¿Por qué convertir archivos físicos con DDS a DDL?

¿Alguna vez ha pensado en **modernizar su base de datos**? Muchas empresas con IBM i han reemplazado la típica pantalla verde con una interfaz gráfica y han convertido sus monolíticos programas heredados en procedimientos y funciones escritas en modernos lenguajes de programación. Pero ¿y qué sucede con la base de datos?

Con frecuencia los datos todavía se siguen almacenando en archivos físicos DDS (Data Description Specification) que han ido creciendo a lo largo del tiempo y que incluyen mucha redundancia, lo que requiere un esfuerzo de programación adicional. Evidentemente esto hace necesaria su modernización. Sin embargo, la base de datos no se puede convertir ni rediseñar en un solo paso: la actividad de negocio diaria debe continuar. Aun así, debe iniciarse en algún punto y lo que primero se puede hacer es convertir los archivos físicos DDS en tablas definidas por DDL (Data Definition Language), para luego convertir los archivos lógicos DDS con clave en índices SQL.

Cuando hablamos con clientes o programadores acerca de pasar de DDS a DDL, a menudo no lo ven totalmente necesario. Argumentan que los archivos físicos descritos por DDS y las tablas definidas por SQL

pueden ser accedidas y utilizadas mediante SQL y E/S nativas exactamente de la misma manera.

ENTONCES ¿POR QUÉ CONVERTIR LOS ARCHIVOS FÍSICOS EN TABLAS SQL?

- DDS para definir objetos de base de datos solo es utilizado en IBM i, mientras que SQL es estándar y todos los fabricantes de bases de datos están comprometidos con este estándar. Aunque es cierto que hay algunas variantes de SQL, las diferencias entre los mandatos DDL para diferentes bases de datos son marginales.

- Cualquier programador o administrador de base de datos recién salido de un instituto o que provenga de un sistema operativo o base de datos distinta, tiene que aprender primero la sintaxis de DDS, mientras que la enseñanza y uso de SQL está totalmente generalizado.

- DDS está estabilizado, es decir, la última mejora fue en la versión V5R3M0 para permitir que los campos empaquetados y de zona puedan tener hasta 63 dígitos.

- Todas las mejoras actuales se realizan en SQL, e incluso hay mejoras en versiones anteriores (por ejemplo, Columnas de Identidad introducidas con la versión V5R1M0) que no están disponibles en DDS. La siguiente lista es solo una breve selección de características importantes que solo pueden ser usadas en SQL:
  - Tipos de datos adicionales: CLOB, DBCLOB, BLOB (objetos grandes), DECFLOAT, XML, ROWID
  - Columnas de Auditoría: por ejemplo, ROW CHANGE TIMESTAMP que almacena automáticamente la última fecha/hora de actualización
  - Columnas de Identidad: genera automáticamente valores de clave única

- Muchas compañías todavía siguen utilizando nombres crípticos de columnas de 6 a 10 caracteres, porque RPGIII solo podía manejar 6 caracteres y RPGIV (antes de la versión 6.1) solo 10 caracteres. Aunque era posible generar nombres más largos en DDS (mediante el uso de la palabra clave ALIAS), estos rara vez se definían debido a las restricciones propias de RPG.

Muy a menudo, estos nombres de columna son crípticos o consisten solo de unos pocos caracteres seguidos de un número, lo que dificulta que los nuevos programadores entiendan cuál es el contenido del campo.

Con SQL, se pueden crear columnas largas y nombres de objetos de bases de datos de hasta 128 caracteres, y estos pueden coexistir con nombres de sistema cortos (máximo de 10 caracteres).

Trabajar con nombres descriptivos más largos hace la vida mucho más fácil, pero si desea agregar nombres largos de columna, primero debe planificarlos basándose en convenciones de nomenclatura predefinidas. De lo contrario, es probable que termine con diferentes nombres la columna de número de cliente como CustNo, CustomerNumber, Customer_No o Customer #.

ALGUNAS DIFERENCIAS ENTRE LOS ARCHIVOS FÍSICOS DDS Y LAS TABLAS SQL QUE DEBE CONOCER

1.- Al escribir, actualizar o eliminar registros con E/S nativa se necesita un nombre de formato diferente del nombre de la tabla.
Según el estándar SQL, una tabla SQL no incluye un formato de registro. Pero debido a una mejora no estándar en Db2 for i, es posible agregar un formato de registro a una tabla SQL especificando la palabra clave RCDFMT en la instrucción CREATE TABLE.
2.- **Un archivo físico DDS puede incluir una definición de clave.**
Las tablas SQL siempre están sin clave. Pero las claves únicas se pueden convertir en una clave primaria, y no hay forma de incluir claves no únicas en una tabla SQL.

Al convertir de DDS a DDL, primero debe verificar los programas con acceso con clave en el archivo físico. Luego, cree un nuevo archivo lógico o, aún mejor, un índice SQL con los campos clave apropiados y cambie sus programas RPG para acceder al nuevo índice o archivo lógico en lugar del archivo físico. Luego puede convertir su archivo físico DDS en una tabla SQL, ignorando la información de clave ... sin necesidad de utilizar su clave principal, lo cual puede ser importante para implementar en el futuro integridad referencial basada en Columnas de Identidad.

Por cierto, si tiene una clave única en su archivo físico y no crea un índice adicional, su programa RPG puede gestionar la restricción de clave primaria exactamente de la misma manera que gestionó la clave única en el archivo físico.

3.- Las diferencias arquitectónicas menores en los objetos de la base de datos ofrecen ventajas significativas.
Al insertar datos en un archivo físico DDS, los datos no se verifican cuando se escriben en el archivo sino que se verifican cuando se leen.

De esta manera, es posible ingresar valores numéricos no válidos (por ejemplo, * BLANKS) en los campos numéricos. Esta situación puede ocurrir ejecutando el comando CPYF CL con la opción de formato (FMTOPT) * NOCHK. En programas antiguos, a menudo nos encontramos con esta situación, específicamente cuando la estructura de datos de registro dentro del programa RPG no se inicializó correctamente. Incluso si sus programas están funcionando correctamente, es posible que tenga registros con valores numéricos no válidos en sus archivos físicos DDS. Estos datos no válidos deben detectarse y rectificarse antes de convertir el archivo físico a una tabla SQL.

Por el contrario, al insertar datos en una tabla definida por DDL, los datos son verificados cuando se escriben y no se verifican cuando se leen, porque la tabla solo puede incluir datos válidos. De esta manera no es posible escribir datos numéricos no válidos en campos numéricos, ni siquiera con un CPYF con la opción de formato * NOCHK. Por lo tanto, una tabla definida por SQL incluye solo datos válidos, lo cual es una gran ventaja.

Pero también hay otra ventaja de convertir archivos físicos descritos por DDS en tablas SQL:

¿Qué sucede si los datos se verifican cuando se escriben, pero no cuando se leen en comparación con lo contrario? ¿Cuántas operaciones de lectura y cuántas operaciones de escritura / actualización se realizarán? ¿Quizás 80% de lectura y 20% de escritura / actualización?

Aunque hablamos de nano segundos, verificar los datos lleva tiempo. Si el 80% de las operaciones de la base de datos ya no se verifican (porque los datos se leen de una tabla SQL) y solo se verifica el 20% de las operaciones de la base de datos (escritura / actualización), el acceso al archivo se vuelve más rápido, independientemente de si el acceso se realiza con SQL o E/S nativa.

En resumen, **la conversión de archivos físicos DDS a tablas SQL es muy beneficiosa.**

Después de la conversión, podemos comenzar a incluir las nuevas funciones que fueron añadidas en las últimas versiones, como Columnas de Identidad, Columnas de Auditoría, Tablas temporales, etc. También podemos comenzar o continuar incluyendo la lógica del programa en la base de datos, por ejemplo agregando Constraints, Integridad Referencial y Triggers. Y con todo esto, ya hemos puesto la primera piedra para un futuro rediseño de la base de datos.

LOUPREY
INTERNATIONAL

(55) 5543 6515
marketing@louprey.com
www.louprey.com
Kansas #7 Piso 2, Col. Nápoles, C.P. 03810,
Benito Juárez, Ciudad de Mexico