UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

PRINCIPALES NOVEDADES DE SQL3 FRENTE A SQL2

Miguel Ángel Moreno Carpintero

Profesor:	Francisco Ruiz
Asignatura:	Bases de Datos
Titulación:	Ingeniería Técnica en Informática de Sistemas
Fecha:	10-05-2001

1.- ANTECEDENTES:

Desde hace varios años, todos hemos estado escuchando y leyendo sobre la aparición de un estándar que todos hemos estado llamando SQL3. Prometió ser un aumento de la actual segunda generación de estándar SQL, comúnmente conocido como SQL-92, debido al año de su publicación, SQL3 fue originalmente planeado para su uso sobre el año 1996, pero los planes no salieron bien.

Debemos ser conscientes de que SQL3 está caracterizado como "SQL orientado a objetos" y es la base de algunos sistemas de manejo de bases de datos orientadas a objetos(incluyendo ORACLE, Informix'Universal Server, IBM's DB" Universal Database y Cloudscape, además de otros).

Esto es una convicción general de que es buena cosa, pero tardó 7 años en desarrollarse en vez de los tres o cuatro que se pensaba iba a tardar.

Como nosotros mostraremos, SQL:1999 es mucho más que la mera tecnología de objetos de SQL92 plus. Esto envuelve características adicionales que consideramos herencia de los SQL relacionales, así como también una reestructuración de los documentos de los estándar con vista a una mayor progresión hacia normas más efectivas.

2.- PROCESO DE DESARROLLO DE NORMAS:

Las dos organizaciones de jure que se involucraron en la estandarización de SQL, y por lo tanto en el desarrollo de SQL:1999 son ANSI e ISO. Más específicamente, la comunidad internacional de trabajos mediante ISO/IEC JTC1(Joint Technical Committee 1), un comité formado por la organización internacional de estandarización junto con la comisión internacional electrotécnica. La responsabilidad de las JTC1 es desarrollar y mantener la información relativa a la tecnología. Dentro de JTC1, el subcomité SC32, cuya función era el intercambio y gestión de datos se formó para la estandarización de normas relativas a varias bases de datos y metadatos que habían sido desarrollados por otras organizaciones (tal como el ahora disuelto SC21). SC32,es a la vez, un número de grupos de trabajo que actualmente realizan los trabajos técnicos-WG3(lenguajes de bases de datos) es responsable de las normas de SQL, mientras WG4 está desarrollando el SQL/MM (SQL multimedia, un departamento de normas que especifiquen las bibliotecas de tipos usando facilidades de SQL orientado a objetos).

En los Estados Unidos, IT estándares son manejados por la institución de acreditación y desarrollo de normas nacionales estadounidenses, el comité NCITS(Comité nacional para la estandarización de tecnología de la información, anteriormente conocido como X3). NCITS comité técnico H2 (anteriormente X3H2) es responsable de varios estándares relativos a la gestión de datos, incluyendo SQL y SQL/MM.

Cuando la primera generación de SQL fue desarrollada (SQL-86 y su aumento menor SQL-89), casi todo el desarrollo se hizo en Estados Unidos por X3H2 y otras naciones participaron en su mayor parte en el modo de revisar y criticar el trabajo propuesto por ANSI. Por el momento SQL-89 fue publicado, la comunidad internacional hacía por escrito propuestas para la especificación que al final llegó a ser SQL-92; que no ha cambiado mientras SQL:1999 está siendo desarrollado, no creemos que halla cambios en el futuro –SQL es realmente un esfuerzo de colaboración internacional.

Las primeras versiones de la norma son conocidas como SQL-86 (o SQL-87, porque la versión no fue publicada hasta 1987), SQL-89 y SQL-92, mientas que la versión actual debe llegar a ser conocida como SQL:1999. ¿Por qué no SQL-99?, bien, simplemente porque debemos pensar en el nombre de la próxima generación y SQL-02 puede ser confundido con

SQL2 (que era el proyecto bajo el cual fue desarrollado SQL-92). En otras palabras, no queremos que desde el año 2000 se produzcan problemas con los nombres de SQL.

3.- LOS CONTENIDOS DE SQL:1999:

Reconozco que los lectores de este documento no sabrán los contenidos precisos de SQL-92. Este documento va ha describir los aspectos nuevos de esta generación en desarrollo de SQL. Los aspectos pueden ser divididos en "aspectos relacionales" y "aspectos relacionados con objetos".

4.- ASPECTOS RELACIONALES:

Aunque denominemos esta categoría como aspectos relacionales, usted reconocerá que es más adecuado llamarla como "aspectos que relacionan el papel de SQL en el modelado de datos". Estos aspectos no están estrictamente limitados al modelo relacional, pero no están relacionados con la orientación a objetos.

Estos aspectos se dividen en cinco grupos: nuevos tipos de datos, nuevos predicados, semántica mejorada, seguridad adicional, la base de datos activa. Hablaremos de cada uno de los grupos por separado.

5.- NUEVOS TIPOS DE DATOS:

SQL:1999 tiene cuatro nuevos tipos de datos (aunque alguno de ellos tiene variantes identificables). El primero de estos tipos es LARGE OBJECT(objeto grande) o LOB. Las variantes de este tipo son las siguientes:

- CHARÁCTER LARGE OBJECT(CLOB)
- BINARY LARGE OBJECT(BLOB)

CLOB funciona como una cadena de caracteres, pero tiene restricciones que impiden su uso como PRIMARY KEY o predicados UNIQUE, FOREIGN KEY, y en comparaciones a excepción de las de igualdad o desigualdad.

BLOB tiene restricciones similares. (Por implicación, LOB no puede ser usado en las cláusulas GROUP BY u ORDER BY). Las aplicaciones típicamente no podrán transferir el valor entero de un LOB hacia una base de datos(después del almacenado inicial, esto es); pero podría manipular valores LOB mediante un tipo de cliente especial que lo soporte denominado "LOB locator".

En SQL:1999, un locator es un único valor binario que actúa como sustituto para un valor que está dentro de la base de datos; los locators pueden ser usados en operaciones (tales como SUBSTRING)

Otro tipo de dato nuevo es el BOLEAN, que permite a SQL registrar valores de verdad, falso y desconocido. Complejas combinaciones de predicados pueden ser ahora expresadas de una manera más sencilla que antes. Ejemplo:

```
WHERE     COL1>COL2 AND
          COL3=COL4 OR
          UNIQUE(COL6) IS NOT FALSE;
```

SQL:1999 también tiene dos nuevos tipos compuestos: ARRAY y ROW. El tipo ARRAY permite almacenar una colección de valores directamente en una columna de una tabla. Ejemplo:

```
WEEKDAYS     VARCHAR(10)     ARRAY(7)
```

Podríamos almacenar los nombres de los días de la semana en una columna en la base de datos.

¿Esto significa que SQL:1999 permite bases de datos que no satisfagan la 1ª forma normal?. Desde luego, lo hace en el sentido de que permite los grupos repetitivos, que la primera forma normal prohíbe. (Sin embargo, algunos sostienen que los tipos ARRAY de SQL:1999 meramente permiten almacenar información que puede ser descompuesta, muchas como la función SUBSTRING puede descomponer strings de caracteres y por lo tanto no infringen la 1ª forma normal).
El tipo ROW en SQL:1999 permite el almacenamiento estructurado de datos en columnas únicas de la base de datos. Ejemplo:

```
CREATE TABLE employee(

        EMP_ID      INTEGER,

        NAME        ROW(
                GIVEN       VARCHAR(30)
                FAMILY      VARCHAR(30)),

        ADDRESS     ROW(
                STREET      VARCHAR(50),
                CITY        VARCHAR(30),
                STATE       CHAR(2)),

        SALARY      REAL)

SELECT      E.NAME.FAMILY
FROM employee   E;
```

Mientras usted podría argumentar que esto también infringe la primera forma normal, la mayoría de observadores reconocen que simplemente es otro tipo de datos descompuesto.

SQL:1999 añade también facilidades para las relaciones entre tipos de datos distintos. Reconociendo que sería inverosímil comparar directamente el tamaño de zapato de un empleado con su "IQ", el lenguaje permite declarar SHOE_SIZE e IQ como entero, pero prohíbe que se mezclen en expresiones. Así, una expresión como:

```
WHERE       MY_SHOESIZE > MY_IQ
```

Se reconoce como un error de sintaxis. Cada uno de estos tipos puede ser representado como un entero, pero no permite su mezcla en expresiones así como tampoco multiplicar por otro entero:

```
SET     MY_IQ = MY_IQ * 2
```

Además de estos tipos, SQL:1999 tiene también definidos tipos definidos por el usuario, pero esto lo veremos dentro de la orientación a objetos.

6.- NUEVOS PREDICADOS:

SQL:1999 tiene tres nuevos predicados, uno de los cuales lo veremos conjuntamente con la orientación a objetos. Los otros dos son el predicado SIMILAR y el predicado DISTINTO.

Desde la primera versión del SQL estándar, las cadenas de caracteres están limitadas a simples comparaciones (como =, > ó <>) y las rudimentarias capacidades del predicado like:

```
WHERE       NAME    LIKE    '%SMIT_'
```

Con esto diremos que hay cero o más caracteres que preceden a los cuatro caracteres 'SMIT' y exactamente uno después de ellos (tal como SMITH o HAMMERSMITS). Reconociendo que las aplicaciones requieren capacidades mas sofisticadas, SQL:1999 introdujo el predicado SIMILAR que ofrece la posibilidad de equiparar modelos. Ejemplo:

```
WHERE  NAME  SIMILAR  TO

       '(SQL-(86|89|92|99)) | (SQL(1|2|3))'
```

(que equipararía los diversos nombres dados al SQL a lo largo de los años). Es un poco desafortunado que la sintaxis de las expresiones regulares usadas en el predicado SIMILAR no saquen partido de las expresiones regulares de UNIX, pero algunos caracteres de UNIX son usados para otros propósitos en SQL.

El otro nuevo predicado, DISTINCT, es muy similar al predicado UNIQUE; la diferencia importante es que dos valores nulos eran considerados no iguales el uno del otro y de este modo podrían satisfacer el predicado UNIQUE, pero no en todas las aplicaciones se daba este caso. El predicado DISTINCT considera que dos valores nulos no pueden ser distintos el uno del otro (o son iguales o distintos) así que dos valores nulos podrían hacer que un predicado DISTINCT no sea satisfactorio.

7.- NUEVA SEMÁNTICA EN SQL:1999:

Es difícil saber exactamente fijar un límite a la hora de hablar de la nueva semántica en SQL:1999, pero daré una selección de lo que creo más importante. Una de las demandas de los escritores de aplicaciones era ampliar las clases de vistas. Muchos entornos usan vistas pesadas como mecanismo de seguridad y/o como simplificación de una aplicación vista de la base de datos. Como siempre, si la mayoría de vistas no son actualizables, luego estas aplicaciones frecuentemente tienen que escapar de los mecanismos de vista y confiar directamente en la modificación de las tablas subyacentes; esto es una situación poco satisfactoria.

SQL:1999 ha incrementado significativamente el número de vistas que pueden ser modificadas directamente, usando solo lo que se facilitó en el estándar. Esto depende en gran medida de las dependencias funcionales para ir determinando que vistas pueden ser modificadas, y como hacer estos cambios en la tabla de la base de datos para efectuar estas modificaciones.

Otra deficiencia criticada en gran medida de SQL era la imposibilidad de construir aplicaciones recursivas. SQL:1999 está provisto para hacer llamadas recursivas, cosa que satisface lo dicho con anterioridad. Escribiremos la pregunta en base a la cual deseamos hacer la recursión y daremos un nombre. Luego usaremos el nombre asociando la pregunta a una expresión:

```
WIT RECURSIVE
        Q1   AS  SELECT .....FROM.....WHERE.....,
        Q2   AS  SELECT…….FROM…….WHERE…..
SELECT…….FROM Q1,Q2   WHERE……
```

Nosotros hemos mencionado ya los locators como un valor del cliente que puede representar un valor LOB almacenado en el servidor. Los locator pueden ser usados en algunos casos para representar los valores de un ARRAY, aceptando el hecho de que (como LOB) los array pueden ser frecuentemente demasiado grandes para ser pasado convenientemente entre una aplicación y la base de datos. Los locator pueden ser usados también para representar tipos de valores de datos indefinidos (luego lo discutiremos) que también tienen la potencialidad de ser grandes y abultados.

Finalmente, SQL:1999 ha agregado la noción de savepoints, ampliamente implementado en sus productos. Un savepoint es como una pequeña substracción en la que una aplicación puede deshacer las acciones realizadas después de comenzar el savepoint sin deshace todas las acciones de una transacción entera.

8.- MEJORAS DE SEGURIDAD:

Las nuevas facilidades de seguridad en SQL:1999 tienen un papel muy importante. Los privilegios pueden ser otorgados según un rol y este a su vez puede otorgar privilegios individuales para otros roles. Esta estructura anidada mejora el manejo de la seguridad en el ambiente de una base de datos. Los roles han estado siendo implementados por los productos SQL desde hace varios años (aunque ocasionalmente bajo nombres diferentes).

9.- BASE DE DATOS ACTIVA:

SQL:1999 reconoce la noción de base de datos activa. Esto es facilitado por lo que se conoce como triggers (disparadores). Un trigger, como sabéis, es una facilidad que permite a los diseñadores de bases de datos relizar operaciones seguras siempre que una aplicación realice operaciones en tablas particulares. Por ejemplo, los triggers podrían emplearse para registrar todas las operaciones que cambien salaries en la tabla empleado.

```
CREATE TRIGGER    log_salupdate

        BEFORE  UPDATE   OF   salary
                ON       employees

        REFERRENCING   OLD    ROW    as   oldrow
                       NEW    ROW    as   newrow

        FOR EACH  ROW
        INSERT INTO   log_table

                VALUES(CURRENT_USER,
                       oldrow.salary,
                       newrow.salary)
```

Los triggers pueden ser usados para muchos propositos, no solo para registrar. Por ejemplo, tu puedes escribir triggers que guarden un balance de presupuesto para saber si puedes contratar a nuevos empleados o no.

10.- ORIENTACIÓN A OBJETOS:

Además de las características discutidas hasta ahora, SQL:1999 se caracteriza porque fue desarrollado principalmente para manejar objetos. Algunas de las características que están dentro de esta categoría fueron definidas en el estándar SQL/PSM publicado en 1996 – específicamente para llamadas a funciones y procedimientos desde SQL.. SQL:1999 mejora esta capacidad que llamó SQL-invoked routines, para añadir una tercera clase de rutina comocida como método, que luego veremos.

11.- TIPOS DE ESTRUCTURAS DEFINIDAS POR EL USUARIO:

La mayor facilidad en SQL:1999 que soportan la orientación a objetos son los tipos estructurados definidos por el usuario; Los tipos estructurados tienen un número de características, las más importantes son:

- Pueden ser definidos para tener uno o más atributos, cada uno de ellos pueden ser algún tipo de SQL, incluyendo tipos empotrados como INTEGER, tipos de colección como ARRAY, u otro tipo de estructuras.
- Todos los aspectos de su comportamiento son provistos mediante métodos, funciones y procedimientos.
- Sus atributos son encapsulados mediante el uso del sistema generador observador y mutador de funciones (funciones get y set) que provee el único acceso a sus valores. Sin embargo, este sistema no puede ser sobrecargado; todas las otras funciones y métodos pueden ser sobrecargados.
- Las comparaciones de sus valores son únicamente realizadas mediante funciones definidas por el usuario.
- Ellos pueden participar en jerarquías de tipo, en las cuales más tipos especializados (subtipos) tienen todos sus atributos y usan todas las rutinas asociadas con más tipos generalizados (supertipos), pero pueden agregar nuevos atributos y rutinas.

Veamos un ejemplo de la definición de un tipo estructurado:

```
CREATE        TYPE   emp_type
        UNDER  person_type

AS (EMP_ID       INTEGER,
     SALARY      REAL)

INSTANTIABLE
NOT FINAL
REF(EMP_ID)
INSTANCE   METHOD
        GIVE_RAISE
                (ABS_OR_PCT    BOOLEAN,
                 AMOUNT        REAL)
        RETURNS    REAL
```

Este nuevo tipo es un subtipo de otro tipo estructurado que podría ser usado para describir personas en general, incluyendo atributos comunes parecidos como name y address; los atributos del nuevo emp_type incluyen cosas que personas ancianas no tienen, como un ID de empleado y un salario. Nosotros hemos declarado este tipo para ser instanciable y permitir que tenga subtipos definidos (NOT FINAL). En suma, nosotros diremos que algunas referencias a este tipo son derivadas del valor del ID del empleado.
Finalmente definimos un método que puede ser aplicado a instancias de este tipo.

SQL:1999, después de una extensiva coquetería con la herencia múltiple (en la cual los subtipos se les permitió tener más de un supertipo inmediato), ahora tiene un modelo de tipo estrechamente relacionado con la herencia simple de Java. Los definidores de tipo se permiten especificar que un tipo determinado es instanciable (en otras palabras, valores de qué tipo específico pueden ser creados) o no instanciable (análogo a los tipos abstractos en otros lenguajes de programación). Y naturalmente, algún lugar –tal como una columna- donde un valor de algún tipo estructurados es permitido, un valor de alguno de estos subtipos pueden aparecer; esto provee la clase de substitutibilidad de la cual los programas orientados a objetos dependen.

Por la manera, algunos lenguaje de programación orientada a objetos, tal como C++ permite definidores de tipo para saber el grado de encapsulación de sus tipos: un nivel PUBLIC de encapsulación aplicado a un atributo significa que cualquier usuario del tipo puede acceder al atributo, PRIVATE significa que el atributo no puede ser usado por el código de otro método, y PROTECTED significa que solo el método del tipo y métodos del subtipo del tipo pueden acceder al atributo.

SQL:1999 no tiene este mecanismo, aunque se intentó; nosotros anticipamos que esto puede ser propuesto para una futura revisión del estándar.

12.- FUNCIONES vs MÉTODOS:

SQL:1999 hace una importante distintión entre las típicas llamadas a funciones y las llamada a métodos. En resumen, un método es una función con varias restricciones y aumentos. Dejaremos resumidas las diferencias entre los dos tipos de rutina:

- Los métodos están estrechamente limitados a un simple tipo definido por el usuario.
- Las funciones pueden ser polimórficas (sobrecargadas), pero una función específica es elegida por compilar en tiempo para examinar las declaraciones de tipos de cada argumento de una invocación a una función y eligiendo el mejor partido entre las funciones candidatas (teniendo el mismo nombre y número de parámetros; los métodos pueden ser también polimórficos, pero la mayoría de tipos específicos de sus argumentos distinguidos, determinados en tiempo de ejecución, permite la selección del método exacto para ser invocado.

13.- NOTACIONES FUNCIONALES Y DE PUNTO:

El acceso a los atributos de tipos definidos por el usuario puede hacerse mediante dos notaciones. En muchas situaciones, las aplicaciones pueden parecer más naturales cuando se usa notación de punto.

```
WHERE    emp.salary > 10000
```

Mientras en otras situaciones, una notación funcional puede ser más natural.

```
WHERE salary(emp) > 10000
```

SQL:1999 soporta las dos notaciones; de hecho, están definidas para ser variaciones sintácticas de la misma cosa.

Los métodos son un poco menos flexibles que las funciones en este caso: Solamente se puede usar notación de punto para las llamadas a métodos.

```
emp.salary
```

Un método diferente, digamos give_raise, podria combinar las dos notaciones:

```
Emp.give_raise(amount)
```

14.- OBJETOS……FINALMENTE:

Los lectores cuidadosos habrán observado que hemos evitado el uso de la palabra objeto hasta aquí en nuestra descripción de tipos estructurados. Esto es por que, a pesar de ciertas características como tipo de jerarquías, encapsulación, y un largo etc, instancias de tipos estructurados de SQL:1999 son simplemente valores, como instancias de construcción de tipos del lenguaje. Seguramente, un valor employee es mucho más complejo (tanto en apariencia como en su comportamiento) que una instancia de INTEGER.

En orden a ir ganando terreno a las características que permiten a SQL proveerse de objetos, habrá algún sentido de identificar lo que puede hacer referencia a una variedad de situaciones. En SQL:1999, esta capacidad es facilitada para permitir que los diseñadores de

bases de datos puedan especificar que ciertas tablas son definidas para ser “typed tables”.... esto es, sus definiciones de columna son derivadas de los atributos de un tipo estructurado.

```
CREATE TABLE    empls   OF   employee
```

Tales tablas tienen una columna para cada atributo del tipo estructurado subyacente. ¡Las funciones, métodos y procedimientos definidos para operar en instancias del tipo, ahora operan en columnas de la tabla!. A cada columna se le dá un único identificador que se comporta exactamente como un OID (identificador de objeto), único en el espacio (esto es, dentro de la base de datos) y tiempo (la vida de la base de datos).

SQL:1999 está provisto de un tipo especial, llamado tipo REF, cuyos valores son estos identificadores únicos. Un tipo dado de REF está siempre asociado con un tipo estructurado específico. Por ejemplo, si nosotros definimos una tabla que contiene una columna llamada “manager”, tales valores harán referencia a columnas en un tipo de tabla empleado, lo hace así:

```
Manager    REF(emp_type)
```

En definitiva, un tipo REF es en esencia un puntero que señala a un tipo de objeto definido por el usuario.

```
SELECT emp.manager ◖ last_name
```

La notación del indicador (◖) es aplicada a un valor de algún tipo REF.

BIBLIOGRAFÍA:

"La biblia de oracle 8" ANAYA

"PL/SQL User's Guide and Reference" Release 8.1.5 ORACLE

REFERENCIAS WEB:

American National Standars Institute (ANSI)
http://web.ansi.org

International Organizatión for Standarization (ISO)
http://www.iso.ch

JTC1 SC32 –Data Management and Interchange

http://bwonotes5.wde.pnl.gov/SC32/JTC1SC32.nsf

National Committee for Informatión Technology Standars (NCITS)

http://www.ncits.org

NCITS H2 – Database

http://www.ncits.org/tc_home/h2.htm