O que é um modelo de banco de dados?

Um modelo de banco de dados mostra a estrutura lógica de um banco de dados, incluindo as relações e restrições que determinam como os dados podem ser armazenados e acessados. Modelos de banco de dados individuais são projetados com base nas regras e nos conceitos do modelo de dados mais abrangente que os designers adotam. A maioria dos modelos de dados pode ser representada por um diagrama de banco de dados acompanhante.

Você pode optar por descrever um banco de dados com qualquer um destes modelos dependendo de vários fatores. O fator mais importante é se o sistema de gestão de banco de dados que você usa suporta um modelo específico. A maioria dos sistemas de gestão de banco de dados é construída com um modelo de dados particular em mente e exige que seus usuários adotem esse modelo, embora alguns ofereçam suporte a vários modelos.

Além disso, diferentes modelos se aplicam a diferentes estágios do processo de criação de banco de dados. Os modelos de dados conceituais de alto nível são os melhores para mapear as relações entre os dados de maneira que as pessoas percebam esses dados. Os modelos lógicos baseados em registros, por outro lado, refletem melhor as formas com que os dados são armazenados no servidor.

Selecionar um modelo de dados é também uma questão de alinhar suas prioridades para o banco de dados com os pontos fortes de um determinado modelo, independentemente de essas prioridades incluírem velocidade, redução de custos, usabilidade ou qualquer outra coisa.

Vamos dar uma olhada em alguns dos modelos de bancos de dados mais comuns.

Modelo relacional

O modelo mais comum, o modelo relacional, classifica dados em tabelas, também conhecidas como relações, cada uma das quais consiste em colunas e linhas. Cada coluna lista um atributo da entidade em questão, como preço, código postal ou data de nascimento. Juntos, os atributos em uma relação são chamados de domínio. Um determinado atributo ou combinação de atributos é escolhido como uma chave primária que pode ser consultada em outras tabelas, quando é chamada de chave estrangeira.

Cada linha, também chamada de tupla, inclui dados sobre uma instância específica da entidade em questão, como um determinado colaborador.

O modelo também explica os tipos de relações entre essas tabelas, incluindo relações uma para uma, uma para muitas e muitas para muitas. Eis um exemplo:

Dentro do banco de dados, as tabelas podem ser normalizadas ou levadas a cumprir as regras de normalização que tornam o banco de dados flexível, adaptável e redimensionável. Quando normalizado, cada dado é atômico, ou dividido em pequenos pedaços úteis.

Os bancos de dados relacionais são tipicamente escritos em SQL (Structured Query Language). O modelo foi introduzido por E. F. Codd em 1970.

O modelo relacional foi criado por Edgar F. Codd, nos anos 70 e começou a ser usado com o advento dos bancos de dados relacionais, nos anos 80. A idéia de modelo relacional se baseia no princípio de que as informações em uma base de dados podem ser consideradas como relações matemáticas e que podem ser representadas, de maneira uniforme, através do uso de tabelas onde as linhas representam as ocorrências de uma entidade e as colunas representam os atributos de uma entidade do modelo conceitual.

As relações no modelo relacional são conjuntos de dados vistos como tabelas cujas operações são baseadas na álgebra relacional (projeção, produto cartesiano, seleção, junção, união e subtração) e que manipulam conjuntos de dados ao invés de um único registro, isto é, cada operação realizada afeta um conjunto de linhas e não apenas uma única linha, ainda que algumas operações possam afetar uma única linha (conjunto com um único elemento).

Da mesma forma, a resposta das operações de consulta são sempre na forma de uma tabela. As operações da álgebra relacional são implementadas por linguagens não procedurais de alto nível, sendo a SQL a linguagem padrão para os bancos de dados relacionais e universalmente usada, tendo sido padronizada pelo ANSI (American National Standard Institute).

Principais Vantagens do Modelo Relacional

Entre as principais vantagens do modelo relacional podemos citar:

- Independência total dos dados;
- Visão múltipla dos dados;
- Redução acentuada na atividade de desenvolvimento. Particularmente para extração de dados para relatórios e consultas específicas do usuário;
- Maior segurança no acesso aos dados;
- Maior agilidade para consulta/atualização;
- Qualidade dos dados garantida por restrições de integridade (identidade, referencial e de domínio)

As 12 Regras de Codd

Ao definir o modelo relacional, Codd estabeleceu 12 regras para determinação de um banco de dados relacional. Estas regras são usadas portanto para se verificar a fidelidade de um banco de dados ao modelo relacional. Na prática são poucos os gerenciadores de banco de dados que atendem a todas as 12 regras. Na maior parte dos casos são atendidas no máximo 10 regras.

1. Toda informação num banco de dados relacional é apresentada a nível lógico na forma de tabelas;
2. Todo dado em um banco de dados relacional tem a garantia de ser logicamente acessível, recorrendo-se a uma combinação do nome da tabela, um valor de chave e o nome da coluna;
3. Tratamento sistemático de valores nulos; (ausência de informação)
4. O dicionário de dados, catálogo, do banco de dados é baseado no modelo relacional;
5. Há uma uma linguagem não procedural para a definição, manipulação e controle dos dados;
6. Tratamento das atualizações de visões dos dados;
7. Tratamento de alto nível para inserção, atualização e eliminação de dados;
8. Independência física dos dados; (mudança na memória e no método de acesso, criação de um novo índice, criação de uma nova coluna)
9. Independência lógica dos dados; (mudança no tamanho de uma coluna)
10. Restrição de Integridade; (Identidade, Referencial e Domínio)
11. Independência de Distribuição dos dados;
12. Não subversão das regras de integridade ou restrições quando se usa uma linguagem hospedeira;

O Conceito de Chave no Modelo Relacional

Chaves e Índices

Chave - O conceito de chave designa um item de busca, ou seja, um dado que será usado para efetuar uma consulta no banco de dados. É um conceito lógico que só faz sentido para a aplicação e não existe fisicamente no banco de dados.

Índice - O conceito de índice está associado a um recurso físico usado para otimizar uma consulta no banco de dados. É um recurso físico, ou seja, um índice é uma estrutura de dados, (endereços), que existe fisicamente no banco de dados.

Existem diferentes tipos de chave em um modelo relacional. Vamos ver cada um dos tipos de chave abaixo:

Chave Primária: A chave primária é usada para identificar univocamente uma linha em uma tabela. A chave primária pode ser composta, ter vários atributos, ou simples, um único atributo. Por exemplo, o atributo CPF pode ser usado como chave primária para a tabela CLIENTES pois identifica um único cliente considerando que não existe mais de um cliente com o mesmo CPF.

Chave Secundária: A chave secundária é usada para acessar um conjunto de informações. Pode ser formada por um único atributo ou mais de um atributo que identifica(m) um subconjunto de dados em uma tabela. Normalmente, se cria um índice para uma chave secundária como forma de otimizar a consulta feita por aquela chave ao banco de dados. Por exemplo, podemos ter uma chave secundária formada pelo CEP para a tabela de CLIENTES pois esta chave identifica um subconjunto de clientes que residem em uma rua.

Chave Candidata: A chave candidata é formada por um atributo que identifica uma única linha na tabela. Como uma tabela pode possuir mais de um atributo identificador único podemos ter várias chaves candidatas em uma única tabela, sendo que apenas uma das chaves candidatas pode ser escolhida para ser a chave primária da tabela. As demais chaves permanecem como chaves candidatas na tabela. Por exemplo, podemos ter uma chave candidata formada pela coluna NIT (PISPASEP) na tabela FUNCIONARIOS que possui como chave primária a coluna MATRICULA. Ambas identificam univocamente um linha na tabela FUNCIONARIOS, porem a chave NIT é candidata e a chave MATRICULA é a chave primária.

Chave Estrangeira: A chave estrangeira é formada por atributos que são chave primária em outra tabela, servindo assim para estabelecer relacionamentos entre as tabelas de um banco de dados. Assim, quando dizemos que duas tabelas estão relacionadas através de uma coluna devemos observar que em uma tabela esta coluna será chave primária e na outra tabela ela será uma chave estrangeira que fará a ligação entre as duas tabelas, estabelecendo o relacionamento. Por exemplo, podemos ter na tabela FUNCIONARIOS uma chave estrangeira COD_DEPTO que estabelece um relacionamento entre a tabela FUNCIONARIOS e a tabela DEPTOS, sendo que na tabela DEPTOS a coluna COD_DEPTO é a chave primária.

Regras de Integridade do Modelo Relacional

O modelo relacional possui duas regras de integridade descritas a seguir.

Integridade de Identidade - A chave primária nao pode conter valores nulos. Como toda informação em um banco de dados relacionam precisa ter uma identidade exclusiva, a chave primária deve ser obrigatoriamente preenchida. Alem disso, a chave primária não deve ter valores repetidos em um tabela, de forma a garantir que exista apenas uma linha para cada valor definido para a chave primária.

Integridade Referencial - Se uma determinada tabela A possui uma chave estrangeira que estabelece relacionamento com uma tabela B, entao o valor da chave estrangeira da tabela A deve ser igual a um valor de chave primária na tabela B. Esta regra garante que as referencias de uma tabela para outra tabela sejam válidas, de forma que os relacionamentos sejam consistentes e não ocorra inconsistencia nos dados, como haver um funcionário alocado em um departamento que não existe. Assim, para todo valor de uma coluna que é chave estrangeira em uma tabela, deve haver um valor correspondente na coluna que é chave primária da tabela com a qual a chave estrangeira faz referencia.

Como nem sempre o relacionamento entre tabelas é obrigatório uma chave estrangeira pode possuir valor nulo.

É importante ressaltar que em um modelo relacional estas regras de integridade são garantidas pelo gerenciador de banco de dados de forma automática, sem a necessidade de se tratar estas regras no código da aplicação. Ou seja, o programador não precisa programar validações no sistema para garantir que estas regras sejam atendidas pois o próprio gerenciador de banco de dados cuida disso.

Integridade de Domínio - Restringe o conjunto de valores que podem ser gravados em uma coluna de uma tabela. Desta forma, somente os valores que pertencem ao domínio podem ser gravados na coluna da tabela. Outros valores não são permitidos e a atualização é desfeita pelo gerenciador de banco de dados. O domínio define um conjunto de valores. Quando este domínio é associado a uma coluna de uma tabela, somente os valores definidos para o domínio podem ser gravados na coluna. Este tipo de restrição garante a qualidade de dados na base de dados.

Características do Modelo Relacional

- Uma tabela deve ser acessível por qualquer coluna, mesmo que não tenha sido definida como chave;
- O relacionamento entre duas tabelas não existe fisicamente, pois o relacionamento é lógico e representado através de chaves estrangeiras;
- Uso de linguagens não-procedurais e auto-contidas; (SQL)
- Um otimizador de consultas para definição do melhor plano de acesso aos dados;

Regras para Derivação do Modelo Conceitual para o Modelo Relacional

Nesta etapa é feita a transformação das entidades e relacionamentos do modelo E-R para o modelo relacional, no qual os dados são representados por tabelas. Para tanto, foram definidas regras para esta transformação de forma a atender às características do modelo relacional.

Estas regras garantem que o modelo relacional estará adequado, alinhado com o modelo conceitual e sem inconsistências. O resultado desta etapa é um diagrama de tabelas, contendo as tabelas, chaves primárias, chaves estrangeiras e restrições de integridade, formando assim o modelo lógico que servirá de base para o projeto físico do Banco de Dados.

Mapeamento das Entidades - Toda entidade torna-se uma tabela levando todos os atributos definidos na entidade que tornam-se colunas na tabela criada. O identificador da entidade torna-se a chave primária da tabela que não permitirá repetição de valores e nem valores nulos.
Mapeamento de Atributos - Os atributos das entidades e dos relacionamentos devem ser gerados de forma que minimizem o consumo de espaço de armazenamento e torne mais eficiente a consulta de dados. Devem ser consideradas as características do gerenciador de banco de dados que será utilizado para implementar o banco de dados físico. Devem ser escolhidos o tipo de dado e tamanho adequados para cada coluna criada na tabela.
Mapeamento de Relacionamentos - O mapeamento dos relacionamentos implica na transformação de atributos das entidades em colunas nas tabelas e, em casos específicos, implica também na criação de novas tabelas a partir de relacionamentos e entidades associativas. Existem diferentes abordagens dependendo da cardinalidade do relacionamento:
Relacionamentos que possuem atributos - Estes relacionamentos se tornam tabelas no caso de relacionamentos n:n. No caso de relacionamentos 1:n os atributos do relacionamento são transferidos para a tabela que possui cardinalidade n;
Relacionamentos são representados por chaves estrangeiras (Foreign Key) – Todo atributo correspondente à chave primária de outra relação, base para a integridade referencial, é definido como uma chave estrangeira);
Relacionamento 1 para 1 (1:1) - Uma das entidades envolvidas no relacionamento carrega o atributo identificador que deve ser definido com chave estrangeira na tabela criada para a entidade fazendo referencia à chave primária da tabela criada para a outra entidade. O Critério para escolher qual tabela receberá a chave estrangeira depende do negócio que está sendo modelado, sendo necessária análise caso a caso. Porém em geral se escolhe a tabela onde faz mais sentido colocar o atributo, entidade mais importante para o negócio.
Relacionamento 1 para Muitos (1:N) - A entidade cuja cardinalidade é N recebe o atributo identificador da entidade com cardinalidade 1 que será mapeado como uma chave estrangeira na tabela criada para a entidade com

cardinalidade N. Alem disso, recebe os atributos do relacionamento se houve. Caso seja a entidade com cardinalidade N seja uma entidade fraca, então ela recebe o atributo identificador da entidade com cardinalidade 1 que deve ser mapeado de forma a compor a chave primária da tabela criada para a entidade com cardinalidade N, como forma de manter a dependência funcional em relação à entidade com cardinalidade 1.

Relacionamento Muitos para Muitos (M:N) - Deve ser criada uma tabela que recebe os atributos identificadores das entidades que participam do relacionamento, sendo criada a chave primária composta pelas colunas derivadas dos atributos identificadores. Alem disso, a tabela recebe todos os atributos do relacionamento, se existirem. Este é o único caso em que um relacionamento se torna uma tabela, em todos os demais casos, são criadas chaves estrangeiras nas tabelas a fim de estabelecer os relacionamentos.

Relacionamentos Múltiplos (Ternário, Quaternário, etc.)- Deve ser criada uma tabela que recebe tantos atributos identificadores quantas foram as entidades que participam do relacionamento. A chave primária desta tabela é composta por todos os atributos identificadores. É o caso de relacionamentos ternário, quaternários, etc.

Relacionamento Auto-relacionamento - Incluir a chave primária da entidade na própria entidade como chave estrangeira, gerando uma estrutura de acesso a partir dessa chave.

Mapeamento de Generalização/Especialização - Deve ser criada uma tabela para a entidade pai e uma tabela para cada entidade filha. Os atributos comuns às entidades filhas devem ser mapeados na tabela criada para a entidade pai. As tabelas criadas para cada entidade filha devem receber o atributo identificador da entidade pai na composição da chave primária e receber também os atributos específicos da entidade filha correspondente. A entidade pai e a entidade filha também podem ser mapeadas para uma única tabela.

Mapeamento de Agregações - normalmente gera uma nova tabela que representa a agregação. Esta tabela normalmente faz relacionamento com uma tabela associativa;

Modelo hierárquico

O modelo hierárquico organiza dados em uma estrutura do tipo árvore, onde cada registro tem um único "pai" ou raiz. Registros "irmãos" são classificados em uma ordem específica. Essa ordem é usada como a ordem física para armazenar o banco de dados. Este modelo é bom para descrever muitas relações do mundo real.

Esse modelo foi usado principalmente pelos Sistemas de Gestão de Informações da IBM nos anos 60 e 70, mas são raramente vistos hoje devido a certas ineficiências operacionais.

Um Banco de dados hierárquico consiste em uma coleção de registros que são conectados uns aos outros por meio de ligações. Um registro é uma coleção de campos, cada qual contendo apenas um valor de dados. Uma ligação é uma associação entre exatamente dois registros. O modelo hierárquico é, portanto similar ao modelo de rede, no sentido de que dados e relacionamentos entre dados são também representados por registros e ligações, respectivamente. O modelo hierárquico difere do modelo de rede na organização de registros como coleção de árvores em vez de como grafos arbitrários.

Um diagrama com estrutura de árvore é um esquema para um banco de dados hierárquico. Tal diagrama consiste em dois componentes básicos: retângulos, que correspondem a tipos de registro, e linhas, que correspondem a ligações. O diagrama com estrutura de árvore serve para os mesmos propósitos que um diagrama entidade-relacionamento; a saber, ele especifica a estrutura lógica geral do banco de dados. Um diagrama com estrutura de árvore é similar ao diagrama de estrutura de dados no modelo de rede. A principal diferença é que, no primeiro, tipos de registro são organizados na forma de uma árvore enraizada. Para todo diagrama entidade-relacionamento, existe um diagrama com estrutura de árvore correspondente.

O esquema de banco de dados é, portanto representado como uma coleção de diagramas com estrutura de árvore. Para cada diagrama, existe uma única instância de uma arvore do banco de dados. A raiz dessa árvore é um nó auxiliar. Os filhos desse nó são instâncias de fato do tipo de registro apropriado. Cada instância pode, por sua vez ter diversas instâncias de vários tipos de registro, como especificadas no diagrama com estrutura de árvore correspondente.

A linguagem de manipulação de dados consiste em uma série de comandos que são embutidos em uma linguagem hospedeira. Esses comandos fazem o acesso e manipulam itens do banco de dados assim como variáveis declaradas localmente. Para cada programa de aplicação o sistema mantém uma área de trabalho de programa que contém gabaritos de registro, ponteiros correntes e indicadores de estado.

Os itens de dados são buscados através do comando get, que localiza um registro o banco de dados e posiciona o ponteiro corrente para apontar para ele, e então copia aquele registro do banco de dados para o gabarito de área de trabalho do programa apropriado. Existe uma série de diferentes formas do comando get. A principal distinção entre elas é se um registro deve ser localizado dentro de toda a árvore de um banco de dados ou dentro de uma subárvore.

Vários mecanismos estão disponíveis para atualizar informações no banco de dados. Eles incluem a criação e a remoção de registros (via operações insert e delete) e a modificação (via operação replace) do conteúdo dos registros existentes.

No caso de relacionamentos muitos-para-muitos, a duplicação de registros é necessária para preservar a estrutura de árvore do banco de dados. A duplicação de registros tem dois inconvenientes principais: atalizações podem levar a inconsistência de dados e o desperdício de espaço é inevitável. A solução é o registro virtual, tal registro não contém valores de dado, ele contém um ponteiro lógico para um registro físico partícula. Quando um registro é duplicado em diversas árvores de banco de dados, uma única cópia daquele registro é mantida em uma das árvores e todas as outras ocorrências do mesmo são substituídas por um registro virtual contendo um ponteiro para aquele registro físico. A linguagem de manipulação de dados para essa nova configuração leva ao mesmo caso em que a duplicação de registro é permitida. Assim, um usuário não precisa preocupar-se com essas mudanças.

A implementação de banco de dados hierárquicos não usa ponteiros pai-filho, uma vez que eles requerem o uso de registros de tamanho variável. Em vez disso, são usadas cadeias em pré-ordem. Essa técnica permite que cada registro contenha exatamente dois ponteiros. Opcionalmente, um terceiro ponteiro filho – para – pai pode ser adicionado.

Modelo de rede

O modelo de rede se baseia no modelo hierárquico, permitindo relações muitas para muitas entre registros vinculados, implicando em vários registros "pai". Baseado na teoria de conjuntos matemáticos, o modelo é construído com conjuntos de registros relacionados. Cada conjunto consiste em um registro proprietário, ou "pai", e um ou mais registros de membro, ou "filho". Um registro pode ser um membro, ou "filho", em vários conjuntos, permitindo que esse modelo transmita relações complexas.

Foi mais popular nos anos 70, depois de ter sido formalmente definido pela Conferência sobre Linguagens de Sistemas de Dados (CODASYL).

Vantagens e Desvantagens dos Bancos de Dados de Rede

bancos de dados têm uma ampla gama de aplicações em sistemas de negócios . Eles ajudam a automatizar tarefas de personalização de documentos repetitivos , eles permitem que as operadoras de telefonia para obter acesso rápido aos dados do cliente e acelerar a consolidação das informações financeiras. Databases categorizar os tipos de dados e , em seguida, estabelecer relações entre essas categorias , os três principais sistemas de gerenciamento de bancos de dados são relacionais para , hierárquico e de rede. Cada um tem seus próprios méritos. Existem vantagens e desvantagens no modelo de rede de dados em comparação com os outros dois sistemas de gestão de dados . Organização

As informações são agrupadas em entidades ou registros e cada entidade tem atributos , que correspondem a títulos de coluna . Por exemplo, a entidade " cliente" teria nome da empresa e número de telefone como dois atributos . Cada entidade tem um atributo que define que identifica unicamente cada registro na tabela . Isto é chamado de chave , o qual é um índice . As tabelas são unidas por esses atributos -chave para expandir a gama de dados disponíveis para cada consulta.

Alternativas

O sistema de gerenciamento de banco de dados relacional domina aplicativo de negócios. Os dados é " normalizado " e , em seguida, armazenados em tabelas . Dados normalizados se encaixa em uma estrutura que erradica repetição e redundância de dados. É a uniões entre as tabelas de dados normalizado que cria uma relação entre os atributos e, portanto, dá a este sistema de gerenciamento de banco de dados o seu nome. Bancos de dados hierárquicos são menos comuns. Ligam entidades juntos, novamente por atributos -chave, mas organizado como relações pai-filho . Isso cria uma estrutura de árvore de dados e é adequado para informações geográficas, ou dados arquivados utilizados para " mineração de dados ".

Relacionamentos

Um banco de dados de rede é semelhante a uma base de dados hierárquica . No entanto , enquanto que um banco de dados hierárquico tem apenas um -para-muitos relacionamentos entre entidades , um sistema de gerenciamento de banco de dados de rede permite que muitos -para-muitos relacionamentos. Esta é a característica definidora de SGBDs rede. Em um banco de dados hierárquico , uma entidade pai pode ter filhos muitas entidades , mas cada criança pode ter apenas um pai . Esta é a relação de um- para-muitos. O relacionamento muitos-para- muitos em um DBMS rede permite que uma entidade pai para ter crianças muitas entidades , e uma entidade filho tenha muitas entidades pai .

Benefícios

a principal vantagem do modelo de DBMS rede é o relacionamento muitos- para-muitos. O banco de dados hierárquico não leva em conta a partir de muitos eventos organizacionais humanos que exigem que uma entidade tem muitas ligações ascendentes para outras entidades. Por exemplo , em uma base de dados médica , um médico pode ser atribuído a diversos enfermarias e cuidar de muitos pacientes , enquanto que o paciente é uma divisão que não tem ligação por meio do relacionamento da entidade médico e por isso a entidade paciente também precisa de um link tanto para a entidade médico ea entidade ala. Assim, uma série de cruzadas as relações se desenvolvem rapidamente.
Desvantagens

Bancos de dados relacionais têm estruturas que resolvem muitos -para-muitos relacionamentos entre instâncias , ou registros , de uma entidade , mas os bancos de dados de rede não. Por exemplo , no exemplo de banco de dados de

rede, o paciente pode ter muitos médicos , e um médico tem muitos pacientes , mas os DBMS rede só poderia estabelecer um médico para muitos relação paciente ou um paciente para muitos relacionamentos médico, não tanto.
O **Modelo Hierárquico** tem como representantes fundamentais os IMS ("Information Management System") e o System 2000/MRI. A sua concepção original está perfeitamente adaptada ao processamento sequencial, suportado por banda magnética. Com o aparecimento de dispositivos de acesso directo, este modelo desenvolveu-se de modo a ultrapassar alguns dos problemas postos por uma estrutura rigidamente hierárquica. Este modelo, na sua forma original, estrutura-se segundo uma 7hierarquia de segmentos (entidades-tipo), em que os de nível mais baixo dependem dos de nível superior. Verifica-se neste modelo a existência de informação redundante e alguns problemas na inserção, eliminação e actualização dos dados. Assim, os problemas de redundância, inconsistência e perda de informação (que andam todos normalmente associados) devem ser resolvidos o melhor possível para se obter uma boa implementação de uma Base de Dados. Surgiram assim as Bases de Dados lógicas, permitindo definir novas e diferentes hierarquias, sobre as hierarquias físicas, ganhando em flexibilidade sem, por isso, aumentar a redundância. O Modelo Hierárquico comporta-se bastante bem se estiver a representar uma estrutura que, na realidade, seja apreendida como hierárquica, ou seja, em que as correspondências sejam N->1.

- O **Modelo Rede** ("network") visa ultrapassar alguns dos problemas postos pelo Modelo anterior. As realizações mais conhecidas assentam na proposta do grupo de trabalho sobre Bases de Dados (DBTG) reuindo na conferência CODASYL ("Conference on data systems languages"), que é responsável pela normalização da linguagem COBOL. Como exemplos mais importantes temos: TOTAL, IDMS, ADABAS. Para minimizar o problema da redundância e as anomalias de inserção e apagamento e obter maior flexibilidade, desenvolveu-se o Modo Rede (ou reticulado). Neste Modelo as associações entre entidades tipo funcionam essencialmente nos dois sentidos, ao contrário do Modelo Hieráquico, em que funciona num único sentido, do segmento superior para o inferior. Cada entidade (registo) pertence a um determinado Record (entidade-tipo) e pode estar associada a outras através de várias ligações, tendo portanto vários superiores imediatos, sem ser necessário, para isso, repetir a sua representação. As associações são chamadas de ligações (SET) e representam-se por cadeias de apontadores, no Modelo de Estados. Fisicamente podem ser implementadas também por apontadores ou por outros processos, tais como registos de comprimento variável. Aqui a inserção, eliminação e actualização é mais eficiente. Várias características se podem referir:
- Devido ao facto de o Modelo de dados Rede se basear em estruturas de apontadores, a pesquisa de informação na Base de Dados processa-se através destes e não dos registos propriamente ditos, os quais só são trazidos para a área de trabalho quando necessário.
- Torna-se assim conveniente saber em que ponto da Base de Dados nos encontramos em cada momento (função desempenhada pelos pontos correntes – para cada record e cada ligação existe um ponto corrente, onde se guarda a chave do registo correspondente acedido em último lugar).
- O facto das associações serem representadas por cadeias de apontadores torna o Modelo muito próximo da representação física, dificultando a independência dos dados. Além disso, o programador tem de saber claramente quais as ligações que existem e quais as que não existem, para poder organizar as suas rotinas.
- Os problemas de redundância e as anomalias de inserção e eliminação não são aqui tão evidentes como no Modelo anterior (resultante mais de uma questão de normalização do que uma característica do Modelo).
- O **Modelo Relacional**, o mais simples em termos de concepção, é também o mais recente, pois a sua aplicação eficiente só foi possibilitada pelos avanços tecnológicos da década de 70. A norma estabelecida foi designada ANSI/SPARC. Existem vários sistemas baseados em abordagens diferentes deste Modelo: RendezVous, Oracle, ISBL, QUEL, QBE ("query by example"), INGRESS, SQL/DS, INFORMIX, DB2, DBASE IV, UNIFY, Paradox, Access. Este Modelo foi proposto por CODD que enunciou os objectivos da abordagem relacional: Fornecer um elevado grau de independência dos dados.
- Proporcionar uma definição comum dos dados, com grande simplicidade, de forma que uma vasta gama de utilizadores numa empresa (desde "não informáticos" a "informáticos") possa com ela interactuar (sem prejuízo da sobreposição de vistas para fins específicos).
- Simplificar o potencialmente gigantesco trabalho do administrador da Base de Dados.
- Introduzir uma fundamentação teórica na gestão de Bases de Dados (um domínio carente de princípios sólidos e linhas mestras).

- Fundir os campos de pesquisa de factos e de gestão de fichas, preparando a posterior adição de serviços de inferência na realidade comercial.
- Elevar a programação de aplicações de Bases de Dados para um novo nível, em que conjuntos (e mais especificamente relações) sejam tratados como operandos, em vez de serem processados elemento a elemento.