2010

Banco de Dados

Vanderlei Biava
Escola Profissionalizante ESSEI
06/08/2010

Sumário

Banco de Dados.

Banco de dados (ou **base de dados**), é um conjunto de registros dispostos em estrutura regular que possibilita a reorganização dos mesmos e produção de informação. Um banco de dados normalmente agrupa registros utilizáveis para um mesmo fim.

Um banco de dados é usualmente mantido e acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD). Normalmente um SGBD adota um modelo de dados, de forma pura, reduzida ou estendida. Muitas vezes o termo **banco de dados** é usado, de forma errônea, como sinônimo de SGDB.

O modelo de dados mais adotado hoje em dia é o modelo relacional, onde as estruturas têm a forma de **tabelas**, compostas por tuplas (linhas) e colunas.

Um Sistema de Gestão de Bases de Dados, (SGBD) não é nada mais do que um conjunto de programas que permitem armazenar, modificar e extrair informação de um banco de dados. Há muito tipos diferentes de SGBD. Desde pequenos sistemas que funcionam em computadores pessoais a sistemas enormes que estão associados a mainframes. Um Sistema de Gestão de Base de Dados implica a criação e manutenção de bases de dados, elimina a necessidade de especificação de definição de dados, age como interface entre os programas de aplicação e os ficheiros de dados físicos e separa as visões lógica e de concepção dos dados. Assim sendo, são basicamente três as componentes de um SGBD:

1. Linguagem de definição de dados (especifica conteúdos, estrutura a base de dados e define os elementos de dados);
2. Linguagem de manipulação de dados (para poder alterar os dados na base);
3. Dicionário de dados (guarde definições de elementos de dados e respectivas características – descreve os dados, quem os acede, etc. [questões de informação]). (Gouveia; 2004).

Utilização

Os bancos de dados são utilizados em muitas aplicações, abrangendo praticamente todo o campo dos programas de computador. Os bancos de dados são o método de armazenamento preferencial e baseiam-se em tecnologias padronizadas de bancos de dados.

Um **banco de dados** é um conjunto de informações com uma estrutura regular. Um banco de dados é normalmente, mas não necessariamente, armazenado em algum formato de máquina legível para um computador. Há uma grande variedade de bancos de dados, desde simples tabelas armazenadas em um único arquivo até gigantescos bancos de dados com muitos milhões de registos, armazenados em salas cheias de discos rígidos.

Bancos de dados caracteristicamente modernos são desenvolvidos desde os anos da década de 1960. Um pioneiro nesse trabalho foi Charles Bachman.

Apresentação dos dados

A apresentação dos dados geralmente é semelhante à de uma planilha eletrônica, porém os sistemas de gestão de banco de dados possuem características especiais para o armazenamento, classificação, gestão da integridade e recuperação dos dados. Com a evolução de padrões de conectividade entre as tabelas de um banco de dados e programas desenvolvidos em linguagens como Java, Delphi, Visual Basic, C++ etc, a apresentação dos dados, bem como a navegação, passou a ser definida pelo programador ou o designer de aplicações. Como hoje em dia a maioria das linguagens de programação fazem ligações a bancos de dados, a apresentação destes tem ficado cada vez mais a critério dos meios de programação, fazendo com que os bancos de dados deixem de restringir-se às pesquisas básicas, dando lugar ao compartilhamento, em tempo real, de informações, mecanismos de busca inteligentes e permissividade de acesso hierarquizada.

Direitos de propriedade

A Directiva CE de Bases de Dados (*EU Database Directive*), estabelecida pelo Parlamento Europeu em de 11 de março de 1996, fixa os termos de protecção jurídica e fisica do sitema de bancos de dados, em particular os *direitos de propriedade* sobre a base.

Mesmo para os países que não a adoptam explicitamente, ou não possuam normas mais específicas sobre o tema, como o Brasil, tem sido a principal referência.

Visão de negócio

Todas as organizações têm quantidades, por vezes, astronômicas de dados e informação que têm de armazenar. Contudo, o papel tem problemas ao nível da persistência (tempo e tipo de visualização) e da recuperação (validação e verificação), ou seja, dura pouco. Neste sentido, torna-se mais fácil encontrar a informação numa base de dados que recorre a uma das tecnologias de informação de maior sucesso. Ou seja, as bases de dados estendem a função do papel ao guardar a informação em computadores. Qualquer empresa que pretenda garantir um controle efetivo sobre todo o seu negócio, tem obrigatoriamente de recorrer a sistemas de gestão de bases de dados. O Microsoft Excel continua a ser uma ferramenta de controle extremamente poderosa porque consegue operacionalizar os dados e assim criar informação útil ao planejamento diário das empresas. Contudo, existem outro tipo de ferramentas, mais completas e com funcionalidades acrescidas que elevam para outros níveis, a capacidade operacional de gerar informação de valor para a organização.

Modelos de base de dados

O modelo plano (ou tabular) consiste de matrizes simples, bidimensionais, compostas por elementos de dados: inteiros, números reais, etc. Este modelo plano é a base das planilhas eletrônicas.

O modelo em rede permite que várias tabelas sejam usadas simultaneamente através do uso de apontadores (ou referências). Algumas colunas contêm apontadores para outras tabelas ao invés de dados. Assim, as tabelas são ligadas por referências, o que pode ser visto como uma rede. Uma variação particular deste modelo em rede, o modelo hierárquico, limita as relações a uma estrutura semelhante a uma árvore (hierarquia - tronco, galhos), ao invés do modelo mais geral direcionado por grafos.

Bases de dados relacionais consistem, principalmente de três componentes: uma coleção de estruturas de dados, nomeadamente relações, ou informalmente tabelas; uma coleção dos operadores, a álgebra e o cálculo relacionais; e uma coleção de restrições da integridade, definindo o conjunto consistente de estados de base de dados e de alterações de estados. As restrições de integridade podem ser de quatro tipos: domínio (também conhecidas como type), atributo, relvar (variável relacional) e restrições de base de dados.

Diferentemente dos modelos hierárquico e de rede, não existem quaisquer apontadores, de acordo com o Princípio de Informação: toda informação tem de ser representada como dados; qualquer tipo de atributo representa relações entre conjuntos de dados. As bases de dados relacionais permitem aos utilizadores (incluindo programadores) escreverem consultas (*queries*) que não foram antecipadas por quem projetou a base de dados. Como resultado, bases de dados relacionais podem ser utilizadas por várias aplicações em formas que os projetistas originais não previram, o que é especialmente importante em bases de dados que podem ser utilizadas durante décadas. Isto tem tornado as bases de dados relacionais muito populares no meio empresarial.

O **modelo relacional** é uma teoria matemática desenvolvida por **Edgard Frank Codd**, matemático e pesquisador da IBM, para descrever como as bases de dados devem funcionar. Embora esta teoria seja a base para o software de bases de dados relacionais, muito poucos sistemas de gestão de bases de dados seguem o modelo de forma restrita ou a pé da letra - lembre-se das 13 leis do modelo relacional - e todos têm funcionalidades que violam a teoria, desta forma variando a complexidade e o poder. A discussão se esses bancos de dados merecem ser chamados de relacional ficou esgotada com o tempo, com a evolução dos bancos existentes. Os bancos de dados hoje implementam o modelo definido como objeto-relacional.

Aplicações de bancos de dados

Sistemas Gerenciadores de Bancos de dados são usados em muitas aplicações, enquanto atravessando virtualmente a gama inteira de software de computador. Os Sistemas Gerenciadores de Bancos de dados são o método preferido de armazenamento/recuperação de dados/informações para aplicações multi-usuárias grandes onde a coordenação entre muitos usuários é necessária. Até mesmo usuários individuais os acham conveniente, entretanto, muitos programas de correio eletrônico e organizadores pessoais estão baseados em tecnologia de banco de dados *standard*.

Aplicativo de Banco de Dados

Um **Aplicativo de Banco de dados** é um tipo de software exclusivo para gerenciar um banco de dados. Aplicativos de banco de dados abrangem uma vasta variedade de necessidades e objectivos, de pequenas ferramentas como uma agenda, até complexos sistemas empresariais para desempenhar tarefas como a contabilidade.

O termo "Aplicativo de Banco de dados" usualmente se refere a softwares que oferecem uma interface para o banco de dados. O software que gerencia os dados é geralmente chamado de sistema gerenciador de banco de dados (SGBD) ou (se for embarcado) de "database engine".

Exemplos de aplicativos de banco de dados são Microsoft Visual FoxPro, Microsoft Access, dBASE, FileMaker, (em certa medida) HyperCard, MySQL,Intpró, PostgreSQL, Firebird, Microsoft SQL Server, Oracle, Informix, DB2, Caché e Sybase.

Em Março, 2004, AMR Research (como citado em um artigo da CNET News.com listado na secção de "Referências") previu que aplicações de banco de dados de código aberto seriam amplamente aceitas em 2006.

Transação

É um conjunto de procedimentos que é executado num banco de dados, que para o usuário é visto como uma única ação.

A integridade de uma transação depende de 4 propriedades, conhecidas como **ACID**.

- **Atomicidade**
  - Todas as ações que compõem a unidade de trabalho da transação devem ser concluídas com sucesso, para que seja efetivada. Qualquer ação que constitui falha na unidade de trabalho, a transação deve ser desfeita (rollback). Quando todas as ações são efetuadas com sucesso, a transação pode ser efetivada (commit).
- **Consistência**
  - Nenhuma operação do banco de dados de uma transação pode ser parcial.O status de uma transação deve ser implementado na íntegra. Por exemplo, um pagamento de conta não pode ser efetivado se o processo que debita o valor da conta corrente do usuário não for efetivado antes, nem vice-versa.
- **Isolamento**
  - Cada transação funciona completamente à parte de outras estações. Todas as operações são parte de uma transação única. O principio é que nenhuma outra transação, operando no mesmo sistema, pode interferir no funcionamento da transação corrente(é um mecanismo de controle). Outras transações não podem visualizar os resultados parciais das operações de uma transação em andamento.
- **Durabilidade**
  - Significa que os resultados de uma transação são permanentes e podem ser desfeitos somente por uma transação subseqüente.Por exemplo: todos os dados e status relativos a uma transação devem ser armazenados num repositório permanente, não sendo passíveis de falha por uma falha de hardware.

Na prática, alguns SGBDs relaxam na implementação destas propriedades buscando desempenho.

Controle de concorrência é um método usado para garantir que as transações sejam executadas de uma forma segura e sigam as regras ACID. Os SGBD devem ser capazes de assegurar que nenhuma ação de transações completadas com sucesso (*committed transactions*) seja perdida ao desfazer transações abortadas (*rollback*).

Uma transação é uma unidade que preserva consistência. Requeremos, portanto, que qualquer escalonamento produzido ao se processar um conjunto de transações concorrentemente seja computacionalmente equivalente a um escalonamento produzindo executando essas transações serialmente em alguma ordem. Diz-se que um sistema que garante esta propriedade assegura a seriabilidade.

Segurança em banco de dados

Os bancos de dados são utilizados para armazenar diversos tipos de informações, desde dados sobre uma conta de e-mail até dados importantes da Receita Federal.

Para tal existem diversos tipos, os quais variam em complexidade e sobretudo em segurança.

- Criptografia
- Senhas
- Backup

Tabelas

- Nos modelos de bases de dados relacionais, a **tabela** é um conjunto de dados dispostos em número finito de colunas e número ilimitado de linhas (ou tuplos).
- As colunas são tipicamente consideradas os *campos* da tabela, e caracterizam os tipos de dados que deverão constar na tabela (numéricos, alfa-numéricos, datas, coordenadas, etc). O número de linhas pode ser interpretado como o número de combinações de valores dos campos da tabela, e pode conter linhas idênticas, dependendo do objectivo. A forma de referenciar inequivocamente uma única linha é através da utilização de uma chave primária.
- Para além do tipo de dados inerente a todas as colunas de uma tabela, algumas podem ter associadas restrições: a unicidade (SQL: `UNIQUE`), proibição de valores NULL (SQL: `NOT NULL`), delimitação de valores, etc. Estas restrições impedem que sejam inseridos valores não desejados que comprometam a validade e integridade dos dados.
- O número de tuplos de uma tabela é virtualmente ilimitado, o que torna as pesquisas por valor potencialmente muito lentas. Para permitir agilizar estas consultas, podem ser associados índices à tabela, que são estruturas de dados independentes da forma e ordem como estão armazenados os dados, embora tenham relação directa com os mesmos. Como consequência, a cada alteração de dados, irá corresponder uma (ou mais) alterações em cada um dos índices, aumentando o esforço necessário ao sistema gestor de base de dados (SGBD) para gerir essa alteração, motivo pelo qual os índices não existam naturalmente para cada coluna. A estrutura usada para a elaboração do índice depende do SGBD e do tipo de dados das colunas usadas no índice: árvore B, árvore R, etc.
- Não obstante o papel principal da tabela ser a de armazenamento de dados, é também utilizada como representação de relações, tipicamente de N para M. Nesse caso específico, essa tabela irá dispor obrigatoriamente de duas relações 1 para N — uma para a tabela N e outra para a tabela M — e, eventualmente, de atributos específicos à relação. Como consequência desta característica, este tipo de tabela nunca poderá conter linhas duplicadas.
- Um outro tipo de tabela especial — por não fazer armazenamento de dados — é a vista, cujas linhas são determinadas dinamicamente através de uma *query* (consulta) de tabelas reais (que armazenam os dados).

Regras

Regras de Banco de Dados são normas estipuladas pelos usuários ou pelos desenvolvedores de sistemas que definem ou restringem o tratamento dos dados armazenados.

Categorias

- Regras de Negócio - Relacionadas a procedimentos e verificações que tem a ver com o conteúdo.Definem como devem ser relacionadas entre si as informações e podem incluir procedimentos automáticos, programados internamente (Gatilhos ou *triggers*) para verificação e validação dos dados de acordo com a necessidade do usuário. Nos bancos de dados relacionais, podem ser criados procedimentos armazenados ou (Stored Procedures) internos que definem o que deve ocorrer quando se insere ou edita-se informação dentro de uma tabela interna.

- Regras de normalização de dados - Relacionadas com a estrutura ou modo como as informações são armazenadas. Incluem princípios ou 'normas formais' que contribuem para a qualidade e melhor performance no uso dos dados.

- Regras de Segurança - Especificam procedimentos e direitos de utilização das informações armazenadas. Incluem definições de usuários e permissões de acesso, edição, alteração e exclusão das informações.

- Procedimentos armazenados (mais conhecidos como stored procedures)

Procedimento armazenado ou **Stored Procedure** é uma coleção de comandos em SQL para dispensamento de Banco de dados. Encapsula tarefas repetitivas, aceita parâmetros de entrada e retorna um valor de status (para indicar aceitação ou falha na execução). O procedimento armazenado pode reduzir o tráfego na rede, melhorar a performance, criar mecanismos de segurança, etc.

Exemplo: (MS-SQL Server)

```
Create  procedure busca
@nomedebusca varchar (50)
as
select nome1, nome2
from nome_da_tabela
where nome = @nomedebusca
```

Características dos Procedimentos no MS-SQL Server

- Procedimentos do Sistema - Armazenadas no banco de dados Master, são identificadas com o prefixo sp_ , executam tarefas administrativas, podem ser executadas em qualquer banco de dados.

- Procedimentos Locais - São criadas em bancos de dados do usuário.

- Procedimentos Temporárias - **Locais** devem começar com #. **Globais** devem começar com ##.

- Procedures Remotas - Apenas por compatibilidade. No seu lugar se usa Queries distribuídas.

- Procedimentos Estendidas - São implementadas como .DLL e executadas fora do ambiente do SQL Server. Identificadas com o prefixo xp_.

Gatilho

- **Gatilho** ou ***trigger*** é um recurso de programação executado sempre que o evento associado ocorrer.
- É muito utilizada para ajudar a manter a consistência dos dados ou para propagar alterações em um determinado dado de uma tabela para outras. Um bom exemplo é um gatilho criado para controle de quem alterou a tabela, nesse caso, quando a alteração for efetuada, o gatilho é "disparado" e grava em uma tabela de histórico de alteração, o usuário e data/hora da alteração.
- **Exemplo: (MS-SQL Server)**

```
CREATE TRIGGER nome_do_gatilho ON dono.Nome_da_tabela
FOR INSERT (ou UPDATE ou DELETE)
AS
Codigo para execucao
```

Visão

- Uma visão, ou vista (em inglês: view), no contexto dos bancos de dados é uma relação que não armazena dados, composta dinamicamente por uma consulta que é previamente analisada e otimizada.
- Entre as principais utilidades estão, a depender do SGBD utilizado, o aumento de segurança por propiciar uma visão limitada e controlada dos dados que podem ser obtidos da base e a performance por utilizar uma consulta previamente otimizada, tornando desnecessário este processo de otimização quando for realizada.
- Fornece mecanismo de segurança, restringindo o acesso de usuários. Simplifica a interação entre usuário final e banco de dados.

Índice

O **Índice** é um arquivo auxiliar associado a uma Tabela. Sua função é acelerar o tempo de acesso às linhas de uma Tabela, cria ponteiros para os dados armazenados em colunas especificas. O Banco de dados usa o Índice de maneira semelhante ao índice remissivo de um livro, verifica um determinado assunto no Índice e depois localiza a sua posição em uma determinada página.

Principais tipos de índices

Índices compostos x Índices simples

- Índices Compostos: fazem referência a mais de uma coluna.
- Índices Simples: fazem referência a uma única coluna.

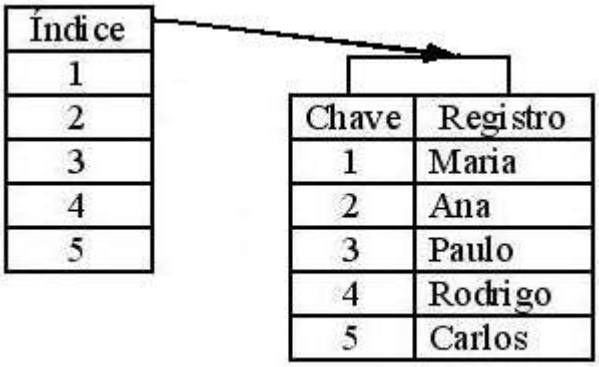

Fig1: Índice composto sobre tabela de clientes

Índices internos x Índices externos

- Índices internos: a chave está contida dentro da tabela.
- Índices externos: quando existe uma tabela de chaves separada que associa ponteiros à registros de uma tabela.

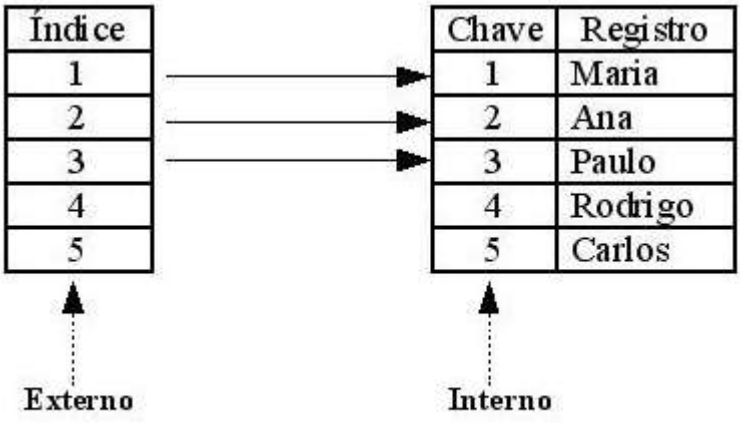

Fig2: Índices Interno e Externo

Índices primários x Chaves primárias

- Índice Primário: associado a uma chave primária (Primary Key) de um arquivo.
- Chave Primária: identificador único de um tabela, utilizado para distinguir um registro de outro.

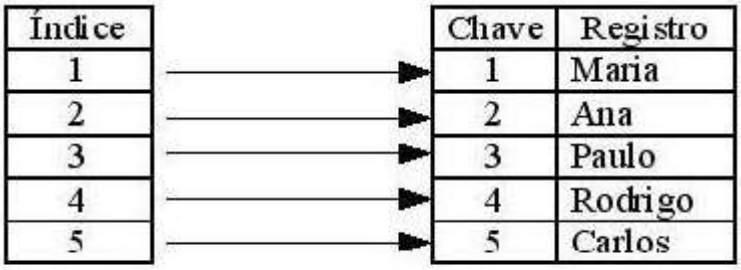

Fig3: Índice primário sobre chave primária

Lista de banco de dados

Esta **Lista de bancos de dados** mostra aplicativos que gerenciam banco de dados:

- **Ants**
- **Apache**
- **BlackfishSQL**
- **Caché**
- **Derby**
- **Dataflex**
- **DB2**
- **Firebird**
- **FileMaker**
- **HSQLDB**
- **H2**
- **Informix**
- **Ingres**
- **InterBase**
- **MaxDB**
- **MSDE**
- **Microsoft SQL Server**
- **MSAccess**
- **MySQL**
- **Oracle**
- **Paradox**
- **PostgreSQL**
- **SmallSQL**
- **SQLBase**
- **SQLite**
- **Sybase**
- **Virtuoso**

Banco de Dados Relacional:

Um **Banco de Dados Relacional** não é um conceito abstrato que define maneiras de armazenar, manipular e recuperar dados estruturados unicamente na forma de tabelas, construindo um banco de dados.

O termo é aplicádo aos próprios dados, quando organizados dessa forma, ou a um **Sistema Gerenciador de Banco de Dados Relacional (SGBDR)** – do inglês **Relational database management system (RDBMS)** – um programa de computador que implementa a abstração.

Os **Bancos de dados relacionais** (BDR) surgiram em meados da década de 1970. Porém, apenas alguns anos mais tarde as empresas passaram a utilizá-los no lugar de arquivos planos (do inglês *flat file*), bancos de dados hierárquicos e em rede.

As 12 regras

Em 1985, Edgar Frank Codd, criador do modelo relacional, publicou um artigo onde definia 12 regras para que um Sistema Gerenciador de Banco de Dados (SGBD) fosse considerado relacional:

1. Regra Fundamental:
   - Um SGBD relacional deve gerenciar seus dados usando apenas suas capacidades relacionais
2. Regra da informação:
   - Toda informação deve ser representada de uma única forma, como dados em uma tabela
3. Regra da garantia de acesso:
   - Todo dado (valor atômico) pode ser acessado logicamente (e unicamente) usando o nome da tabela, o valor da chave primária da linha e o nome da coluna.
4. Tratamento sistemático de valores nulos:
   - Os valores nulos (diferente do zero, da string vazia, da string de caracteres em brancos e outros valores não nulos) existem para representar dados não existentes de forma sistemática e independente do tipo de dado.
5. Catálogo dinâmico on-line baseado no modelo relacional:
   - A descrição do banco de dados é representada no nível lógico como dados ordinários (isso é, em tabelas), permitindo que usuários autorizados apliquem as mesmas formas de manipular dados aplicada aos dados comuns ao consultá-las.
6. Regra da sub-linguagem compreensiva:
   - Um sistema relacional pode suportar várias linguagens e formas de uso, porém deve possuir ao menos uma linguagem com sintaxe bem definida e expressa por cadeia de caracteres e com habilidade de apoiar a definição de dados, a definição de visões, a manipulação de dados, as restrições de integridade, a autorização e a fronteira de transações.
7. Regra da atualização de visões:
   - Toda visão que for teoricamente atualizável será também atualizável pelo sistema.
8. Inserção, atualização e eliminação de alto nível:
   - A capacidade de manipular a relação base ou relações derivadas como um operador único não se aplica apenas a recuperação de dados, mas também a inserção, alteração e eliminação de dados.

9. Independência dos dados físicos:
    - Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as modificações na representação de armazenagem ou métodos de acesso internos.
    - Independência lógica de dados
    - Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as mudanças de informação que permitam teoricamente a não alteração das tabelas base.
10. Independência de integridade:
    - As relações de integridade específicas de um banco de dados relacional devem ser definidas em uma sub-linguagem de dados e armazenadas no catálogo (e não em programas).
11. Independência de distribuição:
    - A linguagem de manipulação de dados deve possibilitar que as aplicações permaneçam inalteradas estejam os dados centralizados ou distribuídos fisicamente.
12. Regra da Não-subversão:
    - Se o sistema relacional possui uma linguagem de baixo nível (um registro por vez), não deve ser possível subverter ou ignorar as regras de integridade e restrições definidas no alto nível (muitos registros por vez).

Por que usar um Banco de Dados Relacional?

Os Bancos de Dados Relacionais foram desenvolvidos para prover acesso facilitado aos dados, possibilitando que os usuários utilizassem uma grande variedade de abordagens no tratamento das informações. Pois, enquanto em um banco de dados hierárquico os usuários precisam definir as questões de negócios de maneira específica, iniciando pela raiz do mesmo, nos Bancos de Dados Relacionais os usuários podem fazer perguntas relacionadas aos negócios através de vários pontos.

A linguagem padrão dos Bancos de Dados Relacionais é a Structured Query Language, ou simplesmente **SQL**(siqüôl), como é mais conhecida.

O Modelo Relacional

Um Banco de Dados Relacional segue o Modelo Relacional.

A arquitetura de um banco de dados relacional pode ser descrita de maneira informal ou formal. Na descrição informal estamos preocupados com aspectos práticos da utilização e usamos os termos tabela, linha e coluna. Na descrição formal estamos preocupados com a semântica formal do modelo e usamos termos como relação(tabela), tupla(linhas) e atributo(coluna).

Tabelas (ou relações, ou entidades)

Todos os dados de um banco de dados relacional (BDR) são armazenados em tabelas. Uma tabela é uma simples estrutura de linhas e colunas. Em uma tabela, cada linha contém um mesmo conjunto de colunas. Em um banco de dados podem existir uma ou centenas de tabelas, sendo que o limite pode ser imposto tanto pela ferramenta de software utilizada, quanto pelos recursos de hardware disponíveis no equipamento.

As tabelas associam-se entre si através de regras de relacionamentos, estas regras consistem em associar um ou vários atributo de uma tabela com um ou vários atributos de outra tabela.

- Exemplo: A tabela *funcionário* relaciona-se com a tabela *cargo*. Através deste relacionamento esta última tabela fornece a lista de cargos para a tabela *funcionário*.

Modelo teórico usado para representar conceitualmente um BD, Idealizado por Codd (1970). Baseado numa estrutura de dados simples chamada relação. É o modelo mais amplamente usado, principalmente em aplicações convencionais de BD.

Registros (ou tuplas)

Cada linha formada por uma lista ordenada de colunas representa um **registro**, ou **tupla**. Os registros não precisam conter informações em todas as colunas, podendo assumir valores nulos quando assim se fizer necessário.

Resumidamente, um registro é uma instância de uma tabela, ou entidade.

- Exemplo: O empregado *Pedro* é uma instância (registro) da tabela *funcionário*, e a função *Analista Comercial* é a instância (registro) da tabela *cargo*. Uma associação entre estas duas tabelas criaria a seguinte instância de relacionamento: *Pedro* ***é*** *Analista Comercial*, onde o verbo **ser** representa uma ligação entre os registros distintos.

Colunas (ou atributos)

As colunas de uma tabela são também chamadas de **Atributos**. Ao conjunto de valores que um atributo pode assumir chama-se *domínio*. Por exemplo: em um campo do tipo numérico, serão somente armazenados números.

O conceito mais similar a domínio é o de Tipo Abstrato de Dados em linguagens de programação, ou seja são meta-dados (dados acerca de dados).

Chave

As tabelas relacionam-se umas as outras através de **chaves**. Uma chave é um conjunto de um ou mais atributos que determinam a unicidade de cada registro.

Por exemplo, se um banco de dados tem como chaves *Código do Produto* e *ID Sistema*, sempre que acontecer uma inserção de dados o sistema de gerenciamento de banco de dados irá fazer uma consulta para identificar se o registro já não se encontra gravado na tabela. Neste caso, um novo registro não será criado, resultando esta operação apenas da alteração do registro existente.

A unicidade dos registros, determinada por sua chave, também é fundamental para a criação dos índices.

Temos dois tipos de chaves:

Chave primária: (*PK - Primary Key*) é a chave que identifica cada registro dando-lhe unicidade. A chave primária nunca se repetirá.
Chave Estrangeira: (*FK - Foreign Key*) é a chave formada através de um relacionamento com a chave primária de outra tabela. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes. Caso a chave primária seja composta na origem, a chave estrangeira também o será.

Relacionamentos

Com o advento do Modelo de Entidades e Relacionamentos foi causada uma confusão entre os termos **relação** e **relacionamento**

O Modelo Relacional, quando descrito de forma matemática, é definido como um modelo formado por relações (no sentido matemático) entre os domínios. Cada tupla é um elemento do conjunto relação. Ou seja, a relação é a tabela.

Um relacionamento do Modelo de Entidades e Relacionamentos é uma associação entre entidades distintas. Não há relação direta entre o nome *relacionamento* e o nome *relação*.

Porém, um relacionamento, do Modelo de Entidades e Relacionamentos é traduzido para a criação de atributos com chaves externas do Modelo Relacional. Esta tradução é feita ligando-se um campo de uma tabela X com um campo de uma tabela Y, por meio da inclusão do campo chave da tabela Y como um campo (conhecido como chave estrangeira) da tabela X.

Por meio das chaves estrangeiras, é possível implementar restrições nos SGBDR.
Existem alguns tipos de relacionamentos possíveis no MER:

- Um para um (1 para 1) - indica que as tabelas têm relação unívoca entre si. Você escolhe qual tabela vai receber a chave estrangeira;
- Um para muitos (1 para N) - a chave primária da tabela que tem o lado 1 vai para a tabela do lado N. No lado N ela é chamada de chave estrangeira;
- Muitos para muitos (N para N) - quando tabelas têm entre si relação n..n, é necessário criar uma nova tabela com as chaves primárias das tabelas envolvidas, ficando assim uma chave composta, ou seja, formada por diversos campos-chave de outras tabelas. A relação então se reduz para uma relação 1..n, sendo que o lado n ficará com a nova tabela criada.

Os relacionamentos *1 para 1* e *1 para N* podem ser mapeados diretamente em chaves estrangeiras nas tabelas originais. Já o relacionamento *N para N* exige o uso de uma tabela auxiliar. No relacionamento N:N não há chave estrangeira.

Modelagem

Normalização

Os bancos de dados relacionais utilizam a normalização de dados para evitar redundâncias e possibilitar um maior desempenho nas pesquisas.

Normalização

É o processo de organização eficiente dos dados dentro de um banco de dados cujos objetivos principais são:

1. Eliminar dados redundantes (por exemplo, armazenando os mesmos dados em mais de uma tabela).
2. Garantir que as dependências entre os dados façam sentido (armazenando apenas dados logicamente relacionados em uma tabela).

Existem cinco estágios de normalização, 1º, o 2º, o 3º, o 4º e o 5º. Para um banco de dados se encontrar em cada um desses estágios ou formas (denominadas formas normais), cada uma de suas tabelas deve atender a alguns pré-requisitos. Os pré-requisitos são cumulativos, isto é, para alcançar a 3ª forma normal (3NF), um banco de dados precisa atender aos pré-requisitos das 1ª e 2ª formas normais, acrescidos dos requisitos exclusivos da 3NF.

Dependência Funcional

Um atributo B possui uma dependência funcional do atributo A se, para cada valor do atributo A, existe exatamente um único valor do atributo B. A dependência funcional é representada por A → B. Exemplo: Observe os conjuntos: CPF Nome 1 - José 2 - João 3 - Rui 4 - Manoel

Observe que existe um dependência entre os valores dos conjuntos, ou seja, nome é função do CPF, se eu estiver com numero do CPF, poderei encontrar o nome da pessoa correspondente.

Essa dependência é expressa por:

```
CPF → Nome
```

Leia da seguinte maneira: - com um número de CPF posso encontrar nome da pessoa, ou - nome depende da funcionalidade do CPF.

Primeira Forma Normal (FN1)

Uma relação está na primeira forma normal se os valores de seus atributos são atômicos (simples, indivisíveis) e monovalorados. Em outras palavras, FN1 não permite “relações dentro de relações” ou “relações como atributos de tuplas”.

Uma tabela está na primeira forma normal quando seus atributos não contêm grupos de repetição. Exemplo:

Cliente		
ClienteID	**Nome**	**Telefone**
123	Rachel Ingram	555-861-2025
456	James Wright	555-403-1659 555-776-4100
789	Maria Fernandez	555-808-9633

Esta tabela não está na primeira forma normal porque apresenta grupos de repetição (possibilidade de mais de um telefone por cliente).

Segunda Forma Normal (FN2)

Uma relação está na FN2 quando duas condições são satisfeitas: 1 - A relação está na 1FN; 2 - Todo atributo da tabela seja dependente funcional da chave completa e não de parte da chave. Todo os atributos não-chave dependem funcionalmente de toda a chave primária.

Uma tabela na 1FN e com uma chave formada por apenas um atributo está 3 contudo código do pedido, isso resumo um pedido chave primaria ex:

Terceira Forma Normal (FN3)

A FN3 exige que não existam atributos transitivamente dependentes da chave.

Um exemplo de uma tabela FN2 que não atende o critério para FN3 é:

Vencedores de Torneios			
Torneio	**Ano**	**Vencedor**	**Data de nascimento do vencedor**
Indiana Invitational	1998	Al Fredrickson	21/7/1975
Cleveland Open	1999	Bob Albertson	28/9/1968
Des Moines Masters	1999	Al Fredrickson	21/7/1975
Indiana Invitational	1999	Chip Masterson	14/3/1977

A chave primária composta é {Torneio, Ano}.

A tabela não está na terceira forma normal porque o atributo *"data de nascimento do vencedor"* é dependente transitivamente de {Torneio, Ano} via o atributo "*Vencedor*". O fato de a *data de nascimento do vencedor* não ser dependente do *vencedor* torna a tabela vulnerável a inconsistências lógicas, já que nada impediria de uma mesma pessoa aparecer com datas de nascimento diferentes em dois registros.

Para atender a terceira forma normal, a mesma informação deveria ser dividida em duas tabelas:

Vencedores de Torneios

Torneio	**Ano**	**Vencedor**
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Datas de nascimento de jogadores

Jogador	**Data de nascimento**
Chip Masterson	14/3/1977
Al Fredrickson	21/7/1975
Bob Albertson	28/9/1968

As duas tabelas acima estão na terceira forma normal.