DIGITAL
INNOVATION
ONE

Aceleração Global Dev #4 everis

Explorando o poder do NoSQL com Apache Cassandra e Apache HBase

Valdir Sevaios
Expert Data Analyst

DIGITAL INNOVATION ONE

Objetivos da Aula

1. Conceito NoSQL e porquê utilizamos no Big Data?
2. Introdução Apache Hbase e Apache Cassandra
3. Comandos (Definição de dados, Manipulação e Segurança)
4. Cenários de utilização NoSQL no BigData
5. Operações de carga massiva
6. Integrações NoSQL com Ambiente Hadoop
7. Próximos passos

DIGITAL INNOVATION ONE

Requisitos Básicos

- ✓ VM disponibilizada pela Everis

DIGITAL
INNOVATION
ONE

1. Conceito NoSQL e porquê utilizamos no Big Data?

NoSQL com Hbase e Cassandra

DIGITAL INNOVATION ONE

Introdução NoSQL

Significados:

NoSQL = “No SQL”, “Não SQL” ou “Não Relacional”

NoSQL = “Not Only SQL”

Termo genérico para representar os bancos de dados não relacionais.

O NoSQL emergiu como uma alternativa de banco de dados não relacionado, normalmente evitando operações de “join”, é distribuído, open-source, escalável na horizontal, livre de modelagens ou schema (não é necessário fixar modelos para as tabelas), suporta replicação, acesso via API de operações e eventualmente consistente.

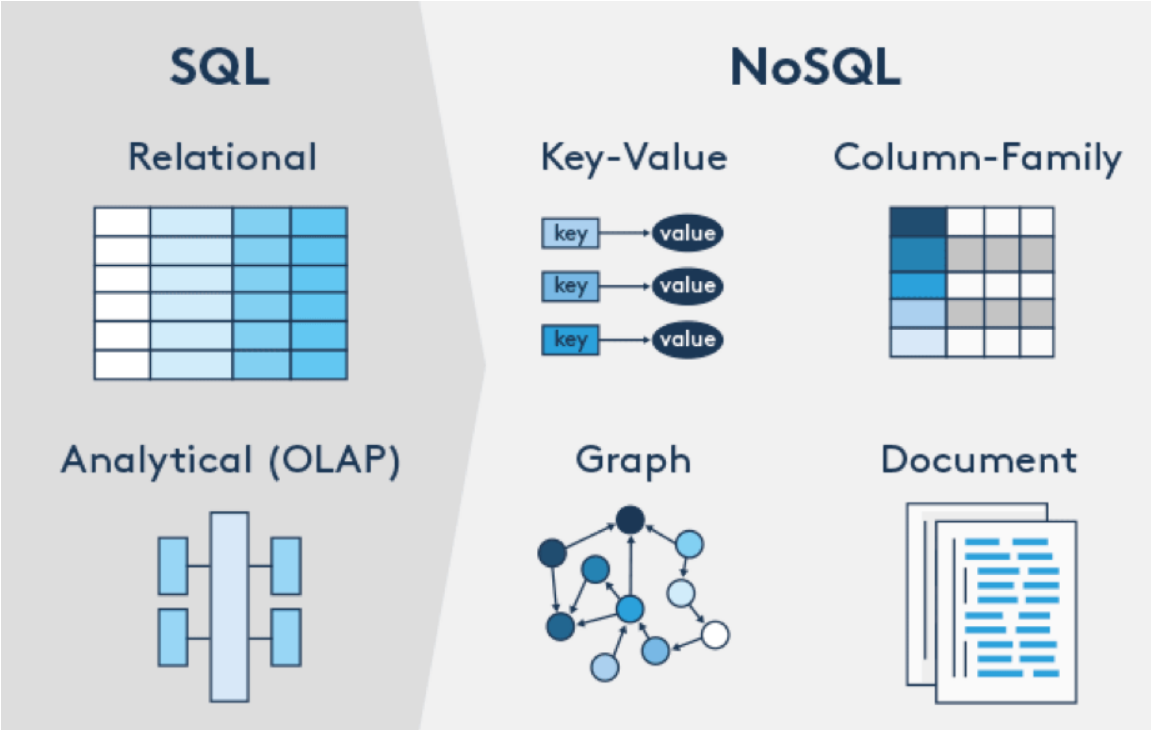
DIGITAL INNOVATION ONE

Relacional vs NoSQL

Quando considerar NoSQL	Quando considerar Relacional
Carga de trabalho de alto volume que exigem grande escala	Carga de trabalho é consistente e requer escala média para grande
Carga de trabalho não exigem garantias do ACID	Garantias de ACID são necessárias
Os dados são dinâmicos e frequentemente alterados	Dados são previsíveis e altamente estruturados
Os dados podem ser expressos sem relações (joins)	Os dados são expressos de maneira relacional
Alto velocidade de gravação e a segurança de gravação não é crítica	A garantia de gravação é um requisito
Consulta de dados é simples e tende a ser simples	Consultas e relatórios complexos
Dados exigem uma ampla distribuição geográfica	Usuários são mais centralizados

DIGITAL INNOVATION ONE

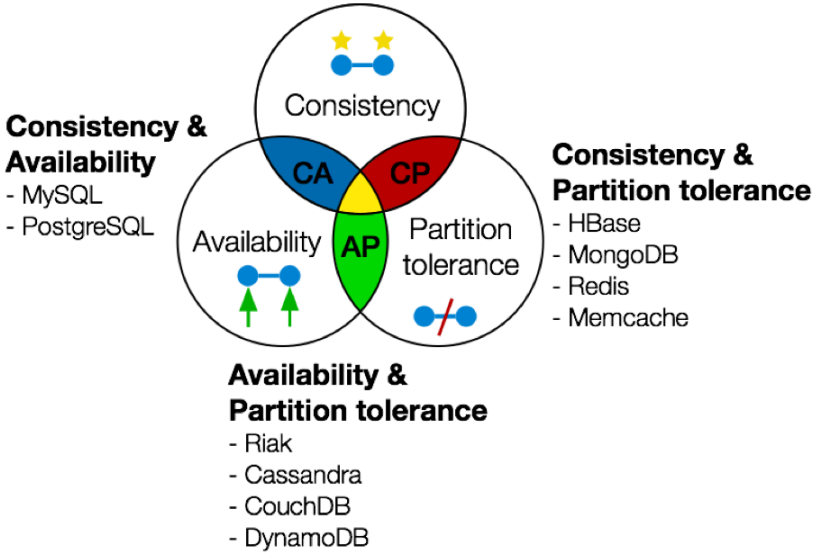
Tipos de NoSQL



Teorema de CAP

O teorema CAP ou teorema de Brewer indica que o armazenamento de dados distribuídos só podem atender dois dos três atributos: **Consistência, Disponibilidade, Partição Tolerante a Falhas**.

Tanto Hadoop e HBase atendem CP, porque possuem um ponto de falha que é respectivamente o NamedNode e HMaster que não possuem redundância dos dados dos próprios serviços para todos os nós do cluster.



DIGITAL INNOVATION ONE

Por quê NoSQL no Big data?

Limitações do Hadoop

- Hadoop (MapReduce) pode executar apenas processamento batch e os dados são acessados de forma sequencial, isso significa que é necessário percorrer todo o conjunto de arquivos (*scan search*) mesmo para os jobs mais simples.

Cenário

- Um grande conjunto de dados (*dataset*) quando processado com um outro conjunto de dados, ambos serão processados de maneira sequencial, nesse momento **uma nova solução é necessária para acessar qualquer ponto do *dataset (linhas)* e que leve um tempo menor para retornar.**
- Aplicações como HBase, Cassandra, Dynamo e MongoDB, etc, são banco dados que armazenam grandes quantidade de dados e os acessos à esses dados são realizados de forma **aleatória em termos de posição do registro e do tempo**.

DIGITAL
INNOVATION
ONE

2. Introdução HBase e Cassandra

NoSQL com Hbase e Cassandra

DIGITAL INNOVATION ONE

O que é o Apache Hbase?

HBase é um banco de dados **distribuído** e
orientado a colunas (Column Family ou Wide Column).

Uma definição mais técnica:

O armazenamento do HBase é um esparso, distribuído, persistente, multidimensional e ordenado **Map**.

As maiores desvantagens do Hbase é não ter uma linguagem própria de SQL, não suportar índices em colunas fora do rowkey e não suportar tabelas secundárias de índices.

A maior vantagem é a facilidade e integração com o ecossistemas Hadoop.

Cont. O que é o Apache Hbase?

Map é indexado por uma linha chave **(row key)**, coluna chave **(column key)**, e um coluna timestamp.

Cada valor no **Map** é interpretado como um vetor de bytes **(array of bytes)**.

O **array of bytes** nos permite gravar portanto qualquer informação se for necessário, inclusive documentos, arquivos JSON, CSV, etc.

Cont. O que é o Apache Hbase?

Portanto podemos entender que o núcleo de dados do HBase é um Map.

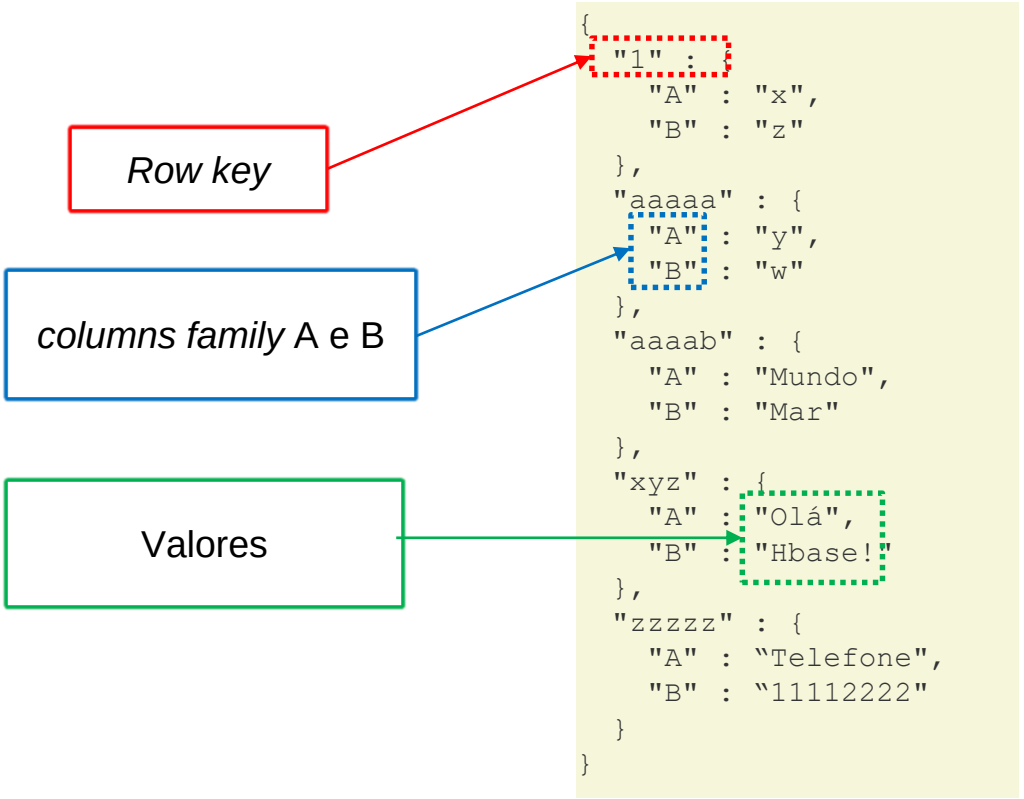
Na maioria das linguagens de programação essa abstração de estrutura de dados existe e pode ser representada como um conjunto de chaves e conjunto de valores. **Cada chave é associada à um valor.**

```
{
  "zzzzz" : "Olá",
  "xyz" : "hello",
  "aaaab" : "world Hbase!",
  "1" : "x",
  "aaaaa" : "y"
}
```

Cont. O que é o Apache Hbase?

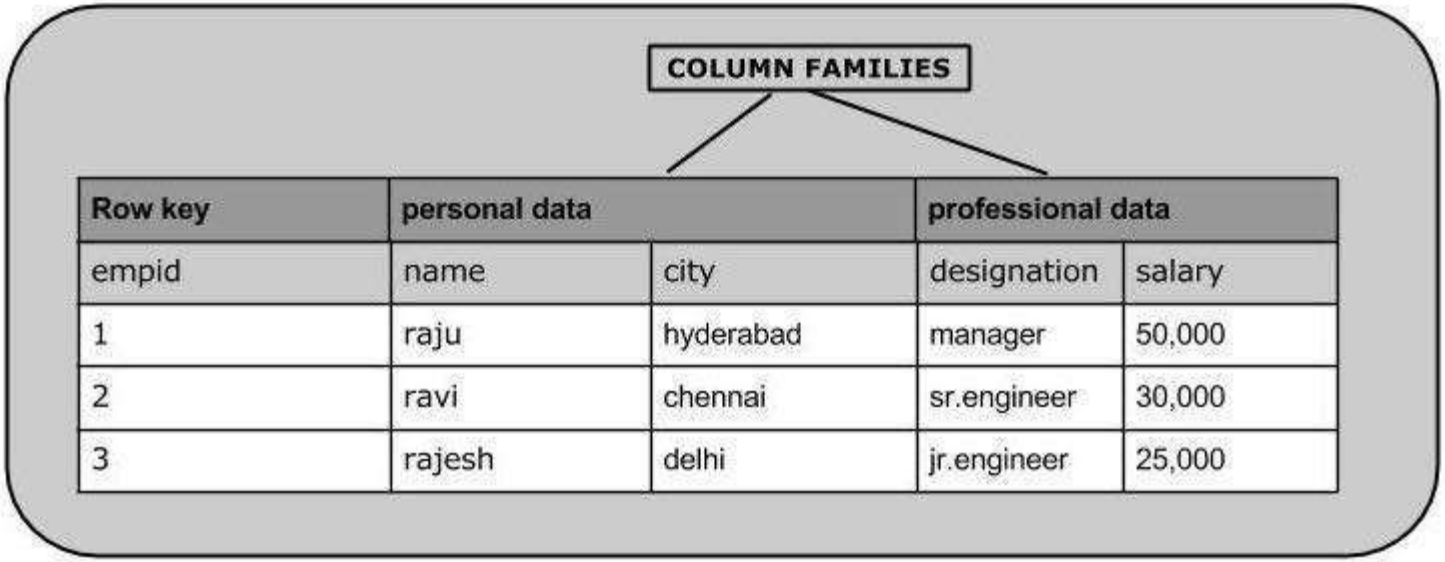
Multidimensional e Ordenado

Esquecendo o conceito tradicional de linhas e colunas do mundo relacional, pense no multidimensional como um **Map** que contém **Maps**.



Cont. O que é o Apache Hbase?

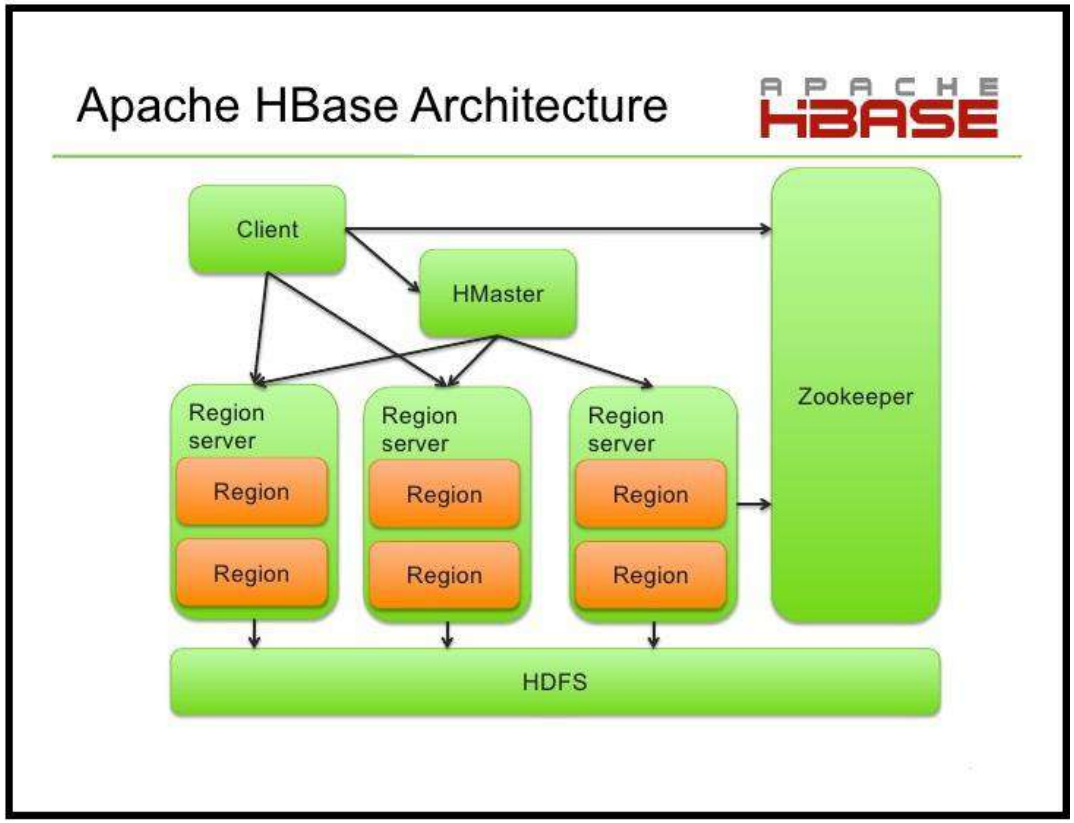
Representação Tabular dos dados no HBase.



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

DIGITAL INNOVATION ONE

Arquitetura do HBase



DIGITAL INNOVATION ONE

Cont. Arquitetura HBase

HBase Client

Responsável por encontrar o RegionServers que estão atendendo as linhas em particular que estão sendo utilizadas, para isso é consultado uma base de dados de metadados interna do Hbase, **hbase:meta** que fica no Zookeeper.

Depois de localizado o RegionServer, o client se comunica para solicitar requisição de leitura/gravação do registro, isso até versão 2.x.y.

Apartir da versão 3.x.y. o HMaster controla as requisições leitura/gravação.

Caso ocorra erro na comunicação com o RegionServer por alguma falha será atribuído pelo load balancer do HMaster um novo RegionServer.

DIGITAL INNOVATION ONE

Cont. Arquitetura HBase

HMaster

Responsável por monitorar todos as instâncias de RegionServer no cluster. É meio para todas as solicitações de mudanças de metadados.

Em um ambiente distribuído de produção esse serviço é executado no NamedNode do Hadoop.

É possível ter vários nós de um ambiente clusterizado atuar como master, porém só um pode ficar ativo, o restante fica passivo, caso ocorra alguma falha no master principal.

DIGITAL INNOVATION ONE

Cont. Arquitetura HBase

RegionServer

Responsável por monitorar todos as instâncias de RegionServer no cluster. É meio para todas as solicitações de mudanças de metadados.

Em um ambiente distribuído de produção esse serviço é executado no NamedNode do Hadoop.

É possível ter vários nós de um ambiente clusterizado atuar como master, porém só um pode ficar ativo, o restante fica passivo, caso ocorra alguma falha no master principal.

Cont. Arquitetura HBase

Regions

Representam os elementos básicos de disponibilidade e distribuição das tabelas e incluem o armazenamento de cada column family.

Hierarquia dos Objetos no HBase

```
Table (Tabela HBase)
    Region (Regiões da tabela)
      Store (Unidade de armazenamento por ColumnFamily para cada região da tabela)
        MemStore (MemStore para cada armazenamento em cada região da tabela)
        StoreFile (StoreFiles para cada armazenamento em cada região da tabela)
          Block (Arquivos que servem como blocos dentro do StoreFile dentro do armazenamento em cada
região de uma tabela)
```

DIGITAL INNOVATION ONE

Cont. Arquitetura HBase

Apache Zookeeper

O tipo de instalação distribuída do Hbase é necessário que o Apache Zookeeper esteja funcionando no cluster.

O Apache Zookeeper é o responsável por dar visibilidade a todos os nós de serviços do HBase de quem é o master atual, são os servidores que atendem ao papel RegionServers, Region. Toda configuração que é padrão para cada um dos papéis são armazenados no Zookeeper, portanto ele tem um papel fundamental de manter sincronizado toda a parametrização em comum para todos os nós.

O cliente HBase se comunica com os RegionServer através do Zookeeper.

Cont. Arquitetura HBase

Apache HDFS

É o sistema de arquivos do ecossistema do Hadoop. Cada arquivo está armazenado em múltiplos blocos e mantem tolerância a falhas, os blocos são replicados pelos cluster Hadoop.

HDFS é utilizado pelo componentes do HBase.

Os arquivos gerenciados pelo HBase são criados dentro do HDFS.

HBASE	HDFS com Hive/MR
Baixa latência nas operações	Alta latência nas operações
Acesso a leitura e gravação aleatória	Grava única e leitura muita vezes
Acessado através de comandos shell, cliente API em Java, REST, Avro ou Thrift	Acessado primeiramente através dos Jobs do MR(Map Reduce)

DIGITAL INNOVATION ONE

O que é o Apache Cassandra?

O Cassandra é um banco de dados **distribuído** e **orientado a colunas** (Wide Column).

Diferente do Hbase, os dados armazenados são tipados e há conceitos mais complexos de modelagem como chave primária composta, partition key e cluster key.

O Cassandra possui a linguagem SQL (**CQL**) contém semelhante com SQL ANSI porém algumas operações não são suportadas / recomendadas, por exemplo: joins, alguns tipos de agrupamentos e tipos de filtros.

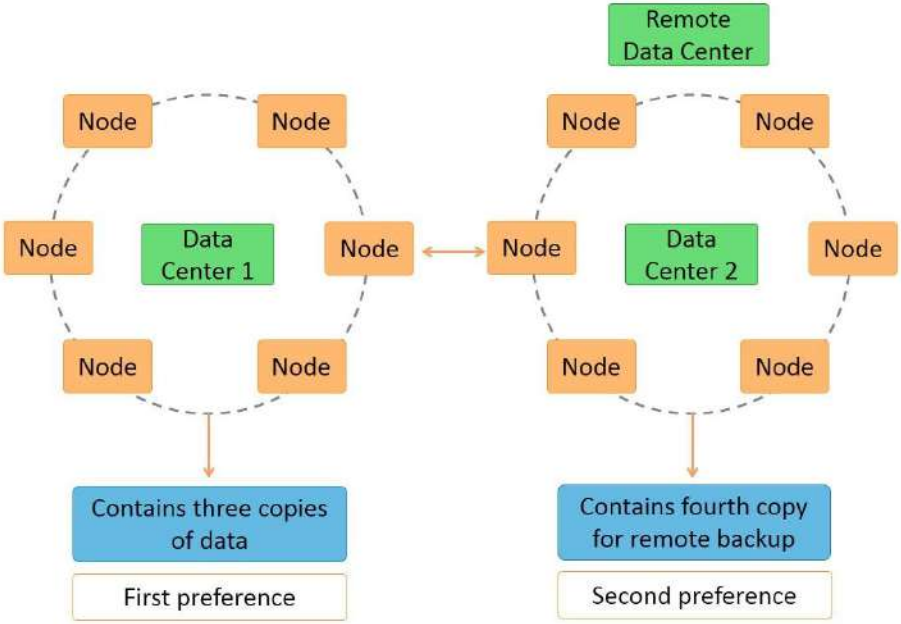
A recomendação para modelagem de dados no Cassandra, é pensar em quais query devem ser consumidas e agregar as informações em uma determinada tabela.

Uma grande diferença para o Hbase, é que o **Cassandra suporta tabela secundárias de índices e permite filtros em colunas fora da primary key.**

Arquitetura do Cassandra

A característica principal do Cassandra é armazenar em múltiplos nós sem nenhum ponto de falha.

Conexão entre os nós é realizada de ponto a ponta, utilizando um protocolo chamado **Gossip**.

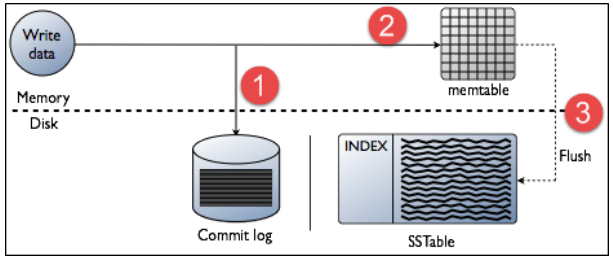


Componentes do Cassandra

Node
Nó responsável por armazenar os dados, é um componente básico da Arquitetura.

Data Center
Representa uma coleção de nós.

Cluster
Representa a coleção de vários Data Centers.



Commit Log
Cada operação é escrita no log de commit. Esse logo é utilizado para recuperação em caso de incidents graves.

Mem-table
Depois que o dado é escrito no log de commit, o dado é escrito no Mem-table. O dado nessa localidade é temporário.

SSTable
Quando o Mem-Table atingi um limite, o dado é gravado no disco no SSTABLE.

DIGITAL
INNOVATION
ONE

3. Comandos gerais

NoSQL com Hbase e Cassandra

DIGITAL INNOVATION ONE

Comandos no HBase

hbase shell

No console do HBase é possível utilizar todos os comandos de manipulação de informação e comandos gerais.

Atualmente o HBase até a versão 3.x.y não suporta uma query de consulta (SQL).

Os comandos mais básicos de manipulação das informação são: put, get, scan, drop, disable, etc.

DIGITAL INNOVATION ONE

Cont.Comandos no HBase

hbase shell

No console do HBase é possível utilizar todos os comandos de manipulação de informação e comandos gerais.

Atualmente o HBase até a versão 3.x.y não suporta uma query de consulta (SQL).

Os comandos mais básicos de manipulação das informação são: put, get, scan, drop, disable, etc.

DIGITAL INNOVATION ONE

Cont.Comandos no HBase

status

Comando dá detalhes sobre o sistema como o número de servidores presente, quantidade de servidores ativos, média de carga, quantidade de Stored ativos, é possível passar qualquer parâmetro dependendo em qual detalhe seja necessário saber sobre o sistema.

Variações

status 'simple'

status 'summary'

status 'detailed'

DIGITAL INNOVATION ONE

Cont.Comandos no HBase

version

Exibe a versão do HBase.

table_help

Comando que auxilia como utilizar comandos que se referenciam a uma tabela.
É possível pegar a referencia de uma tabela cria e utilizar como variável para próximas operações.

DIGITAL INNOVATION ONE

Cont.Comandos no HBase

Os comandos abaixo são utilizados para operar as tabelas no HBase.

create – Cria uma tabela.
list – Lista todas as tabelas no HBase independente do namespace.
disable – Desabilita uma tabela.
is_disabled – Checa se uma tabela está desabilitada.
enable – Habilita uma tabela.
is_enabled – Checa se uma tabela está habilitada.
describe – Exibe informações de definição de uma tabela.
alter – Realiza alterações em uma tabela.
exists – Verifica se uma tabela existe.
drop – Exclui um tabela do HBase.
drop_all – Exclue todas as que se aplicam a um padrão de nomes via regra de Regex.

Cont.Comandos no HBase

CREATE_NAMESPACE

Permite criar uma namespace no HBase.

Sintaxe:

create_namespace '<namespace>' {PROPRIEDADES}

Exemplo:

create_namespace 'empresa'

Cont.Comandos no HBase

CREATE

Permite criar uma tabela no HBase.

Sintaxe:

create '[<namespace>]:<nome tabela>', '<nome da column family>' {PROPRIEDADES}

Exemplo:

Imagine uma tabela de funcionarios

Row Key	Dados Pessoais	Dados Profissional

Cont.Comandos no HBase

Exemplo criação da tabela de funcionários

Comando:

create 'funcionario', 'pessoais', 'profissionais'
create 'funcionario', {NAME=>'pessoais', VERSIONS=>5}, {NAME=> 'profissionais', VERSIONS =>4}
Liste todas as tabelas

list

Com o comando list é possível visualizar todas as tabelas presentes e criados no HBase.
É possível passar expressões regulares para realizar buscas mais personalizadas.

Cont.Comandos no HBase

DESCRIBE

O comando exibe informações sobre as column families presentes na tabela mencionada, também traz informações sobre os filtros associados, versões, se a tabela está em memória, etc.

Sintaxe:

describe '<nome da tabela>'

Comando:

describe 'funcionario'

Cont.Comandos no HBase

DISABLE

Desabilita a tabela mencionada, caso seja necessário que uma tabela seja deletada ou excluída, primeiro é necessário desabilita-la.

Após desabilitada ainda é possível lista-la (list) em conjunto com as demais tabelas e checar sua existência com comando exists, porém não é possível mais escanear (scan).

Sintaxe:

disable '<nome da tabela>'

Comando:

disable 'funcionario'

Cont.Comandos no HBase

DISABLE_ALL

Desabilita as tabelas que atendem dentro do critério de expressão regular.

Para evitar interrupções importantes esse comando tem uma confirmação manual (Sim ou Não) antes de efetivar a desabilitação.

Sintaxe:

disable_all '<prefixo da tabela>.*'

Comando:

disable_all 'func.*'

Cont.Comandos no HBase

ENABLE

Habilita a tabela mencionada no comando. Se a tabela está desabilitada no primeiro momento e não foi deletada ou excluída, e desejamos reutilizar a tabela, então primeiro nos precisamos habilita-la novamente.

Sintaxe:

enable '<nome da tabela>'

Comando:

enable 'funcionario'

Cont.Comandos no HBase

DROP

Para deletar ou dropar uma tabela presente no HBase, primeiro é necessário desabilita-lá com o comando disable.

Sintaxe:

drop '<nome da tabela>'

Comando:

drop 'funcionario'

Cont.Comandos no HBase

DROP_ALL

Esse comando excluirá todas as tabelas que estiverem com o nome dentro da regra da expressão regular.
Todas as tabelas que serão excluídas, precisam estar na situação de desabilitadas, portanto ter passado pelo comando disable_all.

Sintaxe:

drop_all '<expressão regular>'

Comando:

drop_all 'func.*'

Cont.Comandos no HBase

IS_ENABLED

Esse comando só verificará se a tabela está habilitada ou não. Caso esteja desabilitada é necessário acionar o outro comando enable.

Sintaxe:

is_enabled '<nome da tabela>'

Comando:

is_enabled 'funcionario'

Cont.Comandos no HBase

ALTER (Acrescentando column family e limitando versões da coluna)

Esse comando alterará a definição (schema) da família de colunas (column family) de uma tabela especificada.

Alterar uma única ou múltiplas column family.
Deletar column family
Diversas operações são permitidas utilizando atributos específicos de definição de uma tabela.

Sintaxe:

alter '<nome da tabela>', NAME=><column familyname>, VERSIONS => 5

Exemplo comando abaixo para limitar armazenamento até 5 versões da column family

alter 'funcionario', NAME => 'hobby', VERSIONS => 5
alter 'funcionario', 'delete'=>'hobby' (Para deletar)

Cont.Comandos no HBase

ALTER_STATUS

Com esse comando é possível acompanhar com o status das alterações realizada de uma tabela para todos nós RegionServer.

Sintaxe:

alter_status '<nome da tabela>'

Comando:

alter_status 'funcionario'

Cont.Comandos no HBase

Os comandos abaixo são utilizados para operar os dados no HBase.

put – Insere/Atualiza um valor em uma determinada célula de uma específica linha de uma tabela.
get – Consulta todo o conteúdo de uma linha ou célula em uma tabela.
delete – Exclui um valor de uma célula em uma tabela.
deleteall – Exclui todas as células de uma linha em específico.
scan – Varre toda a tabela retornando as dados contidos.
count – Conta e retorna o número de linhas em uma tabela.
truncate – Desabilita, exclui e recria uma tabela em específico.

IMPORTANTE: **Todas as operações de CRUD lista acima estão disponíveis via API Java de utilização sob o pacote org.apache.hadoop.hbase.cliente com os objetos Htable Put e Get.**

Cont.Comandos no HBase

PUT

Comando para inserir um determinado valor em uma célular na coluna ou linha.
Também utilizado para atualizar um determinado valor de uma célula para um determinado rowKey e ColumnFamility:colname já existente.

Sintaxe:

```
put '<nome da tabela>', '<rowkey>', '<columnfamily:colname>', '<valor>'
```

Comando:

```
put 'funcionario', '1', 'pessoais:nome', 'Maria'
put 'funcionario', '1', 'pessoais:cidade', 'São Paulo'
put 'funcionario', '1', 'pessoais:cidade', 'Belo Horizonte'
put 'funcionario', '2', 'profissionais:empresa', 'Everis'

scan 'funcionario', {VERSIONS => 3}
```

*É acionado **upsert** da chave/valor.*

Cont.Comandos no HBase

GET

Comando retorna um determinado valor em uma célula na coluna ou linha inteira do rowKey.

Sintaxe:

get '<nome da tabela>', '<rowkey>', [parâmetros opcionais]

Comando:

get 'funcionario', '1'
get 'funcionario', '1', {COLUMN => 'pessoais:cidade'}

Consultar todas as versões de uma column para um rowkey específico
get 'funcionario', '1' , {COLUMN => 'pessoais:cidade', VERSIONS=> 3}

Cont.Comandos no HBase

COUNT

Comando recupera a quantidade de linhas de uma determinada tabela.
O Intervelo de contagem de linhas pode ser especificado de forma opcional.
A contagem de linhas é exibida a cada 1000 linhas.

Sintaxe:

count '<nome da tabela>', CACHE => 1000

Comando:

count 'funcionario', CACHE=> 1000

Cont.Comandos no HBase

DELETE

Comando remove um determinado valor em uma célula na coluna informadas, também possível remover todas as células de uma rowKey especificada.

Sintaxe:

delete '<nome da tabela>', '<rowkey>', '<column name>'
delete <'nome da tabela>', '<rowkey>', '<column name>', '<timestamp'>
deleteall '<nome da tabela>','<rowkey>'

Comando:

delete 'funcionario', '1', 'pessoais:cidade'
delete 'funcionario', '1', 'pessoais:cidade', 129019101
deleteall 'funcionario', '1'

Cont.Comandos no HBase

SCAN

Comando remove um determinado valor em uma célula na coluna informadas, também possível remover todas as células de uma rowKey especificada.

Sintaxe:

scan '<nome da tabela>', '[parâmetros opcionais]'

Comando:

Exibe as ultimas versões de cada rowKey e respectivas colunas
scan 'funcionario'

Exibe todas as últimas 10 versões de cada rowKey e respectivas colunas
scan 'funcionario', {RAW=>true, version=>10}

Cont.Comandos no HBase

TRUNCATE

Comando remove todas as linhas e colunas presentes na tabela.
Internamente o comando realizada a desabilitação da tabela, drop a tabela se ainda está presente, e recria.

Sintaxe:

truncate '<nome da tabela>'

Comando:

truncate 'funcionario'

TTL – Registro temporário

Colunas com propriedade TTL – Time To Live

É um mecanismo que permite configurar de forma opcional o tempo de permanência de registros em uma tabela ou especificamente de uma coluna.

Configurável em segundos, quantos tempo o registro ficará disponível para consulta.

Os registros são deletados após esse período.

Esse comportamento é feito para todas as versões de cada linha e também válido para quando o HBase é utilizado como intermediários no fluxo de dados ou seja para dados de transição.

A deleção do registro pode ocorrer também como versionamento da linha, como se fosse uma deleção lógica mas os dados vão continuar nas versões anteriores.

TTL – Registro temporário

Exemplo TTL – Time To Live

Criar uma tabela com registros que ficam 20 segundos na base.

```
create 'ttl_exemplo', { 'NAME' => 'cf', 'TTL' => 20}
```

```
put 'ttl_exemplo', 'linha123', 'cf:desc', 'TTL Exemplo'
```

```
get 'ttl_exemplo', 'linha123', 'cf:desc'
```

Aguarde 20 segundos e consulte a linha novamente.

```
get 'ttl_exemplo', 'linha123', 'cf:desc'
```

Comandos do Cassandra

Criação do keyspace (schema, namespace)

```
CREATE KEYSPACE empresa
WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
```

Criação de tabela

```
create table empresa.funcionario
    (
        empregadoid int primary key,
        empregadonome text,
        empregadocargo text,
    );
```

Criação de índice secundário para consulta

```
CREATE INDEX rempresacargo ON empresa.funcionario (empregadocargo);
```

DIGITAL INNOVATION ONE

Cont. Comandos do Cassandra

```
cqlsh:empresa> help

Documented shell commands:
==========================
CAPTURE      COPY  DESCRIBE  EXPAND  LOGIN   SERIAL  SOURCE
CONSISTENCY  DESC  EXIT      HELP    PAGING  SHOW    TRACING

CQL help topics:
================
ALTER                          CREATE_TABLE_OPTIONS  SELECT
ALTER_ADD                      CREATE_TABLE_TYPES    SELECT_COLUMNFAMILY
ALTER_ALTER                    CREATE_USER           SELECT_EXPR
ALTER_DROP                     DELETE                SELECT_LIMIT
ALTER_RENAME                   DELETE_COLUMNS        SELECT_TABLE
ALTER_USER                     DELETE_USING          SELECT_WHERE
ALTER_WITH                     DELETE_WHERE          TEXT_OUTPUT
APPLY                          DROP                  TIMESTAMP_INPUT
ASCII_OUTPUT                   DROP_COLUMNFAMILY     TIMESTAMP_OUTPUT
BEGIN                          DROP_INDEX            TRUNCATE
BLOB_INPUT                     DROP_KEYSPACE         TYPES
BOOLEAN_INPUT                  DROP_TABLE            UPDATE
COMPOUND_PRIMARY_KEYS          DROP_USER             UPDATE_COUNTERS
CREATE                         GRANT                 UPDATE_SET
CREATE_COLUMNFAMILY            INSERT                UPDATE_USING
CREATE_COLUMNFAMILY_OPTIONS    LIST                  UPDATE_WHERE
CREATE_COLUMNFAMILY_TYPES      LIST_PERMISSIONS      USE
CREATE_INDEX                   LIST_USERS            UUID_INPUT
CREATE_KEYSPACE                PERMISSIONS
CREATE_TABLE                   REVOKE

cqlsh:empresa>
```

DIGITAL INNOVATION ONE

Exercício

1. Criar uma tabela que representa lista de Cidades e que permite armazenar até 5 versões na Column Familiy com os seguintes campos:

Código da cidade como rowKey

Column Family=info
- Nome da Cidade
- Data de Fundação

Column Family=responsaveis
- Nome Prefeito
- Data de Posse do Prefeito
- Nome Vice prefeito

Column Family=estatisticas
- Data da última Eleição
- Quantidade de moradores
- Quantidade de eleitores
- Ano de fundação

DIGITAL INNOVATION ONE

Exercício

2. Inserir 10 cidades na tabela criada de cidades.

3. Realizar uma contagem de linhas na tabela.

4. Consultar só o código e nome da cidade.

5. Escolha uma cidade, consulte os dados dessa cidade em específico antes do próximo passo.

6. Altere para a cidade escolhida os dados de Prefeito, Vice Prefeito e nova data de Posse.

7. Consulte os dados da cidade alterada.

8. Consulte todas as versões dos dados da cidade alterada.

DIGITAL INNOVATION ONE

Exercício

9. Exclua as três cidades com menor quantidade de habitantes e quantidade de eleitores.

10. Liste todas as cidades novamente.

11. Adicione na ColumnFamily “estatísticas”, duas novas colunas de “quantidade de partidos políticos” e “Valor em Reais à partidos” para as 2 cidades mais populosas cadastradas.

12. Liste novamente todas as cidades.

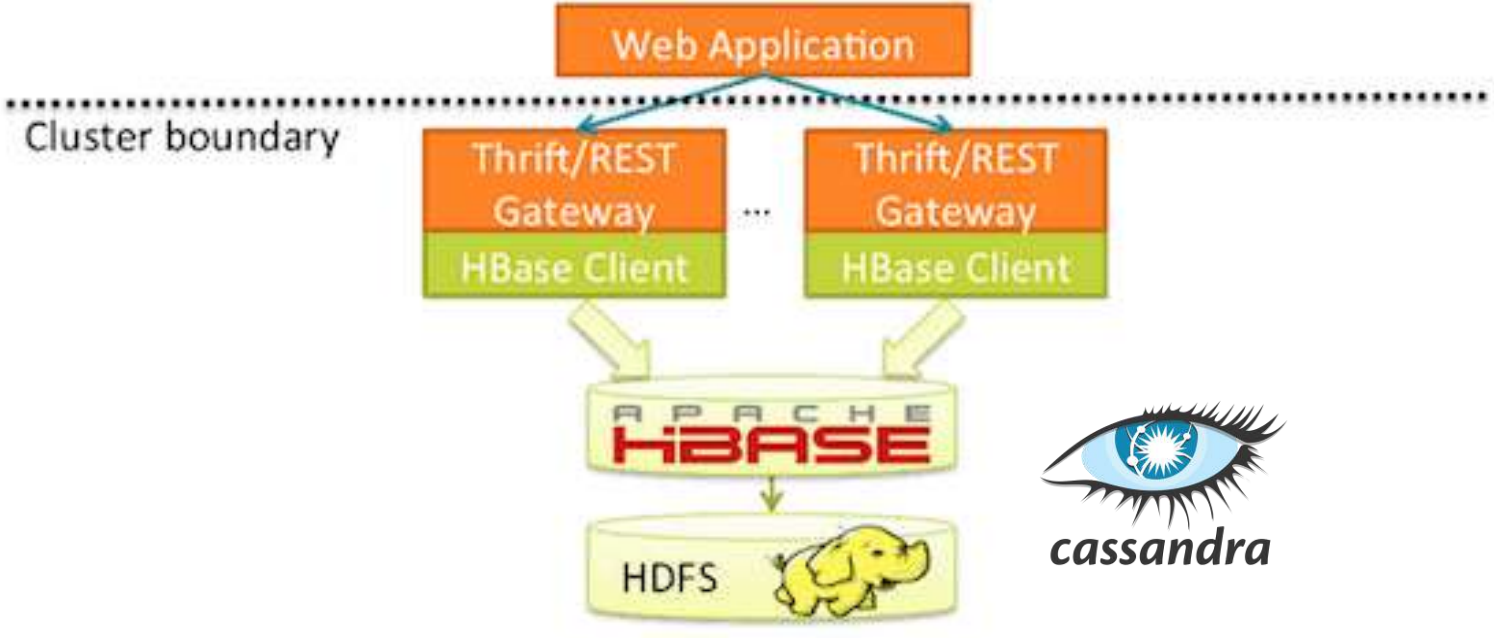
DIGITAL
INNOVATION
ONE

4. Cenários de Utilização

NoSQL com Hbase e Cassandra

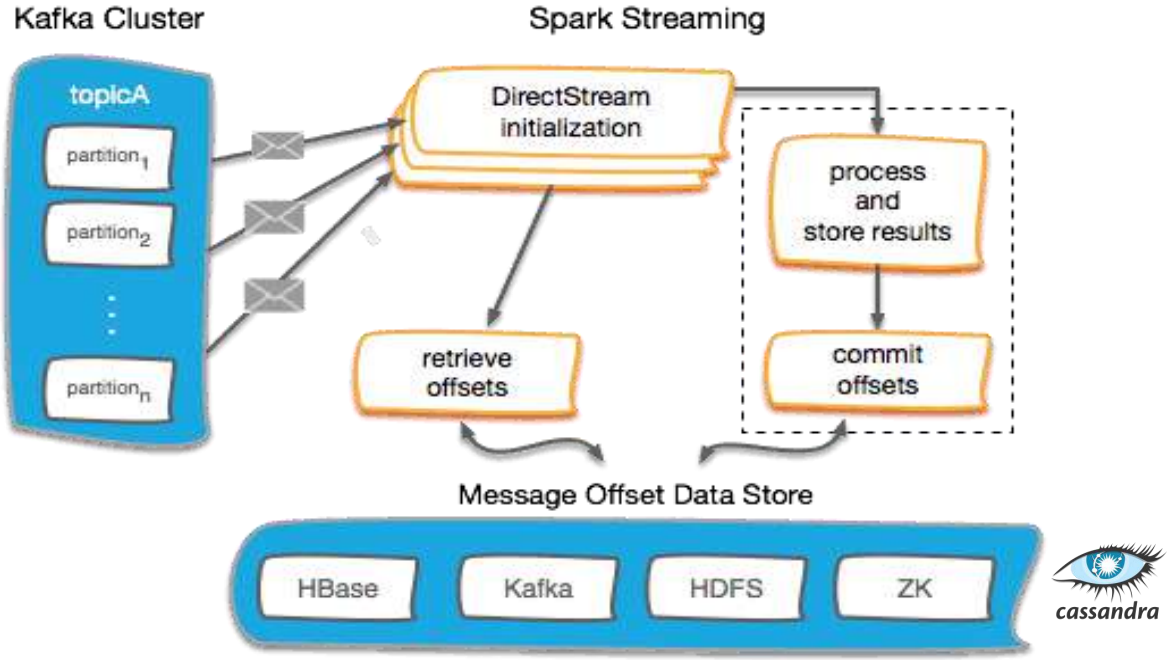
Cenário 1 de Utilização

Utilizado como banco de dados para aplicações Web e aplicativos móveis.
Por meio de integrações REST é possível fazer essa integração com o HBase.



Cenário 2 de Utilização

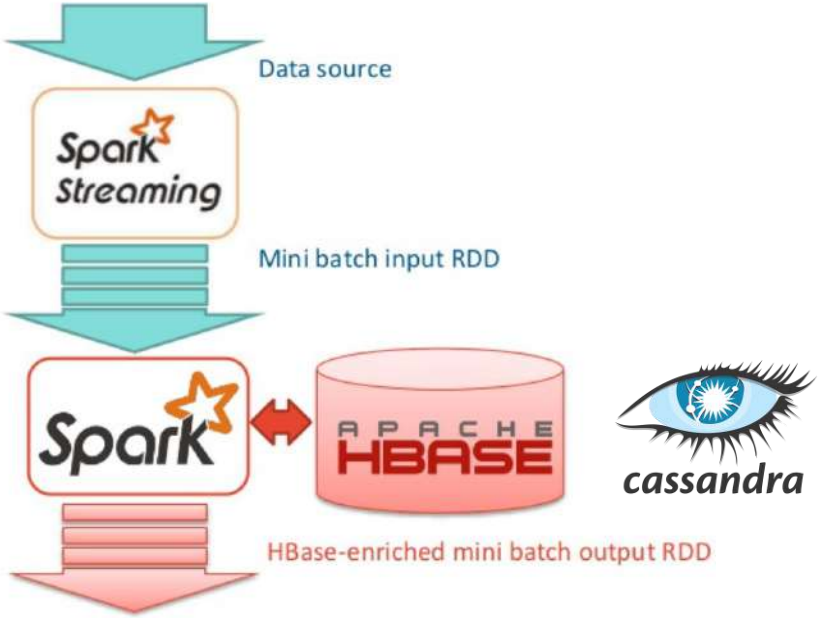
Utilizado como repositório para guardar uma cópia e/ou replicador dos dados vindo de serviços de eventos antes da consolidação final.



DIGITAL INNOVATION ONE

Cenário 3 de Utilização

Enriquecimento dos dados finais com dados armazenados no HBase. Por exemplo: lookups. Como há o versionamento padrão do Hbase, é possível ter o rastro das informações alteradas para cada rowKey.



DIGITAL
INNOVATION
ONE

5. Operações massivas

NoSQL com Hbase e Cassandra

DIGITAL INNOVATION ONE

O que é *Build Insert*?

Nos banco de dados essa é a uma operação que permite fazer a carga massiva de dados.

Normalmente isso ocorre por acionamento de um programa utilitário do banco de dados.

No HBase é possível realizar essa carga por esses meios mais comuns:

- Classe Utilitária do HBase [org.apache.hadoop.hbase.mapreduce.ImportTsv].
- Intermediários em script com Sqoop, aplicações em Spark, Apache Phoenix, e etc, que implementam indiretamente a Hbase API.
- External Table do Hive utilizando StorageHandler para gravar os dados no Hbase/Cassandra.
- API

Exemplo de Builk Insert

1. Confirmar que o arquivo employees.csv esteja em /home/everis/arquivos/employee.csv

2. Criar a tabela employees no Hbase com a column Familty: employee_data

3. Criar uma pasta no HDFS pelo shell do Linux
hadoop fs -mkdir /test
Copia os arquivos exportados para o HDFS pelo shell do Linux
hadoop fs -copyFromLocal /home/everis/arquivos/employees.csv /test/employees.csv

4. Executar a importação no shell do Linux

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=';' -Dimporttsv.columns=HBASE_ROW_KEY,employee_data:birth_date,employee_data:first_name,employee_data:last_name,employee_data:gender,employee_data:hire_date employees /test/employees.csv

DIGITAL INNOVATION ONE

Exercício

1. Criar a tabela salaries no HBASE com o schema ao lado.

2. Efetuar a carga de dados via ImportTsv utilizando o arquivo salaries.csv
3. Verificar a quantidade na tabela carregada versus a quantidade de linhas do arquivo.

4. Crie a tabela salaries_concatenado agora para o arquivo salaries_com_row_key.csv, observe que esse arquivo tem uma coluna a mais que é a row_key concatenada.

5. Porque o primeiro arquivos carregou menos registros?

```
mysql> select * from salaries limit 10;
+--------+--------+------------+------------+
| emp_no | salary | from_date  | to_date    |
+--------+--------+------------+------------+
|  10001 |  60117 | 1986-06-26 | 1987-06-26 |
|  10001 |  62102 | 1987-06-26 | 1988-06-25 |
|  10001 |  66074 | 1988-06-25 | 1989-06-25 |
|  10001 |  66596 | 1989-06-25 | 1990-06-25 |
|  10001 |  66961 | 1990-06-25 | 1991-06-25 |
|  10001 |  71046 | 1991-06-25 | 1992-06-24 |
|  10001 |  74333 | 1992-06-24 | 1993-06-24 |
|  10001 |  75286 | 1993-06-24 | 1994-06-24 |
|  10001 |  75994 | 1994-06-24 | 1995-06-24 |
|  10001 |  76884 | 1995-06-24 | 1996-06-23 |
+--------+--------+------------+------------+
10 rows in set (0.00 sec)
```

6. Integrações NoSQL com Ambiente Hadoop

NoSQL com Hbase e Cassandra

DIGITAL INNOVATION ONE

Integração NoSQL com Hadoop

É possível realizar essa integração utilizando a implementação da interface **StorageHandler** com a classe **org.apache.hadoop.hive.hbase.HBaseStorageHandler** que o Hbase disponibiliza.

No Hive a interface **StorageHandler** permite que outras aplicações externas ao Hive (i.e Cassandra, Azure Table, JDBC (MySQL), MongoDB, ElasticSearch, e etc) implementem e disponibilizem operações dessas estruturas e dados armazenados ao Apache Hive.

A idéia é que o Hive tenha visibilidade do metadados quando a tabela é criado no Hive mas os dados e o controle de armazenamento esteja externo.

É uma evolução do conceito de tabela gerenciada (managed) e externa (external) do Hive.

Saber mais sobre StorageHandlers, saiba mais.

Para saber mais detalhes da Integração do Hive e HBase e quais operações estão suportadas, saiba mais.

Exemplo Integração NoSQL com Hadoop

Utilizando a tabela employee utilizada nos exercícios anteriores, vamos dar visibilidade ao Hive dessa estrutura do HBase.

1. Vamos criar um novo schema no Hive.

```
CREATE DATABASE tabelas_hbases;
```

2. Criar a tabela employee no Hive.

```
CREATE EXTERNAL TABLE tabelas_hbases.employees (
emp_no INT,
birth_date string,
first_name string,
last_name string,
gender string,
hire_date string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES("hbase.columns.mapping"=":key, employee_data:birth_date,
employee_data:first_name, employee_data:last_name,employee_data:gender,employee_data:hire_date")
TBLPROPERTIES("hbase.table.name"="employees", "hbase.mapred.output.outputtable"="employees");
```

DIGITAL INNOVATION ONE

Exercício

1. Criar a tabela externa salaries no Hive que representa a mesma tabela que foi carregada em exercícios anteriores no Hbase.
2. Consultar os empregados com o maior salário em cada ano.
3. Consultar o quanto foi gasto em salários por ano.

DIGITAL
INNOVATION
ONE

7.Próximos passos

NoSQL com Hbase e Cassandra

DIGITAL INNOVATION ONE

Próximos Passos

Utilizar o **Apache Spark** para processamento distribuído de informações vindo de eventos (Kafka) ou gravando em tópicos com as informações de dados no **Hbase/Cassandra**.
Resumindo: olhar para Arquitetura de Eventos no big data.

Aplicação **Apache Phoenix** que habilita capacidades de execução consultas SQL, analítico e OLTP ao HBase.
As duas distribuições mais conhecidas do ecossistema do Hadoop on-premisse Cloudera e Horton suportam e em algumas instalações podem permitir a utilização ou já utilizam.
https://phoenix.apache.org/ e https://phoenix.apache.org/who_is_using.html

Caso queira aprofundar o conhecimento em NoSQL, olhar outros produtos como **Apache Cassandra (versão da DATASTAX) https://www.datastax.com/** que tem versão do Cassandra *as a Service*.

DIGITAL
INNOVATION
ONE

Referências Utilizadas

NoSQL com Hbase e Cassandra

Referências e Bibliografia

Apache HBase Reference Guide

Secondary Indexing on Hbase

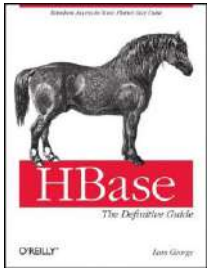
Cloudera Apache HBase Rest Interface

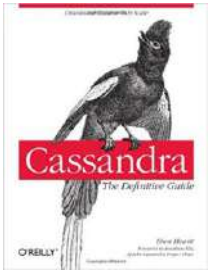
Hbase as Offset Managment for Apache Kafka with Apache Spark Streaming

Ccomo-se-aplica-el-teorema-cap-y-el-reto-de-la-escalabilidad-en-las-rdbms-y-nosql

NoSQL do teorema CAP para PACCL

Apache Cassandra 3.x Reference Guide





DIGITAL
INNOVATION
ONE

Dúvidas?

[Nome do curso]