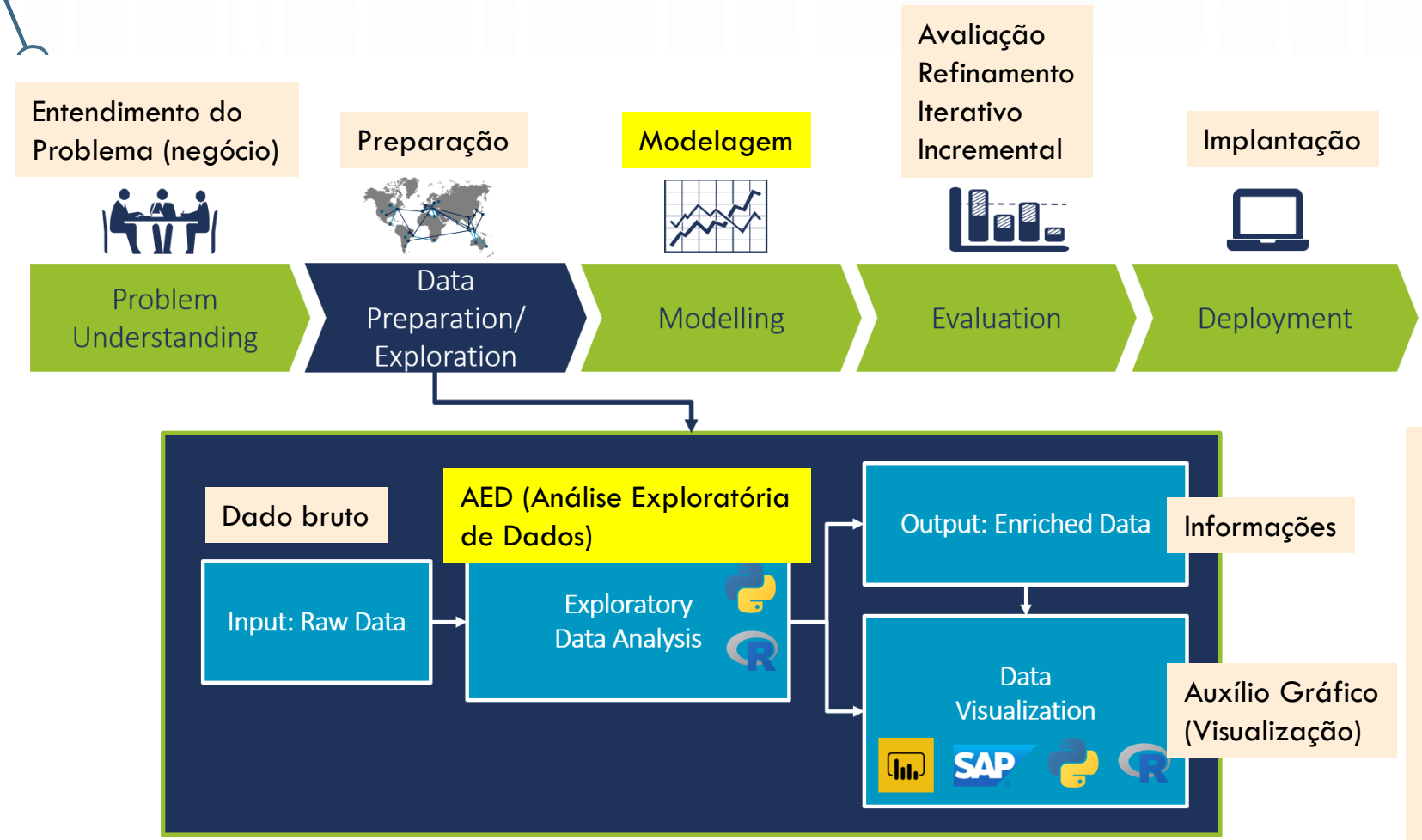
FUNDAMENTOS E TÉCNICAS EM CIÊNCIAS DE DADOS

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br
https://github.com/josenalde/datascience

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

NO QUE CONSISTE UM PROJETO DE CIÊNCIA DE DADOS?



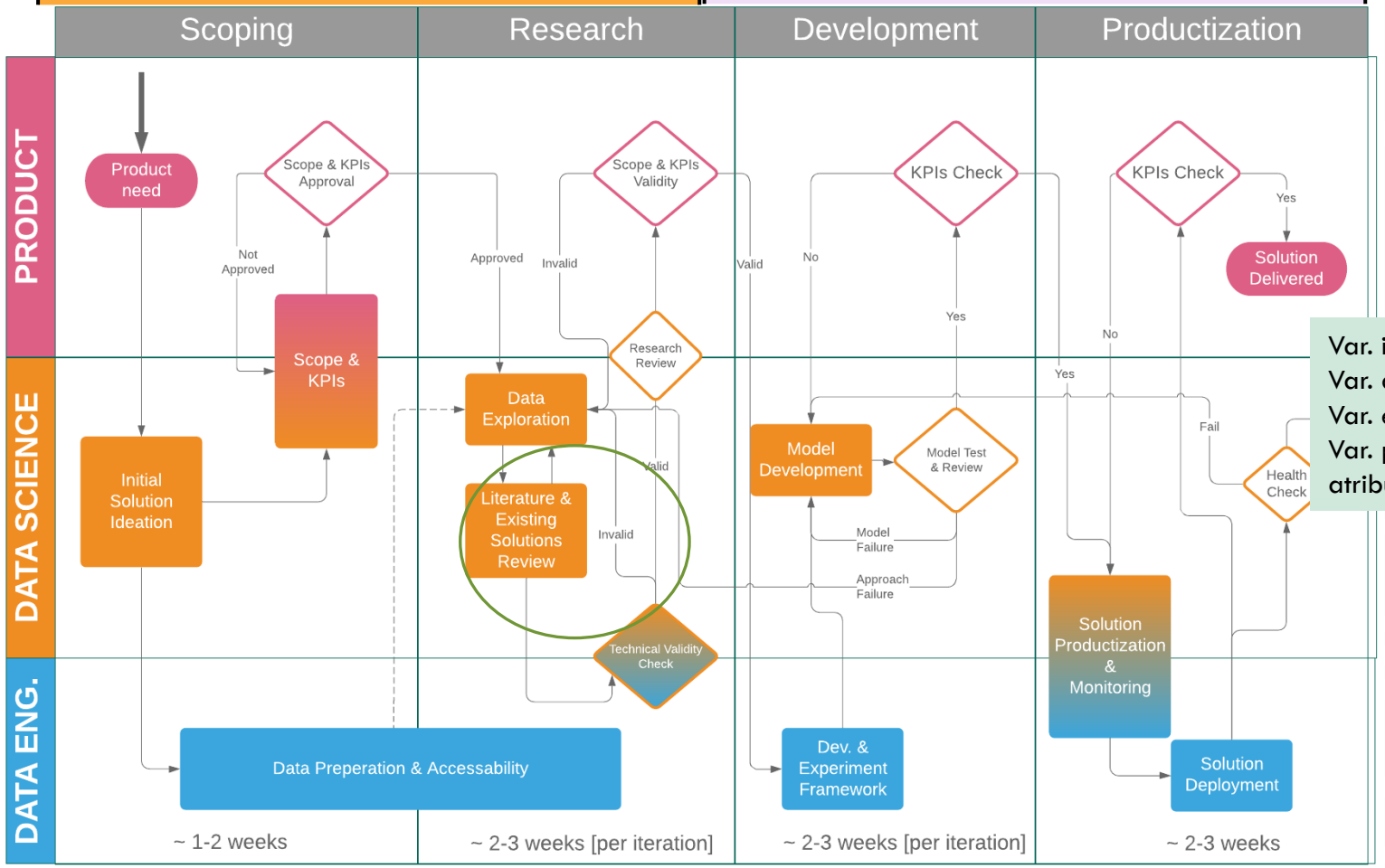
Analisar dados – aplicar algum tipo de transformação nos dados em busca de conhecimento. Dados podem ser produzidos para posterior análise. A ideia é conhecer antes de analisar, para melhor escolha das técnicas.

Exploratória: conhece os dados que irá analisar? Tem noção de como estão distribuídos? Quais suas médias? Desvios padrões? Como estão relacionados? Existem valores anormais?

NO QUE CONSISTE UM PROJETO DE CIÊNCIA DE DADOS?

Escopo/Necessidades/Indicadores

Modelagem/Avaliação/Indicadores



Mas o que é um modelo?

Simplesmente uma especificação de relação matemática (ou probabilística) existente entre variáveis diferentes

Var. independente
Var. de entrada
Var. explicativa
Var. preditora
atributos

x,y,z,... → f(x,y,z,...) → f

Var. resposta
Var. dependente
Target var.

- Lucro nos próximos anos
- Probabilidade de ganhar
- Mensagem é spam ou não
- Transação no cartão é fraudulenta?
- Probabilidade de um banner ser clicado
- Que time ganhará?
- Impacto de ação A ou B em processo X

AS VÁRIAS POSSIBILIDADES E TÉCNICAS

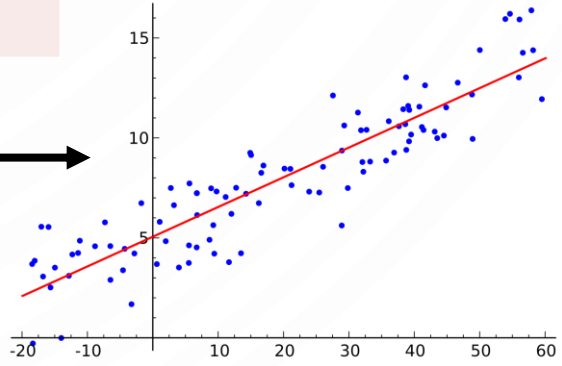
Var Resposta / Var Indep.	Categóricas	Intervalares (numéricas)	Intervalares e categóricas
Intervalar (numérica)	ANOVA Árvore Decisão Random Forest Gradient boosting Rede neural	Regressão linear* Árvore decisão Random Forest Gradient boosting Rede neural	ANCOVA Árvore decisão Random Forest Gradient boosting Rede neural
Categórica	**Regressão logística** Rule induction Árvore Decisão Random Forest Gradient boosting Rede neural	**Regressão logística** Rule induction Árvore Decisão Random Forest Gradient boosting Rede neural	**Regressão logística** Rule induction Árvore Decisão Random Forest Gradient boosting Rede neural

*RL – preço de produtos variando em função da quantidade vendida
**ANOVA – gastos no cartão de crédito em função do gênero
***ANCOVA – salários em função de faixa etária, gênero, anos de empresa

Variáveis:

Qualitativas
Nominal (categoria, descrição, classe), exemplo:
cor dos olhos, tipo de material, gênero, modo de pagamento...
Ordinal (pequeno, médio, grande etc.)

Quantitativas
Discreta: têm um número contável de valores entre quaisquer dois valores. Uma variável discreta é sempre numérica. Por exemplo, o número de reclamações de clientes ou o número de falhas ou defeitos.
Contínua: medições (-inf, +inf)



AS VÁRIAS POSSIBILIDADES E TÉCNICAS

Resumo: exemplos de algoritmos

Gerais: Árvore de decisão (decision tree), floresta aleatória (Random forest), redes neurais, gradient boosting
Classificadores OU Preditores (regressores)

Específicos: Regressão linear (numérico, numérico – contínuos)
Regressão logística (qualquer X, mas Y categórico/qualitativo) – muito usado em diagnóstico
Indução de regras (rule induction) – independente do dado de entrada, extrair regras if-then-else
Algoritmos como lem1, lem2, aq etc.
ANOVA e ANCOVA são análises de variância, um tipo de teste de hipóteses. Ambos tem variável target numérica, mas ANOVA tem X apenas categórico, e ANCOVA X pode ser misto

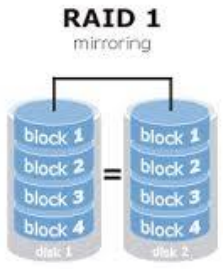
RETORNANDO AO CICLO DO DADO: PRODUÇÃO

1) Dados podem ser coletados, **produzidos** ou simplesmente comprados (*data brokers*)
   1) PRODUZIDOS como resultado de processamento (Ex. folha de pagamento) em sistemas transacionais,
   2) podem ser resultado de TRANSFORMAÇÃO de dados (para análise, armazenamento)
   3) podem ser produzidos por modelagem estatística ou aprendizagem de máquina
   3) podem ser COLETADOS de outros sistemas, pesquisas, dados históricos, arquivos, de um Data WareHouse
   4) Das chaves mecânicas, passando pelos cartões perfurados, às telas touch-screen e sensores de movimento
   5) teclados, mouses, leitores (barras, QR), RFID, mesas digitalizadoras, câmeras
   6) Projetos de computação distribuída, com base em doação de tempo de CPU (SETI, CLIMATE PREDICTION, ROSETTA@)

Acelerômetros – aceleração do objeto em 3 eixos
GPS
Giroscópio – orientação do celular em 3 eixos
Magnetômetro – mede o campo magnético da Terra
Podem se utilizar das interfaces de comunicação: Wifi, bluetooth, nfc etc.
WEARABLES
Em Desenvolvimento APIs permitem manipulação

RETORNANDO AO CICLO DO DADO: ARMAZENAMENTO

1) Uma vez armazenado, pode ser recuperado para replicar processos ou produzir informação ou conhecimento
2) Contempla as premissas de segurança da informação, integridade, minimização da redundância, concorrência, otimização de espaço etc.
3) Podem ser simplesmente REPLICADOS em sua forma original (backup, redundância (RAID 1, clusterização)
4) Já num data warehouse, é armazenado com sua estrutura modificada

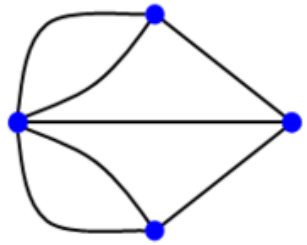


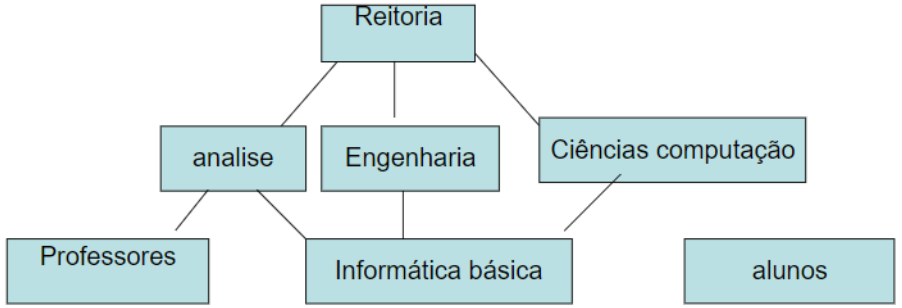
MODELOS DE ARMAZENAMENTO

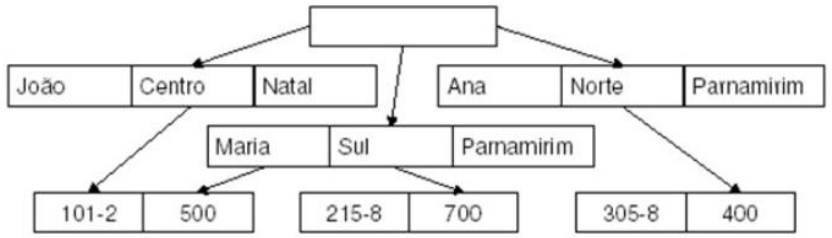
A Medida que o volume e complexidade aumentaram, a partir de 1950 surge necessidade de gerenciadores capazes de incluir, alterar e excluir, mas também de manter integridade, segurança e de prover indexação.

Modelo	Década
Pré-relacionais (hierárquico, rede (codasyl)	60
Relacional	70
Orientado a objetos (caché, Db4o, Versant...)	80
NoSQL (kvs, colunas ordenadas, documentos, grafos)	2000

Modelo de rede baseado em navegação – registros vinculados precisavam ser percorridos numa árvore, Unidirecional, do pai para o filho, não havendo controle efetivo de integridade e redundância –
O consórcio CODASYL é conhecido por ter criado a linguagem COBOL – relacionamentos: LINKS
OBS: *ideia de Grafo e sua teoria remonta a Leonhard Euler (1736) e as pontes de Königsberg*







MODELOS DE ARMAZENAMENTO

RELACIONAL: Criado por Edgar Codd em 1970 *(Relational Model of Data for Large Shared Data Banks)*

Foco na manutenção da integridade das transações (inclui, altera, exclui) e redução da redundância

1) Baseado em álgebra relacional (junções), abstrai implementação física do BD, permitindo realizar consultas através da Linguagem Estruturada de Consulta (SQL) – recuperar dados distribuídos em várias TABELAS
2) Estrutura básica (rígida e fixa): TABELAS, com instâncias (registros) sendo as linhas (row) e os atributos as colunas (column) (schema)
3) Bom para armazenamento, não necessariamente para ANÁLISE (podem haver consultas complexas) 
4) Relacionamentos são mantidos por identificadores únicos denominados CHAVES: primária (PK) e estrangeira (FK), possuindo o conceito de cardinalidade: um-para-um, um-para-muitos, muitos-para-muitos,...
5) Implementações confiáveis, estáveis, eficientes, permitindo backups incrementais, replicação, clusterização, tolerança a falhas, distribuição de carga entre outros: DB2, Oracle, SQL Server, PostgreSQL, MySQL etc.
6) CODD definiu a NORMALIZAÇÃO e suas FORMAS NORMAIS (FN)
    6) 1FN: atributos atômicos, sem valores repetidos ou possuindo mais de um valor. Por exemplo, CLIENTE = {IDC + END + TELEFONES}, como TELEFONES é multivalorado, ou remove ou cria uma outra TABELA com IDC como chave ESTRANGEIRA

CLIENTES		
IDC	END	TELEFONES
12	Av. Jundiaí, 13	9944-3232, 9932-5089

CLIENTES	
IDC	END
12	Av. Jundiaí, 13

+

TELEFONES		
IDT	IDC	NUMTEL
8	12	9944-3232
9	12	9932-5089

MODELOS DE ARMAZENAMENTO

6.2) 2FN: os atributos não chave devem depender unicamente da PK da tabela em questão. As colunas que não são dependentes apenas da PK são removidas e colocadas em outra tabela

professorCurso = {idProf, idCurso, salario, descricaoCurso}: descricaoCurso não depende de idProf, mas de idCurso

professorCurso			
idProf	idCurso	salario	descricaoCurso
12	10	1.500,00	computação

professorCurso		
idProf	idCurso	salario
12	10	1.500,00

+

cursos	
idCurso	descricaoCurso
10	computação

6.3) 3FN: projetada para melhorar desempenho do BD e minimizar custos de armazenamento. Atributos devem ser independentes funcionalmente uns dos outros.

funcionarios = {idFunc, nome, salario, fgts}: fgts (não chave) depende de salario (não chave), logo pode ser retirado e deixar o cálculo para a camada de negócio ou, se necessário, *ir para outra tabela referenciando funcionários*.

6.4) 4FN: remover multiplicidade de campos multivalorados

Solicitação de Exame

Paciente	Plano de Saúde	Exame
João	Amil	Teor alcoólico
João	Blue-Life	Teor alcoólico
João	Amil	Sangue
João	Blue-Life	Sangue

Solicitação de Exame

Paciente	Exame
João	Teor alcoólico
João	Sangue

Suporte de Seguridade

Paciente	Plano de Saúde
João	Amil
João	Blue-Life

MODELOS DE ARMAZENAMENTO

6.4) 4FN: remover multiplicidade de campos multivalorados

ID_CIDADE (PK)	CIDADE
1	Brasília
2	Rio de Janeiro
3	Campinas

Tabela 8 – Exemplo de 4a. forma normal (relação de cidades).

ID_ANO (PK)	ANO_INGRESSO
1	2014
2	2017
3	2018

Tabela 9 – Exemplo de 4a. forma normal (relação de anos).

ID_PROFESSOR (PK)	NOME	ID_CIDADE (FK)	ID_ANO (FK)
1	Thiago Cavalcanti	1	1
2	Renato da Costa	2	3
3	Felipe Luccas	3	2
4	Adriana Figueiredo	3	2
5	André Castro	1	1

Tabela 10 – Exemplo de 4a. forma normal (relação de professores).

https://www.estrategiaconcursos.com.br/blog/banco-dados-forma-normal/

MODELOS DE ARMAZENAMENTO

7) A PK pode ter valor semântico ou incremental e também pode ser simples ou composta

DESNORMALIZADO

Cliente	Cliente ID	Transação
João	1	Tr. ID: 12890, Data: 14/out/2003, Valor: -87 Tr. ID: 12904, Data: 15/out/2003, Valor: -50
Wilson	2	Tr. ID: 12898, Data: 14/out/2003, Valor: -21
Márcio	3	Tr. ID: 12907, Data: 15/out/2003, Valor: -18 Tr. ID: 14920, Data: 20/nov/2003, Valor: -70 Tr. ID: 15003, Data: 27/nov/2003, Valor: -60

Tabela dos clientes

Cliente	Cliente ID
João	1
Wilson	2
Márcio	3

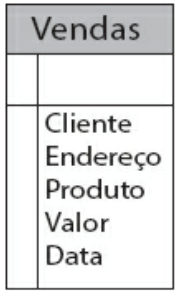
Tabela das transações

Cliente ID	Tr. ID	Data	Valor
1	12890	14/out/2003	-87
1	12904	15/out/2003	-50
2	12898	14/out//2003	-21
3	12907	15/out/2003	-18
3	14920	20/nov/2003	-70
3	15003	27/nov/2003	-60

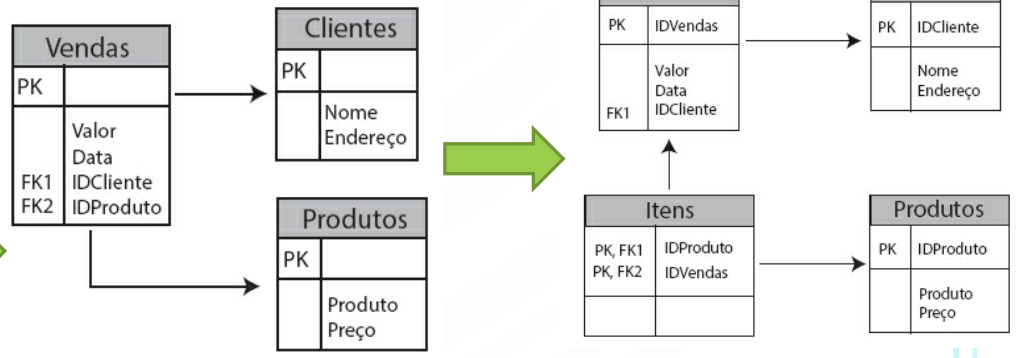
```
SELECT Transacoes.TrID, Clientes.Cliente
FROM Transacoes
INNER JOIN Clientes ON Transacoes.ClienteID = Clientes.ClienteID;
```

Produto e cliente repetidos, não se sabe ao certo se é o mesmo cliente
Como o valor não tem referência única, qual linha tem o valor atual?
Já imaginou a cada compra cadastrar os dados do cliente?

DESNORMALIZADO



Vendas				
Cliente	Endereço	Produto	Valor	Data
José da Silva	Bloco A, Casa 3	Chupeta	5,34	12/09/2014
Maria Cardoso	Rua Brasil, 44	Mamadeira	12,30	13/09/2014
Pedro Henrique	Bloco C, casa 24	Colchão	348,00	12/09/2014
José da Silva	Bloco A, casa 3	Chupeta	6,00	13/09/2014
Pedro H.	Bloco C, casa 24	Mamadeira	12,30	13/09/2014



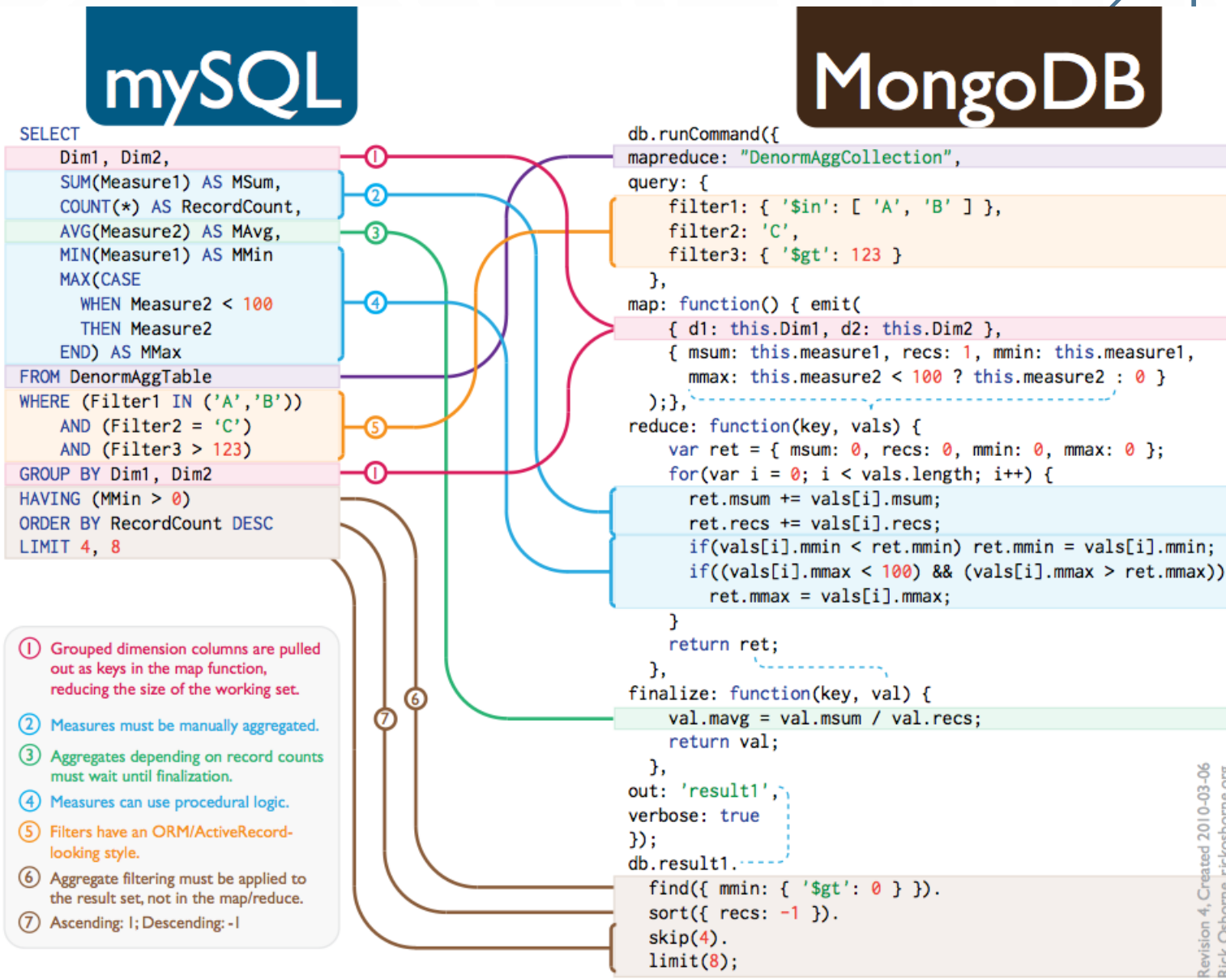
MODELOS DE ARMAZENAMENTO

OBSERVAÇÕES AO MODELO RELACIONAL

- De fato o modelo apresenta boa integridade, reduz redundância, mas possui muitas TABELAS
- Torna assim mais complexa a álgebra das consultas para recuperar dados (para analisar)
- Com o aumento do volume de dados, operações demandam custo computacional elevado
- O desenvolvimento da web passa a demandar armazenamento de dados não estruturados, com escalabilidade e redundância – a partir dos anos 90, o volume aumenta e a dificuldade em analisar consequentemente
- Organizações tinham vários bancos de dados separados e necessitavam agregar estes dados e transformá-los para as operações de análise – surgem os DATA WAREHOUSES
- Os modelos NewSQL ou NoSQL surgem para atender a tais situações, não substituindo o relacional, mas sendo uma alternativa a depender do problema. Para aplicações com demandas de integridade, informações estruturadas e normalizadas, o relacional ainda tem seu espaço
- O NoSQL usa um modelo mais simples, de objetos com atributos, os quais trazem flexibilidade na definição e cada objeto da coleção pode trazer apenas um subconjunto de atributos
- Apropriado para situações onde não é fácil ou possível decompor em conjunto de atributos, como por exemplo texto livre (tweets etc.) ou imagens – contudo necessitam serem colocados em algo estruturado para análise

NOSQL (NOT ONLY SQL)

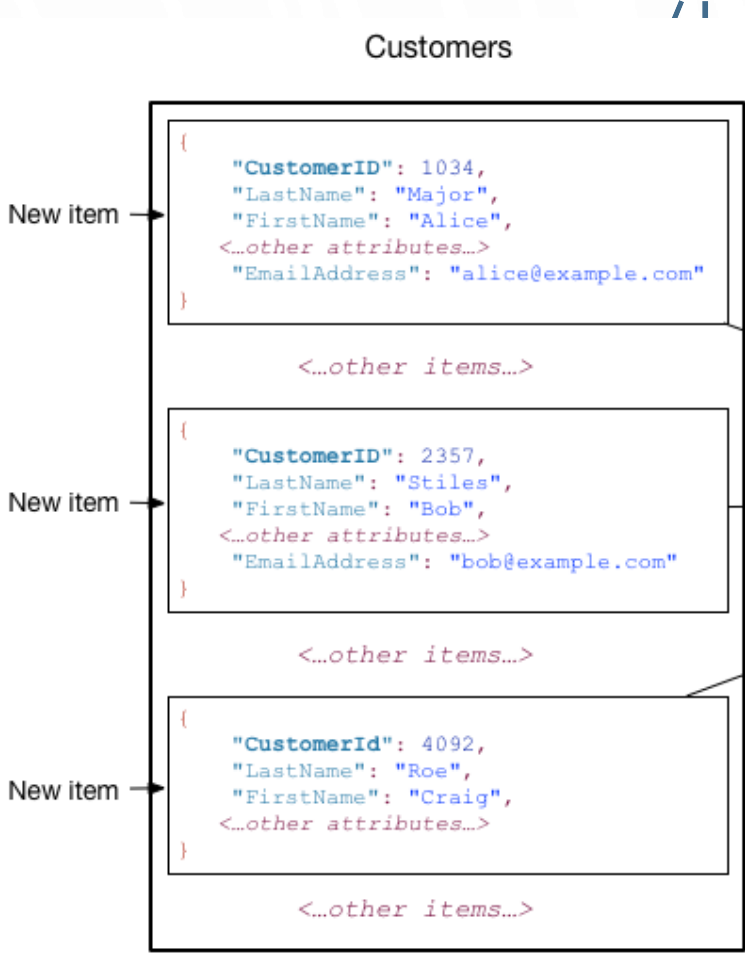
- Aplicações para agregar grandes volumes de dados (gestão de documentos, séries temporais, feed de notícias, dados em painéis em tempo real
- Problema não só de R/W, mas o simples armazenamento em si
- Como armazena-se +1bi de usuários do Facebook e o conteúdo que gerem diariamente?
- Produtos baseados em não normalização, poucas restrições de integridade e controle mínimo de transações
- Contudo também podem processar dados altamente estruturados

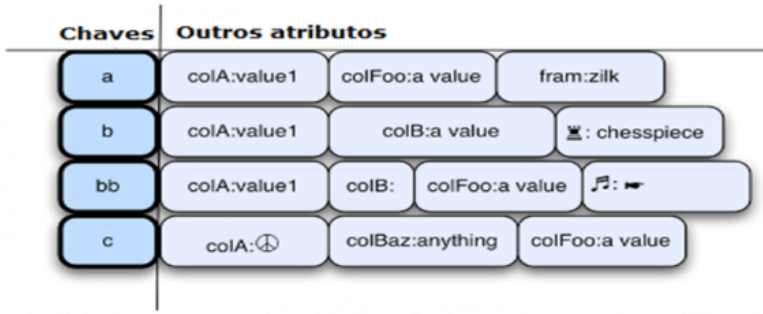


NOSQL - KVS

- TIPOS:
- Key-Value Store (KVS) – (chave-valor) todos os registros fazem parte da mesma coleção de elementos, e a única coisa que todos eles tem em comum é a chave única; invés de incluir um conjunto de atributos, a operação insere apenas uma chave e um valor (Couchbase, Kyoto Cabinet, Redis (open source), DynamoDB - Amazon) – **BD as a service**
- **Estrutura mais SIMPLES**
- **Grande tabela hash (pareamento)**
- Base do tipo DOCUMENTOS – grupo de coleções

NoSQL	Relacional
Esquema	
Não há necessidade de um esquema fixo, o que dá uma maior liberdade de armazenamento.	Tem que ser definido antes de qualquer operação, limitando o armazenamento.
Relação	
Sem relações, a informação é armazenada como um agregado, onde um único arquivo possui tudo sobre a transação.	As relações são estabelecidas por conexões entre tabelas.
Distribuição	
Múltiplos computadores podem armazenar dados de uma mesma base de dados.	Múltiplos computadores podem armazenar e processar dados, porém se especificados.





Chave	Valor
16	nome = NoSQL Essencial, ano - 2014
Connor	idade = 22, interesse = programação
2	nome = True Blood, gênero = fantasia, classificação = 16 anos
Laís	ocupação = estudante

redislabs
HOME OF REDIS

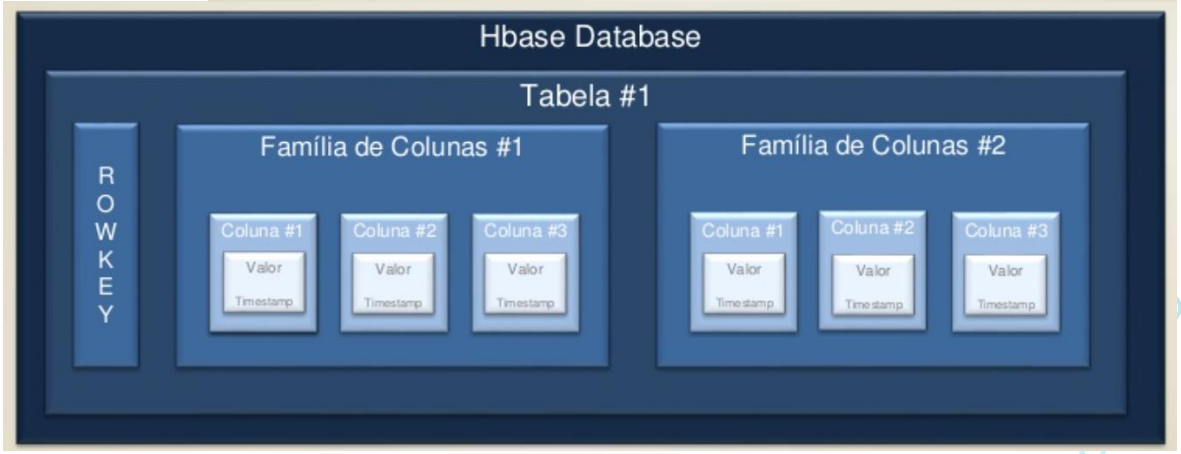
```
import redis
r = redis.Redis(
host= ‘redis.us-east-1-1.ec2.cloud.redislabs.com’,
port= ‘12345’,
Password = ‘thisismypassword’)
r.set(‘foo’,’bar’)
r.get(‘foo’)
```

NOSQL - COLUNAR

- TIPOS:
- Colunas Ordenadas (Colunar)
- Baseado no modelo BigTable do Google, onde o dado é orientado por coluna ao invés de linha. São implementações deste modelo o Hbase (Apache), o HyperTable da Clouddata
- Bancos de dados colunares, de coluna larga ou de famílias de colunas armazenam dados de modo eficiente, consultam linhas de dados esparsos são vantajosos ao consultar em colunas específicas no banco de dados.
- Projeto baseado no que se quer consultar, quais dados são obtidos juntos?
- Desnormalização: evitar JOINS – todos os dados da app numa linha ou conjunto de linhas (regiões – faixas contínuas de rowkeys)

RowKey	FC1			FC2		
	col1	col2	col3	col1	col2	col3
row1	Valor		Valor		Valor	Valor
row2		Valor		Valor		
row3	Valor	Valor		Valor		Valor
row4			Valor		Valor	
row5	Valor					Valor
row6	Valor	Valor	Valor	Valor	Valor	Valor

Roda sobre o HDFS, integrado ao Hadoop e escala linearmente para lidar com grandes conjuntos de dados com bilhões de linhas e milhões de colunas e combina Facilmente fontes de dados que utilizam uma grande variedade de estruturas e esquemas diferentes.



NOSQL - COLUNAR

RowKey	Blog:t1	Blog:t2	....	Blog:t3
userid1	A Amazônia está	O cenário econômico	....	O Campeonato carioca
userid2	Receita de bolos ..	Sobremesas Diet	....	Pratos leves ...
userid3	Projetos de Arduíno	Conhecendo IOT	....	
userid4	Esportes radicais ...	Triathlon	....	Stand up Paddle

1) Todos os blogs armazenados numa única RowKey (modelo Flat-Wide, poucas linhas muitas colunas); Aqui cada blog tem em seu valor o timestamp.
2) oposto ao Tall-narrow, neste caso, com um blog por linha, e o ROWKEY é o userID+timestamp (+detalhe)

RowKey	Blog
userid1_t1	A Amazônia está ....
userid1_t2	O cenário econômico
userid1_t3	O Campeonato carioca
userid2_t1	Receita de bolos ..
userid2_t2	Sobremesas Diet
userid2_t3	Pratos leves ...
userid3_t1	Projetos de Arduíno
userid3_t2	Conhecendo IOT
userid3_t3	Raspberry PI 3
userid4_t1	Esportes radicais ...

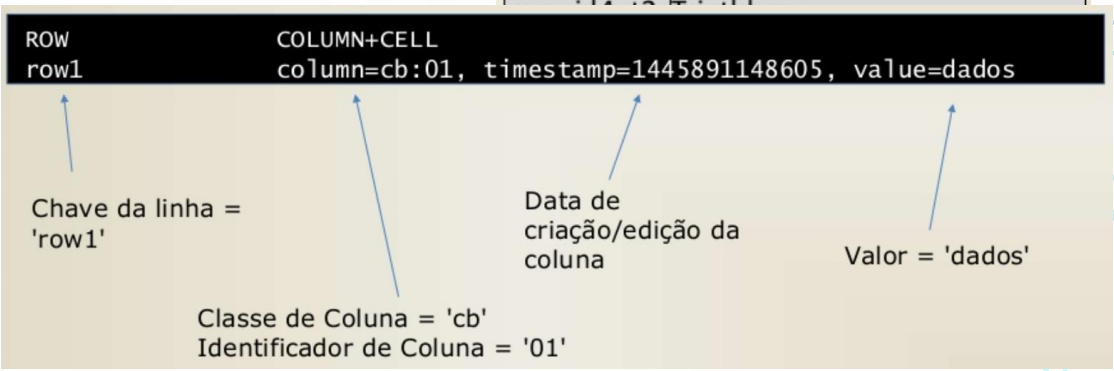
Estas chaves normalmente estão distribuídas no nós da rede
Desnormalizar neste caso acelera leitura mas é custoso para UPDATE

- CRUD, com operações de CREATE TABLE, PUT, GET, SCAN
- Formas de acesso:
  - Hbase shell – (em JRuby)
  - API Java, REST, integração com Spark, Hive, Pig

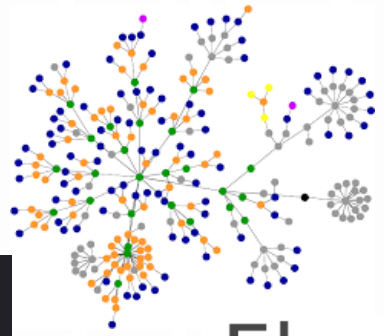
```
hbase(main):002:0> create 'tab1', 'familia1', 'familia2'
0 row(s) in 1.3390 seconds

=> HBase::Table - tab1
```

```
hbase(main):010:0> put 'tab1', 'linha1', 'familia1:colunaA', 'aaaaa'
0 row(s) in 0.0750 seconds
```

```
hbase(main):002:0> get 'test', 'row1'
COLUMN                                  CELL
 cb:01                                  timestamp=1445891148605,
value=dados
1 row(s) in 0.0070 seconds
```



NOSQL - GRAFOS

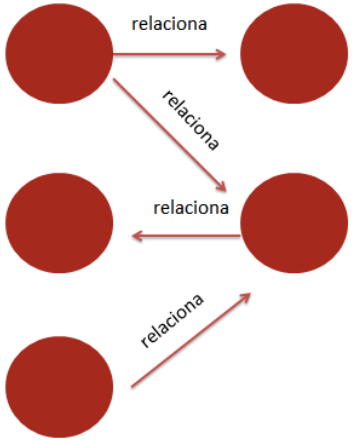


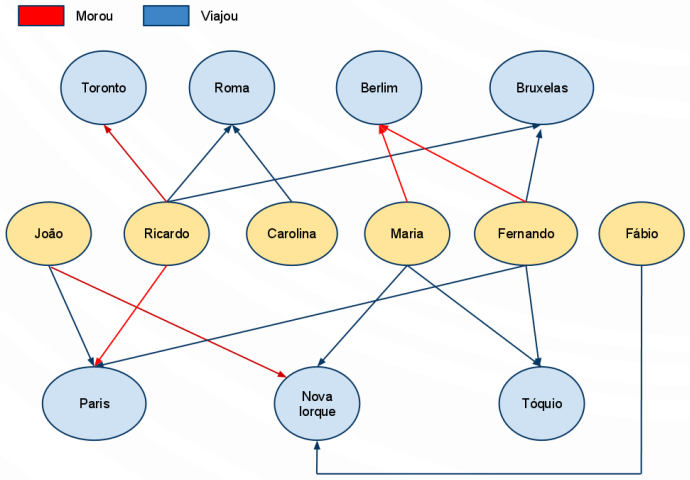
- A grande diferença para o modelo clássico está na representação explícita de relacionamentos entre os **dados**, através de um modelo com vértices, chamados de nós, e arcos ou arestas, chamados de relações
- Para representar relacionamentos complexos, o grafo é eficaz, em parte pela eficiência de seus algoritmos de BUSCA
- Suas inúmeras aplicações na medicina, computação, genética, economia, matemática abstraem os algoritmos e implementações

OrientDB Community Edition

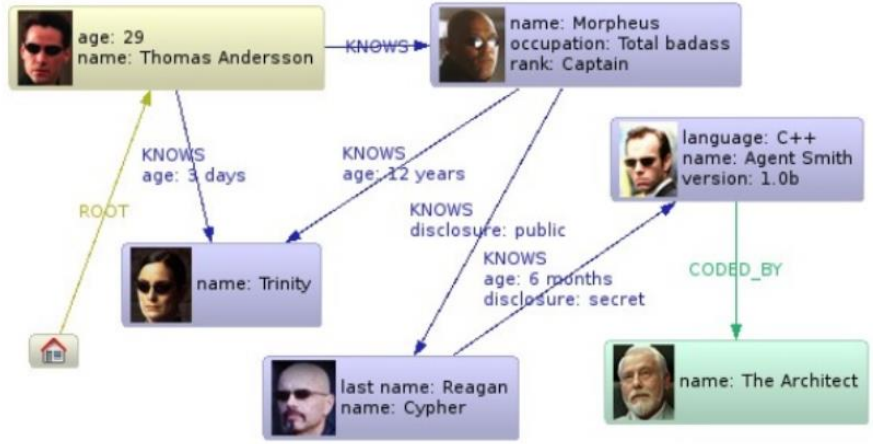
FlockDB

neo4j



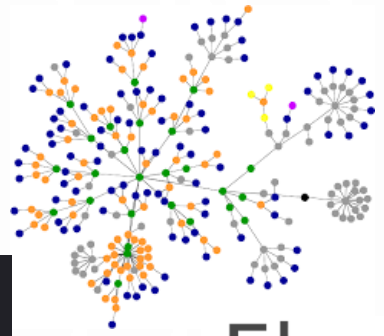


1) Nós e arestas possuem ATRIBUTOS
2) Relacionamentos possuem DIREÇÃO
3) São particularmente interessantes em semântica WEB



NOSQL - GRAFOS

- Itens de um BD de Grafos
  - Vértice (Nó) – conjunto de propriedades chave:valor que representam uma entidade. Exemplo, um usuário do Twitter
    ```
    nome: 'Joaquim Miguel'
    arroba: 'Trinity'
    ```
  - Arestas – são os relacionamentos. Ligam os vértices por meio de ligações semânticas. Exemplo baseado no Twitter: nó 'Priscila' segue o nó 'Lucas' desde 2012 – o nome do relacionamento é 'segue', que possui um sentido: 'Priscila'->'Lucas' e dados: data de início do relacionamento
  - Usualmente acesso por API REST; no caso do neo4j, linguagem Cypher



OrientDB Community Edition

FlockDB

neo4j

AllegroGraph, ArangoDb, Bitsy, DEX, InfiniteGraph, InfoGrid, Oracle Spatial, HyperGraphDB, Titan...

Tabela 1. Comparação sintática entre SQL e Cypher.

Cláusula	SQL	Cypher
Seleção	`SELECT * FROM usuario;`	`MATCH (u:usuario) RETURN u;`
Filtragem	`SELECT * FROM usuario WHERE username LIKE 'alvstricklan8196';`	`MATCH (u:usuario) WHERE u.username = 'alvstricklan8196' RETURN u;`
Junção	`SELECT * FROM usuario u JOIN post p ON u.username = p.usuario;`	`MATCH c=(u:usuario)-[:PUBLICA]->(p:post) RETURN c;`
Agrupamento	`SELECT u.username, count(*) FROM usuario u JOIN post p ON u.username = p.usuario GROUP BY u.username;`	`MATCH (u:usuario)-[:PUBLICA]->(p:post) RETURN u.username, count(u);`
Ordenação	`SELECT * FROM usuario u ORDER BY u.nome DESC;`	`MATCH (u:usuario) RETURN u ORDER BY u.nome DESC;`

```
start programmer=(3)
match (programmer)-[:PAIRED]->(pair)
where pair.age > 30
return pair order by pair.age skip 5 limit 10
```

Comparativo postgresql/node4js

DB-Engines Ranking - popularity ranking of database management systems (db-engines.com)