Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo

Bases de Datos

Pequeños apuntes de SQL[*]

1 SQL

En estas hojas se exponen de manera concisa algunos aspectos adicionales del SQL, así como ejemplos de la sintaxis del lenguaje para estos aspectos. No pretende ser un manual de referencia de SQL, pues no es exhaustivo ni en cuanto a aspectos ni en cuanto a sintaxis.

Para completar el panorama del SQL se debe acudir a un manual de SQL y/o a un texto explicativo del estándar, como "Understanding the New SQL: A complete guide", de Jim Melton y Alan R. Simon (para SQL2).

En cualquier caso, siempre es conveniente comprobar los manuales del SGBD que se utilice en cada momento para conocer exactamente cómo implementa el sistema el SQL, puesto que lo más común es que no se implemente el estándar completo (falten cosas), algunas cosas no se hagan exactamente igual que en el estándar, y existan características adicionales en el producto no estandarizadas.

ESTÁNDARES DE SQL
Existen 3 estándares aprobados de SQL:

> SQL86 - Estándar inicial de SQL. Conjunto limitado de características. Apenas con alguna restricción de integridad.
>
> SQL89 - Revisión mínima del estándar del 86. Añade cierta integridad referencial.
>
> SQL92 (SQL2)- Revisión completa del lenguaje (superconjunto del SQL89). Sobre todo se añade más potencia semántica en la definición de datos (restricciones de integridad, dominios, más tipos de datos). Mejora el lenguaje de consulta (ej. productos natural y externo) y elimina problemas de ortogonalidad. Estandariza el SQL inmerso.
>
> SQL3 – Siguiente revisión del SQL en proceso de estandarización.

NOMENCLATURA
En lugar de la nomenclatura relacional, SQL utiliza la siguiente, que se usará indistintamente.

Tabla por relación
Columna por atributo
Fila (o registro) por tupla

[*] En caso de encontrar alguna errata de cualquier tipo en estos apuntes (que seguro que la habrá), puede comunicarse al que los ha perpretado: Darío Álvarez, correo-e: darioa@uniovi.es

2 LDD – LENGUAJE DE DEFINICIÓN DE DATOS

Se verán los elementos básicos dentro del estándar SQL86, basados en el SQL del dBASE IV 1.5[1].

Para activar el modo SQL desde el punto indicativo del dBASE se usa la orden SET SQL ON

dBASE distingue una sentencia SQL de una dBASE porque las de SQL deben finalizar en punto y coma.

CREACIÓN DE BASES DE DATOS
Un SGBD puede trabajar con varias bases de datos diferentes (incluso a la vez, como es el caso usual de un servidor de bases de datos), aunque en sistemas monousuario se trabaja en cada momento con una de ellas.

Para crear una base de datos se usa

CREATE DATABASE *nombreBaseDatos*;

Internamente, dBASE crea un subdirectorio MS-DOS con el nombre de la base de datos.

Cada tabla de la base de datos (relación) se almacena en ese directorio en un fichero con el formato DBF. Un fichero .DBF es en esencia un fichero secuencial que tiene una cabecera que define el formato de los registros (filas) que vienen a continuación. Estos ficheros son la única estructura física de almacenamiento que usa dBASE.

En este subdirectorio se almacena el **catálogo** o **diccionario de datos**. El catálogo se almacena mediante una serie de tablas normales de la base de datos (pueden consultarse con SQL). Por ejemplo SYSTABLS almacena metadatos acerca de las tablas definidas en la base de datos, SYSCOLS de las columnas, etc.
Este aspecto está estandarizado ya en SQL92 con el concepto de esquema (SCHEMA)

INICIO DEL TRABAJO CON LA BASE DE DATOS
Para que el sistema sepa con qué base de datos se quiere trabajar, se "arranca" la base de datos:

START DATABASE *NombreBaseDatos;*

STOP DATABASE;
detiene el trabajo con la base de datos activa.

Esto es propio de un entorno monousuario como dBASE. En un entorno multiusuario con concurrencia de uso de bases de datos se suele CONECTAR a una base de datos determinada.

[1] Fundamentalmente porque utiliza SQL86 sin extensiones prácticamente y funciona en MS-DOS con 640K de RAM.

Este tipo de trabajo entre clientes y un servidor de base de datos (Acceso Remoto a Bases de Datos) está estandarizado en SQL92.

CREACIÓN DE TABLAS

Esquemas

No existen esquemas de relaciones. En su lugar se define a la vez una tabla con su esquema (columnas que la componen).

Tipos de datos

No se pueden crear dominios, sólo pueden usarse los tipos de datos predefinidos, que se reducen a números y cadenas:

CHAR(x)	Cadenas de caracteres
NUMERIC(x,y)	Números decimales (exactos) – con precisión y escala
INTEGER	enteros
SMALLINT	enteros cortos
FLOAT(x,y)	Números en coma flotante (aproximados) – con precisión y escala
REAL	reales
DOUBLE PRECISION	más precisión que REAL

Cada SGBD suele incorporar tipos de datos adicionales, junto con sus operadores. Por ejemplo dBASE incorpora booleanos (LOGICAL) y fechas (DATE).

CREATE TABLE

Para crear una tabla se especifica el nombre de la tabla, las columnas y el tipo de cada columna:

```
CREATE TABLE deposito (
        n_suc CHAR(15),
        n_cuenta INTEGER,
        n_CH CHAR (20),
        saldo REAL);
```

Restricciones de integridad: NOT NULL

Prácticamente la única restricción de integridad que se puede indicar es que un atributo no pueda tomar el valor nulo (dBASE no tiene esta opción). Esto se hace a continuación del tipo del atributo:

```
        n_CH CHAR(20) NOT NULL, ...
```

En realidad una sentencia CREATE TABLE indica entre paréntesis especificaciones que afectan a la tabla separadas por comas. La especificación más normal es la definición de una columna. En SQL92 se pueden definir más, como claves, integridad referencial, restricciones de integridad, etc.

SQL86 no soporta el concepto de clave primaria (ni otro tipo de claves). Precisamente por eso quizá sea más adecuado hablar de tablas SQL que de relaciones (en SQL92 es OPCIONAL indicar claves primarias).

CREACIÓN DE VISTAS

CREATE VIEW *nombreVista* AS
    <Expresión SELECT que define la vista>

CREACIÓN DE ÍNDICES

Se pueden definir índices sobre cualquier conjunto de columnas de una tabla. Un índice acelera el acceso (búsquedas) a la tabla, siempre que el sistema necesite o pueda utilizarlo para los accesos (si se indexa por nombre, para una consulta que se base únicamente en DNI el índice no se podrá utilizar).

CREATE [UNIQUE] INDEX *nombreÍndice* ON *nombreTabla* (col1 [ASC|DESC], col2 ...);

UNIQUE indica que el índice no admite valores de clave repetidos (elementos repetidos)
En la lista de columnas se indica la clave, los atributos sobre los que se indexará. Se puede indicar la ordenación de cada columna en el índice: ascendente o descendente.

Se puede conseguir el efecto de clave primaria creando un índice único sobre la clave primaria de cada tabla. De esta manera el intento de insertar una clave duplicada será rechazado por violar el UNIQUE del índice.

Los índices son el único elemento que se puede considerar parte de la definición del esquema físico parte del estándar SQL. Sin embargo son elementos lógicos puesto que se pueden eliminar y añadir en cualquier momento sin afectar al funcionamiento del sistema (sólo a su rendimiento). Esto quiere decir que las aplicaciones y consultas funcionarán exactamente igual desde el punto de vista lógico con índices que sin ellos. Cuando existen índices el SGBD puede decidir utilizarlos (o no) para acelerar el acceso.

Hay que tener en cuenta que los índices, si bien aceleran los accesos (consultas) también ocupan un cierto espacio en disco y necesitan un tiempo para actualizarse cuando hay modificaciones en las tablas.

ALTERACIÓN DE TABLAS

La modificación del esquema no está estandarizada en SQL86 (sí en SQL92). dBASE incluye una sentencia para añadir columnas a una tabla (cambio que garantiza la independencia lógica, el borrado de una columna no).

ALTER TABLE CH ADD (telefono CHAR(9), edad INTEGER);

BORRADO DE BASES DE DATOS, TABLAS, VISTAS E ÍNDICES

Eliminan el elemento en cuestión.

DROP DATABASE;
DROP TABLE *nombreTabla*;
DROP VIEW *nombreVista*;
DROP INDEX *nombreÍndice*;

No debe confundirse con borrar todos sus elementos. Por ejemplo, DELETE * FROM CH borra todas las tuplas, pero la tabla CH sigue existiendo. DROP TABLE CH; elimina la tabla.

3 LMD. LENGUAJE DE MANEJO DE DATOS

Aquí se mencionarán algunos detalles adicionales sobre la parte de lenguaje de manejo de datos.

IMPORTACIÓN Y EXPORTACIÓN DE DATOS

Casi todos los SGBD disponen de herramientas para importar/exportar los datos almacenados en sus BD.

Ejemplos que incorpora el dBASE:

LOAD DATA FROM *ficheroDBF* INTO TABLE *tabla;*
Carga los datos del fichero DBF en la tabla de la base de datos, basándose en el nombre de las columnas que casen.

UNLOAD DATA TO *ficheroDBF* FROM TABLE *tabla;*
Crea un fichero DBF a partir de la tabla.

DBDEFINE;
Convierte todos los ficheros DBF de un directorio de una BD SQL a tablas SQL.

OPERADORES ADICIONALES EN SELECT

En SQL2 ya hay definidos los operadores para la diferencia (EXCEPT) y la intersección (INTERSECT).

Además existen muchos más operadores adicionales: productos naturales ([NATURAL] INNER JOIN) y externos (LEFT, RIGHT y FULL OUTER JOIN), en los que se indica sobre qué columnas (o con qué predicado) se quieren unir las relaciones. Por completitud, también se pueden especificar productos cartesianos (CROSS JOIN) con el mismo efecto que colocar las relaciones en la FROM.

Las consultas son más ortogonales en SQL2. En la FROM, además de nombres de relaciones, se puede especificar cualquier expresión que devuelva una relación (como por ejemplo los anteriores productos externos, etc.).

4 DCL. LENGUAJE DE CONTROL DE DATOS

Con el nombre de lenguaje de control de datos se hace referencia a la parte del lenguaje SQL que se ocupa de los apartados de seguridad y de la integridad en el procesamiento concurrente.

SEGURIDAD
Elementos que permiten proteger los datos frente a acceso, modificación o borrado no autorizados.

- Definición de usuarios mediante un mecanismo determinado, que puede ser específico de cada SGBD. En sistemas multiusuario puede usarse el propio usuario del SO, o bien definir usuarios propios del SGBD.
- Concesión de permisos (privilegios) a los usuarios.

dBASE permite también encriptar los datos (ya que se almacenan en simples ficheros MS-DOS)

Concesión de privilegios: GRANT
GRANT ALL | *listaPrivilegios*
ON TABLE *listaTablas*
TO PUBLIC | *listaUsuarios*
[WITH GRANT OPTION]

Se conceden todos (ALL) o un subconjunto de privilegios que permiten borrar, insertar, consultar o actualizar (DELETE, INSERT, SELECT, UPDATE) una tabla o un conjunto de tablas.

Estos se conceden a todos los usuarios (PUBLIC) o a una lista de ellos.

WITH GRANT OPTION indica que aquellos usuarios a los que se ha concedido estos privilegios pueden a su vez traspasarselos (nunca más de los que se tienen actualmente) a otros usuarios (por medio de sentencias GRANT).

También se pueden restringir los privilegios a un subconjunto de las columnas de la tabla:
GRANT SELECT, UPDATE (n_CH, n_suc, n_cuenta)
ON TABLE deposito ...

Sólo el propietario de un objeto puede conceder los privilegios del mismo. El propietario es siempre el creador del mismo.

Las operaciones con el esquema de la base de datos (CREATE, DROP, etc.) sólo pueden ser realizadas por el propietario del esquema. El esquema (SCHEMA) es un concepto de SQL2 que es equivalente al de DATABASE visto para el dBASE y contiene toda la información del esquema lógico de la base de datos.

Revocación de privilegios: REVOKE
Para quitar (revocar) los privilegios concedidos se utiliza REVOKE (se revocan en cascada en caso de que el usuario los haya traspasado a otros con grant option (añadiendo CASCADE al final en SQL2).

REVOKE ALL | *listaPrivilegios*
ON TABLE *listaTablas*
FROM PUBLIC | *listaUsuarios*

La utilización de vistas combinada con una definición de usuarios y concesión juiciosa de privilegios constituye el mecanismo de seguridad que el administrador de la base de datos SQL utiliza para llevar a cabo las políticas de seguridad del sistema.

TRANSACCIONES
Una transacción es un conjunto de acciones que o bien se realizan todas, o bien no se realiza ninguna. La base de datos siempre queda en un estado consistente, nunca puede quedar en un estado intermedio.

SQL dispone de sentencias que permiten que los cambios realizados por una transacción queden reflejados permanentemente en la base de datos (comprometer)

COMMIT [WORK]

También puede deshacer los cambios realizados por la transacción, en caso de que haya habido algún error (o una equivocación)

ROLLBACK [WORK]

Las transacciones se finalizan explícitamente con un COMMIT o un ROLLBACK. Sin embargo se inician de manera implícita al ejecutar una sentencia que necesite el contexto de una transacción (ej. SELECT, UPDATE, etc.). Dependiendo del entorno de programación que se esté utilizando, es posible que esta iniciación y finalización de las transacciones sea realizada por las librerías, software intermedio, etc. que se utilicen, y no sea necesario programarlas directamente.

También es posible que cada entorno de programación y/o SGBD disponga de elementos adicionales para el control de concurrencia que puedan ser utilizados por el usuario, como por ejemplo bloqueos.

En SQL92 las transacciones tienen un modo (sólo lectura, lectura escritura) y un determinado nivel de aislamiento frente a otras transacciones (hay 4 posibles), que se establece en función de los requisitos de concurrencia de la transacción.

En caso de no especificar explícitamente el tipo de transacción con SET TRANSACTION el sistema usa el tipo de transacción por defecto.

5 RESTRICCIONES DE INTEGRIDAD EN SQL2

TIPOS DE DATOS
SQL2 dispone de una gama más amplia de tipos de datos que las versiones anteriores (junto con una serie de operadores para esos tipos de datos):

Exactos

INTEGER
SMALLINT / Entero nunca con menos rango que INTEGER
NUMERIC (n) , (n,d) n dígitos, d decimales
DECIMAL (n) , (n,d) / Como NUMERIC pero puede tener mayor rango

Aproximados

REAL / número en coma flotante de precisión sencilla
DOUBLE PRECISION / mayor precisión que REAL
FLOAT (p) p precisión

Cadenas

CHARACTER, CHAR(x)
CHARACTER VARYING (x), VARCHAR(x) / cadenas que no desperdician el espacio no ocupado si la cadena no alcanza su longitud máxima

Cadenas de bits

BIT(x) / como CHAR, pero los elementos son bits
BIT VARYING (x)

Fechas

DATE (10 posiciones) DD/MM/AAAA / fecha
TIME HH:MM:SS.xxxxxx / hora
TIMESTAMP / Marca de tiempo: señala un instante determinado en el tiempo: fecha con la hora, minutos, etc.
TIME WITH TIMEZONE / La hora con la zona horaria donde se tomó la hora
TIMESTAMP WITH TIMEZONE / Marca de tiempo con su zona horaria

Intervalos de tiempo

DOMINIOS
SQL2 sí permite definir dominios del usuario

CREATE DOMAIN *nombreDominio tipoDatos*
CHECK *predicado*

Define un dominio que puede ser utilizado como tipo de datos de las columnas. Además se puede definir una restricción de integridad para el dominio, mediante una cláusula CHECK que especifica un predicado que siempre se debe cumplir (véase más adelante la parte general de restricciones de integridad).

También se puede especificar un valor que tomar por defecto en caso de ser necesario para los valores del dominio.

```
CREATE DOMAIN  Tipo_n_CH CHAR (20)
CHECK (VALUE IS NOT NULL) /* El valor de un Tipo_n_CH no puede ser nunca nulo */
DEFAULT 'Nombre por defecto'

CREATE TABLE CH ( n_CH Tipo_n_CH, ...
```

Se pueden especificar restricciones adicionales en la propia tabla.

RESTRICCIONES DE INTEGRIDAD REFERENCIAL

SQL89 permite expresar claves primarias, claves candidato (claves únicas) y claves externas.

Claves primarias

La clave primaria de una relación se especifica mediante una cláusula que se coloca en la definición de cada tabla indicando los atributos que la componen:

```
PRIMARY KEY (k1, k2, ...)
```

Claves candidato

```
UNIQUE (u1, u2, ...)
```

Claves externas

```
FOREIGN KEY (a1, a2, ...) REFERENCES tablaReferenciada [(k1, k2, ...)]
```

Se indican los atributos que forman la clave externa y a qué relación referencian. Por defecto harán referencia a la clave primaria de esta relación, pero puede indicarse exactamente a qué atributos se hace referencia.

```
CREATE TABLE depósito (
        n_suc CHAR(15) NOT NULL,
        n_cuenta INTEGER,  /* también n_cuenta INTEGER UNIQUE */
        n_CH CHAR(20) NOT NULL, /* o n_CH CHAR(20) NOT NULL REFERENCES
        sucursal */
        saldo REAL,

        PRIMARY KEY (n_suc, n_CH),
        UNIQUE (n_cuenta),
        FOREIGN KEY (n_suc) REFERENCES sucursal,
        FOREIGN KEY (n_CH) REFERENCES CH (n_CH) );
```

Reglas de integridad referencial

En SQL2 se pueden especificar las reglas de comportamiento de la integridad referencial en caso de borrado o de actualización, añadiéndo a continuación de la declaración de la clave externa:

ON [DELETE | UPDATE] *acción*

Se indica la acción que tomará el sistema en caso de borrado o actualización. Hay cuatro tipos de acciones posibles:

- NO ACTION (restingir) que se toma por defecto
- CASCADE propaga el cambio: borra o actualiza la clave externa de las tuplas que referenciaban
- SET DEFAULT coloca en la clave externa el valor por defecto de la misma
- SET NULL pone a nulos la clave externa.

```
... FOREIGN KEY n_suc REFERENCES sucursal
                ON DELETE SET DEFAULT
                ON UPDATE CASCADE , ...
```

RESTRICCIONES GENÉRICAS

La definición de restricciones genéricas en SQL2 se basa en la utilización de la cláusula CHECK.

CHECK(expresión)

que obliga a que el sistema compruebe que se cumple la expresión (una expresión booleana cualquiera en SQL) dentro del contexto en que se haya colocado la cláusula CHECK.

Además de estos restricciones genéricas existen también restricciones específicas, como dominios, claves primarias, valores no nulos, etc.

Denominación de las restricciones

A cualquier restricción puede asociársele un nombre mediante una definición

CONSTRAINT nombre_restricción restricción

Tipos de restricciones

Dependiendo del lugar en que definamos una restricción tendremos restricciones

- de columna
- de tabla
- globales (asertos).

RESTRICCIONES DE COLUMNA

Referidas a un atributo de una tabla

```
...
n_suc CHAR(15)
CONSTRAINT n_suc_no_nulo
NOT NULL,
...

...
sexo CHAR(10)
  CHECK(VALUE IN ('Varón', 'Mujer'),
...
```

```
...
n_suc CHAR(15)
CONSTRAINT n_suc_no_nulo
CHECK(VALUE IS NOT NULL)
...

...
importe INTEGER
  CHECK
(VALUE BETWEEN 0 AND 1000000)
...
```

RESTRICCIONES DE TABLA

```
sexo CHAR(15)
...
CONSTRAINT clave_primaria_tabla
  PRIMARY KEY (DNI),
...
CONSTRAINT tipos_sexo
  CHECK (sexo IN ('Varón', 'Mujer'),
...
```

Aquí expresamos la misma restricción para el sexo que en el caso anterior pero como restricción de tabla. En lugar de comprobar el valor del atributo asociado al CHECK, al ser a nivel de tabla hay que indicar el nombre exacto del atributo a comprobar.

```
...
n_sup CHAR(10),
n_pers CHAR(10),
...
CONSTRAINT no_autosupervisor
  CHECK(n_sup<>n_pers),
...
```

Un supervisor no puede supervisarse a sí mismo

```
(* tabla depósito *)
...
CHECK (n_CH IN
  (SELECT n_CH FROM CH)),
...
```

Una integridad referencial expresada mediante una restricción genérico: el CH propietario de una cuenta tiene que estar en el conjunto de CH de la tabla CH

```
(* tabla supervisa *)
...
CHECK (n_sup <>ANY
(SELECT n_pers FROM trabaja
WHERE n_comp='Vanesto')),
```

Un supervisor nunca puede trabajar para el Vanesto

En las cláusulas CHECK pueden aparecer otros operadores como UNIQUE, que comprueba que no se repiten valores en una tabla

```
CHECK ( UNIQUE (SELECT ... ) )
```

Comprobación diferida

Puede indicarse el momento de comprobación de una restricción mediante

```
DEFERRABLE / NOT DEFERRABLE  /* al final de la definición del restricción */
```

Si se indica que una restricción es DEFERRABLE , entonces se aplaza la comprobación del mismo hasta el final de la transacción o se indique explícitamente. Esto es muy útil en aquellos casos en los que los estados intermedios de una transacción (no se han ejecutado todos los pasos) hacen que se incumpla (temporalmente) una restricción, pero esto se arregla en los pasos restantes.

Por ejemplo, cuando se retira dinero de una cuenta hay que hacer dos pasos: disminuir el saldo de la cuenta y luego el activo (total de los saldos de la sucursal). En el punto intermedio se incumple el restricción de que el activo debe ser siempre igual a la suma de los saldos, pero esto se arregla en el segundo paso de la transacción. Este restricción debe comprobarse al finalizar la transacción, no en el intermedio.

En otros casos las restricciones que deben cumplirse siempre en todo instante son NOT DEFERRABLE. Por ejemplo el importe debe ser un número entero, el sexo Varón o Mujer, etc. Estos se comprueban al finalizar cada sentencia.

ASERTOS

Algunas restricciones podrían ponerse como restricciones de tabla, aunque en muchos casos quedan más claras expresados como restricciones globales no ligados a una tabla concreta: los asertos. Se definen con

```
CREATE ASSERTION nombre_aserto
        CHECK (expresión)
```

Ej: La suma de los saldos del banco debe ser siempre menor que la de los prestamos

```
CREATE ASSERTION no_prestes_mas_de_lo_que_tienes
        CHECK( (SELECT SUM(importe) FROM prestamo)
                                <
                (SELECT SUM(saldo) FROM deposito) )
```

Ej: El activo de cada sucursal debe ser igual a la suma de los saldos depositados en ella.

```
CREATE ASSERTION activo_igual_suma_saldos
        CHECK( NOT EXISTS (
                        SELECT n_suc
                        FROM sucursal s
                        WHERE activo <>
                                (SELECT SUM(saldo)
                                 FROM deposito d
                                 WHERE d.n_suc = s.n_suc) ) )
```

(el activo de una sucursal s.n_suc no puede ser distinto de la suma de los saldos de las cuentas que tiene. este restricción también puede expresarse como una restricción de la tabla sucursal).

DISPARADORES EN SQL3

Existe una propuesta para la definición de disparadores para el próximo estándar de SQL. Algunos fabricantes ya incorporan disparadores en sus productos siguiendo en alguna medida esta sintaxis.

CREATE TRIGGER nombre_disparador
{BEFORE | AFTER} {INSERT | DELETE | UPDATE | UPDATE OF col1, col2...}
ON relación
[*referencia*]
[*condición*]
acciones [*granularidad*]

El disparador se activa antes o después de una modificación sobre la relación (inserción, borrado, actualización o actualización de unas columnas determinadas. Si se especifica, además de esa modificación se debe de cumplir una condición para la activación. La activación provoca la ejecución de la acción.

Referencia. Sólo en actualizaciones, es la manera de indicar cuál es el valor antiguo y el nuevo de las tuplas tras la actualización.

REFERENCING OLD [AS] n_antiguo [NEW AS n_nuevo] o a la inversa.

Condición. Indica una condición necesaria para la activación del disparador. Si se omite se activa directamente.

WHEN (expresión)

Acciones. Indica las acciones (separadas por comas) del disparador.

(acción, acción, ...)

Granularidad. Indica si la acción se ejecuta una vez por cada evento o bien una vez por cada tupla a la que se refiere el evento.

FOR EACH STATEMENT | FOR EACH ROW

Ej: En una actualización del saldo se pasa a tener números rojos. En este caso lo que se hace es conceder un préstamo por esa cantidad usando los mismos datos de la cuenta, y dejando el saldo en cero:

```
CREATE ASSERTION numeros_rojos BEFORE UPDATE OF saldo ON deposito
  REFERENCING OLD AS a NEW AS n
  WHEN ( n.saldo < 0 )
  ( UPDATE deposito s
    SET saldo=0
    WHERE s.n_cuenta = a.n_cuenta,

    INSERT INTO prestamo
      VALUES ( a.n_suc, a.n_cuenta, a.n_CH, -n.saldo) )
  FOR EACH ROW
```

Cuando el nuevo valor de saldo pasa a ser negativo, se pone a cero el saldo de la cuenta y se inserta un prestamo con los mismos datos de la cuenta: sucursal, cliente, el número del prestamo igual al de la cuenta y por la misma cantidad que se quedaba en números rojos (el valor del nuevo saldo n.saldo).