Fundamento Bases de Datos
Facultad de Ciencias UNAM
M.I. Gerardo Avilés Rosas
gar@ciencias.unam.mx
Laboratorio:
Efraín Hipólito Chamú <*chamugauss@gmail.com* >

PRACTICA 08
INTEGRIDAD

1. Introducción

Hasta ahora se ha estudiado a la estructura de una base de datos como una abstracción que puede representarse de varias formas y la mejor depende del problema que se quiera resolver. También se ha hecho un primer acercamiento a la implementación de la base de datos pero, no se han tomado las consideraciones para crear un buen diseño del modelo de la realidad.

Se considera un buen diseño a aquel que protege, en la medida de lo posible, contra la cometida de errores de inconsistencia y brinda cierto nivel de integridad. Hay que tener en cuenta el no depender de la verificación de los datos de entrada en las aplicaciones que se comunican con la base de datos, sino utilizar las facilidades provistas por el SMBD que lleven a cabo este tipo de verificación.

Integridad: La integridad, es una propiedad que garantiza que el modelo de la realidad sea lo más parecido a esta y se conserve consistente con ella a través del tiempo. Se han definido múltiples tipos de integridad, pero para nuestro estudio sólo se consideran los siguientes:

- Integridad de entidad
- Integridad de dominio
- Integridad de usuario
- Integridad referencial

2. Ejemplo

Para los ejemplos de esta práctica considere el esquema de la Figura1.

```
CREATE TABLE Tabla1(
nIdTabla1 INTEGER,
atributo1 INTEGER,
atributo2 INTEGER,
atributo3 VARCHAR(16),
atributo4 VARCHAR(32),
nIdTabla2 INTEGER
);

CREATE TABLE Tabla2(
nIdTabla2 INTEGER,
atributo1 INTEGER,
atributo2 INTEGER,
atributo3 VARCHAR(16),
atributo4 VARCHAR(32),
nIdTabla1 INTEGER
);
```

Figura 1

A. Integridad de entidad

Se refiere a que cualquier información del sistema se puede identificar por medio de tres elementos.

- La relación a la que pertenece (tabla).
- La columna a la que está relacionado (atributo).
- Identificación individual dentro de la tabla donde se encuentra.

Como ejemplo tenemos:

✓ Llaves Primarias

Se llama llave primaria a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. Una llave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma llave primaria.

Una llave primaria debe identificar a todas las posibles filas de una tabla, no únicamente a las filas que se encuentran en un momento determinado.

Para definir una llave primaria dentro de una tabla, se sigue la sintaxis presentada en la Figura 2.

Es importante notar que dicha restricción de tabla posee el nombre pk_Tabla1_nIdTabla1 y como tal, ahora podrá identificarse dentro del motor de base de datos de Oracle.

```
ALTER TABLE Tabla1 ADD CONSTRAINT pk_Tabla1_nIdTabla1 PRIMARY KEY(nIdTabla1);
ALTER TABLE Tabla2 ADD CONSTRAINT pk_Tabla2_nIdTabla2 PRIMARY KEY(nIdTabla2);
```

Figura 2

✓ Llaves foráneas

Una llave foránea o llave ajena (o Foreign Key FK) es una limitación referencial entre dos tablas. La llave foránea identifica una columna o grupo de columnas en una tabla (tabla hija o referendo), que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la llave primaria u otra llave candidata en la tabla referenciada.

Los valores en una fila de las columnas referendo deben existir solo en una fila en la tabla referenciada. Así, una fila en la tabla referendo no puede contener valores que no existen en la tabla referenciada. De esta forma, las referencias pueden ser creadas para vincular o relacionar información.

Para definir una llave foránea dentro de una tabla, se sigue la sintaxis presentada en la Figura 3.

```
ALTER TABLE tabla1 ADD CONSTRAINT fk_tabla1_nIdTabla2 FOREIGN KEY (nIdTabla1) REFERENCES tabla2(nIdTabla2);
ALTER TABLE tabla2 ADD CONSTRAINT fk_tabla2_nIdTabla1 FOREIGN KEY (nIdTabla2) REFERENCES tabla1(nIdTabla1);
```

Figura 3.

B. Integridad de dominio

Este tipo de integridad se da sobre el conjunto de valores que puede tomar cada una de las tuplas pertenecientes a una tabla (su dominio). Este dominio puede estar o no definido por el SMBD pero, siempre se generan reglas que automáticamente verifican el cumplimiento de este tipo de integridad. Por ejemplo, al querer ingresar un valor alfanumérico en el atributo2 de la Tabla1, el motor regresará una advertencia de que no se logró la inserción ya que los tipos de datos no coinciden o no se pudieron convertir.

Sin embargo, existen distintos tipos de datos que pueden ser objeto de una conversión implícita. Si ahora se quiere insertar un valor numérico o alfanumérico no se muestra mensaje de error, debido a la conversión implícita que se llevó a cabo.

Por otro lado, también es posible restringir el conjunto de valores que podrá contener un atributo en cuestión. Dado que esto corresponde a una definición de restricción, para realizar esta tarea se recurre nuevamente a la cláusula ALTER TABLE. Para este tipo de restricciones, por cada una de las tuplas que se inserten el sistema se encarga de revisar que cumplan con estas restricciones de dominio.

Dentro de 'expresión' puede suministrarse cualquier expresión válida de SQL que regrese un valor apropiado dentro del dominio original del dato, esto es, si se tiene un bigint como tipo de dato en la definición entonces dentro de la expresión se debe tener alguna que regrese un entero. Si se presenta en la definición como tipo de dato un varchar ( ) entonces la expresión debe regresar una cadena, y así sucesivamente. Si se denota una 'restricción', ésta también puede definirse con las cláusulas NULL, UNIQUE, NOT NULL entre otras, pero es de

particular interés la cláusula CHECK, la cual permite definir restricciones sencillas dentro del domino de nuestros atributos.

A modo de ejemplo, si se desea garantizar que todos los valores del atributo2 de la Tabla2 sean un número positivo o el cero, es necesario crear una restricción con la siguiente instrucción:

```
ALTER TABLE tabla2 ADD CONSTRAINT ck_tabla2 CHECK (atributo1 >= 0);
```

Figura 4

Es importante mencionar que la cláusula CHECK no puede manejar sub-consultas como restricción.

- Marcas NULL

Un caso muy particular de restricción de dominio corresponde a la aparición de las marcas NULL. Dichas marcas poseen una variedad de significado y su presencia puede ser o no válida según la semántica que deseemos se represente con el modelo de la base de datos. Para controlar su existencia en la base de datos, se cuenta con la restricción de dominio NULL o NOT NULL, que indica si se permite o no este tipo de marcas para un atributo dado. Para definir una restricción de este tipo, se utiliza nuevamente la sentencia ALTER TABLE.
A modo de ejemplo, si se desea evitar la aparición de marcas NULL en el atributo2 de la Tabla1 , habrá que alterar dicha tabla de la siguiente forma:

```
ALTER TABLE tabla1 ADD CONSTRAINT ck_tabla1 CHECK (atributo1 IS NOT NULL);
```

Figura 5

- Unicidad

Existe una restricción que refiere la unicidad de valores dentro del atributo o columna. Una razón para ello es que posiblemente un atributo debe ser único en el conjunto, pero no es una llave primaria por razones de eficiencia o administración.
Un ejemplo de ello, es la utilización del atributo1 en la Tabla1.

Para identificar este tipo de restricciones, nuevamente se utiliza la cláusula ALTER TABLE con el modificador UNIQUE. A modo de ejemplo, si se desea que el correo electrónico de un cliente sea único, habrá que ejecutar la siguiente instrucción:

```
ALTER TABLE tabla1 ADD CONSTRAINT unq_tabla1 UNIQUE (atributo1);
```

Figura 6

Es importante hacer notar que una restricción UNIQUE no se cumple cuando existen dos o más tuplas en la tabla donde los valores de todas las columnas incluidas en la restricción son idénticos. Sin embargo, las marcas NULL no son consideradas equivalentes en una

comparación, esto significa que aún en la presencia de una restricción UNIQUE es posible encontrar un número ilimitado de tuplas que tengan la marca NULL en la columna de la restricción.

Además, tampoco es posible crear una restricción UNIQUE sobre un atributo cuando ya existen valores duplicados en él.

C. Integridad de usuario

La integridad de usuario está relacionada con la autorización que tiene cada usuario de la base de datos y su interrelación con otros usuarios y con la base de datos misma.
Es tema más avanzado de administración y no se profundizará en él.

D. Integridad referencial

La integridad referencial consiste en un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que las tuplas de tablas relacionadas son válidas y que no se borren o cambien datos relacionados de forma accidental produciendo errores de consistencia.

Cuando se define una columna como llave foránea, las tuplas de la tabla pueden contener en esa columna la marca NULL o bien un valor que existe en la otra tabla.

Eso es lo que se denomina integridad referencial y consiste en que los datos que hacen referencia a otros (llaves foráneas) deben ser correctos. La integridad referencial hace que el SMBD se asegure de que no haya en las llaves foráneas valores que no estén en la tabla referenciada.

La integridad referencial se activa en cuanto es creada una llave foránea y a partir de ese momento se comprueba cada vez que se modifiquen datos que puedan alterarla.

Los escenarios en los cuales se pueden producir errores en los datos contenidos en una llave foránea son:

- Cuando se inserta una nueva tupla en la tabla y el valor de la llave foránea no existe en la tabla referenciada.
- Cuando se modifica el valor de la llave primaria de una tupla que tiene referencias en otras tablas.
- Cuando se modifica el valor de la llave foránea, el nuevo valor debe existir en la tabla referenciada.
- Cuando se elimina una tupla de la tabla principal y esa tupla tiene referencias en otras tablas.

Asociada a la integridad referencial están los conceptos de actualizar los registros en cascada y eliminar registros en cascada. El actualizar y/o eliminar registros en cascada, son opciones

que se indican cuando es definida la llave foránea y que informan al SMBD qué acciones llevar a cabo en los casos comentados en el punto anterior.

Actualizar registros en cascada: Indica al SMBD que al modificar un valor del atributo llave de la tabla referenciada, automáticamente se actualice el valor de la llave foránea de las tuplas relacionadas en la tabla referenciante. Si no se tiene definida esta opción, no se puede cambiar los valores de la llave primaria de la tabla referenciada. En Oracle esta opción no está soportada, sin embargo un código de ejemplo para motores como SQLServer o PostgreSQL se deja a continuación:

```
CREATE TABLE Tabla4(
nIdTabla4 INTEGER,
atributo1 INTEGER,
atributo2 INTEGER,
atributo3 VARCHAR(16),
atributo4 VARCHAR(32),
nIdTabla1 INTEGER
);

ALTER TABLE Tabla4 ADD CONSTRAINT pk_Tabla4_nIdTabla4 PRIMARY KEY (nIdTabla4);
ALTER TABLE Tabla4 ADD CONSTRAINT fk_Tabla4_nIdTabla1 FOREIGN KEY (nIdTabla1) REFERENCES Tabla1(nIdTabla1) ON UPDATE CASCADE;
```

Eliminar registros en cascada: Indica al SMBD que al eliminar una tupla de la tabla referenciada automáticamente se borran también las tuplas relacionadas en la tabla referenciante. Si no se tiene definida esta opción, no se pueden borrar tuplas de la tabla referenciada si estas tienen tuplas relacionadas en la tabla referenciante.

```
CREATE TABLE Tabla3(
nIdTabla3 INTEGER,
atributo1 INTEGER,
atributo2 INTEGER,
atributo3 VARCHAR(16),
atributo4 VARCHAR(32),
nIdTabla1 INTEGER
);

ALTER TABLE Tabla3 ADD CONSTRAINT pk_Tabla3_nIdTabla3 PRIMARY KEY (nIdTabla3);
ALTER TABLE Tabla1 ADD CONSTRAINT fk_Tabla3_nIdTabla1 FOREIGN KEY (nIdTabla1) REFERENCES Tabla1(nIdTabla1) ON DELETE CASCADE;
```

Por lo tanto, para cada llave foránea de la base de datos habrá que contestar a tres preguntas:

- ¿Tiene sentido que la llave foránea acepte nulos?
- Regla de borrado: ¿Qué ocurre si se intenta borrar la tupla referenciada por la llave foránea?
  - ✓ Restringir: no se permite borrar la tupla referenciada.
  - ✓ Propagar: se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la llave foránea.
  - ✓ Anular: se borra la tupla referenciada y las tuplas que la referenciaban obtienen la marca NULL en la llave foránea (sólo si acepta nulos).

- ❖ Regla de modificación: ¿Qué ocurre si se intenta modificar el valor de la llave primaria de la tupla referenciada por la llave foránea?
  - ✓ Restringir: no se permite modificar el valor de la llave primaria de la tupla referenciada.
  - ✓ Propagar: se modifica el valor de la llave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian mediante la llave foránea.
  - ✓ Anular: se modifica la tupla referenciada y las tuplas que la referenciaban obtienen la marca NULL en la llave foránea (sólo si acepta nulos).

Entregables

- Añade las llaves primarias y foráneas a todas las tablas del esquema presentado en el análisis de la práctica 04. (Utilizar el comando ALTER TABLE)
- Añade 30 restricciones que consideres importantes para el mismo esquema y que estén basadas en la lógica planteada en el Caso de Uso “Gimnasio Hércules”.
- Se entregará:
  - creacion_tablas_normalizado.sql
  - integridad_restricciones.sql indicando las 30 restricciones que consideraste necesario para tus tablas (Incluyendo llaves primarias y foráneas).
  - borrado_tablas,sql para borrar cada una de sus tablas.
- **Fecha de entrega: 15 de Octubre del 2017 a las 23:59:59.**