Dados, Informações e conhecimento

Um **dado** é a menor unidade de informação que pensamos no contexto de banco de dados. É um valor que pode ser interpretado dentro de um contexto para obter informações
Informação é uma constatação derivada de um conjunto de dados
Conhecimento: Conjunto de informações logicamente estruturadas. É o resultado de um processo de assimilação e análise crítica da informação.

A criação dos SGBDs

Dados, informações e conhecimento são de altíssimo valor para qualquer empresa. A questão de como lidar com dados e suas derivações incentivou o desenvolvimento de vários sistemas com abordagens diferentes pra alcançar esses objetivos

Um sistema é um conjunto de programas interagindo para chegar a um objetivo, os chamados **SGBD** (Sistema de gerenciamento de banco de dados) surgem com o objetivo de resolver alguns problemas específicos do

armazenamento, leitura e até processamento de dados entre eles, os principais são:

Redundância e inconsistência
Dificuldade de acesso dos dados
Isolamento de dados
problemas de segurança

O que é um SGBD

Um SGBD é formado por uma coleção de dados e um conjunto de programas para acessar adequadamente esses dados. A coleção de dados é comumente conhecida como banco de dados.

A organização de um SGBD relacional

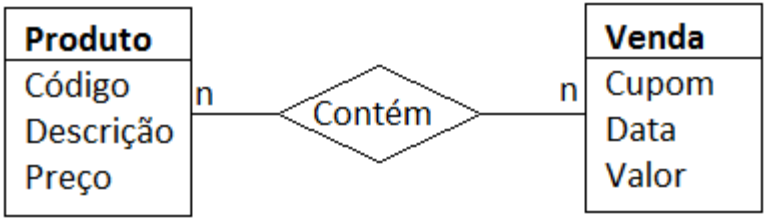
dentro dos bancos de dados temos o que chamamos de schema, o schema é uma definição lógica de que tipo de dados este banco de dados terá

Dentro do modelo relacional um schema é formado por uma série de relações (tabelas) que representam as entidades do sistema
Cada relação é um conjunto de tuplas (linhas) as tuplas são listas de valores

agrupados com um nome de atributo
Cada atributo é um agrupamento de nome e domínio (o conjunto de possíveis valores que o atributo pode assumir)
Um conjunto de tuplas que tem os mesmos atributos é agrupado em uma relação
Tuplas representam fatos e não são ordenadas, por isso elas precisam de um atributo identificador. O principal atributo identificador de uma tabela é chamado de chave primária ou primary key

Schemas na teoria

Entidades também podem ter relações entre si. É comum usar diagramas que representem essas relações para garantir que todos tem o mesmo entendimento sobre o que deve ser programado. Esses diagramas são chamados de Modelo de entidade relacionamento (M.E.R.).



Schemas na prática

SQL

"A medida que os sistemas de banco de dados se tornavam mais sofisticados linguagens melhores eram desenvolvidas para os programadores interagirem com os dados"

Com o tempo uma linguagem passou a dominar o mercado de banco de dados relacional (e mantém esse domínio até hoje)
Essa é a linguagem estruturada de queries, mais conhecida como SQL (Structured query language)

Podemos dividir SQL em alguns subgrupos da linguagem separadas por seus objetivos, mas no momento vamos focar em 2 desses grupos
DDL (Data Definition Language) e DML (Data Manipulation Language)
DDL é a linguagem usada para construir schemas, definimos quais são as tabelas que pertencem à esse schema, que atributos elas tem, qual o domínio de cada atributo, etc.
Já DML é usado para manipular as tuplas que existem nessas tabelas, para inserir, editar e deletar registros

Constraints

Outro problema que os SGBDs buscam resolver são problemas de integridade, esses problemas levam o banco de dados à um estado que não deveria ser permitido. Para evitar esses problemas podemos aplicar **Constraint** (restrições) nos atributos dos nossos bancos de dados.

Algumas constraints normalmente usadas são
Não nulo: Para a tupla ser válida o valor desse atributo deve ser preenchido
Único: Para a tupla ser valida, nenhuma outra tupla pode ter o mesmo valor para o mesmo atributo
Domínio: Restringe o domínio do atributo

Atributos que são não nulos e únicos são consideradas chaves candidatas, já que conseguem identificar uma tupla. Dentre as chaves candidatas uma deve ser escolhida para ser a chave primária, o identificador principal da tupla

Outra restrição comum é a restrição de integridade referencial.
Quando queremos expressar uma relação entre duas tabelas podemos usar um atributo em uma delas que contenha uma chave candidata (normalmente a chave primária) da outra

Por exemplo, para dizermos que uma tupla da entidade carro pertence a uma tupla da entidade pessoa podemos adicionar um atributo CPF_do_dono ao carro. Isso nos permitirá identificar o dono do carro quando necessário

A integridade referencial diz que pra todo carro, o cpf do dono deve ser um cpf real que identifique uma tupla na tabela pessoa.

Cardinalidade

Essas relações podem acontecer entre números diferentes de tuplas. No nosso último exemplo, uma pessoa pode ter mais de um carro, mas um carro só pode ter um dono. O número de tuplas de cada entidade na relação é o que chamamos de cardinalidade

Existem 3 tipos de cardinalidade, a mais simples dela sendo 1:1 (um pra um)
Nessa cardinalidade duas tuplas de tabelas diferentes se ligam entre si e somente entre si. Exemplo: Uma pessoa tem um passaporte e um passaporte pertence a uma pessoa

Outra cardinalidade comum é 1:n (um pra muitos). Nessa cardinalidade cada tupla da primeira tabela se liga com muitas da segunda mas cada tupla da segunda só se liga a uma tupla da primeira. O exemplo do carro se encaixa nessa cardinalidade. Cada carro tem um dono, cada pessoa pode ter mais de um carro

O último tipo de cardinalidade é n:n (muitos pra muitos). Nessa cardinalidade, varias tuplas de cada tabela se relacionam com vária tuplas da outra. Um exemplo é como cada aluno tem várias disciplinas e cada disciplina tem vários alunos

Acesso concorrente

Para melhorar a performance dos SGBDs, uma evolução necessária foi o suporte para acesso concorrente, diferentes usuários acessando os mesmos dados ao mesmo tempo. Esse tipo de acesso trás consigo seus próprios problemas

Problemas comuns

Anomalias de acesso concorrente
Caso um usuário modifique o valor que outro está lendo, esse valor pode se tornar rapidamente inválido. Por exemplo, se eu faço uma compra no cartão e ao mesmo tempo um serviço de assinatura cobra sua mensalidade pode ser que ambos os pagamentos sejam aprovados mesmo que eu não tenha crédito para os dois juntos

problemas de atomicidade
Outro problema comum é a quebra de atomicidade. Atomicidade é a ideia de que cada operação deve ser atômica, indivisível. Só deve existir um estado antes e um depois, nunca um estado diferente durante a operação
Num exemplo prático, caso estejamos registrando um cliente num sistema em que ele obrigatoriamente deve ter um endereço a tupla na tabela cliente e a tupla na tabela endereço devem ser registradas ao mesmo tempo. Nenhum outro usuário deveria ser capaz de ver apenas uma das duas sem conseguir ver a outra. E caso o sistema falhe ele deve falhar para as duas, não criando um registro novo em nenhuma tabela

Transações

Para resolver esses problemas foi desenvolvido o conceito de transação. Uma transação é um conjunto de operações que acontecem juntas, de forma atômica. No nível mais puro de transações o sistema se comportaria como se você fosse o único usuário presente.

Como dito antes uma transação é atômica, indivisível, isso significa que todas as operações tem sucesso juntas, ou todas falham juntas
Quando as operações falham, ocorre o processo de rollback, o sistema volta ao estado que ele estaria caso a transação nunca tivesse começado
Quando as operações tem sucesso, ocorre o commit, o estado oficial do sistema passa a ser o declarado pela transação e as mudanças feitas por ela não podem mais ser facilmente desfeitas

Stored procedures e triggers

Nem todos os problemas das bases de dados podem ser resolvidos com as operações descritas até agora. Para lidar com problemas mais específicos existem procedimentos armazenados (Stored Procedures) conjuntos arbitrários de código que devem ser executados arbitrariamente ou como reação à um evento.
O evento que dispara uma stored procedure é chamado de gatilho ou trigger

Os gatilhos mais comuns são
On Select
Before Insert
After Insert
Before Update
After Update
Before Delete
After Delete

História dos SGBDs

década de 50 e início de 60

Carões perfurados e fitas

final de 60 e década de 70

Discos rígidos
Desenvolvimento de modelos hierárquicos e de rede
Codd escreve o documento que dá origem ao modelo relacional

final de 70 e década de 80

Surgem os primeiros sgbds relacionais comerciais

década de 90

Com a popularização da web bancos de dados tiveram que se adaptar para o desenvolvimento de aplicações, não só de queries analiticas

década de 2000

Inicio do NoSQL e big data

década de 2010

Popularização de NoSQL e bases de dados terceirizadas