¿Desarrollo orientado a procesos u orientado a datos?

Algunas reflexiones en el 40° aniversario de los Sistemas de Gerencia de Bases de Datos

Preparado por Breogán Gonda
bgv@artech.com.uy

Advertencia previa

No quiero que el lector se llame a engaño y se frustre con la lectura de este trabajo: nunca he tenido la posibilidad ni la vocación de mirar la vida pasar. Siempre he asumido fuertes compromisos profesionales y, por todo ello, no pretendo que se me considere objetivo.

Bases de Datos: un poco de historia y reflexiones

El año 1963 fue un año fermental en la informática. ¡Fue el año de la llamada 3ª Generación de computadores!: por primera vez los fabricantes comenzaron a pensar que empresas normales podrían beneficiarse del uso de la informática que, hasta entonces, estaba restringido a enormes proyectos, muchos de ellos de carácter estratégico y ligados con la guerra fría.

Un buen día de 1963 Honeywell lanzó su H200, primer computador de la llamada 3ª Generación. Rápidamente otras empresas respondieron con otros computadores de "3ª Generación", o anuncios de los mismos (en ciertos casos tan sólo enunciando algunas de sus características y mostrando maquetas de sus gabinetes, porque habían sido tomadas por sorpresa y sus proyectos estaban en un estado primitivo): el líder IBM con su "/360", General Electric con sus "Compatibles 400" y "Compatibles 600", RCA con su línea "Spectra", Univac lanzando nuevo hardware y su legendario sistema operativo "Exec", etc.

¿Qué fue lo más importante de la 3ª Generación? De acuerdo a sus repercusiones, los elementos más importantes fueron la formalización, como algo independiente del hardware, de los conceptos de software, en general y del sistema operativo, en particular, y un fuerte impulso a los lenguajes de programación estándares (Cobol, Fortran, Algol) que teóricamente ya existían desde 1959 pero que no eran utilizados en el desarrollo real de aplicaciones.

Charles Bachman, las primeras ideas sobre bases de datos, el primer Sistema de Gerencia de Base de Datos (DBMS)

En esos momentos hubo algo muy importante que pasó desapercibido para la mayoría: En forma bastante silenciosa General Electric en EE UU y Bull – General Electric en el resto del mundo liberaron el IDS (Integrated Data Store) que fue el primer Sistema de Gerencia de Base de Datos en el mercado mundial.

El IDS seguía las ideas de Charles Bachman [1], el gran pionero de las Bases de Datos que, en aquel momento, fueron tomadas por la gran mayoría de la comunidad informática como una sofisticación exagerada y sólo destinada a unas pocas aplicaciones muy complejas.

Algunos, sin embargo, desde el principio pensamos que las bases de datos estaban destinadas a soportar **todas** las aplicaciones computacionales, mientras que la mayor parte no tomaba nuestras posiciones muy en serio.

¿Qué ha pasado en estos 40 años?, ¿Cómo ha afectado el lanzamiento del IDS al desarrollo de sistemas?

Repasemos rápidamente, de una forma libre, las ideas fundamentales de Charles Bachman:

La Base de Datos debe contener todos los datos de la empresa.

Todos los sistemas interactuarán con dicha Base de Datos.

El Sistema de Gerencia de Base de Datos debe permitir el almacenamiento, actualización, eliminación y recuperación rápida y sencilla de esos datos y, para ello, debe contener también los mecanismos de acceso y de control / aseguramiento de la integridad.

¿Cuáles eran los mecanismos de integridad previstos por Bachman? Los de "integridad de entidad" (cada entidad tendrá un identificador explícito, no habrá dos instancias de una entidad con el mismo valor del identificador) y los de "integridad referencial" (que, en la implementación del IDS podríamos caracterizar bien como "cada hijo tendrá al menos un padre", un "padre puede tener ninguno, uno o múltiples hijos" y, como corolario: los "hijos" pueden, a su vez, ser "padres".

La implementación del IDS era una red libre, básicamente soportada vía acceso randómico por llave primaria y todos los demás accesos y el control / aseguramiento de la integridad vía cadenas de pointers.

Si cotejamos (sólo desde el punto de vista cualitativo) las prestaciones liberadas por Bachman en el IDS con las de los sistemas de gerencia de base de datos actuales, podría parecer que poca cosa ha cambiado en 40 años: no es así, pero admitamos que las ideas de Charles Bachman, llevadas a la práctica por primera vez en 1963, no mediante un "paper" sino mediante algo mucho más sólido: hechos (el Sistema de Gerencia de Base de Datos IDS en el mercado), siguen teniendo plena vigencia.

La respuesta del mercado al IDS: múltiples sistemas de gerencia de base de datos

En los siguientes años, apareció un conjunto de Sistemas de Gerencia de Base de Datos más o menos formales. Los desarrollos conocidos más importantes ocurrieron dentro de IBM y fueron dos bien diferentes: el BOMP y el IMS.

El BOMP (Bill Of Materials Processor) era un sistema de propósito específico basado en una red de mucho menor poder expresivo que la de Bachman: red de 2 niveles (podemos caracterizarla como una red libre a la que le agregamos la siguiente restricción: los "hijos" no son "padres").

El IMS (Integrated Management System), que tuvo su origen en un gran proyecto de la carrera espacial en el que se vio involucrada IBM, y cuya estructura era un bosque de árboles (podemos caracterizarla como: todo "hijo" tiene un solo "padre").

IBM acabó decidiéndose por el IMS. El equipo del BOMP se sintió muy frustrado lo que acabó, a comienzos de la segunda mitad de los 60, inspirando la creación de la Cincom Systems, una de las primeras empresas de software independientes, que lanzó su Sistema de Gerencia de Base de Datos: TOTAL, retomando las ideas básicas del BOMP.

El TOTAL se implementa inicialmente para mainframes IBM pero, luego, recibe implementaciones simples y sólidas para otras plataformas (incluso para el pequeño mini computador IBM /3, primero en que yo lo utilicé en 1976). Rápidamente TOTAL se transforma en el líder en cantidad de instalaciones.

Poder expresivo de los primeros sistemas de gerencia de base de datos y su facilidad/dificultad de reorganización

A esta altura existía un sistema con gran poder expresivo (IDS) pero cuya implementación muy trabada presentaba un gran problema ¿cómo reorganizar una red, soportada básicamente por la vía de cadenas de pointers? Nunca se encontró una buena solución para esta necesidad (sea cual sea el procedimiento que se emplee, podemos concluir teóricamente que los tiempos serán enormes).

Como consecuencia aparecieron las primeras tendencias hacia la búsqueda de las "bases de datos estables": postular que existe para una determinada empresa, u organización, una "base de datos estable" que satisface todos sus requerimientos, tomar todo el tiempo necesario en los estudios previos de manera de asegurarse que "nunca habrá que reorganizar estructuralmente la base de datos" y, además, dejar campos libres en todos los registros para poder agregarles campos sin tener que recurrir a la tan temida reorganización.

40 años después es claro que no existen las tales "bases de datos estables" (por lo menos en empresas que no estén muertas). ¿Es, sin embargo, ésta una verdad generalmente admitida?: lo dudo, mucha gente sigue pensando y trabajando con los mismos conceptos de hace 40 años. Aún hoy, increíblemente, la mayor parte de las metodologías de desarrollo de sistemas se basan en "bases de datos estables".

¿Qué ocurría con el IMS? Los árboles son mucho más fáciles de reorganizar que las redes libres pero su poder expresivo es mucho más pequeño (se parte de una afirmación: "la realidad es jerárquica", pero esa afirmación es falsa, la afirmación correcta sería "las visiones de la realidad que los humanos podemos manejar con comodidad son jerárquicas"). La falta de poder expresivo de la base de datos recarga los

programas y deja en la mano del programador buena parte de la navegación en la base de datos y del control / aseguramiento de la integridad, lo que es costoso y peligroso. IBM trata de corregir el problema sustituyendo el bosque de árboles independientes original por una extraña red de árboles que se apuntan entre sí a todos los niveles, de forma bastante caótica.

El IMS, a pesar de un cierto éxito comercial alcanzado, y aunque siempre fue utilizado en bases de datos de gran tamaño (medido en cantidad de registros), nunca fue una solución para verdaderas bases de datos corporativas.

La elección de un esquema jerárquico para el IMS fue un error. El hecho de haberlo cometido IBM le dio gran trascendencia y fue un gran freno para al desarrollo de los sistemas de gerencia de base de datos durante muchos años..

¿Y el TOTAL? El TOTAL estaba entre los dos: sus redes de 2 niveles tenían mucho menor poder expresivo que las redes libres del IDS pero mucho mayor que los árboles del IMS. Al mismo tiempo, su reorganización era más complicada y demandaba más tiempo que la del IMS, pero muchísimo menos tiempo que el necesario para reorganizar las redes libres del IDS. Por otra parte, su implementación simple y sin pretensiones viabilizó su disponibilidad para múltiples computadoras y sistemas operativos. Como consecuencia, el TOTAL pasó a dominar el mercado de las empresas medias y lo viabilizó para muchas pequeñas.

La respuesta a la complejidad estructural: sistemas basados en índices

Las dificultades de reorganización y la búsqueda de mayor flexibilidad en la recuperación de datos, dieron lugar a sistemas de gerencia de base de datos con muy poco poder expresivo (en particular sin ninguna capacidad de controlar / asegurar la integridad referencial) pero muy fáciles de reorganizar y con razonable flexibilidad para la recuperación de datos y, paulatinamente, mucho más eficientes en el acceso: los sistemas basados en archivos con índices múltiples (fundamentalmente el Datacom de la Applied Data Research y el Adabas de la Software AG y, en cierto sentido, el VSAM de IBM – esencialmente un sistema de administración de archivos).

Este esquema implica atribuirle al programador un conjunto de funciones para el aseguramiento de la integridad, lo que es ineficiente en costos (dinero, tiempo) y peligroso: ¿qué ocurre cuando el programador en un programa cualquiera, olvida o interpreta mal una regla?, ¿quién sabe cuales son las reglas que realmente regulan nuestra base de datos y nuestros sistemas?

La justicia de los EE UU y la industria del software

A fines de la década del 60, decisiones de la justicia de los EE UU que obligan a IBM a cotizar y vender separadamente Software y Hardware, constituyen un gran impulso para la industria independiente de software.

Los sistemas de gerencia de base de datos son el primer territorio de la lucha entre múltiples empresas, generalmente independientes de los fabricantes, y cada una de ellas agrega su propia casuística.

La industria se desarrolla (anárquicamente) y la complejidad de los sistemas de gerencia de base de datos crece cada día.

La búsqueda de la simplificación y la "usabilidad", el sueño de llevar el poder al usuario final: Codd y el modelo relacional

Pero, al mismo tiempo, dentro de IBM aparece una tendencia simplificadora: Edgar F. Codd [2] parece preguntarse cosas del tipo "¿la realidad es tan complicada o es que los humanos nos complicamos inútilmente para representarla?".

Codd quiere tornar disponibles las bases de datos – en todos los aspectos - para todo el mundo, quitándolas del ámbito de los súper especialistas. Trabaja con la intención de simplificar el problema del diseño y uso de las bases de datos y, entonces, introduce su modelo relacional sobre la base de: representación simple de los datos (tablas con columnas formadas por elementos uniformes y atómicos), criterios para detectar (y eliminar o - en su defecto – controlar) la redundancia (normalización), reglas y operadores para manipular automáticamente los datos (álgebra relacional, cálculo relacional).

IBM plasma una parte de las ideas de Codd en la especificación del lenguaje SQL (Structured Query Language) – especificación que publica y libera al uso público - y que luego se constituiría en el estándar mundial (y, con modificaciones menores, lo es hasta hoy).

La contribución de Codd es enorme. El Laboratorio Santa Teresa de IBM publica y/o envía a todos los investigadores interesados en el tema, generosa y desinteresadamente, múltiples papers sobre los que pasamos a basarnos todos los que, de alguna manera, hemos trabajado en el desarrollo de sistemas de gerencia de bases de datos relacionales.

Las bases de datos relacionales

La comunidad informática toma la idea de las bases de datos relacionales con entusiasmo, snobismo y sin el menor pragmatismo.

Todo el mundo se pronuncia a favor de ellas, todo el mundo implementa algún tipo de software siguiendo el modelo relacional.

Nadie se preocupa por la eficiencia y, en particular, nadie asume que los mecanismos de acceso (en general) y los índices (en particular) son esenciales para la eficiencia de las bases de datos relacionales (lo eran en 1970, lo son hoy y lo seguirán siendo en un futuro previsible). Nadie afirma que pueden establecerse procedimientos determinísticos para diseñarlos y construirlos óptimamente y la mayoría duda de que puedan y deban utilizarse en forma totalmente transparente al programador y a los programas (llevó tiempo, pero los optimizadores lo hacen automáticamente desde hace años y cada vez mejor).

El nivel del SQL es demasiado bajo: el hecho de que el usuario deba saber de que tablas tomar los datos y como ligar esas tablas no es razonable y constituye un problema importante hasta hoy.

Veámoslo con un ejemplo. Supongamos que tenemos una base de datos con las siguientes tablas:

Clientes (Cliente, Nombre, Direccion)
Productos (Producto, Descripcion, Precio, Stock)
Facturas (Factura, Fecha, Cliente)
LineasFacturas (Factura, Producto, Cantidad)

Y se quiere obtener el siguiente tabulado:

Cliente	Nombre	Factura	Fecha	Producto	Descripcion	Cantidad

El comando SQL que debemos construir para lograrlo es:

Select Clientes.Cliente, Nombre, Facturas.Factura, Fecha, Productos.Producto, Descripción, Cantidad
Where Clientes.Cliente = Facturas.Cliente and
Facturas. Producto = Productos.Producto

Sin embargo, tan sólo agregando al SQL el soporte de una buena nomenclatura de nombres (por ejemplo la URA: Universal RelationalAssumption) y dotándolo de una mínima inteligencia, ese comando podría sustituirse por el siguiente:

Select Cliente, Nombre, Factura, Fecha, Producto, Descripción, Cantidad

Se obtendría el mismo resultado escribiendo menos. Si, desde luego, pero la diferencia principal no está en escribir más o menos: el primero es un claro caso de comando a ser escrito por un informático, que sabe en que tablas están los diferentes atributos y como ligar válidamente esas tablas

El segundo es un comando que puede ser escrito por un usuario: simplemente especifica cuales son las columnas que quiere en su listado en el orden que las quiere: dice lo que necesita y sabe, no necesita recurrir a nadie para que lo ayude.

Parece claro que la segunda sintaxis es mejor porque permite que lo usen naturalmente muchos más usuarios.

No acaban aquí, sin embargo, las diferencias: ¡casi siempre una especificación de alto nivel tiene ventajas sobre una de bajo nivel, más allá de lo que se escribe!

La gran diferencia es que si alguna de las columnas referidas en el select cambia de tabla, la primera sentencia se torna incorrecta, mientras que la segunda permanece válida.

Nadie se preocupa de la integridad referencial: Codd no la definió explícitamente y "todo lo que Codd no dijo" pasa a ser exageradamente importante.

Realmente el SQL debería resolver estos dos últimos problemas: nada se ha hecho con el primero y tan sólo hace pocos años se dio soporte a la integridad referencial.

La ineficiencia de los prototipos de sistemas de gerencia de bases de datos relacionales. Siguen las implementaciones casuísticas

Ante la ineficiencia que mostraban los prototipos del SQL, se dicen cosas del tipo "cuando existan las memorias de burbujas magnéticas, el problema estará resuelto" (realmente muchos años después del probado fracaso de las memorias de burbujas magnéticas, el argumento seguía siendo utilizado) o sea: se ignora deliberadamente el problema dejándolo para después sin tener la menor idea de cómo resolverlo y sin realizar esfuerzos serios para hacerlo.

Estamos en la segunda mitad de la década de los 70. Como he dicho, todo el mundo se pronuncia a favor del modelo relacional (como algo que "queda bien") pero, paralelamente, se implementan los sistemas de gerencia de base de datos más casuísticos.

Ha terminado la guerra comercial entre IBM por un lado y Honeywell, General Electric, RCA, Univac, etc., por el otro. IBM ha ganado ampliamente.

Buena parte de las aplicaciones más sofisticadas estaban soportadas por el IDS y sus usuarios comienzan a temer por el desarrollo y el soporte futuro de los computadores General Electric y desean una versión del IDS para plataforma IBM, lo que da como resultado el surgimiento del IDMS, producto isomorfo con el IDS, implementado por empresa independiente para mainframe IBM.

Una reflexión importante al pasar: reorganización de los datos / procesos

Obsérvese que, hasta ahora, me he referido con especial énfasis a los problemas de reorganización de las bases de datos y no he hablado de los problemas de inadecuación de los programas existentes cuando ocurren modificaciones estructurales en dichas bases de datos. En realidad aquellos eran tan graves que no dejaban ver la enorme importancia de éstos.

¿Desarrollo orientado a datos o desarrollo orientado a procesos?

Paralelamente al desarrollo de las bases de datos y, en especial, de los sistemas de gerencia de base de datos, se trabaja mucho en la construcción de metodologías de desarrollo de sistemas. Aparece una dicotomía: "**desarrollo orientado a datos**" o "**desarrollo orientado a procesos**".

Recordemos que las bases de datos no eran relevantes más que para los enormes usuarios: las empresas de tamaños normales no las utilizaban y los informáticos, en general, pensaban que se trataba de una sofisticación inútil y una pérdida de tiempo impulsada por "teóricos que nunca habían visto una aplicación".

Quizás esta situación haya hecho que la balanza entre orientación a datos y orientación a procesos se inclinara decididamente hacia el desarrollo orientado a procesos. Puede argumentarse con razón a favor de la orientación a procesos que es más general: con ella toda casuística puede contemplarse, todo puede hacerse. Al mismo tiempo, su nivel es mucho más bajo y los costos de desarrollo y mantenimiento son mucho mayores.

¿Quiénes fueron los líderes del desarrollo orientado a procesos? Djikstra que introdujo la programación estructurada y Edward Yourdon, Larry Constantine, Tom De Marco, Chris Gane, Trish Sarson y otros, que introdujeron el proyecto estructurado y lo popularizaron en todo el mundo [3].

¿Qué podemos decir del desarrollo orientado a datos? Los líderes de esta tendencia han sido Jean-Dominique Warnier, Ken Orr y Michael Jackson [4].

A diferencia del desarrollo orientado a procesos, que es casuístico y trata de soportar por medio de la programación manual todas las particularidades de cada aplicación, el desarrollo orientado a datos parte de otras premisas:

Los datos y programas siempre tienen estructuras – muchas veces conocidas a priori - como, por ejemplo: los datos que ve el usuario; los datos que necesita un programa para procesar una determinada operación; los datos almacenados; un programa.

¿No podríamos pensar en reglas y operadores para trabajar con esas estructuras? Sí, podríamos y ello simplificaría mucho el desarrollo y mantenimiento de sistemas, y mejoraría su calidad: Warnier, Orr y Jackson lo hicieron.

Algunas reflexiones personales sobre el desarrollo orientado a datos.

He trabajado mucho - a fines de la década de los 60 y en la de los 70 - con los métodos de Warnier-Orr: seguíamos programando a mano, pero había criterios claros para identificar las estructuras de datos y visiones de usuarios y derivar de ellas las estructuras de los programas y establecer los comandos de los mismos.

De todas maneras mi experiencia básica con este abordaje es de una época en que las aplicaciones eran fundamentalmente "batch" y utilizaban archivos convencionales. Si en vez de archivos se hubieran usado bases de datos relacionales creo que este abordaje se hubiera impuesto claramente. Pero, ¡no se puede re escribir la historia!

Debo confesar que no tuve la suficiente fe o dedicación al uso de este abordaje - que consideraba de mucho mayor nivel - cuando, de pronto, las aplicaciones de mis clientes pasaron a ser fundamentalmente interactivas y soportadas por bases de datos, lo que lamento: en ese momento opté – erróneamente – por el desarrollo orientado a procesos.

Al mismo tiempo me pregunto ¿por qué Warnier, Orr o Jackson – que parecían tener todo mucho más claro que los demás - no dieron ciertos pasos adicionales para establecer algunas cosas mucho más avanzadas (que hoy me parecen obvias, pero que claramente no lo son – o, por lo menos, no lo eran en aquel momento)?: diseño de la base de datos partiendo de las visiones de los usuarios, generación automática de los programas, etc. Realmente, sus metodologías, "informalmente" nos llevaban a hacer todo esto de una manera manual pero sistemática, natural y simple.

Dejo especial constancia de todo esto no como queja por lo que no hicieron, sino como agradecimiento por la invalorable introducción de una suerte de representación rigurosa de las estructuras de datos y, en particular, de las visiones de datos de usuarios y programas, elementos esenciales para cualquier esquema orientado a datos (o, a un nivel mayor, basado en conocimiento) y, en particular, para Genexus [5].

En la década de los 70 la disputa la ganaron los que enarbolaron la bandera de la orientación a procesos que era la bandera de la casuística, de la orientación a la programación manual y no sistemática y configuraba la prescindencia de todo tipo de operador de alto nivel.

Pero, volviendo a las bases de datos: en 1979 ¡ORACLE!

Un buen día de 1979 una pequeña empresa cuyo nombre original no recuerdo y que después tomó el de su producto, modificó el mundo: lanzó un Sistema de Gerencia de Base de Datos relacional basado en el lenguaje SQL que funcionaba eficientemente en una computadora pequeña. ¡El producto se llamaba Oracle! De repente las bases de datos relacionales de las que todos hablábamos se habían convertido en realidad: ¡Nada volvería a ser igual (afortunadamente)!

El lanzamiento de Oracle tuvo enorme repercusión: todos los demás fabricantes debieron bajar a la tierra y asumir que con la teoría relacional **y los medios disponibles** se podían hacer cosas muy importantes y pasaron a tratar de hacerlas.

Durante la década de los 80, luego que todos los fabricantes reivindicaran que sus productos (tal como estaban) "eran relacionales", se trabajó mucho en la implementación de sistemas de gerencia de bases de datos **realmente** relacionales. Los usuarios potenciales, sin embargo, no tomaban conocimiento y el uso real de sistemas de gerencia de base de datos era muy limitado.

Cuando en 1989 lanzamos Genexus, el llevar todo el desarrollo de nuestros clientes a bases de datos relacionales pareció a muchos una sofisticación exagerada que los obligaría a una pérdida de eficiencia en tiempo de ejecución. Algunos nos decían "nos gusta Genexus, pero no aceptamos que se nos obligue a utilizar una base de datos".

Esta situación pasa a modificarse decididamente en el comienzo de la década de los 90: en un cierto momento, **todos** los sistemas comienzan a desarrollarse sobre bases de datos relacionales.

Supra

En la segunda mitad de la década de los 80 Cincom Sistems lanza un Sistema de Gerencia de Base de Datos revolucionario: **SUPRA.**

Supra presenta niveles de "independencia de datos" mucho mayores que el SQL: permite visiones multi-tabla actualizables lo que determina una buena independencia entre los programas y las bases de datos, con una potencial disminución muy importante de los costos de mantenimiento de los sistemas.

Sin embargo, el mercado no lo adopta. ¿Por qué?

Es difícil saberlo: quizás haya sido porque se apartó del lenguaje SQL (Supra no fue introducido como un súper conjunto del SQL, sino que tuvo una sintaxis totalmente diferente), o quizás haya sido porque Cincom Systems permaneció demasiado años con el TOTAL y ello desanimó a sus clientes.

¿Qué ocurrió desde 1990?

¿Qué ocurrió desde 1990?, ¿qué impacto tuvieron (y tendrán) estos acontecimientos en el mercado, en las tendencias generales y en la posible dilucidación de la vieja polémica entre orientación a procesos y orientación a datos?

En el área de bases de datos los cambios fueron muy importantes pero relativamente poco visibles mientras que en el área de procesos se introdujo la orientación a objetos que ha obtenido una muy buena acogida en todo el mundo. Sin embargo, quizás lo más importante en este tiempo han sido otras cosas: el surgimiento de las "plataformas de ejecución" y el del XML o, más propiamente, de una nueva orientación: "la orientación a mensajes".

Sistemas de gerencia de base de datos

Los sistemas de gerencia de base de datos han evolucionado mucho. ¿En qué aspectos? Fundamentalmente en lo que está fuera de la vista del usuario. Hoy son mucho más sólidos ¡son realmente sólidos! Su disponibilidad, seguridad, eficiencia y escalabilidad han progresado mucho y funcionan muy bien sobre las más variadas combinaciones de software y hardware, generalmente de precios mucho menores que los tradicionales.

En lo relativo a la funcionalidad existe una nueva característica estándar muy importante: la definición y control automático, a nivel del Sistema de Gerencia de Base de Datos, de la integridad referencial. Y poco o nada más realmente importante.

Todos los fabricantes soportan alguna suerte de procedimientos almacenados pero en forma incompatible entre ellos.

Como consecuencia de lo anterior, ha disminuido mucho la cantidad de casas de software dedicadas a los sistemas de gerencia de base de datos: sólo enormes empresas, con inversiones muy grandes, pueden

brindarnos productos suficientemente sólidos, eficientes y escalables. Los principales fabricantes y productos hoy son IBM (DB2 e Informix), Microsoft (SQL Server) y Oracle.

La eliminación de las pequeñas empresas fabricantes de sistemas de gerencia de base de datos, junto con la existencia de un robusto estándar del SQL hacen que la innovación sea cada vez menor.

Los líderes, a veces, en vez de profundizar el liderazgo en su mercado vertical por la vía de la innovación, tratan de utilizar el poder que les da esa posición para hacerse fuertes en otros mercados laterales como plataformas de ejecución, servidores de aplicación y, aún, paquetes de aplicación o servicios de desarrollo y outsourcing. Estas políticas, muchas veces, los aíslan de las casas de desarrollo de software (sus naturales socios de negocios) y acaban actuando en perjuicio del desarrollo de nuevas funcionalidades en los sistemas de gerencia de base de datos. Todo se vuelve más conservador: gran solidez y poca innovación.

El SQL no ha resuelto dos temas importantes:

Un buen nivel de "independencia de datos" que permita que los programas sean inmunes a las modificaciones estructurales en la base de datos.

Algún tipo de "inteligencia" que permita definir de manera declarativa reglas y operadores de modo de independizar el desarrollo de los programadores: de permitir que un usuario pueda hacer todo aquello que quiera y esté autorizado a hacer, sin necesidad de programar.

A esta altura se plantea una pregunta obvia: ¿se modificará sustancialmente y en tiempo razonable el actual estándar SQL de manera de dotarlo de nuevos operadores de alto nivel y otras características que viabilicen un comportamiento "inteligente" de las bases de datos?

La tendencia general, hoy, es la de soportar, para la escritura de los procedimientos almacenados, lenguajes de programación comunes (Java, C#, etc.). O sea que es posible que en un futuro cercano podamos asociar mucha más lógica transportable a la base de datos, pero escribiéndola en lenguajes algorítmicos, lo que implica programación manual y dependencia de la estructura de dicha base de datos.

Al mismo tiempo, aparecen otros problemas: se han resuelto bien los tradicionales de seguridad por la vía de una buena identificación de los usuarios y especificación de lo que estos están autorizados a hacer con los datos, pero nuestros sistemas de gerencia de base de datos no son, por ejemplo, inmunes a los virus. Ésta es un área en que los grandes fabricantes deberán trabajar mucho.

Por otra parte, la propagación del sistema operativo Linux ha traído como consecuencia el éxito de nuevos sistemas de gerencia de base de datos de muy bajo precio o gratuitos y sin las pretensiones de solidez y escalabilidad de los grandes líderes. Entre estos sistemas cabe destacar a Postgres y MySQL.

Pero hoy no se discute más el uso o no de bases de datos: todas las aplicaciones, en todo el mundo, se desarrollan sobre bases de datos, la eficiencia y escalabilidad son incomparablemente mayores de las que se obtenían con archivos convencionales y existen líderes claros que dominan el mercado.

Las formas de utilización han ido variando con el tiempo, un poco por las tecnologías disponibles, un poco por las necesidades de los usuarios y un mucho por los gustos y tendencias de los profesionales de la informática, desde la arquitectura centralizada original pasando por una arquitectura Cliente / Servidor y siguiendo, ahora, por arquitecturas multiservidor orientados a la red y los Web Services bien plasmados en las plataformas de ejecución que dominarán el mundo de la informática en un futuro previsible: Java y .net:

Al final de la primera mitad de la década de los 80 y cuando todas las aplicaciones funcionaban en arquitectura centralizada, apareció un nuevo concepto: la "arquitectura cliente / servidor".

Se trataba de racionalizar y dividir el procesamiento de datos entre el servidor central y los microcomputadores (clientes) y representaba adelantos importantes, más allá de los errores que todos cometimos al principio en su implementación.

La firma que introdujo y desarrolló el concepto fue Sybase. Pero el mercado demoró mucho: sólo en 1995 adoptó esta arquitectura y ya Sybase no estaba en su mejor momento, como para luchar con grandes adversarios.

En 1996 Sun con el apoyo de múltiples empresas de software, entre las que se destacan IBM y Oracle, lanza el lenguaje Java y la plataforma de desarrollo y ejecución Java.

En 2001 Microsoft lanza el lenguaje C# y la plataforma de desarrollo y ejecución .net, sofisticada implementación de las mismas ideas de Java con propósitos y resultados cualitativos similares.

En 2001 IBM compra Informix lo que, sumado a su liderazgo en sus sistemas OS/390 y OS/400, la transforma en el mayor fabricante de Sistemas de Gerencia de Base de Datos.

En 2002 Microsoft lanza, para el .net, un nuevo estándar para la comunicación entre las aplicaciones y la base de datos (DataSets y DataAdapters) que facilitan la programación y ayudan mucho a la escritura de aplicaciones muy escalables.

Orientación a procesos

La orientación a procesos, que en su momento estuvo sustentada por las metodologías estructuradas ha dado paso al desarrollo orientado a objetos.

El desarrollo orientado a objetos, apoyado en la existencia de lenguajes de programación orientados a objetos, se ha impuesto claramente y ha traído enormes ventajas en la programación cuando ésta se refiere a datos en memoria. En estas condiciones, las ventajas sobre los procesos tradicionales son enormes.

Existe un eslabón aún débil en la orientación a objetos: ¿Cómo comunicarse bien con la base de datos? Durante cierto tiempo se experimentó bastante con los llamados sistemas de gerencia de base de datos orientados a objetos (OODBMS), pero no se obtuvieron resultados adecuados.

Hoy es claro para todos que los sistemas de gerencia de base de datos relacionales dominan el mundo de los datos y lo seguirán haciendo en un futuro previsible y se trabaja mucho en la creación de mecanismos de convivencia entre programas orientados a objetos y bases de datos relacionales (ORM: Object Relational Mapping). Existen logros aislados pero se está muy lejos de un estándar.

Plataformas de ejecución

Hace 40 años existían múltiples sistemas operativos. Cada fabricante tenía el suyo. La contraprestación necesaria era que cada fabricante debía ser capaz de atender directamente todas las necesidades de sus clientes.

Cuando surgió la industria del software, esta situación comenzó a cambiar: el cliente llenaba sus necesidades a partir de un conjunto de proveedores. Pero un sistema operativo sólo sería atractivo para la industria independiente del software si llegara a tener muchos usuarios.

Muchos fabricantes se vieron forzados por los hechos, desde mediados de la década de los 70, a abandonar sus sistemas operativos y adoptar otros. Subsisten aquellos de fabricantes que tienen una base instalada muy grande (como IBM OS/390, IBM OS/400) y crecen fuertemente algunos relativamente independientes de los fabricantes de hardware Unix, Windows y, últimamente, Linux.

Cada sistema operativo tiene sus propias complejidades. Un paso más es pensar en plataformas de ejecución que funcionen encima de ellos y que sean mucho más amigables al usuario.

En 1996 Sun lanza su lenguaje Java (orientado a objetos) y su plataforma de desarrollo y ejecución Java. Como lenguaje, más allá de sus importantes características, es uno más, pero como plataforma de ejecución implica algo totalmente nuevo: por primera vez se ofrece un ambiente amigable para desarrollar y ejecutar aplicaciones con total independencia del hardware. La adopción del concepto fue rápida y general aunque el desarrollo de reales aplicaciones Java llevó más tiempo del previsible. Hoy es una realidad.

En 2001 Microsoft lanza su plataforma de desarrollo y ejecución .net con sus propios lenguajes orientados a objetos C# y Visual Studio .net y una pluralidad de lenguajes de terceros.

¿Por qué el cliente utilizaría alguna de estas plataformas? Por muchas razones (que, en general, exceden el propósito de este trabajo) pero, fundamentalmente, por una: nos permiten la instalación y actualización automática de las aplicaciones en las estaciones de trabajo "clientes" con una disminución muy importante de los costos operativos.

Entiendo que entre estas dos plataformas se va a distribuir el mercado en los próximos años y que los sistemas operativos se tornarán commodities. Al mismo tiempo, la adopción de estas plataformas refuerza el uso de los lenguajes orientados a objetos.

XML y la "orientación a mensajes"

En 1999 aparece otro jugador muy importante: el XML, sistema que nos permite definir y manipular mensajes autodescritos. Se inicia otra tendencia muy importante: la "orientación a mensajes".

Desde su anuncio el XML rápidamente fue adoptado por todos los fabricantes lo que lo convirtió en un estándar de hecho y, luego, de derecho y su uso se ha propagado con una velocidad mucho mayor de la tradicional en las innovaciones: ¡realmente existía una enorme necesidad insatisfecha de un sistema estándar de mensajes!

El XML tiene las utilizaciones más diversas, la mayor parte de ellas fuera de la vista del usuario, pero se transforma en actor principal, por ejemplo, en los Web Services y tendrá derivaciones muy importantes en la constitución de verdaderas bases de datos extendidas: mi base de datos, las de mis proveedores / clientes, que accedo vía XML y un conjunto de Web Services de interés general que "consumo" en la forma de mensajes XML.

Pero si profundizamos un poco veremos que el XML puede convertirse en un actor importante en la vieja puja entre orientación a procesos y orientación a datos: muchos complicados procesos, en el fondo, tienen como objetivo proveernos un dato ¿no podremos pensar que ese dato nos sea entregado por un mensaje XML (data provider) que podamos especificar en forma 100% declarativa? Mi opinión es que sí.

Una vez más, sin embargo, aparecen las "bases de datos jerárquicas": los fabricantes de sistemas de gerencia de base de datos están anunciando ampliaciones del SQL que nos permitan almacenar y manipular mensajes XML. Espero que esta característica se utilice con la debida prudencia (que quede circunscripta a casos muy específicos, como el soporte de documentos textuales).

Resumen:

En los últimos años se han solidificado los sistemas de gerencia de base de datos relacionales: hoy es impensable hablar de otro tipo de Sistema de Gerencia de Base de Datos.

Se han configurado estándares robustos que, utilizados estrictamente, permiten la portabilidad de las aplicaciones.

La existencia de esos estándares dificulta la adopción de modificaciones sustanciales a los mismos. En particular, es muy poco probable que tengamos en un futuro previsible bases de datos cualitativamente mucho más evolucionadas.

Es previsible, en cambio, que se nos siga dando cada vez más solidez: disponibilidad, seguridad, eficiencia, escalabilidad, optimización de recursos y que se neutralicen las nuevas amenazas como las representadas por virus innovadores.

Paralelamente se ha generalizado la programación orientada a objetos (en particular fuertemente impulsada por el previsible auge de las plataformas Java y .net) y es vital que se resuelva rápidamente el problema de la vinculación natural de los programas con la base de datos (Object Relational Mapping)..

La orientación a mensajes puede ser el componente que falta. **Quizás no tendremos un futuro claramente orientado a procesos ni a datos sino, más bien, una realidad que se comporte como un conjunto de mensajes.**

Los sistemas de gerencia de base de datos de hoy y las ideas de Bachman y Codd

Pero, con esto, ¿hemos llegado al final?, ¿nuestra pretensión – hoy – es la misma de Charles Bachman en 1963?.

Si es la misma, si queremos a acceder las bases de datos a partir de programas donde hacemos casi todo a mano, está todo resuelto (y bien resuelto porque tenemos muy buenas eficiencia, disponibilidad, seguridad, escalabilidad, recuperación ante accidentes, etc. y, además, medios razonables para interrogar a la base de datos). ¡El sueño de Bachman se ha realizado, pero demoró casi 40 años para hacerse realidad!

Si, en cambio, pensamos como pensaba Edgar F. Codd en 1970 y queremos llevar el uso de las bases de datos a todo el mundo, estamos muy lejos.

¡La tan mentada y pregonada "era del usuario" parece aún muy lejana!

Mi idea sobre el futuro

Creo firmemente en un paradigma diferente del vigente, del representado por el "Estado del Arte" actual.

Creo en "bases de datos inteligentes" que nos permitan alimentar en forma declarativa todo el conocimiento necesario (fundamentalmente "reglas del negocio", "reglas de precedencia y/o de flujo" y "reglas de autorización" con toda la generalidad que esos conceptos representan) de manera que cualquier usuario sin la necesidad de conocimiento técnico alguno, de una manera simple, pueda hacer en cualquier momento todo aquello que quiera y esté autorizado a hacer, sin necesidad de programar.

Se podrá decir que lo que quiero es muy difícil.

Puedo concordar con ello. Pero refinemos esta respuesta un poco más: ¿es muy difícil porque carecemos de la tecnología necesaria? No: es muy difícil porque no se ajusta al Estado del Arte actual, y por la tónica general del mercado y de la industria, que se han mostrado muy conservadores.

A veces es mucho más difícil cambiar de paradigma que desarrollar la tecnología necesaria para hacer realidad el nuevo paradigma.

Sólo alguno de los grandes jugadores del área de los Sistemas de Gerencia de Base de Datos (IBM, Microsoft, Oracle, ¿algún otro?) puede, con éxito, abandonar el estándar SQL decidirse a buscar soluciones mucho más evolucionadas. Pero es un gran riesgo para un líder, que con el éxito casi siempre tiende a volverse conservador.

¿Cuál de ellos se animará?

Si alguno de estos líderes así lo resuelve, pienso que tendremos "bases de datos inteligentes" rápidamente.

De no ser así, el actual estándar SQL tendrá una vida larga y relativamente pacífica. En ese caso el problema no se resolverá con él, sino con sistemas de más alto nivel, basados en conocimiento, que "operen" automáticamente el SQL.

Creo firmemente que es inevitable la sustitución del paradigma actual (como 40 años atrás orientado a los programadores y a la programación algorítmica y manual) por un nuevo paradigma orientado al conocimiento.

Creo que esa sustitución es inevitable porque la programación manual se muestra cada vez más inviable y hoy implica muy baja productividad del desarrollo y dramáticos costos (dinero y tiempo) del mantenimiento de sistemas y, muchas veces, obliga a las empresas a sacrificar su individualidad y renunciar a ventajas competitivas, para adoptar paquetes estándar con las consiguientes rigideces, porque les es imposible construir y mantener los que realmente necesita.

Algunos de los países más desarrollados tratan de solucionar el problema importando mano de obra, de manera de bajar sus costos.

Paralelamente, muchas empresas, para tratar de disminuir sus grandes costos, han optado por transferir su programación y, paulatinamente, su desarrollo (en algunos casos, llegando a un outsourcing total) a países donde la mano de obra es relativamente calificada y muy barata. Cada vez más países pujan por ese mercado.

Esta tendencia lleva a los usuarios finales a una paulatina perdida de libertad y constituye una fuerte amenaza para la industria de software de la mayor parte de los países más desarrollados e, incluso, para la latinoamericana.

Sin embargo, estos intentos para disminuir los costos son simples paliativos: la real solución es la inserción de tecnología tal que haga que profesionales bien pagados puedan ser realmente eficientes en términos mundiales sin necesidad de desplazarse de sus países.

¿Cuanto demoraremos en tener bases de datos inteligentes?, ¿cuanto demoraremos en llegar realmente a la tan pregonada "era del usuario"?.

No lo sé, pero **no es un problema de tecnología: la tecnología existe. Es un problema de actitud**.

Desde nuestra empresa hemos dado los primeros pasos con Genexus y hoy existen miles de grandes aplicaciones en todo el mundo, fundamentalmente sistemas de misión crítica, que son desarrollados y mantenidos automáticamente en base a conocimiento, sin escribir una sola línea de código en lenguajes de bajo nivel. Hay mucho más para hacer, pero se ha demostrado la viabilidad del nuevo paradigma. ¡No es poco!

Bibliografía

En lo relativo a la bibliografía utilizada para la elaboración de este trabajo cabe hacer las siguientes puntualizaciones:

Es complejo proporcionar una bibliografía concreta cuando no se trata de temas puntuales sino de una intensa experiencia de vida de más de 40 años de profesión orientados desde el comienzo a cosas no convencionales.

Más compleja aún se torna esta tarea cuando el autor, por su actividad profesional, ha tenido el privilegio de participar en seminarios e intercambios informales de ideas con muchas de las principales figuras que hicieron la historia de las bases de datos y de las metodologías de desarrollo de sistemas y/o, acceso a documentos no publicados de los mismos (en algunos casos años antes de que los autores los acabaran publicando).

Existe, sin embargo otro problema creciente cada día: hace 40 años había un razonable equilibrio entre la investigación realizada por los fabricantes y la realizada por las universidades de todo el mundo. Esa situación se ha modificado paulatinamente y hoy la participación relativa de las universidades en la investigación informática se ha minimizado.

Esta situación no es buena, pero constituye la realidad mundial y tiene una primera consecuencia negativa muy importante: la falta de publicaciones oportunas.

La industria se interesa por hacer cosas innovadoras, por superar a sus competidores y, pocas veces, por publicar oportunamente sus descubrimientos tecnológicos. Hoy, para mantenernos actualizados en la frontera de la tecnología, debemos recurrir a investigaciones propias y a nuestras buenas relaciones con otras empresas y con otros investigadores ya que, generalmente, cuando algo se publica ya es tarde para tomar determinaciones oportunas.

Por todo ello listar simplemente una bibliografía sería totalmente insuficiente y, además, a esta altura de los tiempos, cuando muchas de las incertidumbres que se tuvieron en su momento se han despejado, sería inútil para el lector por lo que, además de unos pocos documentos y libros que siguen teniendo interés hoy, quiero referirme, como explicación y como agradecimiento, a eventos y a personas que me han aportado sus puntos de vista.

Libros y "papers" publicados:

[1] Charles Bachman

The Integrated Data Store, a General Purpose Programming System for Random Access Memories, General Electric, 1964.

Integrated Data Store Application Manual, General Electric, 1966.

[2] Edgar F. Codd

Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks, IBM Research Report, Santa Teresa Lab, San José, California, RJ599, 1969.
A Relational Model of Data for Large Shared Data Banks, CACM 13:6 1970.

Further Normalization of the Data Base Relational Model, IBM Research Report, Santa Teresa Lab, San José, California, RJ909, 1971.

The Relational Model for Database Management, Addison-Wesley, 1990

[3] Desarrollo orientado a procesos

Edward Yourdon

A Case Study in Structured Programming – Redesign of a Payroll System, Proceedings of the IEEE Compcon conference, 1975, New York, IEEE 1975

Techniques of Program Structure and Design, Englewood Cliffs, N.J.: Prentice-Hall, 1975
Managing the Structured Techniques, New York, Yourdon Press, 1979

Edward Yourdon & Larry Constantine:

Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, New York, Yourdon Press, 1978

Tom De Marco

Structured Analysis and System Specification, New York, Yourdon Press, 1979

Chris Gane & Trish Sarson:

Structured Systems Analysis: Tool and Techniques, New York, Improved Systems Technologies, 1977

[4] Desarrollo orientado a datos

Jean-Dominique Warnier

The Logical Construction of Programs, New York, van Nostrand Reinhold, 1974

Les procedures de traitement et leurs donnes, Les Editions D´Organization, Paris, 1975

La transformation des programes, Les Editions D´Organization, Paris, 1976

Entrainement a la programmation, Les Editions D´Organization, Paris, 1979

Ken Orr

Structured Systems Development, New York, Yourdon Press, 1977

Michael Jackson

Principles of Program Design, New York, Academy Press, 1975

System Development, Prentice-Hall International, 1983

[5] Administración del conocimiento

ARTech

Genexus General View, 1989- 2003, www.genexus.com/documents/generalview.pdf

Breogán Gonda, Juan Nicolás Jodal

Desarrollo Incremental, 1989-2003, ARTech

Algunas reflexiones sobre los Modelos de Datos a los 35 años de su introducción, 1997, 1er Congreso Uruguayo de Informática, www.genexus.com/whitepapers

Genexus: Philosophy, 2002-2003, Artech, www.genexus.com/whitepapers

[6] Otros:

C.J. Date: An Introduction to Database Systems, Addisson-Wesley, 1976

David Maier: The Theory of Relational Databases,1983 Computer Science Press, Pitman, Publishing, 1983

Jeffrey D. Ullman:Principles of Database and Knowledge-Base Systems, Computer Science Press, 1988

Hermann E. Dolder: Diseño de Bases de Datos utilizando conceptos y técnicas de Inteligencia Artificial, EUDEBA, 1987

Morton M. Astrahan, Mike W. Blasgen, Donald D. Chamberlin, Kapali P. Eswaran, Jim Gray, Patricia P. Griffiths, W. Frank King III, Raymond A. Lorie, Paul R. McJones, James W. Mehl, Gianfranco R. Putzolu, Irving L. Traiger, Bradford W. Wade, Vera Watson: System R: Relational Approach to Database Management. TODS 1(2): 97-137 (1976)

Gerald Held, Michael Stonebraker, Eugene Wong: INGRES: A Relational Data Base Management System. AFIPS NCC 1975: 409-416

Bo Sundgren: Theory of Databases, Petrocelli/Charter, New York, 1975

Borge Langefords, Bo Sundgren: Information Systems Architecture, Mason/Charter, New York, 1974

Donald E. Knuth: The Art of Computer Programming, Addison-Wesley, 1968-1973

Documentos no publicados.

Varios de **Jean Dominique Warnier**

Varios de **Charles Bachman**

Varios trabajos desarrollados por el **Laboratorio Santa Teresa de IBM**

Múltiples trabajos de otros laboratorios de investigación

Múltiples manuales y trabajos no publicados de fabricantes de los diferentes sistemas de gerencia de base de datos

Cursos, seminarios y discusiones personales.

Seminario Avanzado de Bases de Datos celebrado en la Pontificia Universidad Católica de Río de Janeiro en 1976 con la participación de **Codd, Date, Blasgen, Stonebraker, Furtado**, etc.

Múltiples seminarios organizados por SCI en Brasil y, en mi carácter de Director de Tecnología de la empresa organizadora, discusiones técnicas con los expositores como, por ejemplo, **Yourdon, Gane, De Marco y Date**

Discusiones personales con mi amigo **Hermann Dolder**

Discusiones personales con mi amigo **Ken Orr**

Y, muy especialmente, discusiones personales con Nicolás Jodal y con el equipo de ARTech