Evolução das Bases de Dados

Autor: Jacinto Estima
Revisto: Cláudio Sapateiro

15:00

Exercícios

1. O que é uma BD NoSQL?
2. Que variantes existem?
3. Em que se distingue(m) das relacionais que temos trabalhado?
4. Que requisitos podem motivar a sua adoção?

Exercícios

Discussão

1. O que é uma BD NoSQL?
2. Que variantes existem?
3. Em que se distingue(m) das relacionais que temos trabalhado?
4. Que requisitos podem motivar a sua adoção?

Conteúdos

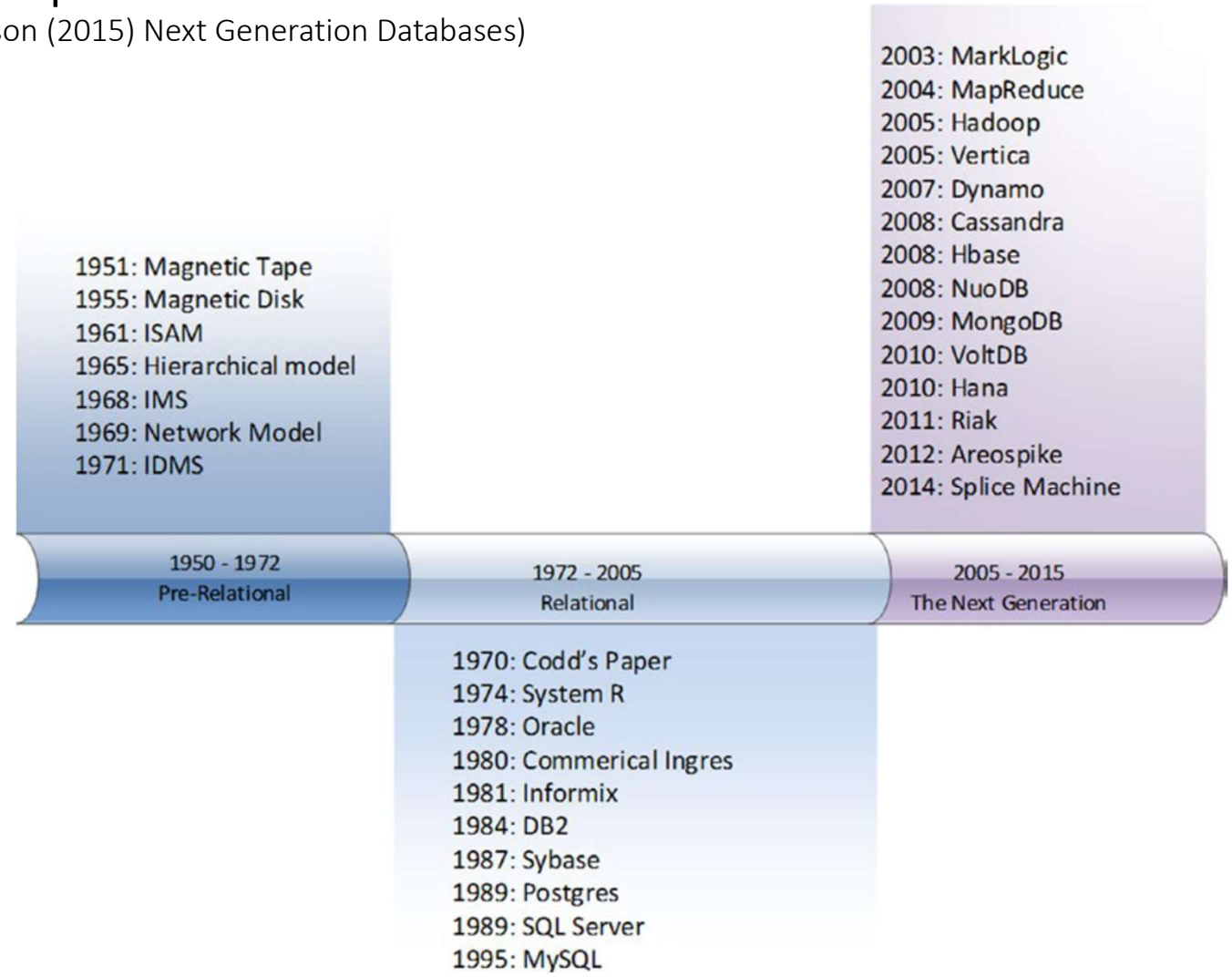
As 3 revoluções das Bases de Dados:

- Primeiros sistemas de “BD”
- Primeira revolução das BD
- Segunda revolução das BD
- Terceira revolução das BD

As 3 revoluções das Bases de Dados

3 maiores períodos na historia das Bases de Dados

(Fonte: Guy Harrison (2015) Next Generation Databases)



Principais determinantes das revoluções

1 Revolução

- Surgimento dos computadores eletrónicos

2 Revolução

- Desenvolvimento das Bases de dados relacionais

3 Revolução

- Necessidades de alcance global e disponibilidade constante das aplicações modernas

\+

- ✓ Volume de dados e acessos simultâneos
- ✓ Estrutura *flexível*

Primeiros sistemas de Bases de Dados

- O termo Bases de Dados entrou no nosso vocabulário no final da década de 60
- Uma definição simplista pode referir que uma Base de dados é um conjunto organizado de dados
- No entanto:
  - A recolha e organização de dados tem feito parte integral do desenvolvimento da civilização humana e da tecnologia:
  - Exemplos:
    - Livros representam conjuntos de dados em formato físico
    - Bibliotecas e outros arquivos indexados representam o equivalente pré-industrial dos sistemas modernos de Bases de dados
    - Census

Primeiros sistemas de Bases de Dados

- Os primeiros sistemas usavam:
  - Fitas de papel e mais tarde fitas magnéticas para armazenar dados sequencialmente
  - Ate 1950, quando surgiram os primeiros discos rotativos, não era possível aceder aos dados individuais com uma velocidade aceitável
  - Mais tarde foram desenvolvidos métodos de indexação que tornaram possível o acesso rápido a registos e consequentemente estiveram na origem do primeiro sistema OLTP (*On-Line Transaction Processing*)
  - Surgem assim as primeiras bases de dados eletrónicas completamente controladas pelas aplicações – ***DB sem DBMS***
- Problemas:
  - Cada aplicação tinha que desenvolver os seus métodos de manipulação de dados
  - Erros no código de algumas aplicações levavam ao aparecimento de dados corrompidos
  - Acesso concorrente exigia o desenvolvimento de aplicações complexas

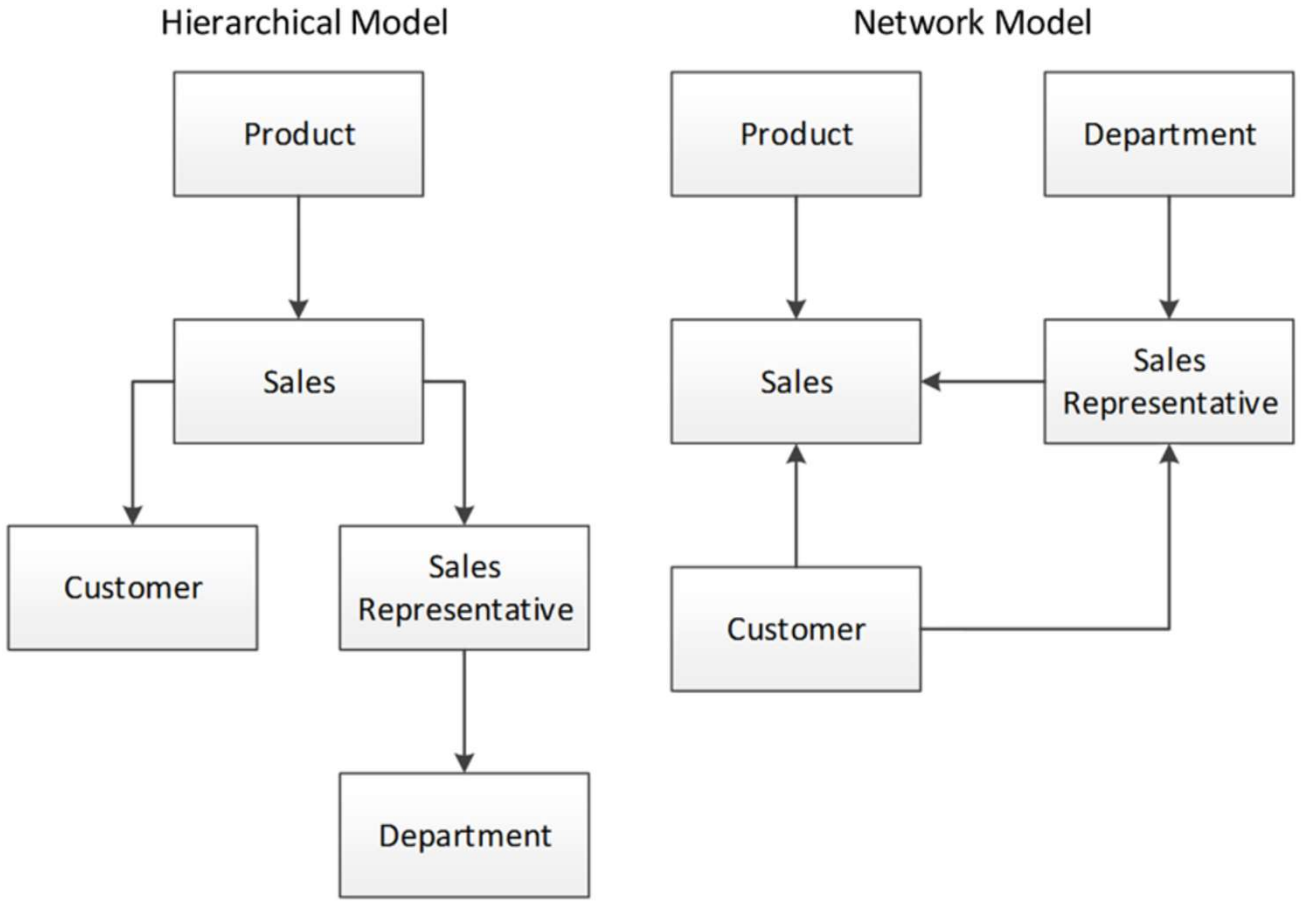
A primeira revolução de Bases de dados

- Ficou clara a necessidade de separar
  - **logica de gestão de Bases de dados** <-> **Aplicações**

- Surgem assim os Sistemas de Gestao de Bases de Dados (SGBD - *DBMS*):
  - Funcionavam em computadores de grandes dimensoes (mainframes)

- Nos inicios da decada de 1970, tinham subjacentes 2 paradigmas principais:
  - Rede
  - Hierarquico

Modelos em rede e hierárquico

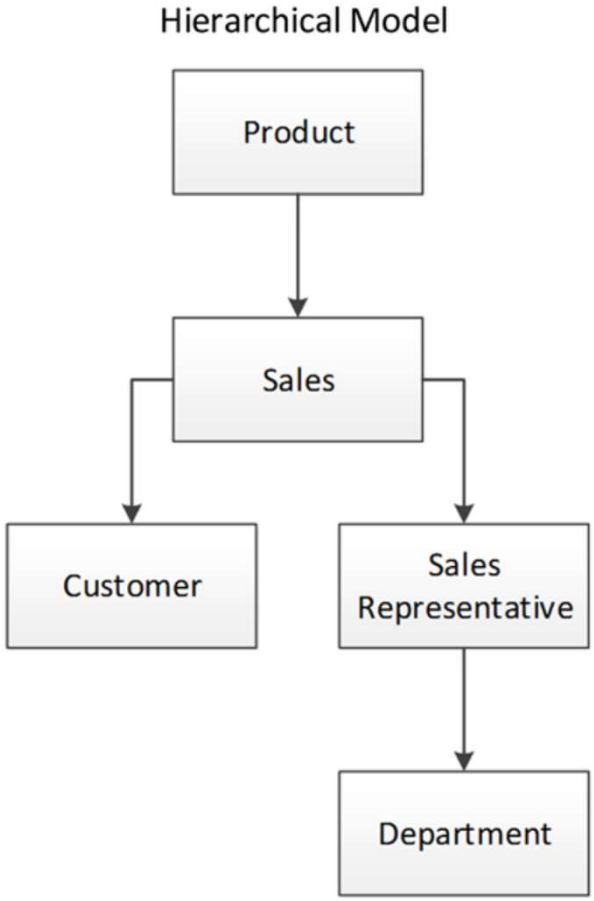


Modelo hierárquico

- O modelo assenta num conjunto de relações de *1 para muitos*
- Os níveis inferiores podem ter várias instancias do nível superior, mas no nível superior cada instancia só corresponde a uma instancia do nível inferior
- Exemplo:
  Estrutura implementada no IBM IMF (Information Management System)
  percussor dos seus DBMS (SGBDs)
- Pressupõe um armazenamento linear que integra as **vantagens**:
  - Acesso rápido aos dados no topo da hierarquia
  - Adição e remoção estão facilitadas
  - Responde bem a solicitações de *1 para muitos*

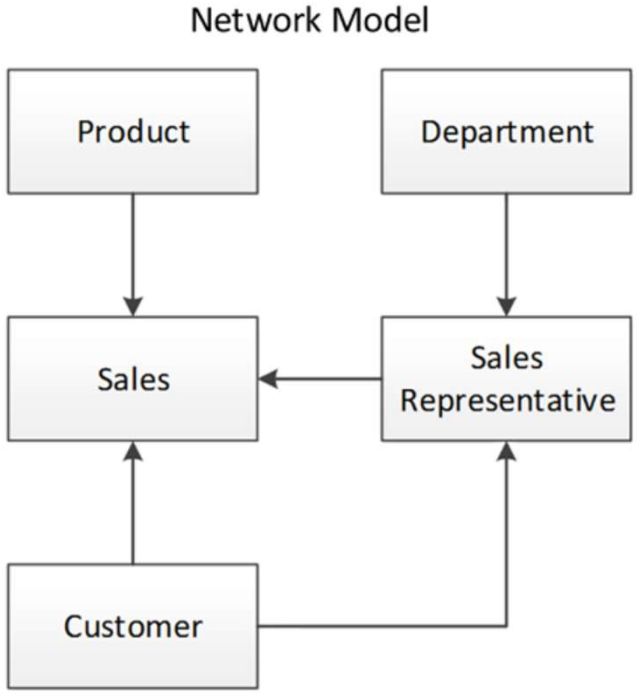
…**desvantagens**:

- Implementação e recuperação de relacionamentos mais complexos
  (e.g. muitos para muitos com propriedades, ...)
- O suporte segundo persistência linear está ultrapassado (desempenho)



Modelo em rede

- Os dados mantêm simultaneamente vários relacionamentos entre várias “entidades tipo”
- **Vantagens**:
  - Promove a integridade
  - Facilita a recuperação de variados tipos de consultas sobre relacionamentos diversificados
- **Desvantagens**:
  - Contudo estrutura de relacionamentos pode ser complexa e difícil de dominar e manter
  - Modelo fortemente acoplado à estrutura (difícil de fazer alterações) – <u>Ausência de independência estrutural</u>



Modelos em rede e hierarquico

Problemas

- Inflexibilidade na estrutura de dados e capacidades de interrogação:
  apenas *queries* pensadas durante o desenho eram possíveis (ou razoavelmente eficientes)

- Sistemas de BD centrados em processamento/transações **de um registo de cada vez**:
  queries complexas, como temos hoje em sistemas de BI, exigiam código complexo, e com compromissos de performance

A segunda revolução de Bases de dados

- **Edgar Codd** foi o impulsionador desta segunda revolução.

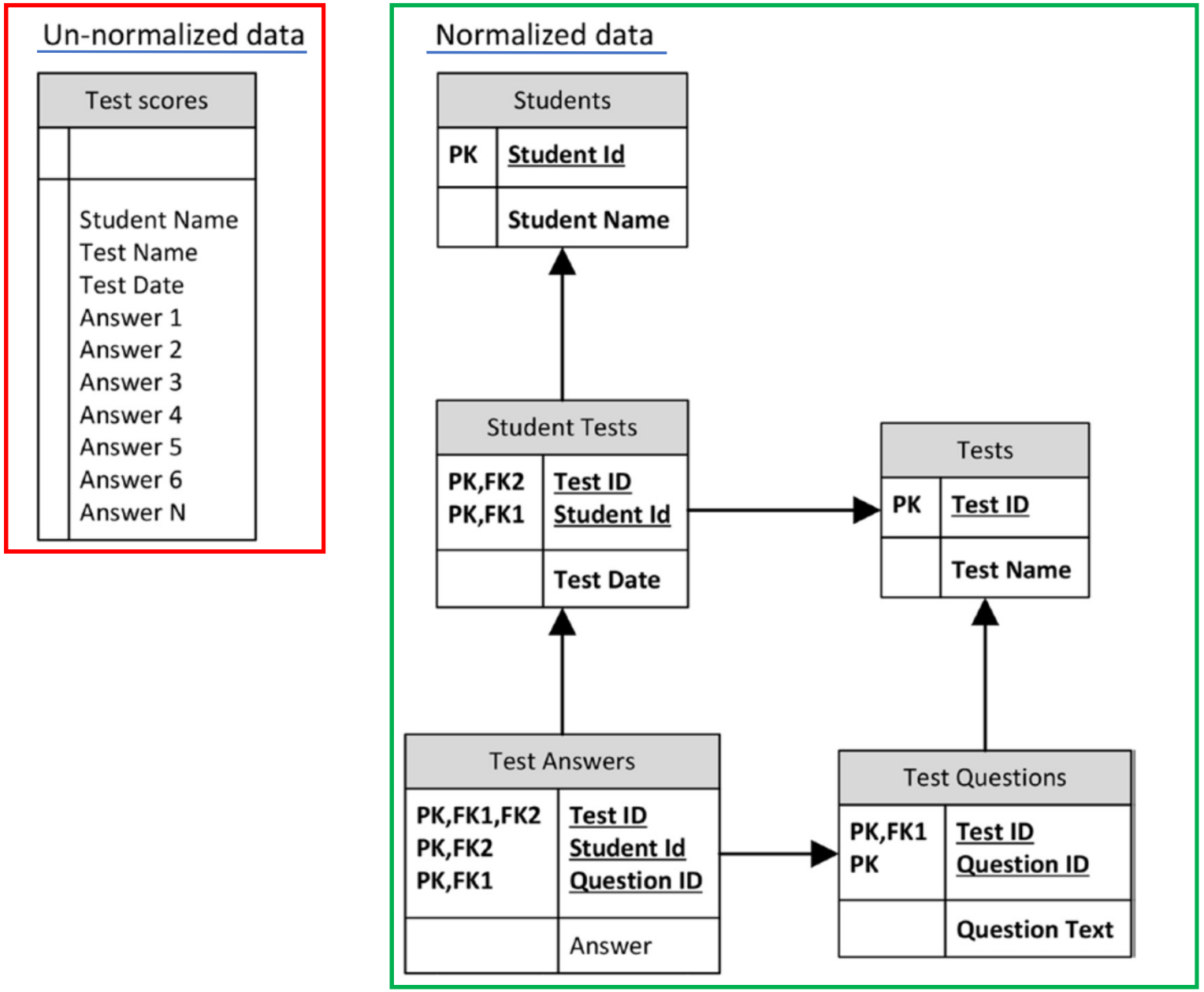
- Motivação - as BD's existentes :
  - Eram demasiado difíceis de usar
  - Não possuíam fundamentação teórica
  - Misturavam as implementações logica e física

- Foi o pai do **Modelo Relacional** que ainda usamos largamente nos dias de hoje

A segunda revolução de Bases de dados

Teoria relacional

- Na sua essência, o modelo relacional descreve como um conjunto de dados, deve ser apresentado ao utilizador, em vez de descrever como os dados devem ser guardados em disco/memória
- Alguns conceitos chave:
  - **Túpulos**, um conjunto não ordenado de atributos
  - **Relações**, que são conjuntos de túpulos distintos que correspondem a uma tabela nas implementações de Bases de dados relacionais
  - **Restrições**, que forçam a consistência da Base de Dados
  - **Operações**, como uniões, projeções, etc. etc.
- Cada túpulo (linha ou registo) deve ser identificada e acedida de forma eficiente pelo valor de uma chave única, e cada atributo (coluna) dessa linha é dependente apenas dessa chave
- Terceira *Forma Normal* é o nível de conformidade mais comum, lembrada tradicionalmente pela regra de que "todos os atributos não-chave devem depender da chave, de toda a chave, e nada mais do que a chave"


Un-normalized data
Test scores
Student Name
Test Name
Test Date
Answer 1
Answer 2
Answer 3
Answer 4
Answer 5
Answer 6
Answer N
Normalized data
Students
PK
Student Id
Student Name
Student Tests
PK,FK2
PK,FK1
Test ID
Student Id
Test Date
Tests
PK
Test ID
Test Name
Test Answers
PK,FK1,FK2
PK,FK2
PK,FK1
Test ID
Student Id
Question ID
Answer
Test Questions
PK,FK1
PK
Test ID
Question ID
Question Text

A segunda revolução de Bases de dados

Modelos Transacionais

- A uma transação corresponde uma transformação de estado observante das propriedades ACID

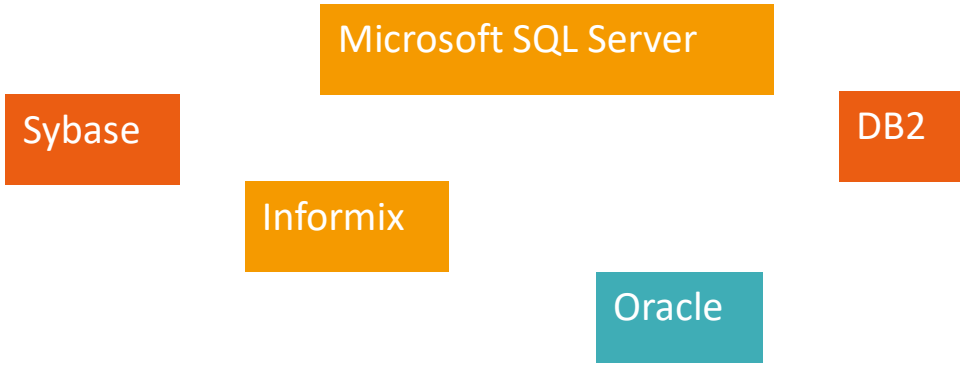
- Ficou popularizado como *ACID Transactions*:
  - **A**tomic: a transação é indivisível
  - **C**onsistent: a base de dados fica num estado consistente antes e depois
  - **I**solated: uma transação não deve ver os efeitos de outras transações ainda em progresso
  - **D**urable: depois de concluída, as alterações devem persistir, mesmo que haja uma falha do sistema operativo ou *hardware*
- As transações **ACID** tornaram-se o standard para as implementações de BD mais maduras, e ficaram fortemente associadas às BD relacionais

❖As restrições de escalabilidade impostas pelas transações ACID tem sido a chave para o desenvolvimento de novas arquiteturas de BD

A segunda revolução de Bases de dados

Bases de Dados Relacionais

- A aceitação das regras de Cood não foram imediatamente aceites
- O primeiro protótipo surge em 1974, denominado **System R**, e resultou de um programa de investigação lançado pela IBM, que também deu origem à linguagem SQL
- Nos anos seguintes acontece uma luta de diferentes sistemas de BD



A terceira revolução de Bases de dados

- Surge com a explosão da Web durante o período 1995-2005 e posterior
- Entre outros motivadores:
- WWW 2.0
- E-commerce:
  - Necessidade de capacidade de processamento transacional capaz de operar com grandes volumes de dados – e-commerce (ex. Amazon)
- Redes sociais:
  - MySpace e mais tarde o Facebook, enfrentam desafios para escalar as suas infraestruturas de milhares para milhões de utilizadores

A terceira revolução de Bases de dados

Google and Hadoop

- Bases de dados relacionais não eram adequadas para lidar com o volume, (falta de) estrutura e velocidade de alguns tipos de dados dos dados
  - 2003 – revelou detalhes do seu sistema distribuído de ficheiros (**GFS**)
  - 2004 – revelou detalhes do algoritmo de processamento paralelo distribuído, denominado **MapReduce**, utilizado para criar índices para a WWW
  - 2006 – revelou detalhes sobre o seu sistema de Bases de Dados de estrutura distribuída "BigTable"
- Estes conceitos, juntamente com outras tecnologias, formaram a base do projeto **Hadoop**, que se tornou a tecnologia facilitadora do ecossistema Big Data

✓Tornando-se mais eficiente a persistência distribuída de dados e a recuperação/processamento desses dados

A terceira revolução de Bases de dados

- Surgem varias soluções alternativas, cada uma com as suas aplicações, vantagens e desvantagens
  - Distributed object classes: para aliviar a carga na BD
  - Database replication: replicação para pesquisas mais rápidas
  - **Sharding**: partir os dados por múltiplas BD

❖Amazon revela detalhes do seu modelo Dynamo, que mais tarde dá origem ao que vem a ser conhecido como "***Key-value databases***"

A terceira revolução de Bases de dados

- Emergem tecnologias como:
  - AJAX (*Asynchronous JavaScript and XML*)
  - JSON (*JavaScript Object Notation*): depressa ultrapassou o XML e foi considerado o **standard de facto** para armazenamento e serialização de objetos em disco, e comunicação
- Websites começam a usar JSON para comunicar e armazenam como documentos em colunas de bases de dados relacionais
- A componente relacional é eliminada, ou deferida, e recorreu-se a bases de dados onde *documentos* **JSON** podiam ser armazenados diretamente as: *Document Databases*

A terceira revolução de Bases de dados

Cloud Computing

- Por volta de 2008, com a mudança de aplicações desktop baseadas em arquiteturas cliente-servidor para aplicações baseadas na Web, e apesar de existir desde o final da década de 1990, aparece abruptamente como uma:
  - Preocupação para grandes organizações
  - Oportunidade para as start-up's

- Surge o conceito de infraestrutura e plataformas **como um serviço**

A terceira revolução de Bases de dados

A explosão não-relacional e o aparecimento do NoSQL

- Entre 2008 e 2009 acontece uma explosão de novos sistemas de Bases de Dados, integrando os avanços tecnológicos

- Algumas persistem:
  - MongoDB, Cassandra, neo4j, …

- Surge a designação para este tipo de Bases de dados: "***Distributed Non-Relational Database Management System (DNRDMS)***", mas não era claro relativamente ao conceito subjacente
  - ✓Surge então o termo **NoSQL** a referir sistemas de bases de dados que quebravam com as tradicionais bases de dados relacionais

✓O termo NoSQL é, refere-se a Bases de Dados que priorizam a **escalabilidade** e **disponibilidade**, em oposição à **atomicidade** e **consistência transacionais**

Lista de Bases de Dados NoSQL segundo diferentes paradigmas

Document	Key-value	Column	Graph	Multi-storage
MongoDB	Redis	BigTable	Neo4J	OrientDB
CouchDB	Membase	Hadoop / Hbase	FlockDB	ArangoDB
RavenDB	Voldemort	Cassandra	InfiniteGraph	Aerospike
Terrastore	MemcacheDB	SimpleDB		
		Cloudera		
…	…	…	…	…

Modelos de armazenamento de dados
Column-oriented store

- Dados guardados em colunas em oposição às linhas dos SGBDR

ID_empregado	PrimeiroNome	UltimoNome	Idade	Salario
SM1	João	Azevedo	45	1500
MM2	Maria	Lopes	34	900
T3	Manuel	Guerreiro	39	1100
E4	Andreia	Pereira	32	700

Armazenamento em SGBDR
SM1, João, Azevedo, 45, 1500
MM2, Maria, Lopes, 34, 900
T3, Manuel, Guerreiro, 39, 1100
E4, Andreia, Pereira, 32, 700

Armazenamento em Column-oriented
SM1, MM2, T3, E4
João, Maria, Manuel, Andreia
Azevedo, Lopes, Guerreiro, Pereira
45, 34, 39, 32
1500, 900, 1100, 700

Modelos de armazenamento de dados
Document store

- Permite inserir, aceder e manipular dados semi-estruturados, em oposição à estrutura rígida dos SGBDR
- Dados armazenados em “documentos”, equiparados a linhas de uma tabela nos SGBDR
- Dois documentos pode ter atributos (colunas) diferentes
- Os registos podem ou não aderir a uma estrutura

Modelos de armazenamento de dados
Document store

- Os documentos podem ter uma estrutura diferente entre si

```
{
"ID_empregado"  : "SM1“,
"PrimeiroNome"  : "João“,
"UltimoNome"    : "Azevedo“,
"Idade"         : "45“,
"Salario"       : "1500“
}
```

```
{
"ID_empregado"    : “MM”,
"PrimeiroNome"    : “Maria“,
"UltimoNome"      : “Lopes“,
"Idade"           : “34“,
"Salario"         : “900“,
“Endereco“        : {
      “Rua”       :”Avenida de Baixo”,
      “CodigoPostal” : “1800-123”
            },
“Projetos”              :{
      “nosql-migration”,
      “top-secret-007”
            }
}
```

Modelos de armazenamento de dados
Key-value store

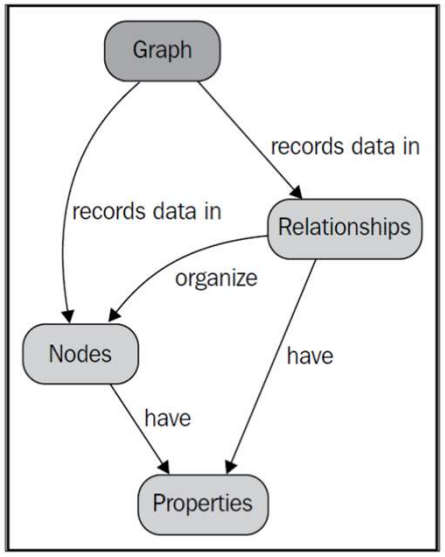
- Possibilita o armazenamento de um valor (***value***) relacionado com uma chave (***key***)

- Não força nenhum esquema para o valor, mas existem alguma restrições:
  - Requer que a chave (key) seja especifica, por oposição às *Document stores*
  - A recuperação do valor é feita por queries que predominantemente lhe acedem pelo valor da chave
    - A chave tem que ser conhecida para aceder ao valor

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Modelos de armazenamento de dados
Graph store

- Representam uma categoria especial de Bases de Dados NoSQL, onde as relações são representadas como grafos – a representar as múltiplas relações que dois nós partilham



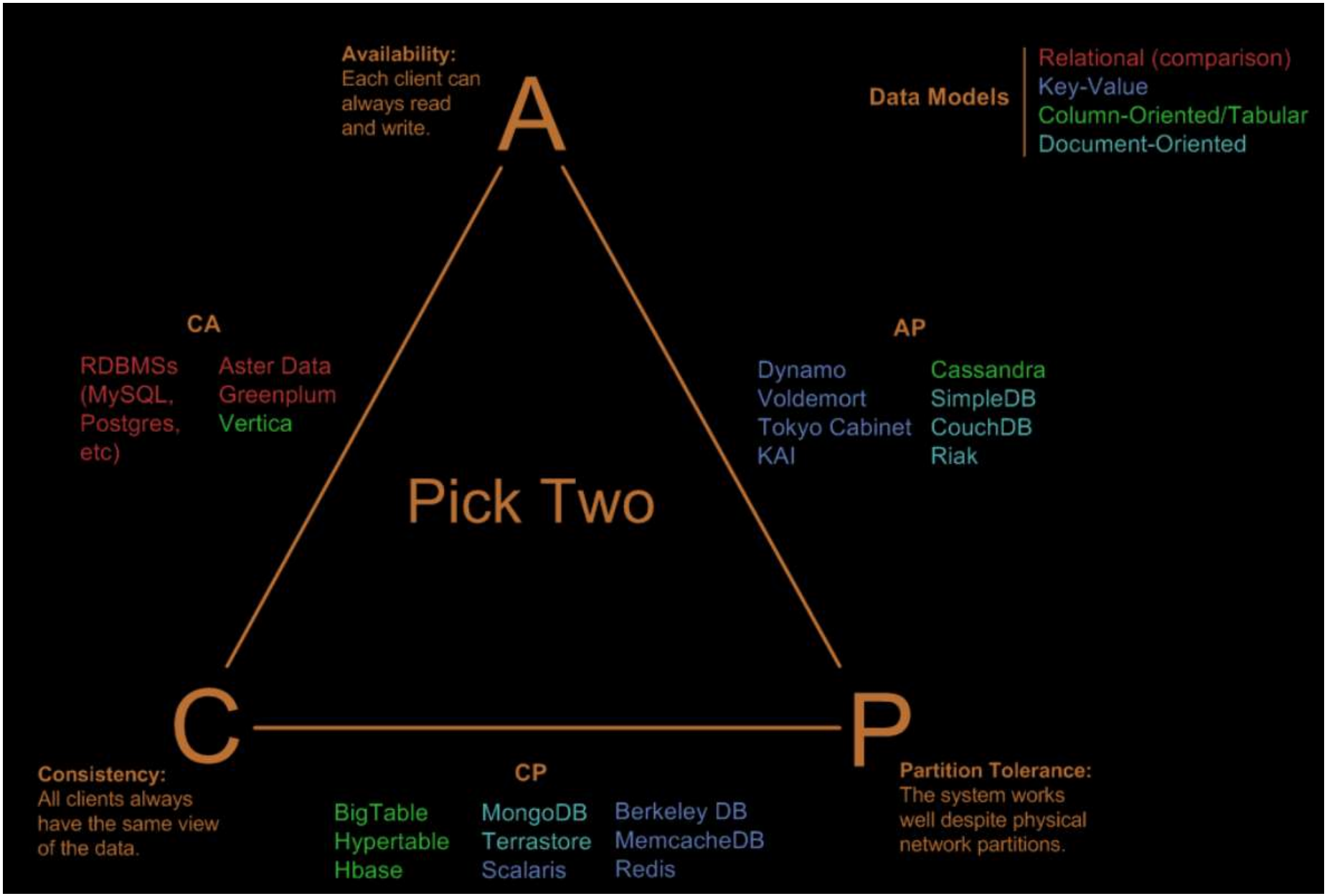
- As relações podem representar:
  - Relações sociais entre pessoas
  - Ligações de transporte entre locais
  - Topologias de rede entre sistemas conectados

- São otimizadas para dados com fortes multi-relações

- A grande vantagem são:
  - Fácil representação, recuperação (acesso) e manipulação de relações entre entidades de um sistema

Vantagens e desvantagens dos modelos de dados NoSQL

- Questões a responder
  - **Requisitos de esquema de relações** – Verificar a densidade das relações entre entidades, e se a estrutura altera e com que frequência
  - **Requisitos de acesso aos dados** – Verificar se os dados têm que estar sempre consistentes ou podem por algum período estar em estado inconsistente, e se o acesso aos dados é mais horizontal ou vertical.
  - **O que NoSQL pode fazer** – o que alguma das famílias das BDs NoSQL pode oferecer face aos requisitos
  - **O que NoSQL não pode fazer** – Onde NoSQL falha relativamente aos requisitos do domínio em causa


Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

Vantagens e desvantagens dos modelos de dados NoSQL

Análise de diferentes aplicações:

- **Transacionais** – aplicações com dados relacionais que requerem transações
- **Computacionais** – aplicações "pesadas" do ponto de vista computacional
- ***Web-scale*** – aplicações orientadas para a Web com escala

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Transacionais

Descrição da aplicação

- Contém dados com uma natureza altamente relacional
- Características principais:
  - Depende da consistência e integridade dos dados
  - A utilização em concorrência é controlada
  - Pode ser servida por um numero reduzido de nós
- Exemplos deste tipo de aplicações:
  - Pontos de venda num retalhista
  - *Entreprise Resource Management Applications*

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Transacionais

- **Requisitos de esquema de relações**
  - Definição altamente estruturada—propriedades, tipos e restrições
  - Possibilidade de definir relacionamentos
  - Esquema não evolui nem varia muito ao longo do tempo

- **Requisitos de acesso aos dados**
  - Consistência – qualquer leitura deve trazer os dados mais recentes
  - Linhas inteiras são acedidas com frequência
  - O acesso cruzado de tabelas pode ser frequente

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Transacionais

- **O que NoSQL pode fazer**
  - Column-oriented database – ajuda a definir a estrutura, que, se necessário, pode ser alterada ao longo do tempo e facilitar algumas computações
- **O que NoSQL não pode fazer**
  - Impossibilidade de definir relações assegurando de forma expedita integridade referencial
  - Tipicamente ausência de transações (Neo4j é uma exceção)
  - Propriedades ACID não estão disponíveis nas operações (CouchDB e Neo4j são exceções)
  - Não existe suporte prático para JOIN's e interrogações cruzadas de tabelas/"entidades" (document stores podem suportar através do MapReduce, mas exigem demasiado esforço quando as queries se tornam complexas)

❖ **Usar ou não usar NoSQL**

- Preferível escolher um SGBDR
- São mais as desvantagens do que as vantagens de escolher NoSQL

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Computacionais

Descrição da aplicação

- Tipo de aplicação que usa bastantes recursos computacionais e forte processamento dos dados
- Características principais:
  - Maioria das operações ao longo de um conjunto de propriedades através dos atributos
  - Os dados armazenados podem ser relacionais, mas a computação centrada nos dados possui, quando existem, relações esparsas
- Exemplos deste tipo de aplicações:
  - Questionário online, que por vezes precisa de definir campos customizados
  - Aplicação de gestão/análise de registos de vendas

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Computacionais

- **Requisitos de esquema de relações**
  - Definição estruturada de esquema —propriedades, tipos, e restrições, se aplicável
  - Esquema não evolui nem varia muito ao longo do tempo
- **Requisitos de acesso aos dados**
  - Acesso parcial dos registos
  - Acesso vertical (processamento sobre as colunas)
    - Dada uma tabela com dados diários de despesas e receitas num dado local de operações, os cálculos mais frequentes aconteceriam através das colunas de receitas e/ou despesas
  - Acesso cruzado a varias tabelas pouco frequente
    - Usado o ID da localização para aceder aos detalhes depois de calcular um balanco final
  - Consistência: Os dados devem ser consistentes. Em alguns casos, latências menores são permitidas
    - Uma vez que os registos são gerados diariamente e não em tempo-real, o utilizador não se importa de usar dados diários “desatualizados” (que não usam os registos mais recentes do dia)

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Computacionais

- **O que NoSQL pode fazer**
  - DB *Column-oriented* ajudam a definir um esquema rigoroso.
    Document ou *Key-value stores* podem também ser usadas
  - *Column-oriented*, *key-value* e *document stores* podem fornecer velocidade e escalabilidade enquanto procuram dados
  - *Document stores* com processamento em MapReduce podem ajudar a realizar processamento computacional junto da camada dos dados e assim aumentar substancialmente a velocidade de execução, evitando que dados desnecessários andem a circular pela rede
  - *Document stores* podem ajudar a implementar JOIN ou criar vistas (com alguns cuidados para desempenho)

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Computacionais

- **O que NoSQL não pode fazer**
  - Definir relações é problemático. As relações têm que ser mantidas na aplicação, exceto para *graph databases*.
  - Como consequência da não existência de relações na BD, os dados podem ficar inconsistentes

❖ **Usar ou não usar NoSQL**

- A lista de tarefas possíveis com NoSQL são também possíveis com SGBDR. As únicas vantagens do NoSQL aqui, e que podem ser bastante diferenciadoras, são a velocidade e a escalabilidade

- Uma pesquisa como esta:

  SELECT revenue, expense FROM location_finance WHERE location_id=1234

  a. Necessita de procurar dados onde a coluna tenha o id=1234, pelo que uma *column store* teria algumas vantagens
  b. Uma escolha entre SGBDR e NoSQL não é evidente

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações *Web-scale*

Descrição da aplicação

- São mais relevantes nas aplicações de consumo, sejam elas *web-based* ou *mobile-native* ou uma combinação das duas
- Características principais:
  - A aplicação tem que ser escalável, dado volume de dados que envolve, o numero de utilizadores, e a distribuição geográfica
  - Os utilizadores não se importam de estar a trabalhar com (alguns) dados que não são tempo-real, ou são de alguma forma obsoletos (diferenças de entre milissegundos a poucos dias). Ex. o numero de likes no Facebook
  - O esquema pode evoluir ao longo do tempo, uma vez que a aplicação permite a integração com outras aplicações
- Exemplos de aplicações:
  - Plataforma de *Micro Blogging* social

Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Web-scale

- **Requisitos de esquema de relações**
  - Definição estruturada de esquema
  - Possibilidade de alterar o esquema com o tempo sem afetar os registos existentes
  - Relacionamentos podem ser opcionais ao nível da base de dados e podem ser passados para a layer de aplicação, dada a baixa densidade

- **Requisitos de acesso aos dados**
  - Acesso parcial aos registos
  - Velocidade nas operações CRUD
  - Dados inconsistentes, por períodos curtos de tempo, são toleráveis

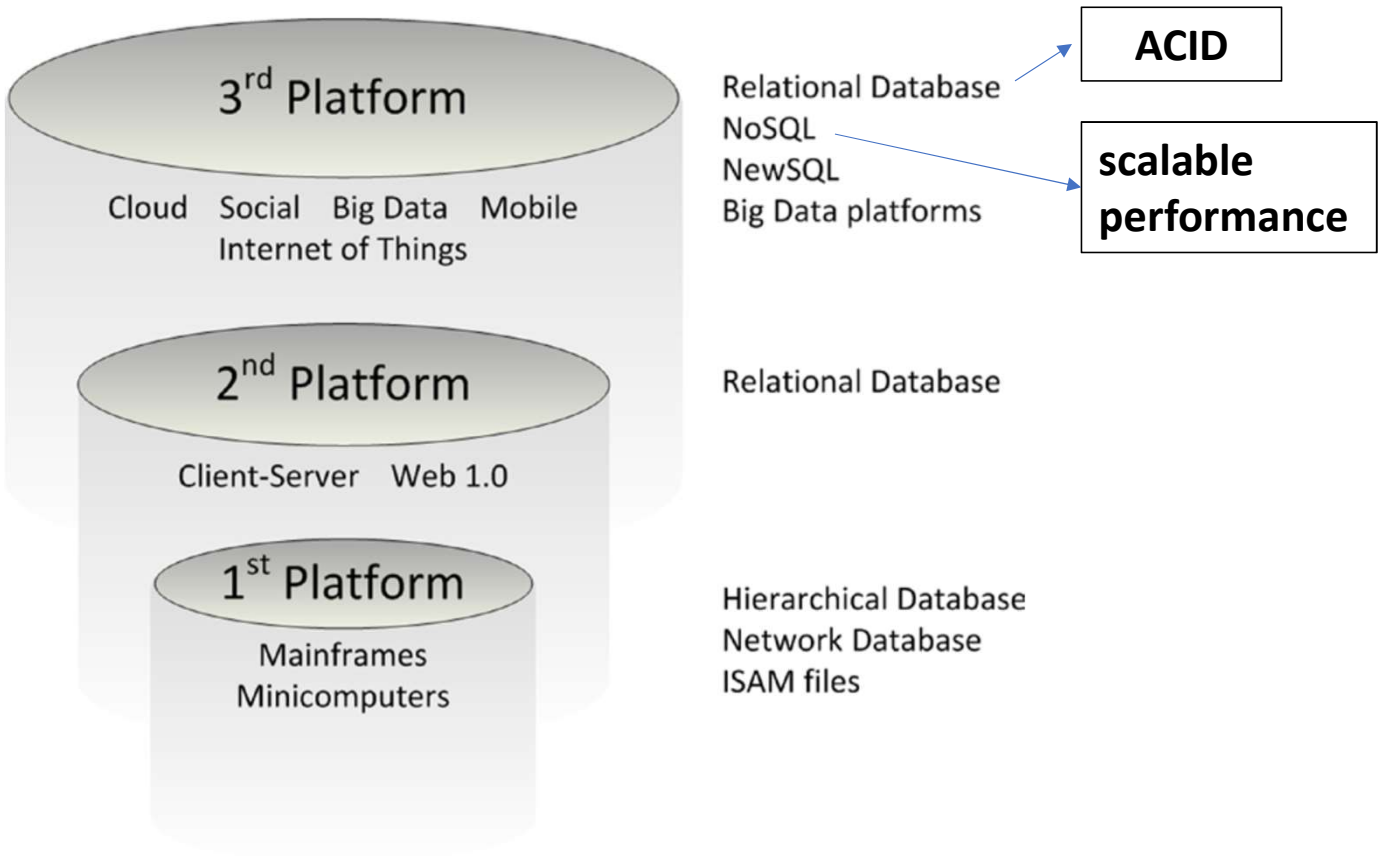
Vantagens e desvantagens dos modelos de dados NoSQL – Aplicações Web-scale

- **O que NoSQL pode fazer**
  - Tudo o que já foi descrito no cenário anterior
  - Operações em escala
- **O que NoSQL não pode fazer**
  - Aqui não existem restrições ACID

❖ **Usar ou não usar NoSQL**

- Tendencialmente usar NoSQL
- A utilização do modelo de dados depende dos requisitos da aplicação

“One Size Doesn’t Fit All”



“One Size Doesn’t Fit All”

Polyglot Persistence

- Nalguns casos necessidade de combinar vantagens:
  - ✓ Sistemas hibridos

Functionality	Considerations	Database Type
User Sessions	Rapid Access for reads and writes. No need to be durable.	Key-Value
Financial Data	Needs transactional updates. Tabular structure fits data.	RDBMS
POS Data	Depending on size and rate of ingest. Lots of writes, infrequent reads mostly for analytics.	RDBMS (if modest), Key Value or Document (if ingest very high) or Column if analytics is key.
Shopping Cart	High availability across multiple locations. Can merge inconsistent writes.	Document, (Key Value maybe)
Recommendations	Rapidly traverse links between friends, product purchases, and ratings.	Graph, (Column if simple)
Product Catalog	Lots of reads, infrequent writes. Products make natural aggregates.	Document
Reporting	SQL interfaces well with reporting tools	RDBMS, Column
Analytics	Large scale analytics on large cluster	Column
User activity logs, CSR logs, Social Media analysis	High volume of writes on multiple nodes	Key Value or Document

Exercícios

Discussão

1. O que é uma BD NoSQL?
2. Que variantes existem?
3. Em que se distingue(m) das relacionais que temos trabalhado?
4. Que requisitos podem motivar a sua adoção?

10:00

Exercícios

1. Aprsente os principais paradigmas de BD's NoSQL?

2. O que significa o teorema CAP?

3.

   a. Dê um exemplo de um requisito que pode levar à optar por uma BD RDMS?

   b. Dê um exemplo de um requisite que pode levar à optar por uma BD NoSQL?

Exercícios Correção

1. Aprsente os principais paradigmas de BD's NoSQL?

2. O que significa o teorema CAP?

3.

   a. Dê um exemplo de um requisito que pode levar à optar por uma BD RDMS?

   b. Dê um exemplo de um requisite que pode levar à optar por uma BD NoSQL?

Fim