Introdução a Banco de Dados

Histórico de Bancos de Dados

Prof. Leandro Batista de Almeida

2023

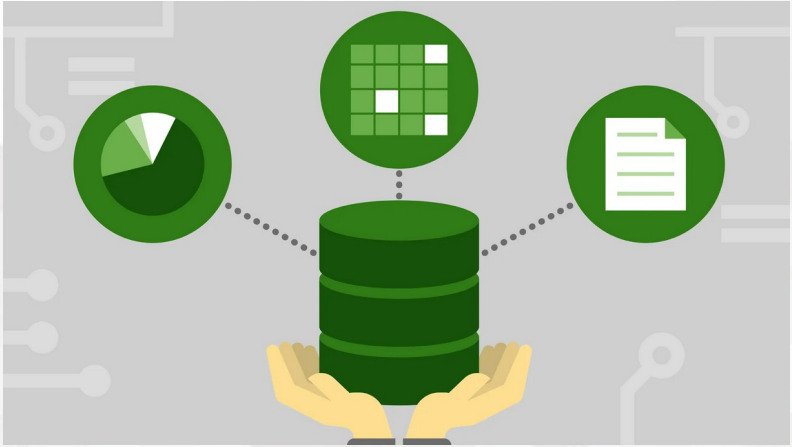
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Bancos de Dados

Databases

Coleção organizada de dados inter-relacionados que modelam algum aspecto do mundo real

- E são tratados por um Sistema Gerenciador de Bancos de Dados (SGBD)



Bancos de Dados são componentes cruciais na maioria dos sistemas de computação

Porque sistemas de BD?

Necessidade de persistência de dados em aplicações

- De sistemas corporativos a aplicativos móveis e IoT

Dificuldade em se fazer isso de maneira proprietária e/ou específica

- Tratar arquivos com as APIs convencionais de sistemas e linguagens
- Otimização em ciclo de desenvolvimento

Tratamento de funções relacionadas

- Que de outra maneira tomariam um tempo excessiv
  - Segurança
  - Deployment
  - Estabilidade
  - Escalabilidade

Porque sistemas de BD?

Como fazer isso com arquivos convencionais?

- Como um CSV, por exemplo?
  - Representando cada entidade como um CSV diferente?

Como realizar as leituras?

- Com filtros, ordenação, buscas otimizadas (índices), etc
- Combinando entidades diferentes no mesmo resultado?

Como garantir a integridade dos dados

- Evitar a repetição das mesmas entidades nos arquivos, por exemplo?

O que vai acontecer se a aplicação cair no meio de um processo?

Como organizar o acesso múltiplo (múltiplas threads ou múltiplos usuários) ao mesmo arquivo?

Necessidades

Persistência

Concorrência

Transações

Linguagem de acesso padronizada

- SQL

Integração de dados

Suporte da indústria

- Fornecedores, APIs, linguagens, frameworks, etc

Histórico

A história sempre se repete

Antigos problemas de sistemas de bancos de dados são ainda relevantes hoje

Disputas (reais ou não) também se repetem frequentemente
- RDBS x "todos os outros"
- OODBs x RDBS
- SQL x NoSQL
    - A cada 10 anos alguém tenta matar o SQL de novo...

Novos conceitos em sistemas de bancos de dados podem não ser assim tão novos...
- Por isso uma sólida pesquisa bibliográfica (e ferramental) é importante

Leitura recomendada:
- What goes around comes around
    - Michael Stonebraker & Joseph Hellerstein

RDBS
Relational Database System
OODB
Object Oriented Database System
SQL
Structured Query Language

Histórico

Até 1960s

- Sistemas de arquivos integrados
  - ISAM (Indexed Sequential Access Method)
  - VSAM (Virtual Storage Access Method)
    - Na realidade, VSAM é uma evolução do ISAM dos anos 1970s

1960s – IDS (Integrated Data Store)

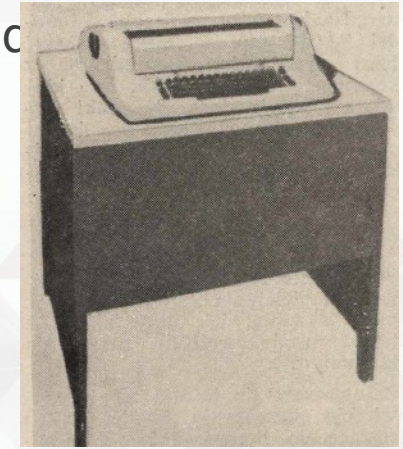
- Desenvolvido como parte de uma solução pela GE
  - Se percebeu como um produto relevante
- Vendido para a Honeywell em 1969
- Um dos primeiros DBMSs
  - Modelo em rede
    - Queries baseadas em uma tupla por vez
    - Tuple-at-a-time

Honeywell

Histórico

1966 – IBM IMS (Information Management System)

- Um dos primeiros sistemas de banco de dados no mercad
  - Desenvolvido para controlar as ordens de compra do programa Apollo
    - Modelo hierárquico de dados
      - Possível duplicação de dados
      - Sem independência de hardware ou software
    - Estrutura física de armazenamento definida em programação
    - Queries baseadas em uma tupla por vez
      - Tuple-at-a-time
  - Existe como produto da IBM até hoje



IBM IMS

Histórico

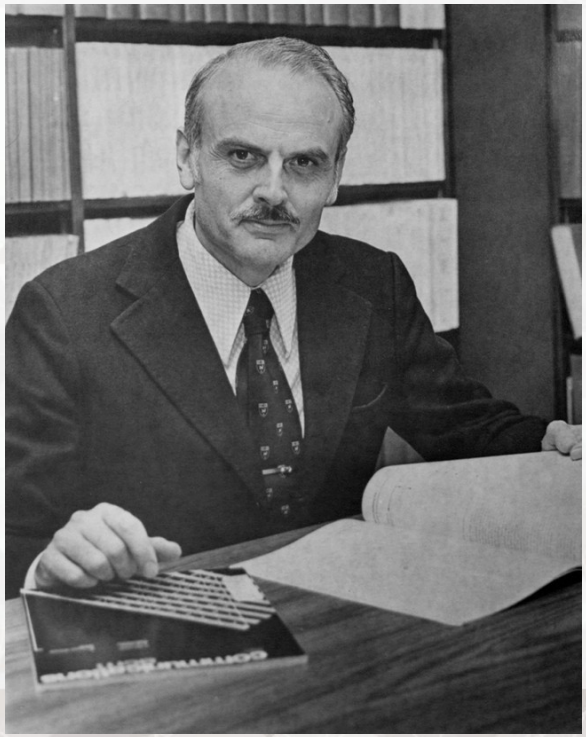
1970s – CODASYL

- Padrão para acesso a dados em linguagem COBOL
  - Como os programas devem acessar um banco de dados]
- Desenvolvimento liderado por Charles Bachman
  - Recebeu o Turing Award por isso
- Modelo de dados em rede
  - Queries complexas
  - Dados podem se corromper
- Queries tuple-at-a-time

Histórico

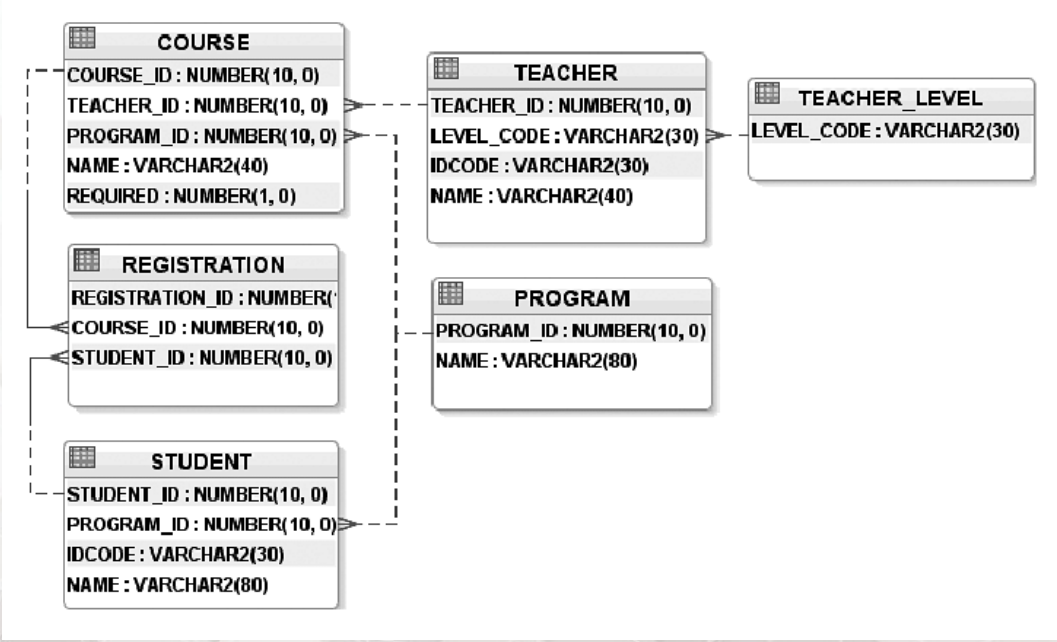
1970s – Modelo Relacional

- Concebido por Ted Codd
  - Matemático trabalhando para a IBM Research
  - Recebeu um Turing Award por isso
- Percebeu que os programadores perdiam muito tempo reescrevendo código IMS e Codasyl a cada vez que o schema ou layout do banco de dados mudava
  - Usou uma abstração de banco de dados para evitar isso
  - Armazenar o banco de dados em estruturas de dados simples
  - Acessar dados por uma linguagem de alto nível
  - Armazenamento físico é deixada para a implementação do serviço
    - Ao contrário da implementação do programa
- 12 regras de Codd
  - A relational model of data for large shared data banks
    - Communications of the ACM, 1970 e 1983



Histórico

1970s – Modelo Relacional



A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.*

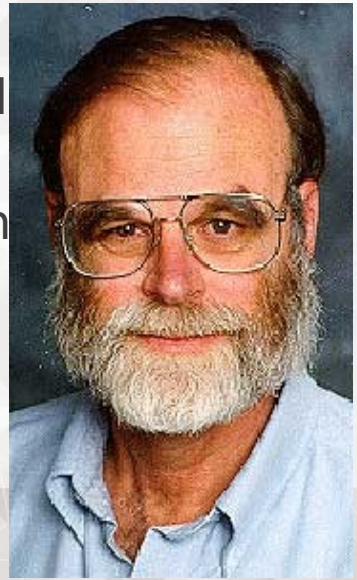
Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Histórico

1970s – Modelo Relacional

Implementações iniciais do SGBDs (Sistemas Gerenciadores de Bancos de Dados) relacionais

- System R – IBM Research
  - Jim Gray
- INGRES – U.C. Berkel
  - Michael Stonebraker
- Oracle – Larry Ellison



Histórico

1980s – Modelo Relacional

- O modelo relacional vence
  - IBM lança SQL/DS em 1981 e DB2 em 1983
- SQL (inicialmente SEQUEL) vence a disputa com QUEL
  - System-R e Oracle usavam SQL, Ingres usava QUEL
  - SQL passa a ser o padrão em consultas

Muitos novos SGBDs "corporativos" aparecem, mas Oracle vence a disputa de mercado

Stonebraker cria o Postgres (PostgreSQL)

- Ingres disputava mercado com Oracle, mas a mudança para SQL tomou 3 anos de desenvolvimento
  - Como código fonte do Ingres estava parcialmente disponível, outros produtos nascem a partir dele
    - Sybase é um deles

Histórico

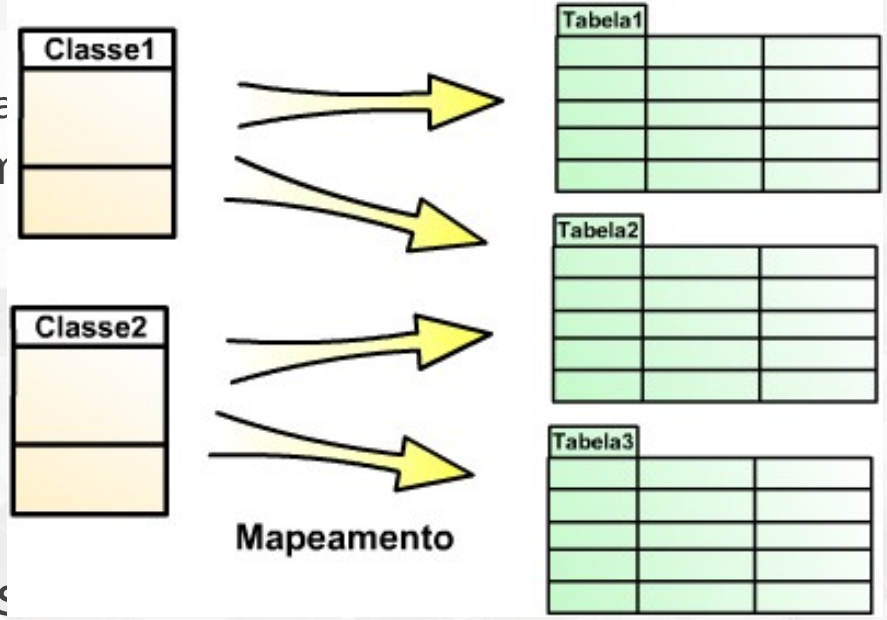
Bancos de dados relacionais

- Modelo relacional
- Persistência
- Concorrência
- Transações
- SQL (linguagem de query unificada)
  - Padrão ANSI
- Ferramenta para integração de dados
- Enorme suporte da indústria e academia
  - Profissionais treinados e experientes
  - Produtos maduros e testados

Histórico

1980s – Bancos de Dados Orientados a Objetos

- Divisão de dados de maneira tabular
  - Não é natural para linguagens e tecnologias orientada
- Evitam a "relational-object impedance mism
  - Acoplamento forte de objetos e o banco de dados
- Soluções
  - Object Oriented Database Systems
  - ORM (Object-Relational Mapping)



Poucos dos SGBDs OO originais ainda exis

- Parte da sua tecnologia passou a ser usada em outros modelos
  - XML, JSON

Histórico

1990s (boring days)

Sem maiores avanços em sistemas de bancos de dados ou workloads

- Microsoft faz um fork do Sybase e cria o SQL Server
- MySQL é escrito como um substituto do mSQL
- Postgres ganha suporte a SQL
- SQLite é criado no final da déca

Microsoft® SQL Server®

2000s – Internet Boom

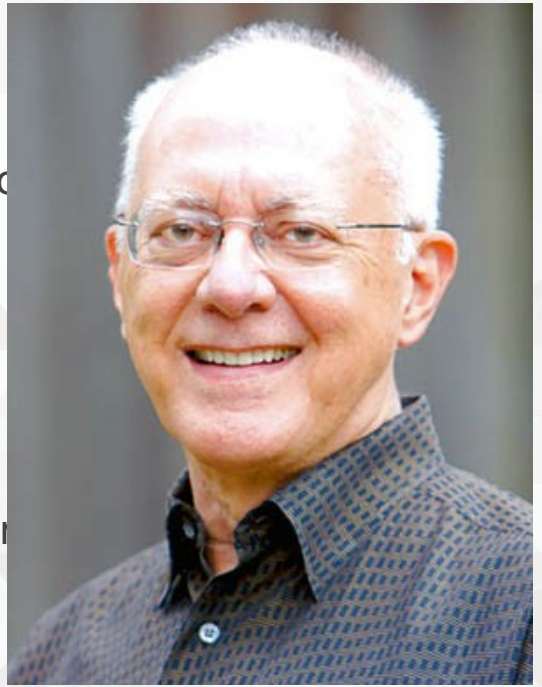
- Grandes players se tornam cada vez maiores (e mais caros)
- Bancos de dados open-source ainda não tinham todas as features necessárias
- Muitas empresas escreviam o próprio middleware para escalar bancos de dados em várias instâncias single-node

UTFPR UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ <lapti>

Histórico

2000s – Data Warehouses

- Data Warehouses, Bussines Intelligence, Data Mining
  - Dr. Ralph Kimball
- Surgimento de SGBDs especializados para OLAP (On Line Analytical Proc
  - Em oposição aos tradicionais OLTP (On Line Transaction Processing)
  - Distribuídos / shared-nothing
  - Relational / SQL
  - Geralmente proprietários
- Bancos de dados semi-estruturados
  - Bancos XML (final dos anos 90, na realidade)
- Melhoras significativas em desempenho pelo uso de Decomposition Sto Model
  - Bancos de dados colunares
    - 100 x mais rápido que row-based em queries para data warehouse



monetdb

VERTICA

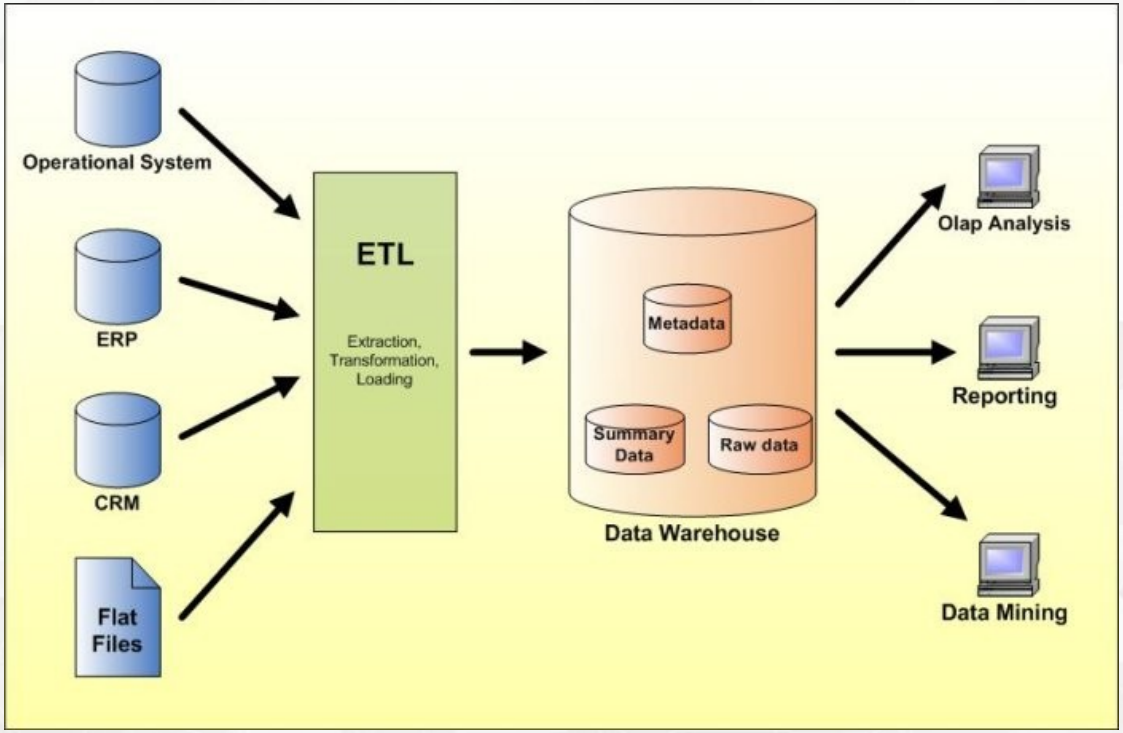
NETEZZA

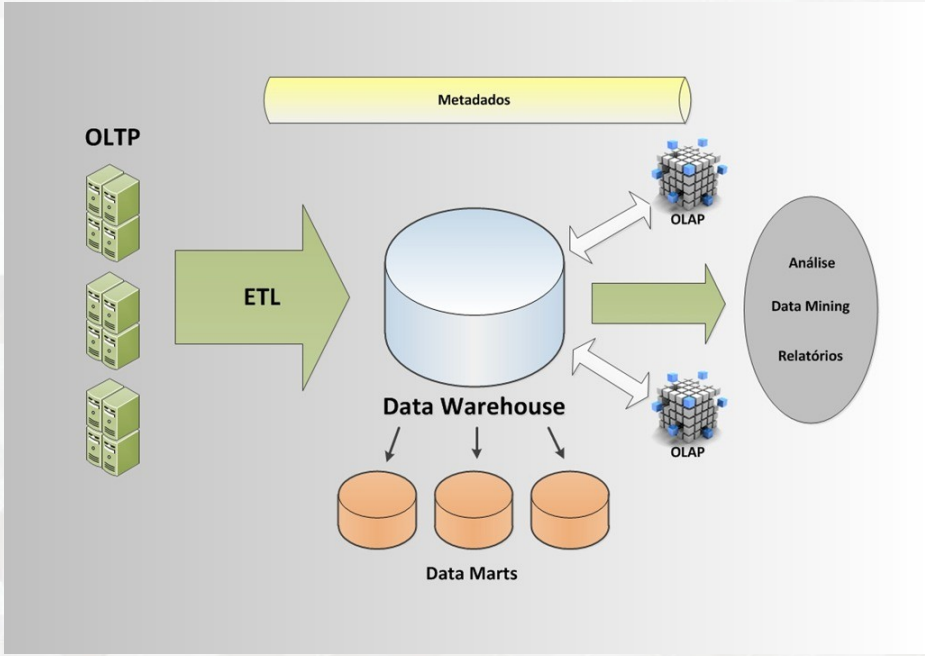
DATAllegro

UTFPR UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

<lapti>

Histórico

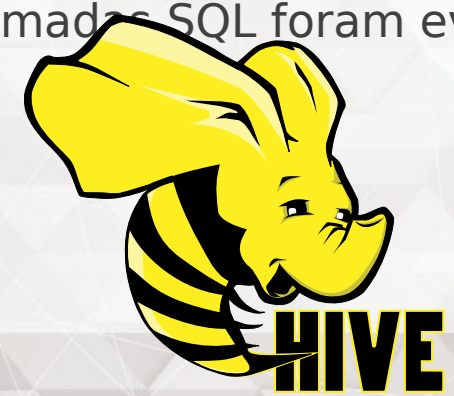

Operational System
ERP
CRM
Flat Files
ETL
Extraction, Transformation, Loading
Metadata
Summary Data
Raw data
Data Warehouse
Olap Analysis
Reporting
Data Mining


Metadados
OLTP
ETL
Data Warehouse
OLAP
OLAP
Análise
Data Mining
Relatórios
Data Marts

Histórico

2000s – Sistemas baseados em Map Reduce

- Big Data
- Proposto em papers pelo Google
- Implementado no Hadoop (Yahoo)
  - HDFS + YARN
- Funções (mapper, reducer, shuffle, etc) para cada etapa de processamento dos dados
- Camadas SQL foram eventualmente colocadas no topo

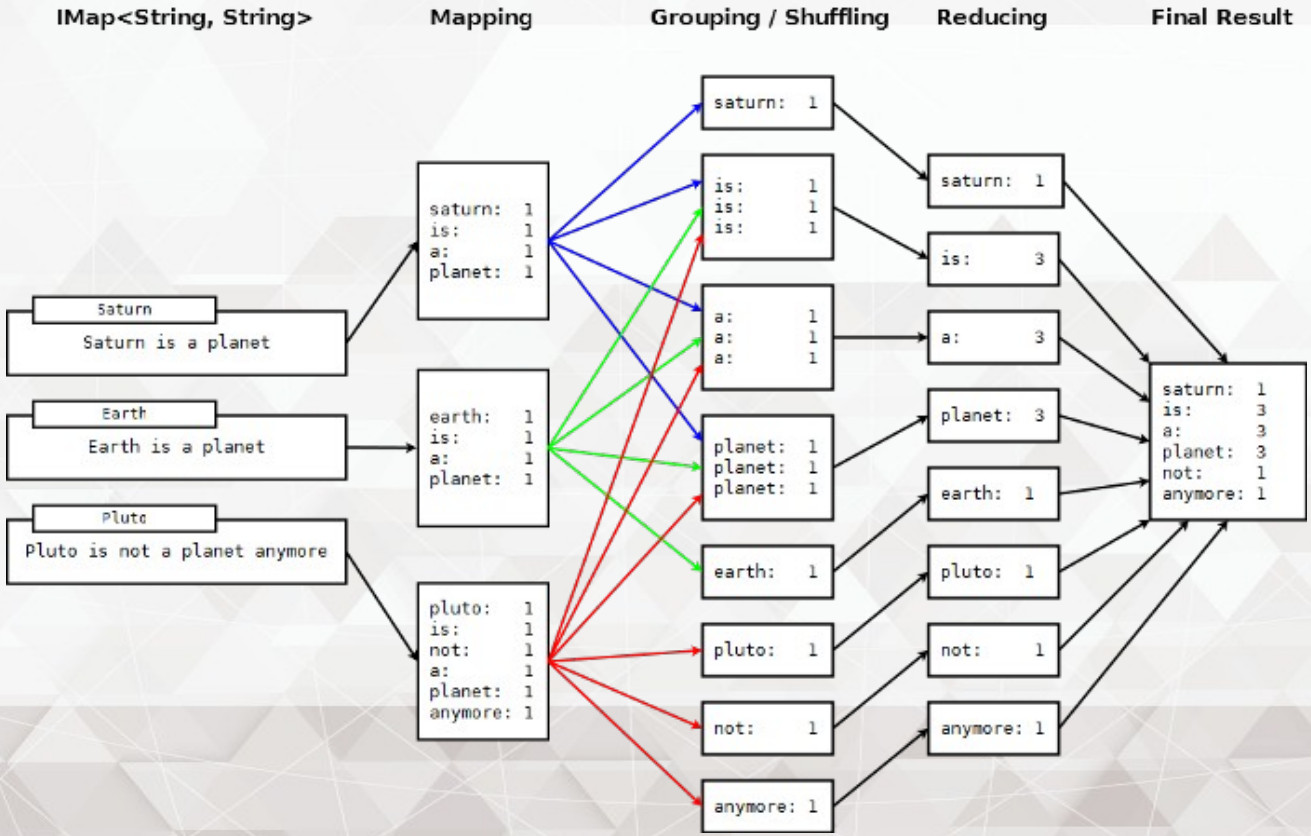


MAPR-DB

APACHE HBASE

Histórico

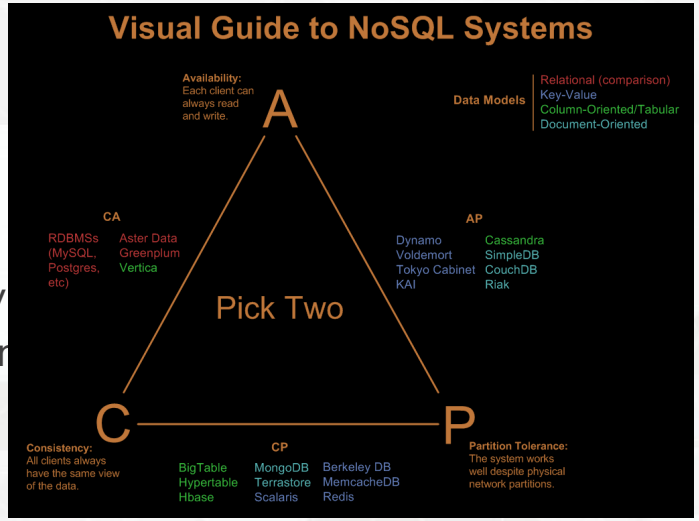
2000s – Sistemas baseados em Map Reduce



Histórico

2000s – NoSQL

- Foco em alta disponibilidade e escalabilidade
  - Schemaless (ou "schema last")
  - Modelos de dados não relacionais (document, key-value, grafos)
  - Sem transações ACID (Atomicity, Consistency, Isolation, Durability
    - BASE no lugar (Basically Available, Soft state, Eventual consister
    - Teorema CAP (Consistency, Availability, Partition tolerance)
  - APIs de acesso customizadas
    - Em vez de SQL
    - No entanto, surgem várias maneiras de usar SQL em bancos NoSQL, por mais que isso pareça contraditório...
  - Open-source, na maioria das vezes



Neo4j

mongoDB

CouchDB
relax

cassandra

redis

Couchbase

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

<lapti>

Histórico

2010 – NewSQL

- Fornece o mesmo desempenho para OLTP como um NoSQL, mantendo ACID
  - Relational / SQL
  - Distribuídos
  - Proprietários, geralmente

Sistemas Híbridos

- Hybrid Transactional-Analytical Processing
  - Executam OLTP com o mesmo desempenho de um NewSQL, mas também executando OLAP como um sistema de data warehouse
  - Distribuídos / shared-nothing
  - Relational / SQL

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
<lapti>

Histórico

2010 – Cloud Systems

- Database-as-a-service (DbaaS)
- Primeiros produtos eram versões em container de DBMSs existentes
- Mas já existem DBMSs projetados do início para executar em ambiente de nuvem

Histórico

2010 – Shared Disk Engines

- O storage manager é delegado a um sistema distribuído
  - Armazenamento distribuído não controlado pelo DBMS
- Implementação de data lakes

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
<lapti>

Histórico

2010 – Timeseries

- Sistemas específicos para armazenar dados com séries de tempo ou eventos
- Precisa tratar de maneira cuidadosa distribuição de dados e padrões de carga de queries

influxdb

TIMESCALE

ClickHouse

VICTORIA METRICS

Histórico

2010 – Blockchain databases

- Log descentralizado e distribuído com checksums incrementais
    - Usa protocolo Byzantine Fault Tolerant (BFT) para determinar a próxima entrada no log
- Praticamente qualquer funcionalidade deles pode ser executada também em um banco de dados relacionais
    - OTLP
    - Políticas externas de autenticação e sincronização

BIGCHAINDB fluree CovenantSQL Condensation

Histórico

2020 – Sistemas especializados

- DBMSs embarcados
- Multi-modelos
- Aceleração por hardware
- Array / matrizes / vector

Histórico

Evolução de Bancos de Dados

- Sistemas mudaram bastante desde os 1970s
  - E também os problemas, para não falar do volume de dados
- "One Size Fits All": An idea whose time has come and gone
  - Michael Stonebraker, 2004
  - Artigo que deu o prêmio Turina a Stonebraker em 2014
- As linhas divisórias entre categorias continuam a se confundir com o temp
  - Sistemas sempre expandem seu funcionamento original

Modelos de dados

- Hierárquico
- Redes
- Relacional 
- NoSQL
  - Document
  - Key-value
  - Column-family
  - Grafos
- Matriz
- NewSQL
- Híbridos

Pesquisa e desenvolvimento em BD

O que se faz hoje em dia em bancos de dados (pesquisa e desenvolvimento)
- Sempre avançar no desempenho de relacionais
- Índices, query optimizer, balanceamento, etc

Desenvolver bancos nosql
- MongoDB já tem transações, por sinal

DBs para machine learning
- Matrizes

DBs embarcados
- IoT e pequenos

Tratamento de dados
- Anonimização
- Cleaning
- Curadoria
- Preenchimento de gaps
- Etc
  - (reconhecimento de que nossos dados nunca estiveram tão bem comportados assim...)

Sites interessantes:

- db-engines.com
- dbdb.io

Obrigado

leandro@utfpr.edu.br
http://lapti.ct.utfpr.edu.br

<lapti> UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ