# Desenvolvimento orientado a processos ou orientado a dados?

**Algumas reflexões no 40º aniversário dos Sistemas de Gerenciamento de Bancos de Dados**

Preparado por Breogán Gonda
bgv@artech.com.uy



## Advertência prévia

Não quero que o leitor se engane e se frustre com a leitura deste trabalho: nunca tive a possibilidade nem a vocação de ver a vida passar. Sempre assumi fortes compromissos profissionais e, por isso, não pretendo ser considerado objetivo.

## Bancos de Dados: um pouco de história e reflexões

O ano de 1963 foi um ano efervescente na informática. Foi o ano da chamada 3ª Geração de computadores: pela primeira vez os fabricantes começaram a pensar que empresas normais poderiam se beneficiar do uso da informática que, até então, estava restringido a enormes projetos, muitos deles de caráter estratégico e ligados à guerra fria.

Em um dia de 1963 Honeywell lançou seu H200, primeiro computador da chamada 3ª Geração. Rapidamente outras empresas responderam com outros computadores de "3ª Geração" ou anúncios dos mesmos (em alguns casos somente enunciando algumas de suas características e mostrando maquetes de seus gabinetes, porque haviam sido tomadas de surpresa e seus projetos estavam em um estado primitivo): a líder IBM com seu "/360", General Electric com seus "Compatibles 400" e "Compatibles 600", RCA com sua linha "Spectra", Univac lançando um novo hardware e seu legendário sistema operacional "Exec", etc.

O que foi mais importante na 3ª Geração? De acordo com suas repercussões, os elementos mais importantes foram a formalização, como algo independente do hardware, dos conceitos de software, em geral e do sistema operacional, em particular, e um forte impulso para as linguagens de programação padrão (Cobol, Fortran, Algol) que teoricamente já existiam desde 1959 mas que não eram utilizadas no desenvolvimento real de aplicativos.

## Charles Bachman, as primeiras idéias sobre Bancos de Dados, o primeiro Sistema de Gerenciamento de Banco de Dados (DBMS)

Nestes momentos houve algo muito importante que passou despercebido para a maioria: De forma bastante silenciosa, a General Electric nos Estados Unidos e a Bull – General Electric no resto do mundo liberaram o IDS (Integrated Data Store) que foi o primeiro Sistema de Gerenciamento de Banco de Dados do mercado mundial.

O IDS seguia as idéias de Charles Bachman [1], o grande pioneiro dos Bancos de Dados que, naquele momento, foram tomados pela grande maioria da comunidade de informática como uma sofisticação exagerada e só destinada a uns poucos aplicativos muito complexos.

Entretanto, alguns, desde o princípio, pensaram que os bancos de dados estavam destinados a suportar **todos** os aplicativos de computadores, enquanto que a maior parte não levava nossas posições muito a sério.

O que aconteceu nestes 40 anos? Como o lançamento do IDS afetou o desenvolvimento de sistemas?

Repassemos rapidamente, de uma forma livre, as idéias fundamentais de Charles Bachman:

O Banco de Dados deve conter todos os dados da empresa.

Todos os sistemas interagirão com tal Banco de Dados.

O Sistema de Gerenciamento de Banco de Dados deve permitir o armazenamento, a atualização, a eliminação e a recuperação rápida e simples destes dados e, para tanto, deve conter também os mecanismos de acesso e de controle / garantia da integridade.

Quais eram os mecanismos de integridade previstos por Bachman? Os de "integridade de entidade" (cada entidade terá um identificador explícito, não haverá duas instâncias de uma entidade com o mesmo valor do identificador) e os de "integridade referencial" (que, na implementação do IDS poderíamos caracterizar bem como "cada filho terá pelo menos um pai", um "pai pode ter nenhum, um ou vários filhos" e, como conclusão: os "filhos" podem, por sua vez, serem "pais").

A implementação do IDS era uma rede livre, basicamente suportada através de acesso randômico por chave primária e todos os demais acessos e o controle / garantia da integridade através de cadeias de *pointers*.

Se compararmos (partindo somente do ponto de vista qualitativo) as contribuições liberadas por Bachman no IDS com as dos sistemas de gerenciamento de banco de dados atuais, poderia parecer que pouca coisa mudou em 40 anos. Não é assim. Mas admitamos que as idéias de Charles Bachman, realizadas pela primeira vez em 1963, não em um "paper" mas em algo muito mais sólido: fatos (o Sistema de Gerenciamento de Banco de Dados IDS no mercado). As contribuições continuam tendo plena vigência.

**A resposta do mercado ao IDS: vários sistemas de gerenciamento de banco de dados**

Nos anos a seguir, apareceu um conjunto de Sistemas de Gerenciamento de Banco de Dados mais ou menos formais. Os desenvolvimentos conhecidos mais importantes ocorreram dentro da IBM e foram dois bem diferentes: o BOMP e o IMS.

O BOMP (Bill Of Materials Processor) era um sistema de finalidade específica baseado em uma rede de poder expressivo muito menor que a de Bachman: rede de 2 níveis (podemos caracterizá-la como uma rede livre à qual agregamos a seguinte restrição: os "filhos" não são "pais").

O IMS (Integrated Management System), que teve sua origem em um grande projeto da corrida espacial no qual a IBM se viu envolvida e cuja estrutura era um bosque de árvores (podemos caracterizá-lo como: todo "filho" tem somente um "pai").

A IBM acabou se decidindo pelo IMS. A equipe do BOMP se sentiu muito frustrada, o que acabou, no início da segunda metade dos anos 60, inspirando a criação da Cincom Systems, uma das primeiras empresas de software independentes, que lançou seu Sistema de Gerenciamento de Banco de Dados: TOTAL, retomando as idéias básicas do BOMP.

O TOTAL foi implementado inicialmente para mainframes IBM, mas logo recebeu implementações simples e sólidas para outras plataformas (inclusive para o pequeno mini computador IBM /3, o primeiro em que eu o utilizei em 1976). Rapidamente, o TOTAL se transforma no líder em quantidade de instalações.

**Poder expressivo dos primeiros sistemas de gerenciamento de Banco de Dados e sua facilidade/dificuldade de reorganização**

Nesta altura existia um sistema com grande poder expressivo (IDS) mas cuja implementação muito travada apresentava um grande problema: como reorganizar uma rede, suportada basicamente por meio de cadeias de *pointers*? Nunca foi encontrada uma boa solução para esta necessidade (seja qual for o procedimento empregado, podemos concluir teoricamente que os tempos serão enormes).

Como conseqüência apareceram as primeiras tendências para a busca de "bancos de dados estáveis": pretensão que existe para uma determinada empresa, ou organização, um "banco de dados estável" que satisfaz todas suas exigências, tomar todo o tempo necessário nos estudos prévios de maneira a assegurar-se que "nunca haverá a necessidade de reorganizar estruturalmente o banco de dados" e, além disso, deixar campos livres em todos os registros para poder agregar a eles campos sem ter que recorrer à tão temida reorganização.

Quarenta anos depois, é claro que não existem os tais "bancos de dados estáveis" (pelo menos em empresas que não estejam mortas). Contudo, esta é uma verdade geralmente admitida? Duvido. Muita gente segue pensando e trabalhando com os mesmos conceitos de 40 anos atrás. Hoje, ainda, incrivelmente, a maior parte das metodologias de desenvolvimento de sistemas se baseia em "bancos de dados estáveis".

O que ocorria com o IMS? As árvores são muito mais fáceis de serem reorganizadas do que as redes livres mas seu poder expressivo é muito menor (parte-se de uma afirmação: "a realidade é hierárquica", mas essa

afirmação é falsa, a afirmação correta seria "as visões da realidade que os humanos podem administrar com comodidade são hierárquicas"). A falta de poder expressivo do banco de dados sobrecarrega os programas e deixa na mão do programador boa parte da navegação no banco de dados e do controle / segurança da integridade, o que é caro e perigoso. A IBM trata de corrigir o problema substituindo o bosque de árvores independentes original por una estranha rede de árvores que são apontadas entre si em todos os níveis, de forma bastante caótica.

O IMS, apesar de um certo sucesso comercial alcançado e do fato de que ainda assim sempre foi utilizado em bancos de dados de tamanho grande (medido em quantidade de registros), nunca foi uma solução para verdadeiros bancos de dados corporativos.

A escolha de um esquema hierárquico para o IMS foi um erro. O fato da IBM tê-lo cometido lhe deu grande transcendência e foi um grande freio para o desenvolvimento dos sistemas de gerenciamento de banco de dados durante muitos anos.

E o TOTAL? O TOTAL estava entre os dois: suas redes de 2 níveis tinham poder expressivo muito menor que as redes livres do IDS mas muito maior que as árvores do IMS. Ao mesmo tempo, sua reorganização era mais complicada e exigia mais tempo que a do IMS, mas muitíssimo menos tempo que o necessário para reorganizar as redes livres do IDS. Por outro lado, sua implementação simples e sem pretensões viabilizou sua disponibilidade para vários computadores e sistemas operacionais. Como conseqüência, o TOTAL passou a dominar o mercado de empresas médias e o viabilizou para muitas pequenas.

**A resposta para a complexidade estrutural: sistemas baseados em índices**

As dificuldades de reorganização e a busca por maior flexibilidade na recuperação de dados, deram lugar a sistemas de gerenciamento de banco de dados com pouquíssimo poder expressivo (particularmente sem nenhuma capacidade de controlar / assegurar a integridade referencial) mas muito fáceis de reorganizar e com flexibilidade razoável para a recuperação de dados e, paulatinamente, muito mais eficientes em acesso: os sistemas baseados em arquivos com índices múltiplos (fundamentalmente o Datacom da Applied Data Research e o Adabas da Software AG e, em determinado sentido, o VSAM da IBM – essencialmente um sistema de administração de arquivos).

Este esquema implica atribuir ao programador um conjunto de funções para assegurar a integridade, o que é ineficiente em termos de custo (dinheiro, tempo) e perigoso: o que ocorre quando o programador, em um programa qualquer, esquece ou interpreta mal uma regra?, quem sabe quais são as regras que realmente regulam nosso banco de dados e nossos sistemas?

**A justiça dos Estados Unidos e a indústria de software**

No final da década de 60, decisões da justiça dos Estados Unidos que obrigaram a IBM a cotizar e a vender separadamente Software e Hardware constituíram um grande impulso para a indústria independente de software.

Os sistemas de gerenciamento de banco de dados são o primeiro território da luta entre várias empresas, geralmente independentes dos fabricantes e cada uma delas agrega sua própria casuística.

A indústria se desenvolve (anarquicamente) e a complexidade dos sistemas de gerenciamento de banco de dados cresce a cada dia.

**A busca pela simplificação e pela "usabilidade", o sonho de levar o poder ao usuário final: Codd e o modelo relacional**

Mas, ao mesmo tempo, dentro da IBM aparece uma tendência simplificadora: Edgar F. Codd [2] parece se perguntar coisas do tipo "a realidade é tão complicada ou são os seres humanos que se complicam inutilmente para representá-la?".

Codd quer tornar disponíveis os bancos de dados – em todos os aspectos - para todo o mundo, tirando-os do âmbito dos superespecialistas. Trabalha com a intenção de simplificar o problema do projeto e uso dos bancos de dados e, então, introduz seu modelo relacional sobre a base de: representação simples dos dados (tabelas com colunas formadas por elementos uniformes e atômicos), critérios para detectar (e eliminar ou -

em seu defeito – controlar) a redundância (normalização), regras e operadores para manipular automaticamente os dados (álgebra relacional, cálculo relacional).

A IBM plasma uma parte das idéias de Codd na especificação da linguagem SQL (Structured Query Language) – especificação que publica e libera para uso público - e que logo se constituiria no padrão mundial (e, com modificações menores, o é até hoje).

A contribuição de Codd é enorme. O Laboratório Santa Teresa da IBM publica e/ou envia a todos os investigadores interessados no tema, generosa e desinteressadamente, vários *papers* sobre os quais passamos a nos basear todos os que, de alguma maneira, trabalhamos no desenvolvimento de sistemas de gerenciamento de bancos de dados relacionais.

**Os bancos de dados relacionais**

A comunidade de informática toma a idéia dos bancos de dados relacionais com entusiasmo, esnobismo e sem o menor pragmatismo.

Todo o mundo se pronuncia a favor deles, todo o mundo implementa algum tipo de software seguindo o modelo relacional.

Ninguém se preocupa com a eficiência e, particularmente, ninguém assume que os mecanismos de acesso (em geral) e os índices (em particular) são essenciais para a eficiência dos bancos de dados relacionais (o eram em 1970, o são hoje e o continuarão sendo em um futuro previsível). Ninguém afirma que é possível estabelecer procedimentos determinísticos para projetá-los e os construir otimamente e a maioria duvida de que possam e devam ser utilizados de forma totalmente transparente ao programador e aos programas (levou tempo, mas os processos para otimização já são feitos automaticamente há alguns anos e cada vez melhor).

O nível do SQL é muito baixo: o fato de o usuário ter que saber de quais tabelas pegar os dados e como ligar essas tabelas não é lógico e constitui um problema importante até hoje.

Vejamos com um exemplo. Suponhamos que temos um banco de dados com as seguintes tabelas:

**Clientes (Cliente, Nome, Endereço)**
**Produtos (Produto, Descrição, Preço, Estoque)**
**Faturas (Fatura, Data, Cliente)**
**Linhas Faturas (Fatura, Produto, Quantidade)**

E que se deseje obter a seguinte tabulação:

| Cliente | Nome | Fatura | Data | Produto | Descrição | Quantidade |
|---|---|---|---|---|---|---|
| | | | | | | |

O comando SQL que devemos construir para conseguir isso é:

**Select Clientes.Cliente, Nome, Faturas.Fatura, Data, Produtos.Produto, Descrição, Quantidade**
**Where Clientes.Cliente = Faturas.Cliente and**
**Faturas. Produto = Produtos.Produto**

Entretanto, só adicionando ao SQL o suporte de uma boa nomenclatura de nomes (por exemplo a URA: Universal Relational Assumption) e o dotando de uma inteligência mínima, esse comando poderia ser substituído pelo seguinte:

**Select Cliente, Nome, Fatura, Data, Produto, Descrição, Quantidade**

Seria obtido o mesmo resultado escrevendo menos. Sim, mas a principal diferença não está em escrever mais ou menos: o primeiro é um caso claro de comando a ser escrito por um profissional de informática, que sabe em quais tabelas estão os diferentes atributos e como ligar validamente essas tabelas.

O segundo é um comando que pode ser escrito por um usuário: simplesmente especifica quais são as colunas que quer em sua lista na ordem em que as quer: diz o que necessita e sabe, não precisa recorrer a ninguém para que o ajude.

Parece claro que a segunda sintaxe é melhor porque permite que seja usado naturalmente por muito mais usuários.

Entretanto, as diferenças não acabam aqui: quase sempre uma especificação de alto nível tem vantagens sobre uma de baixo nível, muito mais do que se escreve!

A grande diferença é que se alguma das colunas referidas no *select* muda de tabela, a primeira sentença se torna incorreta, enquanto que a segunda permanece válida.

Ninguém se preocupa com a integridade referencial: Codd não a definiu explicitamente e "tudo o que Codd não disse" passa a ser exageradamente importante.

Realmente o SQL deveria resolver estes dois últimos problemas: nada se fez com o primeiro e faz somente poucos anos que foi dado suporte à integridade referencial.

**A ineficiência dos protótipos de sistemas de gerenciamento de bancos de dados relacionais. Continuam as implementações casuísticas**

Diante da ineficiência que os protótipos do SQL mostravam, diz-se coisas do tipo "quando existir memórias de bolhas magnéticas, o problema estará resolvido" (realmente muitos anos depois do comprovado fracasso das memórias de bolhas magnéticas, o argumento seguia sendo utilizado) ou seja: ignora-se deliberadamente o problema deixando-o para depois sem ter a menor idéia de como o resolver e sem realizar esforços sérios para o fazer.

Estamos na segunda metade da década de 70. Como disse, todo o mundo se pronuncia a favor do modelo relacional (como algo que "cai bem") mas, paralelamente, os sistemas de gerenciamento de banco de dados mais casuísticos são implementados.

Terminou a guerra comercial entre IBM, de um lado, e Honeywell, General Electric, RCA, Univac, etc., de outro. A IBM ganhou com grande vantagem.

Boa parte dos aplicativos mais sofisticados estava suportada pelo IDS e seus usuários começaram a temer pelo desenvolvimento e suporte futuro dos computadores General Electric e desejam uma versão do IDS para plataforma IBM, o que oferece como resultado o surgimento do IDMS, produto isomorfo com o IDS, implementado por empresa independente para mainframe IBM.

**Una reflexão importante, diga-se de passagem: reorganização dos dados / processos**

*Observe-se que, até agora, me referi com especial ênfase aos problemas de reorganização dos bancos de dados e não falei dos problemas de inadequação dos programas existentes quando ocorrem modificações estruturais em tais bancos de dados. Na verdade, aqueles eram tão graves que não deixavam ver a enorme importância destes.*

**Desenvolvimento orientado a dados ou desenvolvimento orientado a processos?**

Paralelamente ao desenvolvimento dos bancos de dados e, em especial, dos sistemas de gerenciamento de banco de dados, trabalha-se muito na construção de metodologias de desenvolvimento de sistemas. Aparece uma dicotomia: "**desenvolvimento orientado a dados**" ou "**desenvolvimento orientado a processos**".

Lembremos que os bancos de dados só eram relevantes para los enormes usuários: as empresas de tamanho normal não os utilizavam e os estudiosos de informática, em general, pensavam que se tratava de uma sofisticação inútil e uma perda de tempo impulsionada por "teóricos que nunca haviam visto um aplicativo".

Talvez esta situação tenha feito a balança entre orientação a dados e orientação a processos se inclinar decididamente para o desenvolvimento orientado a processos. Pode-se argumentar com razão a favor da orientação a processos que é mais geral: com ela toda casuística pode ser contemplada, tudo pode ser feito.

Ao mesmo tempo, seu nível é muito mais baixo e os custos de desenvolvimento e manutenção são muito maiores.

Quais foram os líderes do desenvolvimento orientado a processos? Djikstra que introduziu a programação estruturada e Edward Yourdon, Larry Constantine, Tom De Marco, Chris Gane, Trish Sarson e outros, que introduziram o projeto estruturado e o popularizaram em todo o mundo [3].

O que podemos dizer sobre o desenvolvimento orientado a dados? Os líderes desta tendência foram Jean-Dominique Warnier, Ken Orr e Michael Jackson [4].

A diferença do desenvolvimento orientado a processos, que é casuístico e suporta por meio da programação manual todas as particularidades de cada aplicativo, o desenvolvimento orientado a dados parte de outras premissas:

Os dados e programas sempre têm estruturas – muitas vezes conhecidas *a priori* – como por exemplo: os dados que o usuário vê; os dados que precisa um programa para processar uma determinada operação; os dados armazenados; um programa.

**Não poderíamos pensar em regras e operadores para trabalhar com essas estruturas? Sim, poderíamos e isso simplificaria muito o desenvolvimento e a manutenção de sistemas e melhoraria sua qualidade: Warnier, Orr e Jackson o fizeram.**

**Algumas reflexões pessoais sobre o desenvolvimento orientado a dados**

Trabalhei muito – no final da década de 60 e na de 70 – com os métodos de Warnier-Orr: continuávamos programando à mão, mas havia critérios claros para identificar as estruturas de dados e as visões de usuários e derivar delas as estruturas dos programas e os comandos dos mesmos.

De todas as maneiras, minha experiência básica com esta abordagem é de uma época em que os aplicativos eram fundamentalmente "batch" e utilizavam arquivos convencionais. Se em vez de arquivos tivessem sido usados Bancos de Dados relacionais creio que esta abordagem teria sido claramente imposta. Mas não se pode reescrever a história!

Devo confessar que não tive a fé ou a dedicação suficiente com relação ao uso desta abordagem - que considerava ser de nível muito maior - quando, derrepente, os aplicativos de meus clientes passaram a ser fundamentalmente interativos e suportados por bancos de dados, o que lamento: nesse momento optei – erroneamente – pelo desenvolvimento orientado a processos.

Ao mesmo tempo me pergunto: por que Warnier, Orr ou Jackson – que pareciam ter tudo muito mais claro que os demais - não deram certos passos adicionais para estabelecer algumas coisas muito mais avançadas (que hoje me parecem óbvias, mas que claramente não são – ou, pelo menos, não o eram naquele momento)?: projeto de banco de dados partindo das visões dos usuários, geração automática dos programas, etc. Realmente, suas metodologias, "informalmente", nos levavam a fazer tudo isto de uma maneira manual mas sistemática, natural e simples.

De tudo isto, quero que conste especialmete, que considero não como queixa pelo que não fizeram, mas sim como agradecimento pela inestimável introdução de um tipo de representação rigorosa das estruturas de dados e, em particular, das visões de dados de usuários e programas, elementos essenciais para qualquer esquema orientado a dados (ou, em um nível maior, baseado em conhecimento) e, em particular, para o Genexus [5].

Na década de 70 a disputa foi ganha pelos que hastearam a bandeira da orientação a processos que era a bandeira da casuística, da orientação à programação manual e não sistemática e configurava a renúncia a todo tipo de operador de alto nível.

**Mas, voltando aos bancos de dados: em 1979: ORACLE!**

Em um dia de 1979, uma pequena empresa, cujo nome original não lembro e que depois passou a usar o de seu produto, modificou o mundo: lançou um Sistema de Gerenciamento de Banco de Dados relacional baseado na linguagem SQL que funcionava eficientemente em um computador pequeno. O produto se chamava Oracle! De repente os bancos de dados relacionais que todos falávamos se converteram em realidade: Nada voltaria a ser igual (felizmente)!

O lançamento do Oracle teve enorme repercussão: todos os demais fabricantes tiveram que descer à terra e assumir que com a teoria relacional **e os meios disponíveis** era possível fazer coisas muito importantes e passaram a tratar de fazê-las.

Durante a década de 80, logo que todos os fabricantes reivindicaram que seus produtos (tal como estavam) "eram relacionais", se trabalhou muito na implementação de sistemas de gerenciamento de bancos de dados **realmente** relacionais. Os usuários potenciais, entretanto, não tomavam conhecimento e o uso real de sistemas de gerenciamento de banco de dados era muito limitado.

Quando, em 1989, lançamos o Genexus, o fato de levar todo o desenvolvimento de nossos clientes para bancos de dados relacionais pareceu a muitos uma sofisticação exagerada que os obrigaria a uma perda de eficiência em tempo de execução. Alguns nos diziam "gostamos do Genexus, mas não aceitamos que sejamos obrigados a utilizar um banco de dados".

Esta situação passa a ser modificada decididamente no começo da década de 90: em um determinado momento, **todos** os sistemas começaram a ser desenvolvidos sobre bancos de dados relacionais.

### Supra

Na segunda metade da década de 80, a Cincom Sistems lança um Sistema de Gerenciamento de Banco de Datos revolucionário: **SUPRA.**

O **Supra** apresenta níveis de "independência de dados" muitos maiores que o SQL: permite visões multi-tabela atualizáveis, o que determina uma boa independência entre os programas e os bancos de dados, com uma potencial diminuição muito importante dos custos de manutenção dos sistemas.

Entretanto, o mercado não o adota. Por quê?

É difícil saber: talvez tenha sido porque se distanciou da linguagem SQL (o Supra não foi introduzido como um super conjunto do SQL, ao contrário, teve uma sintaxe totalmente diferente) ou talvez tenha sido porque a Cincom Systems permaneceu muitos anos com o TOTAL e isto desanimou seus clientes.

### O que ocorreu desde 1990?

O que ocorreu desde 1990? Que impacto tiveram (e terão) estes acontecimentos no mercado, nas tendências gerais e na possível elucidação da velha polêmica entre orientação a processos e orientação a dados?

Na área de bancos de dados as mudanças foram muito importantes mas relativamente pouco visíveis enquanto que na área de processos se introduziu a orientação a objetos que obteve uma recepção muito boa em todo o mundo . Entretanto, talvez o mais importante neste momento foram outras coisas: o surgimento das "plataformas de execução" e o do XML ou, mais propriamente, de uma nova orientação: "a orientação a mensagens".

### Sistemas de gerenciamento de banco de dados

Os sistemas de gerenciamento de banco de dados tiveram uma grande evolução. Em quais aspectos? Fundamentalmente no que está fora da visão do usuário. Hoje são muito mais sólidos, são realmente sólidos! Suas disponibilidade, segurança, eficiência e escalabilidade progrediram muito e funcionam muito bem sobre as mais variadas combinações de software e hardware, geralmente de preços muito menores que os tradicionais.

Com relação à funcionalidade existe uma nova característica padrão muito importante: a definição e o controle automático, no nível do Sistema de Gerenciamento de Banco de Dados, da integridade referencial. E pouco ou nada mais realmente importante.

Todos os fabricantes suportam algum tipo de procedimento armazenado mas de forma incompatível entre eles.

Como conseqüência do anterior, a quantidade de casas de software dedicadas aos sistemas de gerenciamento de banco de dados diminuiu muito, só enormes empresas, com investimentos muito grandes,

podem nos dar produtos suficientemente sólidos, eficientes e escaláveis. Os principais fabricantes e produtos hoje são IBM (DB2 e Informix), Microsoft (SQL Server) e Oracle.

A eliminação das pequenas empresas fabricantes de sistemas de gerenciamento de banco de dados, junto com a existência de um padrão robusto do SQL fazem a inovação ser cada vez menor.

Os líderes, às vezes, em vez de aprofundar a liderança em seu mercado vertical pela via da inovação, utilizam o poder que lhes dá essa posição para serem fortes em outros mercados laterais como plataformas de execução, servidores de aplicativos e, ainda, pacotes de aplicativos ou serviços de desenvolvimento e terceirização. Estas políticas, muitas vezes, os isolam das casas de desenvolvimento de software (seus sócios de negócios naturais) e acabam atuando em prejuízo do desenvolvimento de novas funcionalidades nos sistemas de gerenciamento de banco de dados. Tudo se torna mais conservador: grande solidez e pouca inovação.

O SQL não resolveu dois temas importantes:

Um bom nível de "independência de dados" que permita que os programas sejam imunes às modificações estruturais no banco de dados.

Algum tipo de "inteligência" que permita definir de maneira declarativa regras e operadores de modo a tornar o desenvolvimento independente dos programadores: permitir que um usuário possa fazer tudo aquilo que quiser e que esteja autorizado a fazer, sem necessidade de programar.

A esta altura se estabelece uma pergunta óbvia: o padrão SQL atual será modificado substancialmente e em tempo razoável de maneira a dotá-lo de novos operadores de alto nível e outras características que viabilizem um comportamento "inteligente" dos bancos de dados?

A tendência geral, hoje, é a de suportar, para a escrita dos procedimentos armazenados, linguagens de programação comuns (Java, C#, etc.). Ou seja, é possível que em um futuro próximo possamos associar muito mais lógica transportável ao banco de dados, mas sempre a escrevendo em linguagens algoritmas, o que implica em programação manual e dependência da estrutura de tal banco de dados.

Ao mesmo tempo, aparecem outros problemas: os tradicionais de segurança foram bem solucionados pela via de uma boa identificação dos usuários e especificação do que estes estão autorizados a fazer com os dados, mas nossos sistemas de gerenciamento de banco de dados não são, por exemplo, imunes aos vírus. Esta é uma área em que os grandes fabricantes deverão trabalhar muito.

Por outro lado, a propagação do sistema operacional Linux trouxe como conseqüência o sucesso de novos sistemas de gerenciamento de banco de dados de preços muito baixos ou gratuitos e sem as pretensões de solidez e escalabilidade dos grandes líderes. Entre estes sistemas cabe destacar Postgres e MySQL.

**Mas, hoje não se discute mais o uso ou não dos bancos de dados: todos os aplicativos, em todo o mundo, são desenvolvidos sobre bancos de dados, a eficiência e a escalabilidade são incomparavelmente maiores das que se obtinham com arquivos convencionais e existem líderes claros que dominam o mercado.**

As formas de utilização foram variando com o tempo, um pouco pelas tecnologias disponíveis, um pouco pelas necessidades dos usuários e muito pelos gostos e pelas tendências dos profissionais da informática, desde a arquitetura centralizada original passando por uma arquitetura Cliente / Servidor e seguindo, agora, por arquiteturas multiservidor orientadas à rede e os Web Services bem plasmados nas plataformas de execução que dominarão o mundo da informática em um futuro previsível: Java e .net:

No final da primeira metade da década de 80 e quando todos os aplicativos funcionavam em arquitetura centralizada, apareceu um novo conceito: a "arquitetura cliente / servidor".

Tratava-se de racionalizar e dividir o processamento de dados entre o servidor central e os microcomputadores (clientes) e representava avanços importantes, além dos erros que todos cometemos no princípio em sua implementação.

A empresa que introduziu e desenvolveu o conceito foi a Sybase. Mas o mercado demorou muito: só em 1995 esta arquitetura foi adotada e a Sybase já não estava em seu melhor momento para lutar com grandes adversários.

Em 1996 a Sun, com o apoio de várias empresas de software, entre elas se destacam IBM e Oracle, lança a linguagem Java e a plataforma de desenvolvimento e execução Java.

Em 2001, a Microsoft lança a linguagem C# e a plataforma de desenvolvimento e execução .net, sofisticada implementação das mesmas idéias do Java com propósitos e resultados qualitativos similares.

Em 2001, a IBM compra a Informix o que, somado a sua liderança em seus sistemas OS/390 e OS/400, a transforma no maior fabricante de Sistemas de Gerenciamento de Banco de Dados.

Em 2002, a Microsoft lança, para o .net, um novo padrão para a comunicação entre os aplicativos e o banco de dados (DataSets e DataAdapters) que facilita a programação e ajuda muito a escrita de aplicativos muito escaláveis.

Orientação a processos

A orientação a processos, que em seu momento esteve sustentada pelas metodologias estruturadas deram um passo para o desenvolvimento orientado a objetos.

O desenvolvimento orientado a objetos, apoiado na existência de linguagens de programação orientadas a objetos, foi imposto claramente e trouxe enormes vantagens em programação quando esta se refere a dados na memória. Nestas condições, as vantagens sobre os processos tradicionais são enormes.

Existe um elemento ainda débil na orientação a objetos: Como se comunicar bem com o banco de dados? Durante algum tempo se experimentou muito os chamados sistemas de gerenciamento de banco de dados orientados a objetos (OODBMS) mas não se obteve resultados adequados.

Hoje é claro para todos que os sistemas de gerenciamento de banco de dados relacionais dominam o mundo dos dados e continuarão a fazê-lo em um futuro previsível e trabalha-se muito na criação de mecanismos de convivência entre programas orientados a objetos e bancos de dados relacionais (ORM: Object Relational Mapping). Existem conquistas isoladas, mas estão muito longe de um padrão.

Plataformas de execução

Há 40 anos existiam vários sistemas operacionais. Cada fabricante tinha o seu. A desvantagem era que cada fabricante devia ser capaz de atender diretamente todas as necessidades de seus clientes.

Quando surgiu a indústria do software, esta situação começou a mudar: o cliente atendia suas necessidades a partir de um conjunto de provedores. Mas um sistema operacional só seria atrativo para a indústria independente de software se chegasse a ter muitos usuários.

Muitos fabricantes se viram forçados pelos fatos, desde meados da década de 70, a abandonar seus sistemas operacionais e adotar outros. Subsistem aqueles de fabricantes que têm uma base instalada muito grande (como IBM OS/390, IBM OS/400) e crescem fortemente alguns relativamente independentes dos fabricantes de hardware: Unix, Windows e, ultimamente, Linux.

Cada sistema operacional tem suas próprias complexidades. Um passo a mais é pensar em plataformas de execução que funcionem em cima deles e que sejam muito mais amigáveis ao usuário do que eles.

Em 1996, a Sun lança sua linguagem Java (orientada a objetos) e sua plataforma de desenvolvimento e execução Java. Como linguagem, além de suas importantes características, é uma a mais, mas como plataforma de execução implica algo totalmente novo: pela primeira vez se oferece um ambiente amigável para desenvolver e executar aplicativos com total independência do hardware. A adoção do conceito foi rápida e geral ainda que o desenvolvimento de aplicativos Java reais levou mais tempo do que o previsto. Hoje é uma realidade.

Em 2001, a Microsoft lança sua plataforma de desenvolvimento e execução .net com suas próprias linguagens orientadas a objetos C# e Visual Studio .net e uma pluralidade de linguagens de terceiros.

Por que o cliente utilizaria alguma destas plataformas? Por muitas razões (que, em geral, excedem o propósito deste trabalho) mas, fundamentalmente, por uma: nos permitem a instalação e a atualização automática dos aplicativos nas estações de trabalho "clientes" com uma diminuição muito importante dos custos operacionais.

**Entendo que o mercado vai ser distribuído entre estas duas plataformas nos próximos anos e que os sistemas operacionais se tornarão *commodities*. Ao mesmo tempo, a adoção destas plataformas reforça o uso das linguagens orientadas a objetos.**

XML e a "orientação a mensagens"

Em 1999 aparece outro jogador muito importante: o XML, sistema que nos permite definir e manipular mensagens autodescritas. Outra tendência muito importante é iniciada: a "orientação a mensagens".

Desde seu anúncio, o XML foi rapidamente adotado por todos os fabricantes, o que o converteu em um padrão de fato e, depois, de direito e seu uso foi propagado com uma velocidade muito maior da tradicional nas inovações: realmente existia uma enorme necessidade insatisfeita de um sistema padrão de mensagens!

O XML tem as utilizações mais diversas, a maior parte delas fora da visão do usuário, mas se transforma em ator principal, por exemplo, nos Web Services e terá derivações muito importantes na constituição de verdadeiros bancos de dados estendidos: meu banco de dados, os de meus provedores / clientes, que acesso via XML e um conjunto de Web Services de interesse geral que "consumo" na forma de mensagens XML.

Mas, se aprofundamos um pouco veremos que o XML pode se converter em um agente importante na velha luta entre orientação a processos e orientação a dados: muitos processos complicados, no fundo, têm como objetivo nos prover um dado. Não poderemos pensar que esse dado nos seja entregue por uma mensagem XML (data provider) que possamos especificar de forma 100% declarativa? Minha opinião é que sim.

Uma vez mais, entretanto, aparecem os "bancos de dados hierárquicos": os fabricantes de sistemas de gerenciamento de banco de dados estão anunciando ampliações do SQL que nos permite armazenar e manipular mensagens XML. Espero que esta característica seja utilizada com a devida prudência (que fique circunscrita a casos muito específicos como o suporte de documentos de texto).

**Resumo:**

Nos últimos anos os sistemas de gerenciamento de Banco de Dados relacionais foram solidificados: hoje não se pensa em falar de outro tipo de Sistema de Gerenciamento de Banco de Dados.

Foram configurados padrões robustos que, utilizados estritamente, permitem a portabilidade dos aplicativos.

A existência desses padrões dificulta a adoção de modificações substanciais aos mesmos. Particularmente, é muito pouco provável que tenhamos em um futuro previsível bancos de dados qualitativamente muito mais evoluídos.

Em compensação, é previsível que continue nos dando cada vez mais solidez: disponibilidade, segurança, eficiência, escalabilidade otimização de recursos e que as novas ameaças sejam neutralizadas como as representadas por vírus inovadores.

Paralelamente, a programação orientada a objetos (particularmente impulsionada fortemente pelo previsível auge das plataformas Java e .net) foi generalizado e é vital que o problema do vínculo natural dos programas com o banco de dados (Object Relational Mapping) seja resolvido rapidamente.

A orientação a mensagens pode ser o componente que falta. **Talvez não tenhamos um futuro claramente orientado a processos nem a dados, mas uma realidade que se comporte como um conjunto de mensagens.**

**Os sistemas de gerenciamento de banco de dados de hoje e as idéias de Bachman e Codd**

Mas, com isto, chegamos ao final? Nossa pretensão – hoje – é a mesma de Charles Bachman em 1963?

Se for a mesma, se queremos acessar os Bancos de Dados a partir de programas em que fazemos quase tudo à mão, está tudo resolvido (e bem resolvido porque temos eficiência, disponibilidade, segurança, escalabilidade, recuperação diante de acidentes, etc. muito bons e, além disso, meios coerentes para interrogar o banco de dados). O sonho de Bachman foi realizado, mas demorou quase 40 anos para se tornar realidade!

Se, por outro lado, pensamos como pensava Edgar F. Codd em 1970 e queremos levar o uso dos bancos de dados a todo o mundo, estamos muito longe.

A tão mencionada e anunciada "era do usuário" parece ainda muito distante!

**Minha idéia sobre o futuro**

Creio firmemente em um paradigma diferente do vigente, do representado pela chamada "Tecnologia de Ponta" atual.

**Creio em "bancos de dados inteligentes" que nos permitam alimentar de forma declarativa todo o conhecimento necessário (fundamentalmente "regras do negócio", "regras de precedência e/ou de fluxo" e "regras de autorização" com toda a generalidade que estes conceitos representam) de maneira que qualquer usuário, sem a necessidade de conhecimento técnico algum, de uma maneira simples, possa fazer a qualquer momento tudo aquilo que quiser e esteja autorizado a fazer, sem a necessidade de programar.**

É possível que digam que o que quero é muito difícil.

Posso concordar. Mas refinemos esta resposta um pouco mais: é muito difícil por que carecemos da tecnologia necessária? Não: é muito difícil porque não se ajusta à Tecnologia de Ponta atual e pela tônica geral do mercado e da indústria que se mostraram muito conservadores.

Às vezes é muito mais difícil mudar de paradigma do que desenvolver a tecnologia necessária para tornar realidade o novo paradigma.

Só um dos grandes jogadores da área dos Sistemas de Gerenciamento de Banco de Dados (IBM, Microsoft, Oracle, algum outro?) pode abandonar o padrão SQL com sucesso e se decidir pela busca de soluções muito mais evoluídas. Mas, é um risco grande para um líder, que com o sucesso quase sempre tende a tornar-se conservador. Qual deles se animará?

**Se algum destes líderes assim o resolver, acho que teremos "bancos de dados inteligentes" rapidamente.**

**Caso não seja assim, o atual padrão SQL terá uma vida longa e relativamente pacífica. Nesse caso, o problema não seria resolvido com ele, mas com sistemas de nível mais alto, baseados em conhecimento, que "operem" automaticamente o SQL.**

Creio firmemente que é inevitável a substituição do paradigma atual (como 40 anos atrás orientado aos programadores e à programação algoritma e manual) por um novo paradigma orientado ao conhecimento.

Creio que esta substituição é inevitável porque a programação manual se mostra cada vez mais inviável e hoje implica em produtividade muito baixa do desenvolvimento e custos dramáticos (dinheiro e tempo) de manutenção de sistemas e, muitas vezes, obriga as empresas a sacrificar sua individualidade e renunciar a vantagens competitivas para adotar pacotes padrão com as conseguintes rigidezes, porque é impossível para elas construir e manter os aplicativos que realmente necessita.

Alguns dos países mais desenvolvidos tentam solucionar o problema importando mão-de-obra, de maneira a baixar seus custos.

Paralelamente, muitas empresas, para tratar de diminuir seus grandes custos, optaram por transferir sua programação e, paulatinamente, seu desenvolvimento (em alguns casos, chegando a uma terceirização total) a países em que a mão de obra é relativamente qualificada e muito barata. Cada vez mais países tendem para este mercado.

Esta tendência leva os usuários finais a uma paulatina perda de liberdade e constitui uma forte ameaça para a industria de software da maior parte dos países mais desenvolvidos e, inclusive, para a latino-americana.

**Contudo, estas intenções para diminuir os custos são simples paliativos: a solução real é a inserção de tecnologia tal que faça que profissionais bem pagos possam ser realmente eficientes em termos mundiais sem a necessidade de deslocar-se de seus países.**

Quanto demoraremos para ter bancos de dados inteligentes? Quanto demoraremos para chegar realmente à tão anunciada "era do usuário"?

Não sei, mas **não é um problema de tecnologia: a tecnologia existe. É um problema de atitude**.

Em nossa empresa demos os primeiros passos com o Genexus e hoje existem milhares de grandes aplicativos em todo o mundo, fundamentalmente sistemas de missão crítica, que são desenvolvidos e mantidos automaticamente com base em conhecimento, sem escrever uma só linha de código em linguagens de baixo nível. Há muito mais para fazer, mas a viabilidade do novo paradigma já foi mostrada. Não é pouco!

## Bibliografia

No que se refere à bibliografia utilizada para a elaboração deste trabalho cabem as seguintes observações:

É complexo proporcionar uma bibliografia concreta quando não se trata de temas pontuais mas de uma intensa experiência de vida de mais de 40 anos de profissão orientados desde o começo a coisas não convencionais.

Esta tarefa se torna ainda mais complexa quando o autor, por sua atividade profissional, teve o privilégio de participar de seminários e intercâmbios informais de idéias com muitas das principais figuras que fizeram a história dos bancos de dados e das metodologias de desenvolvimento de sistemas e/ou acesso a documentos não publicados dos mesmos (em alguns casos anos antes de os autores acabarem publicando).

Entretanto, existe, outro problema crescente a cada dia: faz 40 anos que havia um equilíbrio razoável entre a investigação realizada pelos fabricantes e a realizada pelas universidades de todo o mundo. Esta situação foi modificada paulatinamente e hoje a participação relativa das universidades na investigação informática foi minimizada.

Esta situação não é boa, mas constitui a realidade mundial e tem uma primeira conseqüência negativa muito importante: a falta de publicações oportunas.

A indústria se interessa por fazer coisas inovadoras, por superar seus competidores e, poucas vezes, por publicar oportunamente suas descobertas tecnológicas. Hoje, para nos manter atualizados na fronteira da tecnologia, devemos recorrer a investigações próprias e a nossas boas relações com outras empresas e com outros investigadores já que, geralmente, quando algo é publicado já é tarde para tomar determinações oportunas.

Por tudo isso, listar simplesmente uma bibliografia seria totalmente insuficiente e, além disso, a esta altura, quando muitas das incertezas que se teve em seu momento foram esclarecidas, seria inútil para o leitor porque, além disso, uns poucos documentos e livros que seguem tendo interesse hoje, quero me referir, como explicação e como agradecimento, a eventos e a pessoas que me emitiram seus pontos de vista.

**Livros e "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] Desenvolvimento orientado a processos**
**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] Desenvolvimento orientado a dados**

**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] Administração do conhecimento**

**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 dados 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] Outros**:

**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 Bancos de Dados 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 não publicados.**

Vários de **Jean Dominique Warnier**
Vários de **Charles Bachman**
Vários trabalhos desenvolvidos pelo **Laboratório Santa Teresa da IBM**
Muitos trabalhos de outros laboratórios de pesquisa
Muitas manuais e trabalhos não publicados de fabricantes dos diferentes sistemas de gerenciamento de banco de dados

**Cursos, seminários e discussões pessoais.**

Seminário Avançado de Bancos de Dados celebrado na Pontifícia Universidade Católica do Rio de Janeiro em 1976 com a participação de **Codd**, **Date**, **Blasgen**, **Stonebraker**, **Furtado**, etc.

Vários seminários organizados por SCI no Brasil e, como Diretor de Tecnologia da empresa organizadora, discussões técnicas com os expositores como, por exemplo, **Yourdon**, **Gane**, **De Marco** e **Date**

Discussões pessoais com meu amigo **Hermann Dolder**

Discussões pessoais com meu amigo **Ken Orr**

**E, muito especialmente, discussões pessoais com Nicolás Jodal e com a equipe da ARTech**