
```

TABLA ESTUDIANTES (sin partición)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001766284	JUANA NARCISA	BUITRON DOMINGUEZ	F	2
1002222915	MARTHA CENEIDA	LOMAS NIETO	F	1
0401067871	FLOR DEL ROCIO	PABON POZO	F	3
1002673828	MARIA TATIANA	ACOSTA CAICEDO	F	2
1001645009	LOURDES ZENEIDA	VALLES FIERRO	F	4
1704464633	CARLOS VICENTE	PINEDA PALACIOS	M	2
1001607702	LUIS ALFREDO	AVENDAÑO DIAZ	M	4
0801518648	ELISA ESIL	ZAMBRANO DEL VALLE	F	4
1201467113	ARTURO NARCISO	UBE LUCES	M	1
1302112907	EUCLIDES OJILVIE	MENENDEZ LOOR	M	4
1001537321	SILVIA SUSANA	RIVERA GOMEZ	F	3
1600355794	BLADIMIR RUBEN	TERAN NARVAEZ	M	2
0802729202	MARIUXI BETSY	BONE GARRIDO	F	1
0801303348	WILVER ALEXANDER	SALAS GUERRERO	M	4
1002828588	ANGELICA CAROLA	MORAN ACOSTA	F	2
1302871817	PAULA HENOES	SANCHEZ MEZA	F	3
1304148750	JOSE RENSON	ZAMBRANO MOREIRA	M	1
1305584227	SONIA ESTHER	MACIAS LOOR	F	2
1306687565	ANTONIO GEOVANNY	ALCIVAR CHAVEZ	M	2
1307507135	GIANI GUILLERMO	CEDEÑO GARCIA	M	1

Tabla Estudiantes Particionada por Rango (EISIC)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1002222915	MARTHA CENEIDA	LOMAS NIETO	F	1
1201467113	ARTURO NARCISO	UBE LUCES	M	1
0802729202	MARIUXI BETSY	BONE GARRIDO	F	1
1304148750	JOSE RENSON	ZAMBRANO MOREIRA	M	1
1307507135	GIANI GUILLERMO	CEDEÑO GARCIA	M	1

Tabla Estudiantes Particionada por Rango (EITEX)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001766284	JUANA NARCISA	BUITRON DOMINGUEZ	F	2
1002673828	MARIA TATIANA	ACOSTA CAICEDO	F	2
1704464633	CARLOS VICENTE	PINEDA PALACIOS	M	2
1600355794	BLADIMIR RUBEN	TERAN NARVAEZ	M	2
1002828588	ANGELICA CAROLA	MORAN ACOSTA	F	2
1305584227	SONIA ESTHER	MACIAS LOOR	F	2
1306687565	ANTONIO GEOVANNY	ALCIVAR CHAVEZ	M	2

Tabla Estudiantes Particionada por Rango (CIME)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
0401067871	FLOR DEL ROCIO	PABON POZO	F	3
1001537321	SILVIA SUSANA	RIVERA GOMEZ	F	3
1302871817	PAULA HENOES	SANCHEZ MEZA	F	3

Tabla Estudiantes Particionada por Rango (CIERCOM)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001645009	LOURDES ZENEIDA	VALLES FIERRO	F	4
1001607702	LUIS ALFREDO	AVENDAÑO DIAZ	M	4
0801518648	ELISA ESIL	ZAMBRANO DEL VALLE	F	4
1302112907	EUCLIDES OJILVIE	MENENDEZ LOOR	M	4
0801303348	WILVER ALEXANDER	SALAS GUERRERO	M	4

Se puede realizar este particionamiento cuando se tiene datos que tienen rango lógicos y que pueden ser distribuidos por este, como por ejemplo: mes del año o un valor numérico.

1.3.2. PARTICIONAMIENTO POR LISTA

2. Particionamiento por Lista [IMAG.05][2]

El particionamiento por lista no soporta claves de particionamiento formada por varios atributos.

La clave de particionado es una lista de valores, que determina cada una de las particiones, la distribución de datos se define por el listado de valores de la clave de partición. El valor DEFAULT sirve para definir la partición donde iran el resto de registros que no cumplen ninguna condición de las diferentes particiones.

[2] http://www.dataprix.com/blogs/respinosamilla/particionado-tablas-oracle

Ejemplo:

```
create table MATERIAS  (
   ID_MATERIA          INTEGER   not null,
   MATERIA             VARCHAR2(50),
   ID_ESCUELA          INTEGER,
   MATERIA_ANTERIOR    INTEGER,
   ID_NIVEL            INTEGER,
   constraint PK_MATERIAS primary key (ID_MATERIA)
)
PARTITION BY LIST (id_escuela)
   (
     PARTITION EISIC values(1)TABLESPACE TEISIC,
     PARTITION EITEX values(2)TABLESPACE TEITEX,
     PARTITION CIME values(3)TABLESPACE TCIME,
     PARTITION CIERCOM values(4)TABLESPACE TCIERCOM,
     PARTITION ESDYM values(5)TABLESPACE TESDYM,
     PARTITION ESIIN values(6)TABLESPACE TESIIN)
```

TABLA MATERIAS (sin partición)

ID_MATERIA	MATERIA	ID_ESCUELA	MATERIA_ANTERIOR	ID_NIVEL
1	Analisis Matematico I	1		1
2	Analisis Matematico II	1	1	2
3	Ecuaciones Diferenciales	1	2	3
4	Matematicas Aplicadas	1	3	4
5	Modelacion y Simulacion de Computadoras	1		5
6	Fibrologia	2		1
7	Introduccion al Hilado	2		2
8	Quimica Organica I	2		3
9	Quimica Organica II	2	8	4
10	Control de Procesos I	2		5
11	Introduccion a la Ingenieria	3		1
12	Dibujo Mecanico I	3		2
13	Dibujo Mecanico II	3	12	3
14	Maquinas Herramientas	3	13	4
15	Emprendimiento e Innovacion Tecnologica	3		5
16	Tecnicas de Aprendizaje	4		1
17	Expresion Oral y Escrita	4		2
18	SistemasOperativos	4		3
19	Base de Datos	4	18	4
20	Programacion de Sistemas Multimedia	4	19	5

Tabla Materias Particionada por LIST (EISIC)

ID_MATERIA	MATERIA	ID_ESCUELA	MATERIA_ANTERIOR	ID_NIVEL
1	Analisis Matematico I	1		1
2	Analisis Matematico II	1	1	2
3	Ecuaciones Diferenciales	1	2	3
4	Matematicas Aplicadas	1	3	4
5	Modelacion y Simulacion de Computadoras	1		5

Tabla Materias Particionada por LIST (EITEX)

ID_MATERIA	MATERIA	ID_ESCUELA	MATERIA_ANTERIOR	ID_NIVEL
6	Fibrologia	2		1
7	Introduccion al Hilado	2		2
8	Quimica Organica I	2		3
9	Quimica Organica II	2	8	4
10	Control de Procesos I	2		5

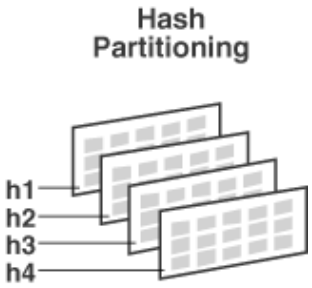
Tabla Materias Particionada por LIST (CIME)

ID_MATERIA	MATERIA	ID_ESCUELA	MATERIA_ANTERIOR	ID_NIVEL
11	Introduccion a la Ingenieria	3		1
12	Dibujo Mecanico I	3		2
13	Dibujo Mecanico II	3	12	3
14	Maquinas Herramientas	3	13	4
15	Emprendimiento e Innovacion Tecnologica	3		5

Tabla Materias Particionada por LIST (CIERCOM)

16	Tecnicas de Aprendizaje	4		1
17	Expresion Oral y Escrita	4		2
18	SistemasOperativos	4		3
19	Base de Datos	4	18	4
20	Programacion de Sistemas Multimedia	4	19	5

1.3.3. PARTICIONAMIENTO POR HASH



3. Particionamiento por Hash [IMAG.06][3]

La correspondencia entre las filas y las particiones se realiza a través de una función de hash. Es una opción útil cuando:

- Se desconoce la correspondencia en función de los rangos o no hay unos criterios de particionado claros.
- El rango de las particiones difiere sustancialmente o es difícil balancearla manualmente.

[3] http://www.dataprix.com/blogs/respinosamilla/particionado-tablas-oracle

La clave de particionado es una función hash, aplicada sobre una columna, que tiene como objetivo realizar una distribución equitativa de los registros sobre las diferentes particiones.

La funcion hash devuelve un valor automatico que determina a que partición irá el registro. Es una forma automática de balancear el particionado.

Se puede definir la partición sin indicar los nombres de las particiones, solo poniendo el número de particiones deseadas.

Ejemplo:

```
create table DOCENTES  (
   CEDULA_DOCENTE      VARCHAR2(12)                not null,
   NOMBRES             VARCHAR2(40),
   APELLIDOS           VARCHAR2(40),
   ID_TITULOS          INTEGER,
   constraint PK_DOCENTES primary key (CEDULA_DOCENTE)
)
 PARTITION BY HASH (cedula_docente)
    partitions 4 store in (tdocente1, tdocente2,tdocente3,tdocente4);
```

El sistema creará automáticamente los nombres asignando cada partición a un diferente tablespace.

Igualmente, se pueden indicar los nombres de cada particion individual o los tablespaces donde se localizaran cada una de ellas:

Ejemplo:

```
create table DOCENTES  (
   CEDULA_DOCENTE      VARCHAR2(12)                not null,
   NOMBRES             VARCHAR2(40),
   APELLIDOS           VARCHAR2(40),
   ID_TITULOS          INTEGER,
   constraint PK_DOCENTES primary key (CEDULA_DOCENTE)
)  PARTITION BY HASH (cedula_docente)
    (
        PARTITION DOCENT1 TABLESPACE tdocente1,
        PARTITION DOCENT2 TABLESPACE tdocente2,
        PARTITION DOCENT3 TABLESPACE tdocente3,
        PARTITION DOCENT4 TABLESPACE tdocente4);
```

TABLA DOCENTES (sin partición)

CEDULA_DOCENTE	NOMBRES	APELLIDOS	ID_TITULOS
1001	JAIME	AGUAS	1
1002	HUMBERTO	BRAVO	1
1003	CATALINA	RAMIREZ	2
1004	'ENDERSON	LARA	2
1005	PAUL	ANDRADE	2
1006	IRVING	REASCOS	1
1007	JORGE	CARAGUAY	1
1008	RODRIGO	NARANJO	1
1009	MARCELO	JURADO	1
1010	HUGO	IMBAQUINGO	4
1011	WIDMAR	AGUILAR	1
1012	DANIEL	JARAMILLO	1
1013	MIGUEL	ORQUERA	1
1014	JORGE	PORRAS	1
1015	JAIME	ALVARADO	1
1016	NANCY	CERVANTES	1
1017	JOSE	HUACA	1
1018	LUIS	ROMAN	5
1019	IVAN	GARCIA	1
1020	CARPIO	PINEDA	1

Tabla Docentes Particionada por HASH (DOCENTE1)

CEDULA_DOCENTE	NOMBRES	APELLIDOS	ID_TITULOS
1001	JAIME	AGUAS	1
1002	HUMBERTO	BRAVO	1
1003	CATALINA	RAMIREZ	2
1004	'ENDERSON	LARA	2
1005	PAUL	ANDRADE	2

Tabla Docentes Particionada por HASH (DOCENTE2)

CEDULA_DOCENTE	NOMBRES	APELLIDOS	ID_TITULOS
1006	IRVING	REASCOS	1
1007	JORGE	CARAGUAY	1
1008	RODRIGO	NARANJO	1
1009	MARCELO	JURADO	1
1010	HUGO	IMBAQUINGO	4

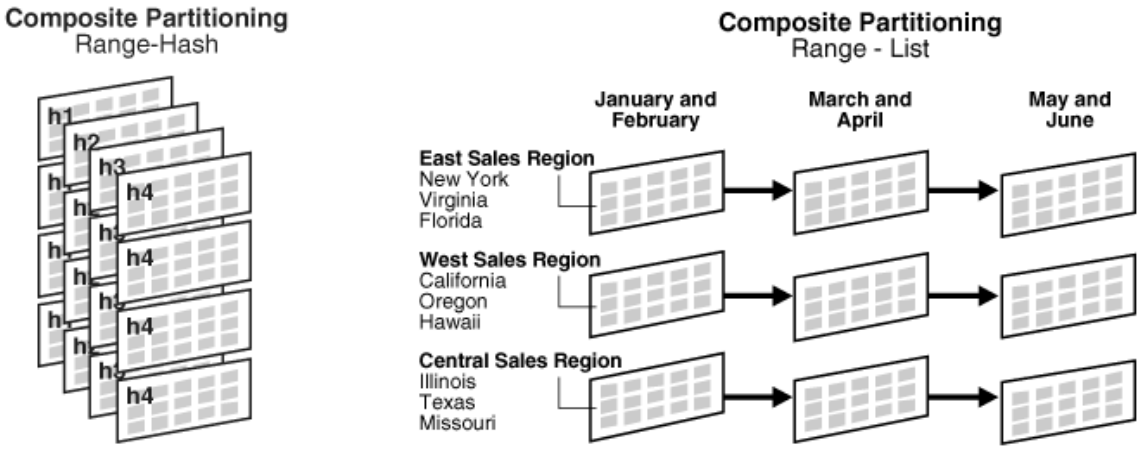
Tabla Docentes Particionada por HASH (DOCENTE3)

CEDULA_DOCENTE	NOMBRES	APELLIDOS	ID_TITULOS
1011	WIDMAR	AGUILAR	1
1012	DANIEL	JARAMILLO	1
1013	MIGUEL	ORQUERA	1
1014	JORGE	PORRAS	1
1015	JAIME	ALVARADO	1

Tabla Docentes Particionada por HASH (DOCENTE4)

CEDULA_DOCENTE	NOMBRES	APELLIDOS	ID_TITULOS
1016	NANCY	CERVANTES	1
1017	JOSE	HUACA	1
1018	LUIS	ROMAN	5
1019	IVAN	GARCIA	1
1020	CARPIO	PINEDA	1

1.3.4. PARTICIÓN POR COMPOSICIÓN



4. Particionamiento por Composición [IMAG.07][4]

Oracle nos permite utilizar métodos de particionado compuestos, ya que se conjuga el uso de dos particionados a la vez. Primero la tabla se particiona con un primer método de distribución de datos

[4] http://www.dataprix.com/blogs/respinosamilla/particionado-tablas-oracle

y luego cada partición se vuelve a dividir en subparticiones utilizando un segundo método de distribución de datos.

Las técnicas de partición compuesta disponibles son:

- Rango - Hash
- Rango - List
- Rango - Rango
- List - Rango
- List - List
- List - Hash

Las particiones más generales se hacen con el método de rango, cada partición se sub-particiona con el método de hash o por lista.

Ejemplo:

```
/*==============================================================*/
/* Table: ESTUDIANTES   PARTITION COMPOSICION  RANGE Y HASH    */
/*==============================================================*/

create table ESTUDIANTES  (
   CEDULA              VARCHAR2(10)  not null,
   NOMBRES              VARCHAR2(40),
   APELLIDOS           VARCHAR2(40),
   SEXO               CHAR(1),
    ID_ESCUELA          INTEGER,
   constraint PK_ESTUDIANTES primary key (CEDULA)
)
  PARTITION BY RANGE (id_escuela)
          SUBPARTITION BY HASH (SEXO)
          SUBPARTITION TEMPLATE  (
          SUBPARTITION MASCULINO,
          SUBPARTITION FEMENINO)

(PARTITION EISIC VALUES LESS THAN (2) TABLESPACE TEISIC,
PARTITION EITEX VALUES LESS THAN (3) TABLESPACE TEITEX,
PARTITION CIME VALUES LESS THAN (4) TABLESPACE TCIME,
PARTITION CIERCOM VALUES LESS THAN (5) TABLESPACE TCIERCOM,);
```

TABLA ESTUDIANTES (sin partición)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001766284	JUANA NARCISA	BUITRON DOMINGUEZ	F	2
1002222915	MARTHA CENEIDA	LOMAS NIETO	F	1
0401067871	FLOR DEL ROCIO	PABON POZO	F	3
1002673828	MARIA TATIANA	ACOSTA CAICEDO	F	2
1001645009	LOURDES ZENEIDA	VALLES FIERRO	F	4
1704464633	CARLOS VICENTE	PINEDA PALACIOS	M	2
1001607702	LUIS ALFREDO	AVENDAÑO DIAZ	M	4
0801518648	ELISA ESIL	ZAMBRANO DEL VALLE	F	4
1201467113	ARTURO NARCISO	UBE LUCES	M	1
1302112907	EUCLIDES OJILVIE	MENENDEZ LOOR	M	4
1001537321	SILVIA SUSANA	RIVERA GOMEZ	F	3
1600355794	BLADIMIR RUBEN	TERAN NARVAEZ	M	2
0802729202	MARIUXI BETSY	BONE GARRIDO	F	1
0801303348	WILVER ALEXANDER	SALAS GUERRERO	M	4
1002828588	ANGELICA CAROLA	MORAN ACOSTA	F	2
1302871817	PAULA HENOES	SANCHEZ MEZA	F	3
1304148750	JOSE RENSON	ZAMBRANO MOREIRA	M	1
1305584227	SONIA ESTHER	MACIAS LOOR	F	2
1306687565	ANTONIO GEOVANNY	ALCIVAR CHAVEZ	M	2
1307507135	GIANI GUILLERMO	CEDEÑO GARCIA	M	1

Tabla Estudiantes Particionada por Rango (EISIC)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1002222915	MARTHA CENEIDA	LOMAS NIETO	F	1
1201467113	ARTURO NARCISO	UBE LUCES	M	1
0802729202	MARIUXI BETSY	BONE GARRIDO	F	1
1304148750	JOSE RENSON	ZAMBRANO MOREIRA	M	1
1307507135	GIANI GUILLERMO	CEDEÑO GARCIA	M	1

Tabla Estudiantes Sub-Particionada por Hash (Masculino-EISIC)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1201467113	ARTURO NARCISO	UBE LUCES	M	1
1304148750	JOSE RENSON	ZAMBRANO MOREIRA	M	1
1307507135	GIANI GUILLERMO	CEDEÑO GARCIA	M	1

Tabla Estudiantes Sub-Particionada por Hash (Femenino-EISIC)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1002222915	MARTHA CENEIDA	LOMAS NIETO	F	1
0802729202	MARIUXI BETSY	BONE GARRIDO	F	1

Tabla Estudiantes Particionada por Rango (EITEX)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001766284	JUANA NARCISA	BUITRON DOMINGUEZ	F	2
1002673828	MARIA TATIANA	ACOSTA CAICEDO	F	2
1704464633	CARLOS VICENTE	PINEDA PALACIOS	M	2
1600355794	BLADIMIR RUBEN	TERAN NARVAEZ	M	2
1002828588	ANGELICA CAROLA	MORAN ACOSTA	F	2
1305584227	SONIA ESTHER	MACIAS LOOR	F	2
1306687565	ANTONIO GEOVANNY	ALCIVAR CHAVEZ	M	2

Tabla Estudiantes Sub-Particionada por Hash (Masculino-EITEX)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1704464633	CARLOS VICENTE	PINEDA PALACIOS	M	2
1600355794	BLADIMIR RUBEN	TERAN NARVAEZ	M	2
1306687565	ANTONIO GEOVANNY	ALCIVAR CHAVEZ	M	2

Tabla Estudiantes Sub-Particionada por Hash (Femenino-EITEX)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001766284	JUANA NARCISA	BUITRON DOMINGUEZ	F	2
1002673828	MARIA TATIANA	ACOSTA CAICEDO	F	2
1002828588	ANGELICA CAROLA	MORAN ACOSTA	F	2
1305584227	SONIA ESTHER	MACIAS LOOR	F	2

Tabla Estudiantes Particionada por Rango (CIME)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
0401067871	FLOR DEL ROCIO	PABON POZO	F	3
1001537321	SILVIA SUSANA	RIVERA GOMEZ	F	3
1302871817	PAULA HENOES	SANCHEZ MEZA	F	3

Tabla Estudiantes Sub-Particionada por Hash (Masculino-CIME)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA

Tabla Estudiantes Sub-Particionada por Hash (Femenino-CIME)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
0401067871	FLOR DEL ROCIO	PABON POZO	F	3
1001537321	SILVIA SUSANA	RIVERA GOMEZ	F	3
1302871817	PAULA HENOES	SANCHEZ MEZA	F	3

Tabla Estudiantes Particionada por Rango (CIERCOM)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001645009	LOURDES ZENEIDA	VALLES FIERRO	F	4
1001607702	LUIS ALFREDO	AVENDAÑO DIAZ	M	4
0801518648	ELISA ESIL	ZAMBRANO DEL VALLE	F	4
1302112907	EUCLIDES OJILVIE	MENENDEZ LOOR	M	4
0801303348	WILVER ALEXANDER	SALAS GUERRERO	M	4

Tabla Estudiantes Sub-Particionada por Hash (Masculino-CIERCOM)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001607702	LUIS ALFREDO	AVENDAÑO DIAZ	M	4
1302112907	EUCLIDES OJILVIE	MENENDEZ LOOR	M	4
0801303348	WILVER ALEXANDER	SALAS GUERRERO	M	4

Tabla Estudiantes Sub-Particionada por Hash (Femenino-CIERCOM)

CEDULA	NOMBRES	APELLIDOS	SEXO	ID_ESCUELA
1001645009	LOURDES ZENEIDA	VALLES FIERRO	F	4
0801518648	ELISA ESIL	ZAMBRANO DEL VALLE	F	4

```
/*==============================================================*/
/* Table: ESTUDIANTES   PARTITION COMPOSICION  RANGE Y LIST     */
/*==============================================================*/

create table ESTUDIANTES  (
   CEDULA              VARCHAR2(10)  not null,
   NOMBRES              VARCHAR2(40),
   APELLIDOS            VARCHAR2(40),
   SEXO                CHAR(1),
    ID_ESCUELA           INTEGER,
   constraint PK_ESTUDIANTES primary key (CEDULA)
)
  PARTITION BY RANGE (ID_ESCUELA)
            SUBPARTITION BY LIST (SEXO)
            SUBPARTITION TEMPLATE
            (SUBPARTITION MASCULINO values('M'),
             SUBPARTITION FEMENINO values('F'))

(PARTITION EISIC VALUES LESS THAN (2) TABLESPACE TEISIC,
PARTITION EITEX VALUES LESS THAN (3) TABLESPACE TEITEX,
PARTITION CIME VALUES LESS THAN (4) TABLESPACE TCIME,
PARTITION CIERCOM VALUES LESS THAN (5) TABLESPACE TCIERCOM,
PARTITION ESDYM VALUES LESS THAN (6) TABLESPACE TESDYM,
PARTITION ESIIN VALUES LESS THAN (7) TABLESPACE TESIIN );
```

1.3.5. EXTENSIONES DE PARTICIONAMIENTO

Oracle Database 11g brinda extensiones de particionamiento, su principal objetivo es:

- Mejorar significativamente la capacidad de administración de una tabla particionada.
- Extender la flexibilidad para definir una clave de particionamiento.

Las extensiones son:

- Particionamiento por Intervalos
- Particionado System
- Particionamiento basado en Columnas Virtuales

PARTICIONAMIENTO POR INTERVALOS

El particionado Interval es complementario a las técnicas de particionado vistas anteriormente.

El particionamiento por intervalos mejora notablemente la capacidad de administración de una tabla particionada.

Oracle extiende las capacidades del método de rangos para definir los rangos igualmente particionados utilizando una definición de intervalo, en lugar de especificar manualmente los rangos individuales.

Oracle creará automáticamente las particiones a medida que sea necesario, cuando se inserta un nuevo registro en la base de datos.

Las técnicas de particionamiento son:

- Interval
- Interval - List
- Interval - Range
- Interval - Hash

Ejemplo:

```
CREATE TABLE interval_tab (
  id           NUMBER,
  code         VARCHAR2(10),
  description  VARCHAR2(50),
  created_date DATE
)
PARTITION BY RANGE (created_date)
INTERVAL (NUMTOYMINTERVAL(1,'MONTH'))
(
   PARTITION part_01 values LESS THAN (TO_DATE('01-NOV-2007','DD-MON-YYYY'))

);
```

Mientras se ingresa valores menores se ingresaran a la partición existente caso contrario se crean nuevas particiones

```
INSERT INTO interval_tab VALUES (1, 'ONE', 'One', TO_DATE('16-OCT-2007', 'DD-MON-YYYY'));
INSERT INTO interval_tab VALUES (2, 'TWO', 'Two', TO_DATE('31-OCT-2007', 'DD-MON-YYYY'));
```

Como son fechas menores no tenemos ningún inconveniente

ID	CODE	DESCRIPCIÓN	CREATED_DATE
1	One	One	16/10/2007
2	two	two	31/10/2007

Si agregamos datos fuera del alcance de la partición existente, se creara una nueva partición.

```
INSERT INTO interval_tab VALUES (3, 'THREE', 'Three', TO_DATE('01-NOV-2007', 'DD-MON-YYYY'));
INSERT INTO interval_tab VALUES (4, 'FOUR', 'Four', TO_DATE('30-NOV-2007', 'DD-MON-YYYY'));
```

ID	CODE	DESCRIPCIÓN	CREATED_DATE
1	One	One	16/10/2007
2	two	two	31/10/2007
3	three	three	01/11/2007
4	four	four	30/11/2007

Las particiones quedan de la siguiente manera:

La partición que nosotros creamos se llama partition(part_01)

ID	CODE	DESCRIPCIÓN	CREATED_DATE
1	One	One	16/10/2007
2	two	two	31/10/2007

El sistema creó una nueva partición con el nombre partition(sys_p41)

ID	CODE	DESCRIPCIÓN	CREATED_DATE
3	three	three	01/11/2007
4	four	four	30/11/2007

PARTICIONADO SYSTEM

El particionado system es útil para aplicaciones nosotros gestionemos la forma en la que se realiza el particionado.

Oracle no realiza la gestión del lugar donde se almacenaran los registros, ya que seremos nosotros los que debemos indicar en qué partición se hacen las inserciones.

Ejemplo:

```
create table ESCUELAS  (
   ID_ESCUELA          INTEGER  not null,
   ID_FACULTAD          INTEGER,
   ESCUELA             VARCHAR2(100),
   SIGLAS              VARCHAR2(10),
   constraint PK_ESCUELAS primary key (ID_ESCUELA)
)
    PARTITION BY SYSTEM  (
         partition p1,
         partition p2,
         partition p3,
         partition p4,
         partition p5,
         partition p6);
```

Al momento de realizar un insert sobre la tabla se debería hacer lo siguiente:

```
INSERT INTO ESCUELAS PARTITION (P1) VALUES (1, 1, 'Ingenieria en Sistemas', 'EISIC');
INSERT INTO ESCUELAS PARTITION (P2)  Values (2, 1, 'Ingenieria Textil', 'EITEX');
INSERT INTO ESCUELAS PARTITION (P3) Values (3, 1, 'Ingenieria Mecatronica', 'CIME');
INSERT INTO ESCUELAS PARTITION (P4) Values (4, 1, 'Ingenieria en Electronica y Redes de
Comunicacion', 'CIERCOM');
INSERT INTO ESCUELAS PARTITION (P5) Values (5, 1, 'Ingenieria Diseño Textil y Moda', 'ESDYM');
Insert into ESCUELAS  PARTITION (P6) Values (6, 1, 'Ingenieria Industrial', 'ESIIN');
COMMIT;
```

Oracle nos permite definir sentencias SQL haciendo referencia a las particiones, como SELECT, INSERT, UPDATE, DELETE, LOCK TABLE :

```
SELECT * FROM schema.table PARTITION(part_name);
```

Esta sintaxis nos proporciona una forma simple de acceder a las particiones individuales como si fueran tablas, y utilizarlas,

PARTICIONAMIENTO BASADO EN COLUMNAS VIRTUALES

El Particionamiento basado en columnas virtuales es soportado con las estrategias básicas de particionamiento, se pueden definir en las tablas columnas virtuales.

Las columnas virtuales pueden ser utilizadas para realizar particionamiento sobre ellas.

Anteriormente una tabla solo podía ser particionada si la clave de partición existía fisicamente en la tabla. Las columnas virtuales, eliminan esa restricción y permite que la clave de particionamiento se defina por una expresión, utilizando una o más columnas existentes de una tabla, y almacenando la expresión como metadatos solamente.

Ejemplo:

Se crea una tabla con una columna virtual que represente la primera letra del nombre de usuario de la tabla, para lo que se va a

particionar mediante la primera letra del nombre, si está entre las letras de la "a" a la "g" se insertara en la partición part_a_g, y asi sucesivamente.

```
CREATE TABLE users (
  id          NUMBER,
  username    VARCHAR2(20),
  first_letter VARCHAR2(1)
    GENERATED ALWAYS AS
      (
        UPPER(SUBSTR(TRIM(username), 1, 1))
      ) VIRTUAL
)
PARTITION BY LIST (first_letter)
(
  PARTITION part_a_g VALUES ('A','B','C','D','E','F','G'),
  PARTITION part_h_n VALUES ('H','I','J','K','L','M','N'),
  PARTITION part_o_u VALUES ('O','P','Q','R','S','T','U'),
  PARTITION part_v_z VALUES ('V','W','X','Y','Z')
);
```

Ingresando los datos de la tabla, debemos tomar en cuenta que la columna virtual no se ingresa ya que ella la obtiene de la misma tabla.

```
INSERT INTO users (id, username) VALUES (1, 'Andy Pandy');
INSERT INTO users (id, username) VALUES (1, 'Burty Basset');
INSERT INTO users (id, username) VALUES (1, 'Harry Hill');
INSERT INTO users (id, username) VALUES (1, 'Iggy Pop');
INSERT INTO users (id, username) VALUES (1, 'Oliver Hardy');
INSERT INTO users (id, username) VALUES (1, 'Peter Pervis');
INSERT INTO users (id, username) VALUES (1, 'Veruca Salt');
INSERT INTO users (id, username) VALUES (1, 'Wiley Cyote');
COMMIT;
```

ID	USERNAME	FIRST_LETTER
1	Andy Pandy	A
1	Burty Basset	B
1	Harry Hill	H
1	Iggy Pop	I
1	Oliver Hardy	O
1	Peter Pervis	P
1	Veruca Salt	V
1	Wiley Cyote	W

Entonces las particiones quedarían de la siguiente manera

PARTICION (PART_A_G)

ID	USERNAME	FIRST_LETTER
1	Andy Pandy	A
1	Burty Basset	B

PARTICION (PART_H_N)

ID	USERNAME	FIRST_LETTER
1	Harry Hill	H
1	Iggy Pop	I

PARTICION (PART_O_U)

ID	USERNAME	FIRST_LETTER
1	Oliver Hardy	O
1	Peter Pervis	P

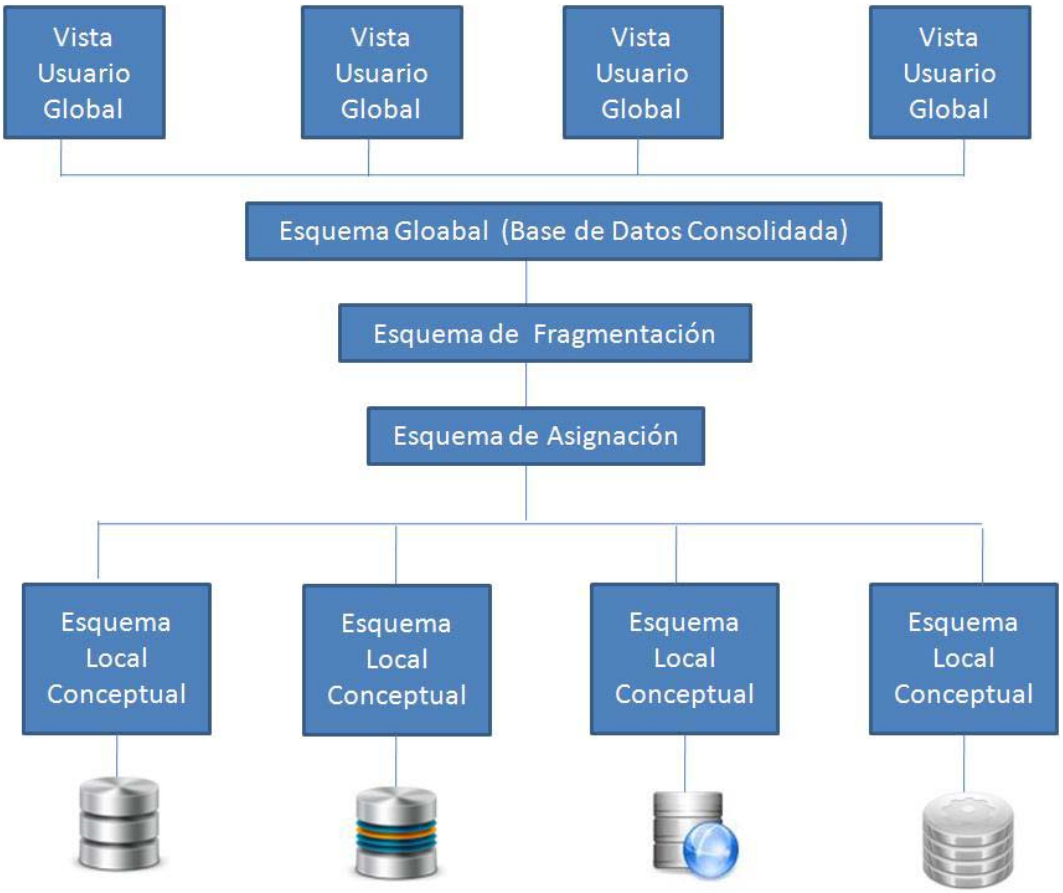
PARTICION (PART_V_Z)

ID	USERNAME	FIRST_LETTER
1	Veruca Salt	V
1	Wiley Cyote	W

1.4. BASES DE DATOS DISTRIBUIDAS EN ORACLE

Un sistema (homogéneo) de bases de datos distribuidas en Oracle es una red de Bases de Datos Oracle que residen en uno o más servidores de modo que es posible acceder a sus datos como si de una única BD se tratara.

El software de red Oracle Net debe ejecutarse en todos los servidores para hacer posible la comunicación entre las Bases de Datos.



1.5. CREACIÓN Y USO DE DBLINKS EN ORACLE

Un Database Link (DBLink) en Oracle permite realizar una conexión desde una base de datos a otra, esta es la manera más sencilla de acceder a tablas y vistas (*views*) de otra base de datos Oracle.

El objetivo principal del DBLink es ocultar los detalles de conexión, facilitando el acceso a los recursos en otras bases de datos, independientemente de que éstas se encuentren instaladas en el mismo servidor o no.

Los DBLinks se crean en la base de datos local utilizando el comando PL/SQL o SQL **CREATE DATABASE LINK.**

El usuario que ejecute el comando anterior debe tener los permisos necesarios para poder hacerlo. La sintaxis del comando es el sigue:

```
CREATE DATABASE LINK Nombre_dblink  CONNECT TO  Nombre_usuario INDENTIFIED BY
Contraseña
USING 'Cadena_conexion';
```

Donde:

Nombre_dblink: Es el nombre del DBLink.

Nombre_usuario y **Contraseña:** Son los identificadores que utilizará el DBLink para conectarse a la base de datos remota.

Cadena_conexion: Identifica a la base de datos remota, este puede ser el nombre de la instancia, esta se define en el archivo tnsnames.ora de la base de datos origen.

```
create database link LINKHILARY connect to academico identified by academico using 'DBHILARY';
```

BORRAR UN DBLINK

```
DROP DATABASE LINK nombreLink;
```

DBLINK COMO PUBLIC

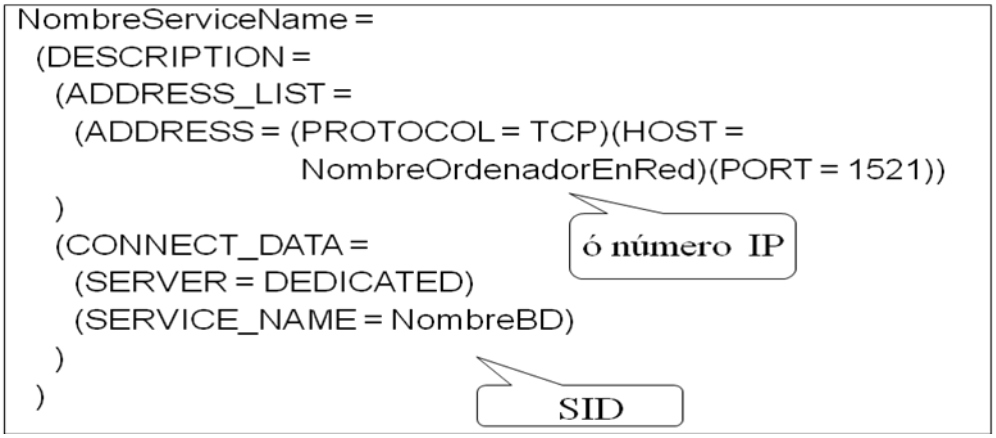
Una vez creado el DBLink, todos los usuarios tendrán acceso al mismo, para referenciar una tabla o vista de la base de datos remota se debe indicar el nombre de la tabla o vista, junto con el carácter "@" y el nombre del DBLink.

Las tablas y vistas remotas pueden usar consultas es decir una sentencia SELECT, se podrán ejecutar también sentencias del tipo DELETE, INSERT, UPDATE o LOCK TABLE.

NOMBRE DE SERVICIO

Cada Base de Datos es identificada únicamente en una Base de Datos Distribuida por un nombre global de Base de Datos, éste consta del nombre de la Base de Datos junto con el nombre del host en la red en la que esta Bases de Datos está ubicada, el nombre se hace transparente al usuario mediante el uso de nombres de servicio como es (service names) en la definición de los enlaces como links.

Los nombres de servicio se definen en el archivo tnsnames.ora de Oracle, cuya ubicación depende del cada ordenador: D:\oracle\product\10.2.0\db_1\NETWORK\ADMIN \tnsnames.ora




```
DBHILARY =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.1)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ORCL)
    )
  )
```

CONSULTA A BASES DE DATOS DISTRIBUIDAS REMOTAS Y OBJETOS REMOTOS VIA LINKS

El nombre de un objeto en una Base de Datos es único dentro del esquema. En una Base de Datos remota puede existir un esquema con el mismo nombre y un objeto con el mismo nombre.

El acceso a través de un link a un objeto remoto de un determinado propietario en una Base de Datos remota se realiza mediante la siguiente sentencia:

Popietario.nombreObjeto@nombreLink

ACADEMICO.ESTUDIANTES@LINKHILARY

Para realizar consultas en una Base de Datos Distribuida podemos utilizar objetos situados en una Base de Datos remota, con la siguiente sentencia.

```
select * from MATERIAS@LINKHILARY  WHERE ID_ESCUELA=6;
```

1.6. VISTAS EN ORACLE

Una vista es una tabla derivada de otras tablas, estas pueden ser básicas o virtuales. Una vista se caracteriza porque, se considera que forma parte del esquema externo, es una tabla virtual la cual no tiene una correspondencia a nivel físico, las vistas se pueden consultar como cualquier tabla básica, sus actualizaciones se realizan a la o las tablas originales

Se puede utilizar vistas por las siguientes razones:

- Para restringir el acceso a la B.D.
- Para realizar consultas complejas de manera fácil.
- Para obtener una independencia de los datos
- Para presentar diferentes vistas de los mismos datos.

La sintaxis para la creación de vistas en SQL es la siguiente:

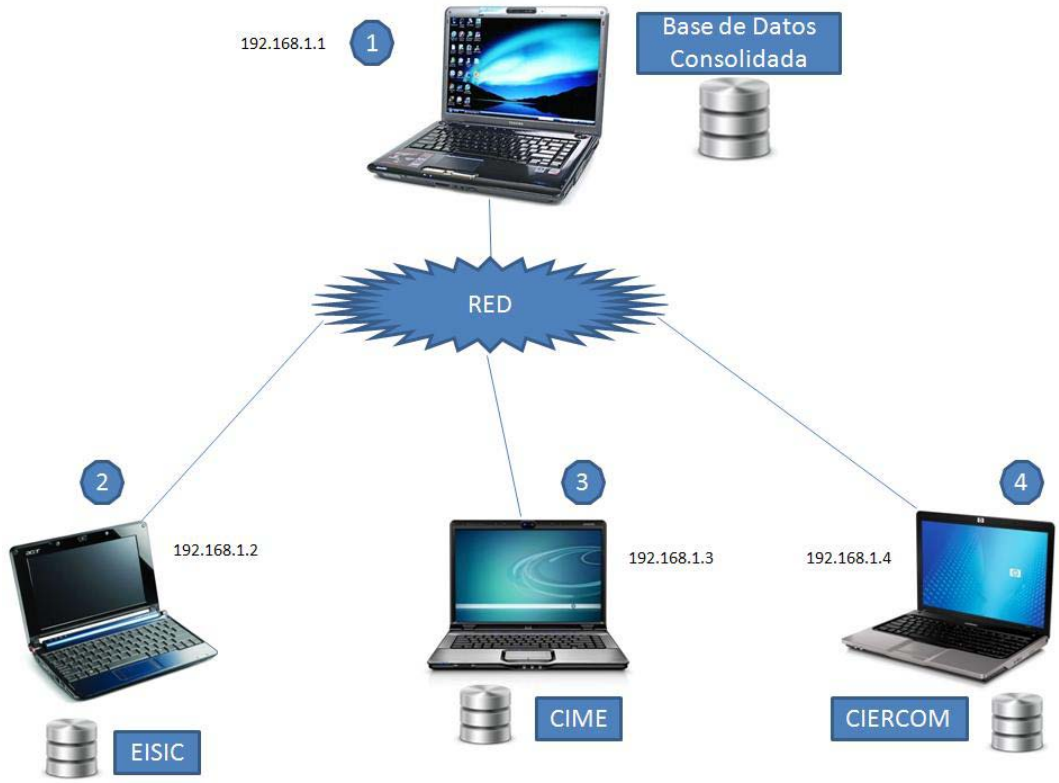
```
CREATE  VIEW vista AS expresión_tabla
Donde:
CREATE VIEW es la orden que permite la creación de la vista.
vista es el nombre de la tabla virtual que se va a crear.
expresión_tabla es una consulta SQL cuyo resultado será el contenido de la vista.
```

CREATE VIEW MATERIAS AS select * from MATERIAS@LINKHILARY WHERE ID_ESCUELA=1;

Para la eliminación de una vista se utiliza la instrucción:

DROP VIEW *vista* [restrict | cascade];

EJEMPLO DE DBLINKS Y VISTAS EN ORACLE



5. Ejemplo de Dblink Y Vistas en Oracle

ORDENADOR 1

Aquí se encuentra la Base de Datos Consolidada en la cual tendremos las diferentes particiones.

Creamos el esquema de nuestra base con su respectivo tablespace, la sentencia es la siguiente:

```
CREATE TABLESPACE TACADEMICO DATAFILE 'C:\oracle\academico\TACADEMICO.DBF' SIZE 300M;

CREATE USER academico IDENTIFIED BY academico DEFAULT TABLESPACE TACADEMICO;

GRANT dba, connect, resource TO academico;

GRANT CREATE ANY VIEW TO academico WITH ADMIN OPTION;
```

No debemos olvidar la creación de los diferentes tablespace en los que serán almacenados las diferentes particiones. Las sentencias son las siguientes:

```
CREATE TABLESPACE TEISIC DATAFILE 'H:\oracle\eisic\TEISIC.DBF' SIZE 300M;
CREATE TABLESPACE TEITEX DATAFILE 'H:\oracle\eitex\TEITEX.DBF' SIZE 300M;
CREATE TABLESPACE TCIME DATAFILE 'H:\oracle\cime\TCIME.DBF' SIZE 300M;
CREATE TABLESPACE TCIERCOM DATAFILE 'H:\oracle\ciercom\TCIERCOM.DBF' SIZE 300M;
CREATE TABLESPACE TESDYM DATAFILE 'H:\oracle\esdym\TESDYM.DBF' SIZE 300M;
CREATE TABLESPACE TESIIN DATAFILE 'H:\oracle\esiin\TESIIN.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE1 DATAFILE 'H:\oracle\docente1\TDOCENTE1.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE2 DATAFILE 'H:\oracle\docente2\TDOCENTE2.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE3 DATAFILE 'H:\oracle\docente3\TDOCENTE3.DBF' SIZE 300M;
CREATE TABLESPACE TDOCENTE4 DATAFILE 'H:\oracle\docente4\TDOCENTE4.DBF' SIZE 300M;
```

Una vez creado el esquema, podemos ingresar a nuestra base de datos para crear las tablas necesarias, recordemos que el nombre es academico y su contraseña academico.

Ahora vamos a crear las diferentes tablas, tablas particionadas y paquetes que necesitemos para el funcionamiento de nuestra base de datos. El scrip es el siguiente, como es un ejemplo no hemos incluido todas las tablas :

```
/*==============================================================*/
/* Table: PARTITION  DOCENTES                                   */
/*==============================================================*/
create table DOCENTES  (
   CEDULA_DOCENTE       VARCHAR2(12)                not null,
   NOMBRES              VARCHAR2(40),
   APELLIDOS            VARCHAR2(40),
   TELEFONO             VARCHAR2(15),
   CELULAR              VARCHAR2(15),
   ESTADO_CIVIL         CHAR(1),
   DIRECCION            VARCHAR2(80),
   MAIL                 VARCHAR2(80),
   ESCALAFON            INTEGER,
   USUARIO              VARCHAR2(20),
   CLAVE                VARCHAR2(20),
   ID_TITULOS           INTEGER,
   constraint PK_DOCENTES primary key (CEDULA_DOCENTE)
)PARTITION by hash (cedula_docente)
    (PARTITION DOCENT1 TABLESPACE tdocente1, PARTITION DOCENT2 TABLESPACE tdocente2,
       PARTITION DOCENT3 TABLESPACE tdocente3, PARTITION DOCENT4 TABLESPACE
tdocente4);

/
/*==============================================================*/
/* Table: PARTITION  ESCUELAS                                   */
/*==============================================================*/
create table ESCUELAS  (
   ID_ESCUELA           INTEGER                     not null,
```

```
   ID_FACULTAD          INTEGER,
   ESCUELA             VARCHAR2(100),
   SIGLAS              VARCHAR2(10),
   constraint PK_ESCUELAS primary key (ID_ESCUELA)
)PARTITION BY RANGE (id_escuela)

   (PARTITION EISIC VALUES LESS THAN (2) TABLESPACE TEISIC,
    PARTITION EITEX VALUES LESS THAN (3) TABLESPACE TEITEX,
            PARTITION CIME VALUES LESS THAN (4) TABLESPACE TCIME,
            PARTITION CIERCOM VALUES LESS THAN (5) TABLESPACE TCIERCOM,
            PARTITION ESDYM VALUES LESS THAN (6) TABLESPACE TESDYM,
            PARTITION ESIIN VALUES LESS THAN (7) TABLESPACE TESIIN );

/
/*==============================================================*/
/* Table: PARTITION  ESTUDIANTES                                */
/*==============================================================*/
create table ESTUDIANTES  (
   CEDULA              VARCHAR2(10)                not null,
   NOMBRES              VARCHAR2(40),
   APELLIDOS            VARCHAR2(40),
   SEXO                CHAR(1),
   FECHA_NACIMIENTO     TIMESTAMP,
   NOTA_GRADO           NUMERIC(4, 2),
   ESPECIALIDAD         VARCHAR2(50),
   TIPO_COLEGIO         CHAR(1),
   EXTRANJERO           CHAR(1),
   ESTADO_CIVIL         CHAR(1),
   DOMICILIO            VARCHAR2(50),
   COLEGIO              VARCHAR2(70),
   TELEFONO             VARCHAR2(15),
   MAIL                VARCHAR2(100),
   CLAVE                VARCHAR2(20),
   ESTADO               VARCHAR2(2),
   ID_ESCUELA           INTEGER,
   constraint PK_ESTUDIANTES primary key (CEDULA)
)
PARTITION BY RANGE (id_escuela)
            SUBPARTITION by hash (SEXO)
            SUBPARTITION TEMPLATE
   (SUBPARTITION MASCULINO,
    SUBPARTITION FEMENINO)

   (PARTITION EISIC VALUES LESS THAN (2) TABLESPACE TEISIC,
    PARTITION EITEX VALUES LESS THAN (3) TABLESPACE TEITEX,
            PARTITION CIME VALUES LESS THAN (4) TABLESPACE TCIME,
            PARTITION CIERCOM VALUES LESS THAN (5) TABLESPACE TCIERCOM,
            PARTITION ESDYM VALUES LESS THAN (6) TABLESPACE TESDYM,
            PARTITION ESIIN VALUES LESS THAN (7) TABLESPACE TESIIN );
/
/*==============================================================*/
/* Table: FACULTADES                                            */
/*==============================================================*/
create table FACULTADES  (
   ID_FACULTAD          INTEGER                     not null,
   FACULTAD            VARCHAR2(100),
   SIGLAS              VARCHAR2(10),
   constraint PK_FACULTADES primary key (ID_FACULTAD)
)
/
```

Una vez creadas nuestras tablas procedemos a configurar el nombre del servicio para poder luego acceder desde nuestras bases remotas mediante el dblink.

Recordemos que para configurar el nombre del servicio debemos cambiar el archivo tnsnames.ora que se encuentra en la siguiente dirección, dependiendo de cada ordenador de la siguiente manera:

```
D:\oracle\product\10.2.0\db_1\NETWORK\ADMIN \tnsnames.ora
ORCLHP =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.2)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ORCL)
    )
  )
```

Ahora creamos las vistas que deseamos que nuestros diferentes nodos tengan, en las siguientes sentencias estamos creando las vistas de cada partición según cada carrera (ID_ESCUELA=1).

VISTA DE ESTUDIANTES

```
Create view ESTUDIANTES_EISIC AS SELECT *FROM ESTUDIANTES WHERE ID_ESCUELA=1;
Create view ESTUDIANTES_EITEX AS SELECT *FROM ESTUDIANTES WHERE ID_ESCUELA=2;
Create view ESTUDIANTES_CIME  AS SELECT *FROM ESTUDIANTES WHERE ID_ESCUELA=3;
Create view ESTUDIANTES_CIERCOM AS SELECT *FROM ESTUDIANTES WHERE ID_ESCUELA=4;
Create view ESTUDIANTES_ESDYM AS SELECT *FROM ESTUDIANTES WHERE ID_ESCUELA=5;
Create view ESTUDIANTES_ESIIN AS SELECT *FROM ESTUDIANTES WHERE ID_ESCUELA=6;
```

VISTA DE MATERIAS

```
Create view MATERIAS_EISIC AS SELECT *FROM MATERIAS WHERE ID_ESCUELA=1;
Create view MATERIAS_EITEX AS SELECT *FROM MATERIAS WHERE ID_ESCUELA=2;
Create view MATERIAS_CIME  AS SELECT *FROM MATERIAS WHERE ID_ESCUELA=3;
Create view MATERIAS_CIERCOM AS SELECT *FROM MATERIAS WHERE ID_ESCUELA=4;
Create view MATERIAS_ESDYM AS SELECT *FROM MATERIAS WHERE ID_ESCUELA=5;
Create view MATERIAS_ESIIN AS SELECT *FROM MATERIAS WHERE ID_ESCUELA=6;
```

ORDENADOR 

Aquí se encuentra la Base de Datos Remota en la cual tendremos la partición de esta carrera como es EISIC

Antes de nada debemos configurar el nombre del servicio para poder luego acceder desde nuestra base remota mediante el dblink.

Recordemos que para configurar el nombre del servicio debemos cambiar el archivo tnsnames.ora que se encuentra en la siguiente dirección, dependiendo de cada ordenador de la siguiente manera:

F:\oracle\product\10.2.0\db_1\NETWORK\admin \tnsnames.ora

```
DBHILARY =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.1)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ORCL)
    )
  )
```

Ahora procedemos a creamos el esquema de nuestra base con su respectivo tablespace, la sentencia es la siguiente:

```
CREATE TABLESPACE TACADEMICO DATAFILE 'H:\oracle\academico\TACADEMICO.DBF' SIZE 300M;

CREATE USER academico1 IDENTIFIED BY academico1 DEFAULT TABLESPACE TACADEMICO;

GRANT dba, connect, resource TO academico1;

GRANT CREATE ANY VIEW TO academico1 WITH ADMIN OPTION;
```

Una vez credo el esquema, podemos ingresar a nuestra base de datos para crear las diferentes vistas de las tablas necesarias, recordemos que el nombre es academico1 y su contraseña academico1.

La manera más sencilla de acceder a tablas y vistas (*views*) de otra base de datos Oracle es creando un DBLink, en la siguiente sentencia estamos accediendo al ordenador 1, al esquema que tiene por nombre académico y su contraseña es académico, para poder acceder a las vistas creadas en ese ordenador.

```
create database link LINKHILARY connect to academico identified by academico using 'DBHILARY';
```

Recordemos que las tablas y vistas remotas pueden usar consultas es decir una sentencia SELECT, como se muestra.

```
select * from MATERIAS@LINKHILARY  WHERE ID_ESCUELA=1;
select * from ESTUDIANTES@LINKHILARY  WHERE ID_ESCUELA=1;
```

En esta sentencia estamos listando a las materias y estudiantes que tienen un id_escuela=1, esto es decir solo material y estudiantes de la carrera EISIC.

Ahora vamos a proceder a crear las vistas, las que serán como una tabla normal en el esquema academico1, pudiendo así realizar una inserción, borrado o edición. La sentencia es la siguiente:

```
CREATE VIEW ESTUDIANTES AS select * from ESTUDIANTES@LINKHILARY  WHERE ID_ESCUELA=1;
CREATE VIEW MATERIAS AS select * from MATERIAS@LINKHILARY  WHERE ID_ESCUELA=1;
```

En el ordenador  y  se realizan los mismos paso que en el ordenador  teniendo en cuenta el nombre del servicio que será distinto según el ordenador y al momento de crear las vistas debemos tener claro que id_escuela de escuela le corresponde.

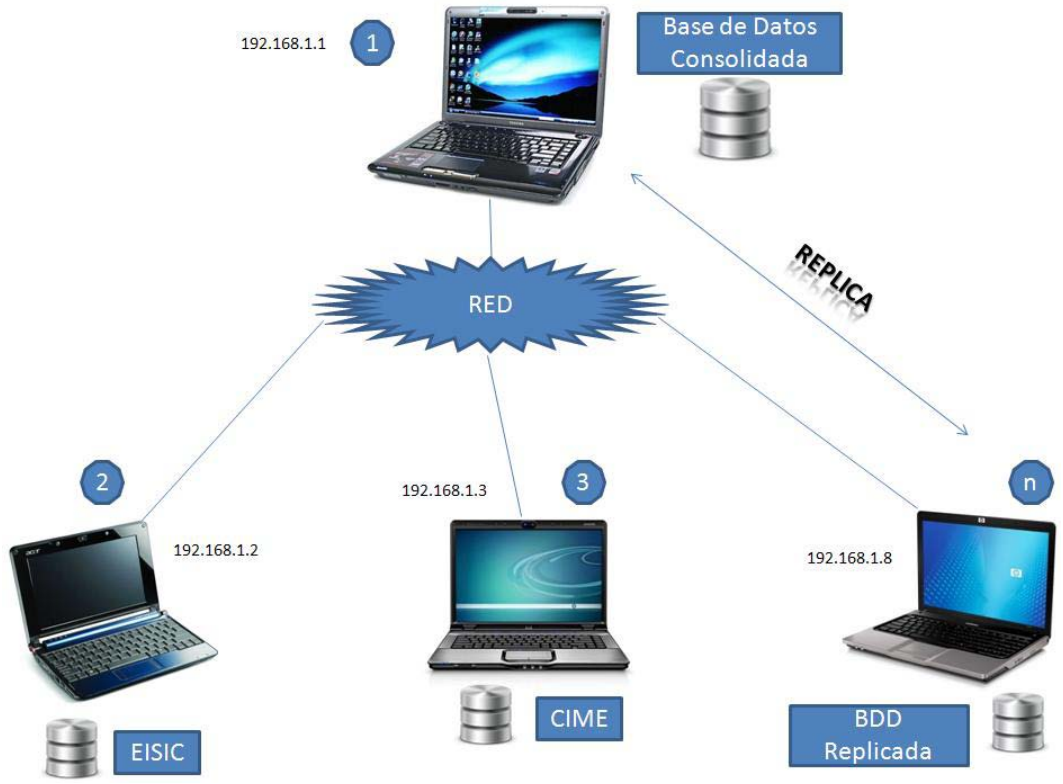
1.7. REPLICACIÓN DE BASE DE DATOS EN ORACLE

Es el proceso de copiar y mantener objetos de bases de datos como tablas que constituyen una base de datos distribuida.

Los cambios aplicados en un sitio son almacenados localmente para posteriormente ser enviados y aplicados al sitio remoto.

En una base de datos distribuida, existen datos disponibles en muchos lugares, pero un objeto en particular (una tabla) solo existe en un nodo de la BD.

En la replicación sincrónica los cambios que ocurran en un master site son replicados inmediatamente a los otros participantes del sistema. Si una transacción no puede ser procesada por un master site, se producirá un rollback de la transacción en todos los otros master sites.



6. Ejemplo de Replica en Oracle

A continuación se muestra la forma de replicar de manera sencilla los datos de una base de datos en oracle hacia otro servidor oracle, mediante triggers.

La replicación permite tener una copia exacta de una base de datos alojada en un servidor (maestro) que se guardará en otro servidor (esclavo). Todas las modificaciones que se hagan en la base de datos del servidor maestro se actualizarán inmediatamente en el servidor esclavo.

Esto no es una copia de seguridad, ya que si borramos una fila en la base de datos maestra, también se borrará en la base de datos esclava.

Estos son los pasos para realizar una réplica.

Recordemos que previamente debe estar configurado del servicio en el archivo tnsnames.ora para poder realizar un dblink

- Una vez creada nuestra base de datos (esquema) en el servidor (maestro) procedemos a crear cada uno de los triggers, esto es INSERT, DELETE, UPTATE para que cuando hagamos una inserción, borrado o edición se refleje inmediatamente en nuestro servidor (esclavo). La sentencia es la siguiente

INSERT TRIGGER

```
CREATE OR REPLACE TRIGGER INSERT_MATERIAS
AFTER INSERT ON MATERIAS
FOR EACH ROW
BEGIN
   INSERT INTO MATERIAS@LINKHP
VALUES(:new.id_materia,:new.materia,:new.creditos,:new.horas_programadas,:new.fecha_creacion,
:new.estado,:new.id_escuela,:new.materia_anterior,:new.id_nivel);
  NULL;
END;
```

UPDATE TRIGGER

```
CREATE OR REPLACE TRIGGER UPDATE_MATERIAS
AFTER UPDATE ON MATERIAS
FOR EACH ROW
BEGIN
```

```
    UPDATE MATERIAS@LINKHP SET
id_materia=:new.id_materia,materia=:new.materia,creditos=:new.creditos,horas_programadas=:new.
horas_programadas,fecha_creacion=:new.fecha_creacion,

estado=:new.estado,id_escuela=:new.id_escuela,materia_anterior=:new.materia_anterior,id_nivel=:ne
w.id_nivel
  WHERE id_materia=:new.id_materia;
  NULL;
END;
```

DELETE TRIGGER

```
CREATE OR REPLACE TRIGGER DELETE_MATERIAS
AFTER DELETE ON MATERIAS
FOR EACH ROW
BEGIN
   DELETE MATERIAS@LINKHP WHERE id_materia=:old.id_materia;
  NULL;
END;
```

- Ahora de la misma manera debemos crear en nuestro servidor (esclavo) los triggers para que cuando realizamos cualquier cambio aquí también ser reflejen en nuestro servidor (maestro).