ción de unicidad del mismo modo que fuerza la restricción de clave primaria. Cualquier intento de insertar o actualizar una fila en la tabla que viole la restricción de unicidad fallaría.

El estándar SQL ANSI/ISO utiliza la sentencia CREATE TABLE para especificar restricciones de unicidad en columnas o combinaciones de columnas. Sin embargo, las restricciones de unicidad fueron implementadas en DB2 mucho antes de la publicación del estándar ANSI/ISO y DB2 las hace parte de su sentencia CREATE INDEX. Esta sentencia es una de las sentencias de administración SQL que tiene que ver con el almacenamiento físico de la base de datos en el disco. Normalmente, el usuario de SQL no tiene que preocuparse de estas sentencias en absoluto; estas sentencias son utilizadas únicamente por el administrador de la base de datos. Desgraciadamente, si el usuario desea imponer una restricción de unicidad en un tabla DB2, deberá utilizar la sentencia CREATE INDEX. Tanto CREATE TABLE como CREATE INDEX se describen en el Capítulo 13.

La mayoría de los productos SQL comerciales siguen hoy día la práctica de DB2 en lugar del estándar ANSI/ISO en cuanto a las restricciones de unicidad. Específicamente, SQL Server, Oracle, Ingres, SQLBase y VAX SQL utilizan la sentencia CREATE INDEX para implementar restricciones de unicidad. Sin embargo, la mayoría de ellos tienen previsto incluir las restricciones de unicidad según la sintaxis ANSI/ISO cuando incluyan las características de SQL2.

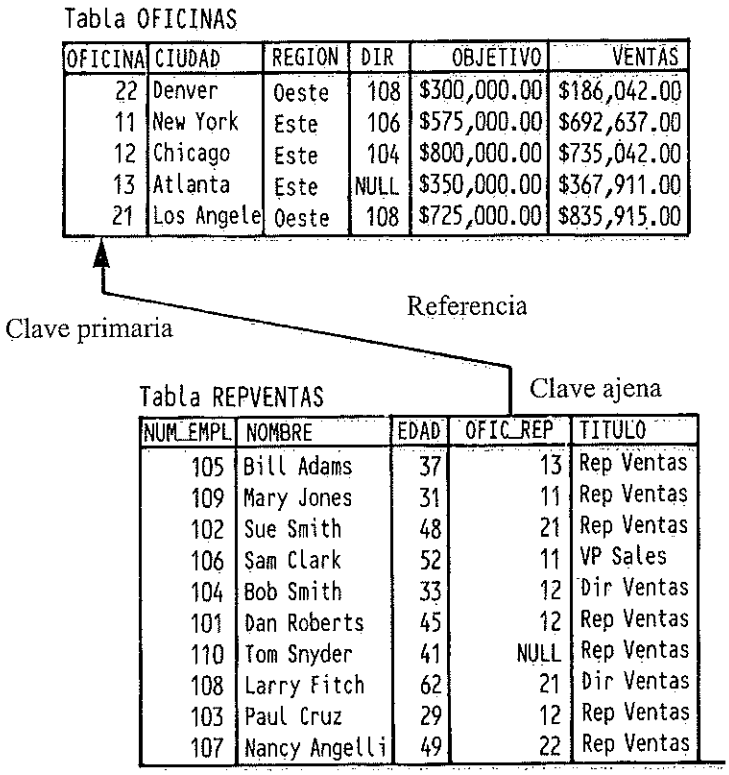
Unicidad y valores NULL

Los valores NULL presentan un problema cuando aparecen en la clave primaria de una tabla o en una columna que está especificada en una restricción de unicidad. Supongamos que se intentase insertar una fila con una clave primaria que fuera NULL (o parcialmente NULL, si la clave primaria está compuesta por más de una columna). Debido al valor NULL, el DBMS no puede decidir concluyentemente si la clave primaria está o no duplicada con respecto a otra ya existente en la tabla. La respuesta debe ser «quizá», dependiendo del valor «real» del dato que falta (NULL).

Por esta razón, SQL requiere que toda columna que forma parte de una clave primaria y toda columna designada en una restricción de unicidad deben ser declaradas NOT NULL.

INTEGRIDAD REFERENCIAL

En el Capítulo 4 se discutieron las claves primarias, las claves ajenas y las relaciones padre/hijo que estas claves crean entre tablas. La Figura 11.1 muestra las Tablas REPVENTAS y OFICINAS, e ilustra una vez más cómo operan las claves ajenas y las claves primarias. La columna OFICINA es la clave primaria para la Tabla OFICINAS, e identifica unívocamente a cada fila. La columna OFICINA_REP, en la Tabla REPVENTAS, es una clave ajena para la Tabla OFICINAS. Identifica la oficina a la cual está asignado cada vendedor.



OFICINA	CIUDAD	REGION	DIR	OBJETIVO	VENTAS
22	Denver	Oeste	108	$300,000.00	$186,042.00
11	New York	Este	106	$575,000.00	$692,637.00
12	Chicago	Este	104	$800,000.00	$735,042.00
13	Atlanta	Este	NULL	$350,000.00	$367,911.00
21	Los Angele	Oeste	108	$725,000.00	$835,915.00

NUM_EMPL	NOMBRE	EDAD	OFIC_REP	TITULO
105	Bill Adams	37	13	Rep Ventas
109	Mary Jones	31	11	Rep Ventas
102	Sue Smith	48	21	Rep Ventas
106	Sam Clark	52	11	VP Sales
104	Bob Smith	33	12	Dir Ventas
101	Dan Roberts	45	12	Rep Ventas
110	Tom Snyder	41	NULL	Rep Ventas
108	Larry Fitch	62	21	Dir Ventas
103	Paul Cruz	29	12	Rep Ventas
107	Nancy Angelli	49	22	Rep Ventas

Figura 11.1. Referencia clave ajena/clave primaria.

Las columnas OFICINA_REP y OFICINA crean una relación padre/hijo entre las Tablas OFICINAS y REPVENTAS. Cada fila OFICINAS (padre) tiene cero o más filas REPVENTAS (hijo) con números de oficina coincidentes. Análogamente, cada fila REPVENTAS (hijo) tiene exactamente una fila OFICINAS (padre) con un número de oficina coincidente.

Supongamos que se intenta insertar una nueva fila en la Tabla REPVENTAS que contenga un número de oficina inválido, como en este ejemplo:

```
INSERT INTO REPVENTAS (NUM_EMPL, NOMBRE, OFICINA_REP, EDAD, CONTRATO, VENTAS)
    VALUES (115, 'George Smith', 31, 37, '01-ABR-90', 0.00)
```

Aparentemente, no hay nada erróneo en esta sentencia INSERT. De hecho, la mayoría de las implementaciones SQL actuales añadirían sin problemas la fila. La base de datos mostrará que George Smith trabaja en la oficina número 31, aun cuando no existe la oficina número 31 en la Tabla OFICINAS. La fila recién insertada «rompe» claramente la relación padre/hijo entre las Tablas OFICINAS y REPVENTAS. De hecho, el número de oficina en la sentencia INSERT es probablemente un error —el usuario puede haber pretendido escribir el número de oficina 11, 21 o 13.

Parece claro que todo valor legal de la columna OFICINA_REP debería ser forzado para que corresponda a algún valor de los que aparezcan en la columna OFI-

CINA. Esta regla se conoce como restricción de *integridad referencial*. Asegura la integridad de las relaciones padre/hijo creadas mediante claves ajenas y claves primarias.

La integridad referencial ha sido una parte esencial del modelo relacional desde que fue propuesto por primera vez por Codd. Sin embargo, las restricciones de integridad referencial no fueron incluidas en el prototipo System/R de IBM, ni en las primeras ediciones de DB2 o SQL/DS. IBM añadió el soporte de integridad referencial a DB2 en 1989, y la integridad referencial fue añadida al estándar SQL1 después de su versión inicial. La mayoría de los vendedores de DBMS han implementado la integridad referencial o han hecho planes para incluir el soporte de integridad referencial en las versiones futuras de sus productos.

Problemas de integridad referencial

Existen cuatro tipos de actualizaciones de bases de datos que pueden corromper la integridad referencial de las relaciones padre/hijo en una base de datos. Utilizando para ilustración las Tablas OFICINAS y REPVENTAS de la Figura 11.1, estas cuatro situaciones de actualización son:

- *La inserción de una nueva fila hijo.* Cuando se inserta una nueva fila en la tabla hijo (REPVENTAS), su valor de clave ajena (OFICINA_REP) debe coincidir con uno de los valores de clave primaria (OFICINA) en la tabla padre (OFICINAS). Si el valor de clave ajena no coincide con ninguna clave primaria, la inserción de la fila corromperá la base de datos, ya que habrá un hijo sin un padre (un «huérfano»). Observe que insertar una fila en la tabla padre nunca representa un problema; simplemente se convierte en un padre sin hijos.

- *La actualización de la clave ajena en una fila hijo.* Esta es una forma diferente del problema anterior. Si la clave ajena (OFICINA_REP) se modifica mediante una sentencia UPDATE, el nuevo valor debe coincidir con un valor de clave primaria (OFICINA) en la tabla padre (OFICINAS). En caso contrario la fila actualizada será huérfana.

- *La supresión de una fila padre.* Si una fila de la tabla padre (OFICINAS), que tiene uno o más hijos (en la Tabla REPVENTAS) se suprime, las filas hija quedarán huérfanas. Los valores de clave ajena (OFICINA_REP) en estas filas ya no se corresponderán con ningún valor de clave primaria (OFICINA) en la tabla padre. Observe que suprimir una fila de la tabla hijo nunca representa un problema; el padre de esta fila simplemente tendrá un hijo menos después de la supresión.

- *La actualización de la clave primaria en una fila padre.* Esta es una forma diferente del problema anterior. Si la clave primaria (OFICINA) de una fila en la tabla padre (OFICINAS) se modifica, todos los hijos actuales de esa fila quedan huérfanos, puesto que sus claves ajenas ya no corresponden con ningún valor de clave primaria.

Las características de integridad referencial de DB2 y del estándar SQL ANSI/ISO se ocupan de cada una de estas cuatro situaciones. El primer problema (INSERT en la tabla hijo) se maneja comprobando los valores de las columnas de clave ajena antes de permitir ejecutar la sentencia INSERT. Si no coincide con un valor de clave primaria, la sentencia INSERT se rechaza con un mensaje de error. En la Figura 11.1 esto significa que antes de que un nuevo vendedor pueda ser añadido a la Tabla REPVENTAS, la oficina a la cual el vendedor está asignado debe existir ya en la Tabla OFICINAS. Como puede verse, esta restricción «tiene sentido» en la base de datos ejemplo.

El segundo problema (UPDATE de la tabla hijo) es análogamente tratado mediante comprobación del valor de clave ajena actualizado. Si no hay un valor de clave primaria con el que coincida, la sentencia UPDATE se rechaza acompañada de un mensaje de error. En la Figura 11.1 esto significa que antes de que un vendedor pueda ser reasignado a una oficina diferente, esa oficina debe ya existir en la Tabla OFICINAS. De nuevo, esta restricción tiene sentido en la base de datos ejemplo.

El cuarto problema (UPDATE de la clave primaria en la tabla padre) también se controla por prohibición. Antes de que la clave primaria pueda ser modificada, el DBMS efectúa comprobaciones para asegurarse de que no haya filas hijo que tengan valores de clave ajena correspondientes. Si hay tales filas hijo, la sentencia UPDATE se rechaza junto con un mensaje de error. En la Figura 11.1 esto significa que un número de oficina no puede ser modificado en la Tabla OFICINAS a menos que no haya vendedores en la Tabla REPVENTAS actualmente asignados a esa oficina. En la práctica, esta restricción no suele provocar problemas, ya que los valores de clave primaria casi nunca se modifican.

El tercer problema (DELETE de una fila padre) es más complejo. Por ejemplo, supongamos que se cierra la oficina de Los Ángeles y que se desea suprimir la fila correspondiente de la Tabla OFICINAS en la Figura 11.1; ¿qué le sucedería a las dos filas hijo en la Tabla REPVENTAS que representan a los vendedores asignados a la oficina de Los Ángeles? Dependiendo de la situación, se podría desear:

- impedir que la oficina fuera suprimida hasta que los vendedores fueran reasignados;
- suprimir automáticamente a los dos vendedores de la Tabla REPVENTAS también; o
- poner la columna OFICINA_REP de los dos vendedores a NULL, indicando que la asignación de oficina es desconocida;
- poner la columna OFICINA_REP de los dos vendedores a algún valor por defecto, como pueda ser el número de oficina de la oficina central de Nueva York, indicando que los vendedores automáticamente son reasignados a esa oficina.

El cuarto problema (UPDATE de la clave primaria en la tabla padre) tiene una complejidad similar. Por ejemplo, supongamos que por alguna razón se desea cambiar el número de oficina de la oficina de Los Ángeles del 21 al 23. Al igual que en el ejemplo anterior, la pregunta es qué ocurre con las dos filas hijo de la

Tabla REPVENTAS que representan los vendedores de la oficina de Los Ángeles. Otra vez hay cuatro posibilidades lógicas:

- impedir que el número de la oficina se modifique hasta que los vendedores sean reasignados. En este caso, primero se añadiría una nueva fila en la Tabla OFICINAS con el nuevo número de oficina para Los Ángeles; después se actualizaría la Tabla REPVENTAS y finalmente se elimina la antigua fila de Los Ángeles en la Tabla OFICINAS;
- actualizar automáticamente el número de la oficina de los dos vendedores en la Tabla REPVENTAS, con lo que sus filas todavía estarían relacionadas con la fila de Los Ángeles en la Tabla OFICINAS, a través de su nuevo número de oficina;
- poner la columna OFICINA_REP de los dos vendedores a NULL, indicando que la asignación de oficina es desconocida;
- poner la columna OFICINA_REP de los dos vendedores a algún valor por defecto, como pueda ser el número de oficina de la oficina central de Nueva York, indicando que los vendedores automáticamente son reasignados a esa oficina.

Aunque algunas alternativas pueden parecer más lógicas que otras en este ejemplo, es relativamente fácil proponer ejemplos en los que una de las cuatro posibilidades es lo «correcto» que se debe hacer si se desea que la base de datos sea un modelo exacto de la situación del mundo real. El estándar SQL1 solamente proporciona la primera opción en los ejemplos anteriores —está prohibido modificar el valor de una clave primaria que esté «en uso» y está prohibido eliminar una fila que contenga tal clave primaria. Sin embargo, DB2 permite otras opciones a través de su concepto de *reglas de supresión*. El estándar SQL2 ha ampliado estas reglas de supresión en *reglas de supresión* y *actualización* que cubren tanto la supresión de filas padre como la actualización de claves primarias.

Reglas de supresión y actualización *

Por cada relación padre/hijo creada mediante una clave ajena en una base de datos DB2, se puede especificar una regla de supresión asociada y una regla de actualización asociada. La regla de supresión le dice al DBMS qué debe hacer cuando un usuario trate de suprimir una fila de la tabla padre. Se puede especificar una de las cuatro reglas de supresión posibles:

- La regla de supresión RESTRICT impide suprimir una fila de la tabla padre si la fila tiene alguna hijo. Una sentencia DELETE que intente suprimir una fila padre tal como ésa es rechazada junto con un mensaje de error. Las supresiones de la tabla padre están por tanto restringidas a filas sin hijos. Aplicada a la Figura 11.1, esta regla puede resumirse como «no se puede suprimir una oficina si tiene vendedores asignados».