UNIVERSIDAD TECNOLÓGICA DE PEREIRA

Ingeniería en Sistemas y Computación

Bases de Datos. (IS644 Gr 100)

Secuencias-Disparadores

Vistas

- Restringir el acceso a los datos
- Hacer consultas complejas fácilmente
- Suministrar independencia de datos
- Presentar la misma información pero de diferentes formas

Sintaxis

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
   [(alias[, alias]...)]
 AS subquery
        [WITH CHECK OPTION [CONSTRAINT constraint]]
        [WITH READ ONLY [CONSTRAINT constraint]];
```

Ejemplo:

```
CREATE VIEW          empvu80    AS
        SELECT    employee_id, last_name, salary
        FROM      employees
        WHERE      department_id = 80;

CREATE OR REPLACE VIEW empvu80
          (id_number, name, sal, department_id)
        AS
        SELECT    employee_id, first_name || ' ' ||
                  last_name,salary, department_id
        FROM      employees
        WHERE      department_id = 80;
```

```
SELECT    *
FROM        empvu80;
```

```
        Vista EMPVU80
SELECT employee_id,
       last_name, salary
FROM   employees
WHERE  department_id=80;
```

Crear 5 vistas con el nombre que deseen para las consultas que han realizada de DREAMHOUSE. Y dar sus conclusiones relacionadas con las vistas.

Datos base:

Crear 2 tablas:

Tabla 1 llamada TBbase debe tener dos campos
- id: number
- Nombre: varchar2(50).

Tabla 2 llamada TBbaseAudit debe tener cuatro campos
Id: number
NombreNew: varchar2(50)
NombreOld: varchar2(50)
Fecha: date
Usuario: varchar2(50)

Secuencias

Una secuencia sirve para generar automáticamente números distintos. Son usados como identificadores únicas en las tablas. Estas son almacenadas independientemente de la tabla por lo que la misma secuencia se puede utilizar para varias tablas.

Sintaxis:

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

Donde:

- **Secuencia**: Es el nombre que se le da al objeto de secuencia
- **INCREMENT BY**: Indica cuánto se incrementa la secuencia cada vez que se usa. Por defecto se incrementa de uno en uno
- **START WITH:** Indica el valor inicial de la secuencia (por defecto 1)
- **MAXVALUE:** Máximo valor que puede tomar la secuencia. Si no se toma NOMAXVALUE que permite llegar hasta el 10^{27}
- **MINVALUE:** Mínimo valor que puede tomar la secuencia. Por defecto 10^{26}
- **CYCLE** Hace que la secuencia vuelva a empezar si se ha llegado al máximo valor

Crear las siguientes secuencias:

```
CREATE SEQUENCE SEQ_IDBase
INCREMENT BY 10
START WITH 5;

CREATE SEQUENCE SEQ_IDBaseAudit
INCREMENT BY 1
START WITH 1;
```

Los métodos NEXTVAL y CURRVAL se utilizan para obtener el siguiente número a generar y el valor actual de la secuencia respectivamente. Ejemplo de uso (Oracle)

- NEXTVAL retorna el siguiente valor de la secuencia. Siempre retornará un valor diferente aunque accedan al mismo tiempo diferentes usuarios.
- CURRVAL retorna el valor actual de la secuencia.
  NEXTVAL debe ser usado antes del CURRVAL.

Ejecute los siguientes comandos para consultar las secuencias:

```
SELECT SEQ_IDBase.NEXTVAL FROM DUAL; -- Genera el siguiente valor de la secuencia
SELECT SEQ_IDBase.CURRVAL FROM DUAL; -- Muestra el valor actual de la secuencia.

SELECT SEQ_IDBaseAudit.NEXTVAL FROM DUAL; -- Genera el siguiente valor de la secuencia
```

```
SELECT SEQ_IDBaseAudit.CURRVAL FROM DUAL; -- Muestra el valor actual de la secuencia.
```

Modificando una secuencia
Sintaxis:

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

Ejemplo:

```
ALTER SEQUENCE SEQ_IDBaseAudit
        INCREMENT BY 20
        MAXVALUE 999999
        NOCACHE
        NOCYCLE;
```

Para saber cuáles secuencias han sido creadas se usa la vista USER_SEQUENCES.

```
Select *
From USER_SEQUENCES;
```

La columna LAST_NUMBER muestra cual será el siguiente número de secuencia disponible uso de la secuencia

Un uso de las secuencias es con el comando INSERT
Inserte el siguiente registro:

```
INSERT INTO TBbase VALUES (SEQ_IDBase.NEXTVAL, 'Claudia Torres');
```

Consulte la table TBbase y revise el valor del campo id. De donde salio este valor?

Disparadores o Triggers

Un disparador define una acción que la base de datos debe llevar a cabo cuando se produce algún suceso relacionado con la misma. Los disparadores (triggers) pueden utilizarse para completar la integridad referencial, también para imponer reglas de negocio complejas o para auditar cambios en los datos. El código contenido en un disparador, denominado cuerpo del disparador, está formado por bloques PL/SQL. La ejecución de disparadores es transparente al usuario.
Para crear un disparador en una tabla, el usuario debe tener el privilegio ALTER para la tabla ó ALTER ANY TABLE. Además, del privilegio CREATE TRIGGER.
Existen varios tipos de disparadores, dependiendo del tipo de transacción de disparo y el nivel en el que se ejecuta el disparador (trigger):

- Disparadores de nivel de fila: se ejecutan una vez para cada fila afectada por una instrucción DML. Los disparadores de nivel de fila se crean utilizando la cláusula for each row en el comando create trigger
- Disparadores de nivel de instrucción: se ejecutan una vez para cada instrucción DML. Por ejemplo, si una única instrucción INSERT inserta 500 filas en una tabla un disparador de nivel de instrucción para dicha tabla sólo se ejecutará una vez. Los disparadores de nivel de instrucción son el tipo predeterminado que se crea con el comando create trigger.
- Disparadores Before y After: puesto que los disparadores son ejecutados por sucesos, puede establecerse que se produzcan inmediatamente antes (before) o después (after) de dichos sucesos.
- Disparadores Instead Of: puede utilizar INSTEAD OF para indicar a Oracle lo que tiene que hacer en lugar de realizar las acciones que invoca el disparador. Por ejemplo, podría usar un disparador INSTEAD OF en una vista para gestionar las inserciones en una tabla o para actualizar múltiples tablas que son parte de una vista
- Disparadores de esquema: puede crear disparadores sobre operaciones en el nivel de esquema tales como create table, alter table, drop table, audit, rename, truncate y revoke. Puede incluso crear disparadores para impedir que los usuarios eliminen sus propias tablas. En su mayor parte, los

disparadores de nivel de esquema proporcionan dos capacidades: impedir operaciones DDL y proporcionar una seguridad adicional que controle las operaciones DDL cuando éstar se producen.

- Disparadores en nivel de base de datos: puede crear disparadores que se activen al producirse sucesos de la base de datos, incluyendo errores, inicios de sesión, conexiones y desconexiones. Puede utilizar este tipo de disparador para automatizar el mantenimiento de la base de datos o las acciones de auditoría.

Sintaxis:

```
CREATE [OR REPLACE] TRIGGER <nombre_trigger>
{BEFORE|AFTER}
{DELETE|INSERT|UPDATE [OF col1, col2, ..., colN]
[OR {DELETE|INSERT|UPDATE [OF col1, col2, ..., colN]...]}
ON <nombre_tabla>
[FOR EACH ROW [WHEN (<condicion>)]]
DECLARE
  -- variables locales
BEGIN
  -- Sentencias
[EXCEPTION]
  -- Sentencias control de excepcion
END <nombre_trigger>;
```

Ejecutar las siguientes líneas:

```
CREATE OR REPLACE TRIGGER UTP.TRG_BI_TBBASE
BEFORE INSERT ON UTP.TBBASE
REFERENCING NEW AS New OLD AS Old
FOR EACH ROW
DECLARE
tmpVar NUMBER;
BEGIN
   tmpVar := 0;


   SELECT SEQ_IDBase.NEXTVAL INTO tmpVar FROM dual;

   :NEW.ID := tmpVar; -- NEW.xxxxx  identifica el Nuevo valor del campo.

   EXCEPTION
     WHEN OTHERS THEN
       -- Consider logging the error and then re-raise
       RAISE;
END TRG_BI_TBBASE;
```

Cuál es propósito de crear este disparador??

Inserte 5 registro a la tabla TBbase no incluya el campo de Id.

Ejemplo: `INSERT INTO TBbase (nombre) values ('carlos alzate');`

Que ventajas tiene el uso del disparador?

Cree el siguiente disparador:

```
CREATE OR REPLACE TRIGGER UTP.TRG_AUDIT_TBASE
BEFORE DELETE OR INSERT OR UPDATE
ON UTP.TBBASE
REFERENCING NEW AS New OLD AS Old
FOR EACH ROW
DECLARE
tmpVar NUMBER;

BEGIN

   INSERT INTO TBbaseAudit (ID, NOMBREOLD, NOMBRENEW, FECHA, USUARIO)
      VALUES (SEQ_IDBaseAudit.NEXTVAL, :OLD.NOMBRE, :NEW.NOMBRE, SYSDATE, USER);


   EXCEPTION
     WHEN OTHERS THEN
       -- Consider logging the error and then re-raise
       RAISE;
END TRG_AUDIT_TBASE;
```

Inserte 5 registros más
Actualice 5 registros pero solo el campo de nombre.

Consulte la información de la tabla de TBbaseAudit. Qué conclusiones puede sacar del uso del disparador?.