Introdução

Segue a solução para cada uma das 4 questões do exercício – Aplicação Lavouras Demonstrativas, da Yara International. Será enviado, juntamente deste documento, um diretório em formato .zip com os arquivos fontes das soluções. O mesmo diretório pode ser acessado no link a seguir:

Github: https://github.com/DennCardoso/Yara-International---Lavouras-Demonstrativas

Qualquer dúvida sobre a execução da solução, estou à disposição no e-mail: dennis.cardoso@outlook.com

As necessidades atendidas – Aplicativo Lavouras Demonstrativas

O Lavouras Demonstrativas foi assim criado para poder armazenar dados de duas lavouras: aquela que utiliza os produtos Yara, e aquela que usa os produtos que o fazendeiro deseja.

O desejo, tanto da Yara quanto de seus clientes, é poder armazenar dados de talhões, as características da cultura tratada e os resultados obtidos.

Para isso, o desejo do aplicativo é que ele possa cumprir o papel de armazenar dados imputados pelo funcionário sobre as lavouras, com campos simples e objetivos para ser acessível à diferentes públicos.

O que Yara International Deseja

- Modelo de arquitetura de dados flexível, escalável, normalizada;
- Otimizar o banco para geração de relatórios;
- Quantidades de interesse:
    - produtividade por
        - cultura
        - localização
        - produtos
- diferença de produtividade entre tratamento Yara e controle

Premissas do exercício

- *Cada relatório pode ter um ou mais tratamentos (Yara e/ou Atual)*
- *Cada relatório está associado a um único cliente e a um único talhão (item da tabela Lavoura)*
- *Cada tratamento está associado a apenas um tipo de cultivo*
- *Cada tratamento pode ter um ou mais produtos associados*

- *Cada do item da tabela Lavoura (Talhão) pode ter apenas um tipo de cultivo (SistemaProdução ou CulturaDePousio)*
- *O campo **TamColhido** da Tabela lavoura contém a área do talhão medida em hectares.*
- *A produtividade absoluta de cada Tratamento é dada pelo campo **ProdutividadeDoTratamento** dada em número de sacas.*

Questões e soluções

1. Exercício

Diversos problemas estão presentes na estrutura da base de dados fornecida, alguns deles são por exemplo:

- *Dados de fazendas e clientes na mesma tabela*
- *Tabelas TratamentoYara e TratamentoAtual duplicadas*
- *Nomes de campos repetidos e sem estrutura*

*Com base no contexto fornecido no modelo json (slides anteriores) e campos relevantes, crie um diagrama EER modelando uma nova estrutura de banco de dados com dados normalizados (reduzir a redundância de dados, aumentar a integridade de dados e o desempenho) facilitando a construção de **relatórios de produtividade para diversos agregadores.***

Solução:

Baseado dos dados fornecidos pelo arquivo **Prova Processo Seletivo – Agosto 2018**, temos abaixo o modelo de dados para o aplicativo Lavouras Demonstrativas.

A aplicação utilizada para o desenvolvimento do modelo ER é Visual Paradigm. O arquivo fonte do modelo ER pode ser acessado no Github como também no arquivo zip (***Yara - Sistema de Lavouras Demonstrativas.vpp).***

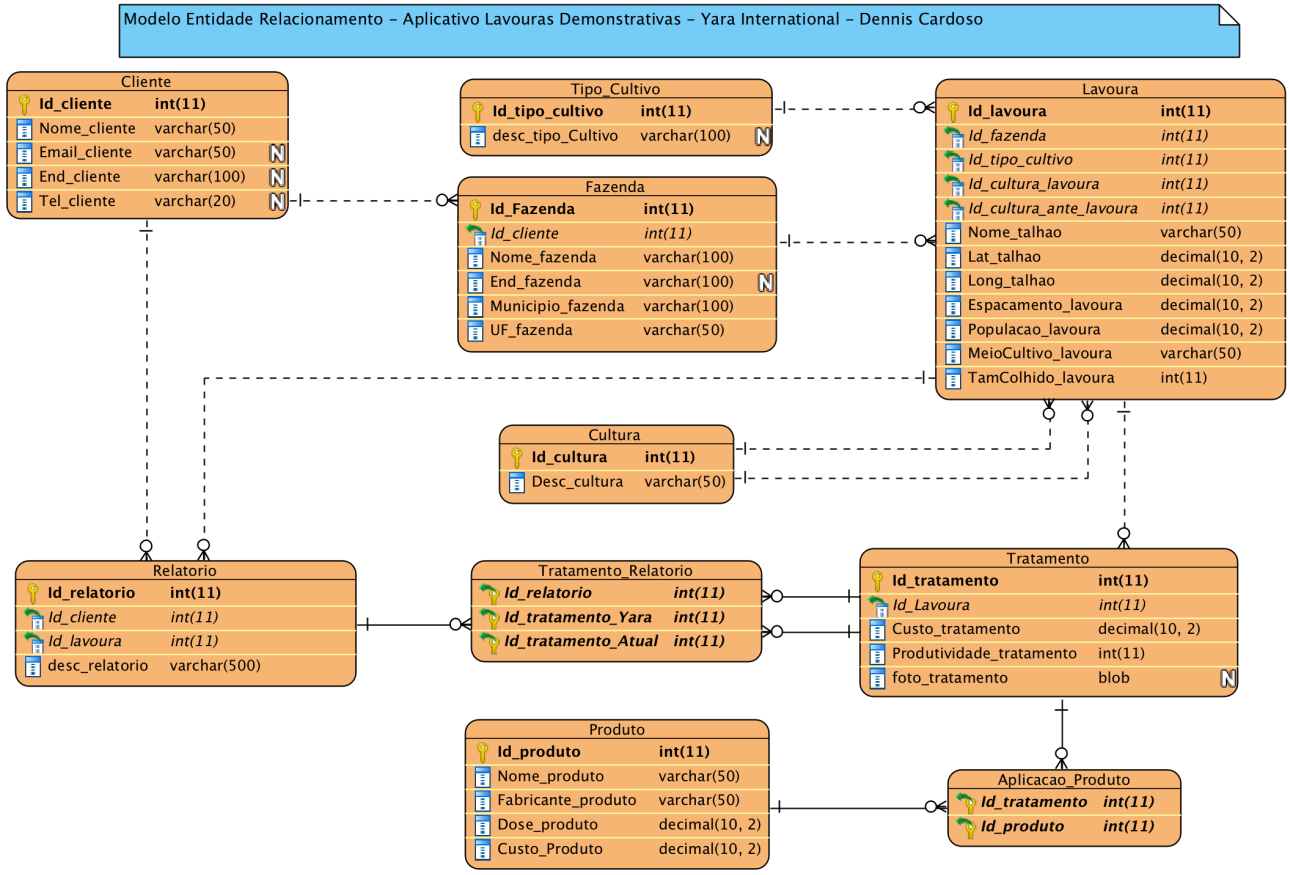

Figura 1 - Modelo Entidade Relacionamento – Lavouras Demonstrativas

Observações:

1. O símbolo N ao lado de cada atributo das entidades significa que o campo em questão é 'Nullable', isto é, permite entrada de valores nulos.
2. Os atributos com uma seta verde ao lado do esquerdo são as chaves estrangeiras correspondentes.
3. Para melhor visualização, o modelo Entidade relacionamento está disponível em formato PNG no diretório zip, bem como também no repositório GitHub (arquivo **ER_Entidade_relacionamento_Yara.png**)
4. Abaixo segue uma breve legenda da estrutura de entidades e relacionamentos do modelo acima.

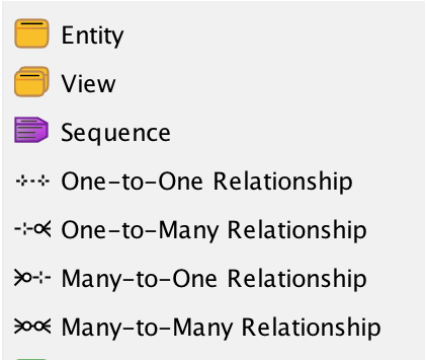

Figura 2 - Legenda do Modelo ER

Resposta do modelo em relação às premissas:

- Cada relatório pode ter um ou mais tratamentos (Yara e/ou Atual)
    - Cada relatório possui 1 ou mais tratamentos, onde o relacionamento é registrado na *Tratamento_Relatório*. Além disso, considerei a premissa que um tratamento pode ter 1 ou mais relatórios, por isso foi criada a tabela de relacionamento *Tratamento_Relatório.*

- Cada relatório está associado a um único cliente e a um único talhão (item da tabela Lavoura)
    - A tabela *Relatório* contém as chaves estrangeiras: *Id_cliente* (tabela cliente) e *Id_lavoura* (Tabela Lavoura) pra garantir a relação entre as três entidades. A chave primária (PK) da tabela relatório é *id_relatório,* o que garante a unicidade do registro e impedindo o cadastro de mais de um cliente e lavoura em um mesmo relatório.

- Cada tratamento está associado a apenas um tipo de cultivo (Sistema Produção ou Cultura De Pousio)
    - A entidade *Lavoura* garante apenas o cadastro de um cultivo, pela coluna *Id_tipo_cultivo (*chave estrangeira FK – tabela referencia Tipo_Cultivo). Cada tratamento, por sua vez, tem relação com apenas uma lavoura, o que garante que exista apenas um único tipo de cultivo vinculado ao tratamento, atendendo a premissa.

- Cada tratamento pode ter um ou mais produtos associados
    - Para atender essa premissa, considerei que para um tratamento pode existir um ou mais produtos e para cada Produto pode existir um ou mais tratamentos. Portanto, a tabela de relacionamento *Aplicacao_Produto* foi criada para compor o relacionamento *many-to-many* entre as duas entidades, *tratamento* e *Produto,* e garantir a premissa que um tratamento pode ter diversos produtos aplicados.

- Cada item da tabela Lavoura (Talhão) pode ter apenas um tipo de cultivo (SistemaProdução ou CulturaDePousio)
    - Considerei por meio de regra de negócio e entendimento do exercício que **Sistema Produção** e **Cultura de Pousio** são *tipos de Cultivo* diferentes. Portanto, para eliminar espaço de memória da tabela Lavoura, criei a tabela domínio *Tipo_Cultivo* para registrar o valor do cultivo e a descrição. Na tabela *Lavoura* foi criada a coluna *id_tipo_cultivo* para armazenar qual o tipo de cultivo utilizado, garantindo a premissa acima.

- O campo **TamColhido** da Tabela lavoura contém a área do talhão medida em hectares.
    - **TamColhido** foi definido como a coluna **TamColhido_lavoura** da tabela *Lavoura*, do tipo Inteiro (Hectares).

- A produtividade absoluta de cada Tratamento é dada pelo campo **ProdutividadeDoTratamento** dada em número de sacas.
    - ProdutividadeDoTratamento foi definido como a coluna ***Produtividade_tratamento*** na tabela *Tratamento*, Decimal(10,2).

2. Exercício

Dada a estrutura que você criou no exercício 1, construa scripts DDL para gerar as tabelas necessárias e crie arquivos com as consultas para produzir um relatório que indique:

- *A média da **diferença de produtividade por área cultivada** entre tratamentos para cada cultura*
- *A média da **diferença relativa de produtividade por área cultivada** entre tratamentos por **UF***

*Em todos itens analise a **performance** das consultas.*

Solução:

A seguir está o código fonte em SQL para criação do DDL (Data Definition Language)
O código fonte ser acessado também no arquivo **Exercicio2_DDL_yara.sql** no arquivo .zip ou no link GitHub.

```
#Criacao do Database yara
#create database yara;

#Usar database yara
use yara;

#Criacao da Tabela Cliente
CREATE TABLE Cliente(
Id_cliente int NOT NULL AUTO_INCREMENT,
Nome_cliente varchar(50) NOT NULL,
Email_cliente varchar(50),
End_cliente varchar(100),
Tel_cliente varchar(20),
CONSTRAINT PK_Cliente PRIMARY KEY (Id_Cliente)
);

#Criacao da Tabela Fazenda
CREATE TABLE Fazenda(
Id_fazenda int NOT NULL AUTO_INCREMENT,
Id_cliente int NOT NULL,
Nome_fazenda varchar(100) NOT NULL,
End_fazenda varchar(100) NOT NULL,
Municipio_fazenda varchar(100) NOT NULL,
UF_fazenda varchar(50) NOT NULL,
CONSTRAINT PK_Fazenda PRIMARY KEY(Id_fazenda),
```

```
FOREIGN KEY (Id_cliente) REFERENCES Cliente(Id_cliente)
);

#Criacao da Tabela Tipo Cultivo
CREATE TABLE Tipo_Cultivo(
Id_tipo_Cultivo int NOT NULL AUTO_INCREMENT,
Desc_tipo_cultivo varchar(100),
CONSTRAINT PK_Tipo_Cultivo PRIMARY KEY(Id_tipo_Cultivo)
);

#Criacao da Tabela Cultura
CREATE TABLE Cultura(
Id_cultura int NOT NULL AUTO_INCREMENT,
Desc_cultura varchar(50),
CONSTRAINT PK_Cultura PRIMARY KEY (Id_cultura)
);

#Criacao da Tabela Lavoura
CREATE TABLE Lavoura(
Id_Lavoura int NOT NULL AUTO_INCREMENT,
Id_fazenda int NOT NULL,
Id_tipo_cultivo int NOT NULL,
Id_cultura_lavoura int NOT NULL,
Id_cultura_ante_lavoura int,
Nome_talhao varchar(50) NOT NULL,
Lat_talhao decimal(10,2) NOT NULL,
Long_talhao decimal(10,2) NOT NULL,
Espacamento_lavoura decimal(10,2) NOT NULL,
Populacao_lavoura decimal(10,2) NOT NULL,
MeioCultivo_lavoura varchar(50) NOT NULL,
TamColhido_lavoura int NOT NULL,
CONSTRAINT PK_Lavoura PRIMARY KEY(Id_Lavoura),
FOREIGN KEY (Id_fazenda) REFERENCES Fazenda(Id_fazenda),
FOREIGN KEY (Id_tipo_cultivo) REFERENCES Tipo_Cultivo(Id_tipo_cultivo),
FOREIGN KEY (Id_cultura_lavoura) REFERENCES Cultura (Id_Cultura),
FOREIGN KEY (Id_cultura_ante_lavoura) REFERENCES Cultura (Id_cultura)
);

#Criacao da Tabela Relatório
CREATE TABLE Relatorio(
Id_relatorio int NOT NULL AUTO_INCREMENT,
Id_Cliente int NOT NULL,
Id_Lavoura int NOT NULL,
Desc_relatorio varchar(500) NOT NULL,
CONSTRAINT PK_relatorio PRIMARY KEY(Id_relatorio),
FOREIGN KEY (Id_cliente) REFERENCES Cliente(Id_Cliente),
FOREIGN KEY (Id_Lavoura) REFERENCES Lavoura(Id_Lavoura)
);
```

```
#Criacao da Tabela Tratamento
CREATE TABLE Tratamento(
Id_tratamento int NOT NULL AUTO_INCREMENT,
Id_lavoura int NOT NULL,
Custo_tratamento decimal(10,2) NOT NULL,
Produtividade_tratamento int NOT NULL,
foto_tratamento blob,
CONSTRAINT PK_tratamento PRIMARY KEY (Id_tratamento),
FOREIGN KEY (Id_lavoura) REFERENCES Lavoura(Id_lavoura)
);

#Criacao da Tabela Tratamento_relatorio
CREATE TABLE Tratamento_Relatorio(
Id_relatorio int NOT NULL,
Id_tratamento_yara int NOT NULL,
Id_tratamento_atual int NOT NULL,
CONSTRAINT PK_tratamento_relatorio PRIMARY KEY (Id_relatorio,
Id_tratamento_yara, Id_tratamento_atual),
FOREIGN KEY (Id_relatorio) REFERENCES Relatorio(Id_relatorio),
FOREIGN KEY (Id_tratamento_yara) REFERENCES Tratamento(Id_tratamento),
FOREIGN KEY (Id_tratamento_atual) REFERENCES Tratamento(Id_Tratamento)
);

#Criacao da Tabela Produto
CREATE TABLE Produto(
Id_produto int NOT NULL AUTO_INCREMENT,
Nome_produto varchar(50) NOT NULL,
Fabricante_produto varchar(50) NOT NULL,
Dose_produto decimal(10,2) NOT NULL,
Custo_produto decimal(10,2) NOT NULL,
CONSTRAINT PK_Produto PRIMARY KEY (Id_produto)
);

#Criacao da Tabela Aplicacao_Produto
CREATE TABLE Aplicacao_Produto(
Id_tratamento int NOT NULL,
Id_produto int NOT NULL,
CONSTRAINT PK_Aplicacao_Produto PRIMARY KEY (Id_tratamento, Id_produto),
FOREIGN KEY (Id_tratamento) REFERENCES Tratamento(Id_tratamento),
FOREIGN KEY (Id_produto) REFERENCES Produto(Id_produto)
);
```

A média da diferença de produtividade por área cultivada entre tratamentos para cada cultura

O desenvolvimento dessa query pode ser encontrado no arquivo **Exercicio2_relatorio_DiffCultura.sql,** também como no código abaixo:

```
/*
O select busca a média da diferença de Produtividade por área entre os
tratamentos Yara e Atual, agrupados por Cultura
Para isso buscamos os seguintes dados: Código da Cultura, Descrição da
cultura.
Realizamos os seguinte calculo.:
- Para cada comparação no relatório, temos a produtividade do tratamento
Yara e do Tratamento Atual.
- É realizado a subtração entre Prod Yara e Prod atual e esse resultado é
dividido pelo TamColhido(Ha)*10000 (transformação e Hectares para Metros).
- Desta maneira temos a quantidade da diferença entre produtividade por
metro quadrado (m2).
- Por fim, agrupamos a média desse valor por Cultura para finalmente
obtermos o valor.
*/
select
l.Id_cultura_lavoura 'Código Cultura',
c.desc_cultura 'cultura',
Round(avg((ty.Produtividade_tratamento -
ta.Produtividade_tratamento)/(l.TamColhido_lavoura*10000)),2)
'media_Diff_Prod_Absoluta_Cultura'
from Tratamento_Relatorio tr
        inner join Tratamento ty
                on tr.id_tratamento_yara = ty.Id_tratamento
        inner join Tratamento ta
                on tr.id_tratamento_atual = ta.Id_tratamento
        inner join relatorio r
                on tr.id_relatorio = r.Id_relatorio
        inner join lavoura l
                on r.id_lavoura = l.id_lavoura
        inner join cultura c
                on l.id_cultura_lavoura = c.Id_cultura
group by l.Id_cultura_lavoura, c.desc_cultura;
```

A performance da query acima pode ser compreendida da seguinte maneira:

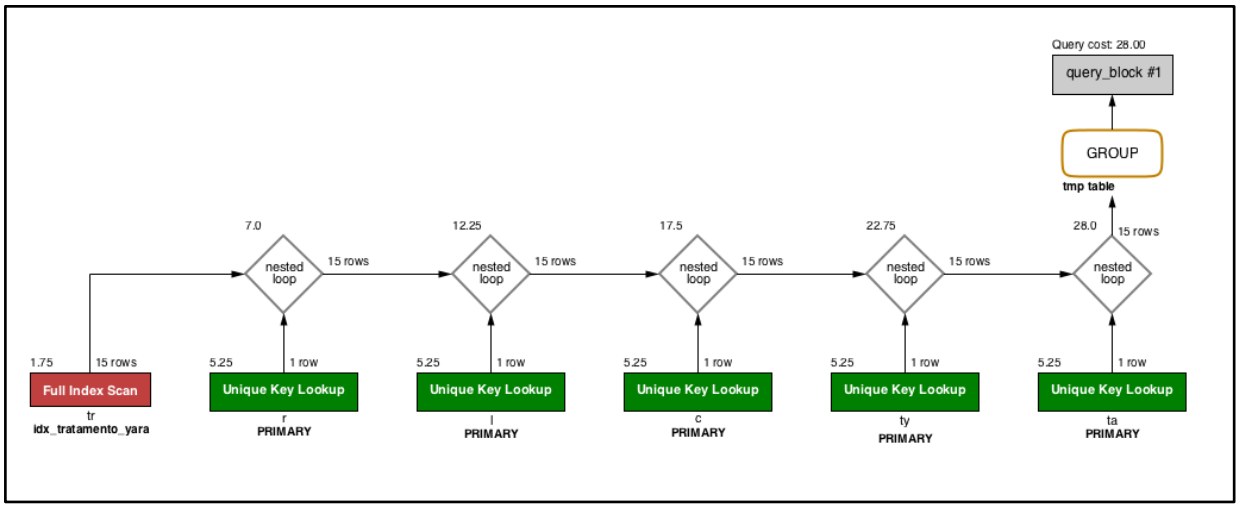

Figura 3 - Query execution plan Image – Select 1

- Foi realizado um Full Index Scan na tabela Tratamento_Relatorio para acesso a todos os itens do tratamento.
- No seguinte fluxo, foram realizadas Unique Key Lookup. Isso significa que, durante a realização do comando inner join, as queries tiveram custo baixo devido ao uso da primary key no relacionamento (e consequentemente, a unique key constraint).

A média da diferença de produtividade por área cultivada entre tratamentos para cada cultura

O desenvolvimento dessa query pode ser encontrado no arquivo **Exercicio2_Relatorio_DiffUF.sql,** também como no código abaixo:

```
/*
O select busca a média da diferença de Produtividade relativa por área
entre os tratamentos Yara e Atual, agrupados por UF
Para isso buscamos os seguinte dado: Descrição da UF
Realizamos os seguinte calculo.:
- Total de Produvidade (sacas/m2) = (Produtividade Tratamento Yara +
Produtividade tratamento Atual)/(Tamanho da Colheita*10000m2)
- Diferença da Produtividade (sacas)  = (Produtividade Tratamento Yara -
Produtividade tratamento Atual)/(Tamanho da Colheita*10000m2)
- Diferença da Produtividade relativa (entre 0.00 a 1.00) = Diferença da
Produtividade/Total de Produvidade
- Media da Diferença da Produtividade Relativa (percentual) = Funcão
Média (Diferença da Produtividade Relativa) * 100
- Desta maneira, temos a Média da Diferença da Produtividade Relativa,
arredondado para duas casas decimais
- Por fim, agrupamos a média desse valor por UF
*/
```

```
select
            f.UF_fazenda 'UF',
            ROUND(avg((ty.Produtividade_tratamento -
ta.Produtividade_tratamento)/(ty.Produtividade_tratamento +
ta.Produtividade_tratamento))*100,2) 'media_Diff_Prod_Relativa_UF'
            from Tratamento_Relatorio tr
                  inner join Tratamento ty
                        on tr.id_tratamento_yara = ty.Id_tratamento
                  inner join Tratamento ta
                        on tr.id_tratamento_atual = ta.Id_tratamento
                  inner join relatorio r
                        on tr.id_relatorio = r.Id_relatorio
                  inner join lavoura l
                        on r.id_lavoura = l.id_lavoura
                  inner join cultura c
                        on l.id_cultura_lavoura = c.Id_cultura
                  inner join fazenda f
                        on l.id_fazenda = f.id_fazenda
Group by f.UF_fazenda
```

A performance da query acima pode ser compreendida da seguinte maneira:

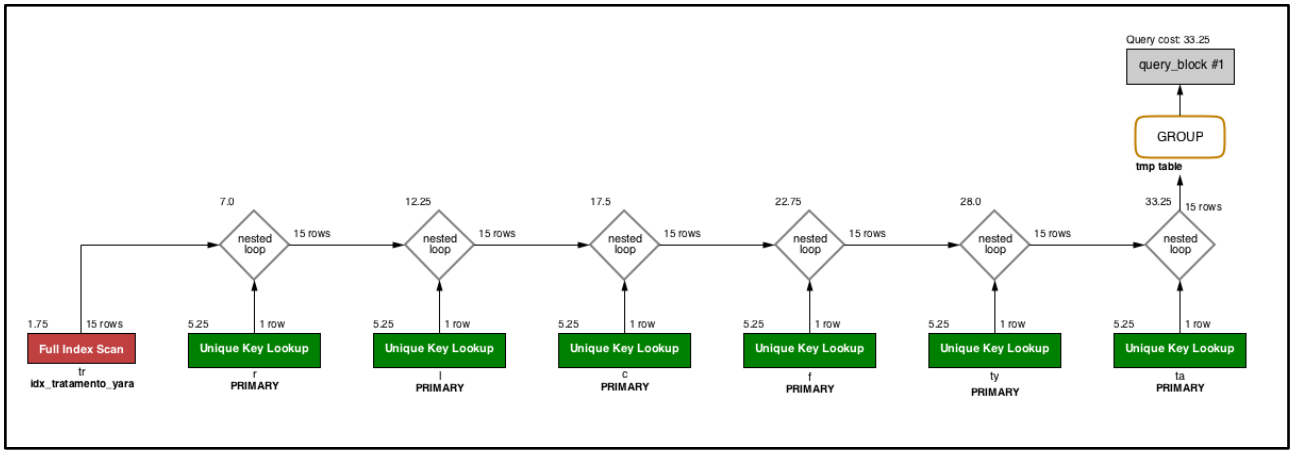

Figura 4 - Query Execution Plan Image - Select 2

Igualmente a query anterior, temos os mesmos dados de execução e performance:

- Foi realizado um Full Index Scan na tabela Tratamento_Relatorio para acesso a todos os itens do tratamento.

- Nas demais tabelas, foram realizadas Unique Key Lookup. Isso significa que, durante a realização do comando inner join, as queries tiveram custo baixo devido ao uso da primary key no relacionamento (e consequentemente, a unique key constraint).

3. Exercício

Qual seria o pipeline que você construiria para migração desses dados para um datalake? Descreva em linhas gerais e justifique suas escolhas.

Solução:

Para criação de um fluxo de dados entre uma base de dados estruturada para um Data Lake, podemos imaginar em uma infraestrutura de Data Lake já desenvolvida, muito similar a essa que podemos ver no exemplo abaixo (imagem 7).

Desta maneira, é possível transferir os dados criando um fluxo ETL (Extract, Transform and Load). Para isso, podemos utilizar aplicações especificas para migração de dados para ambientes big data como, por exemplo, a distribuição Apache Scoop.

O apache Scoop pode ser utilizado como fluxo ETL para transmitir dados de um banco de dados externo para o ambiente interno do Data Lake, isto é, para um banco de dados RDBMS dentro do ambiente Big Data. Existem diversas distribuições do apache Scoop, como por exemplo, a distribuída gratuitamente, bem como de empresas como Hortonworks e Cloudera.

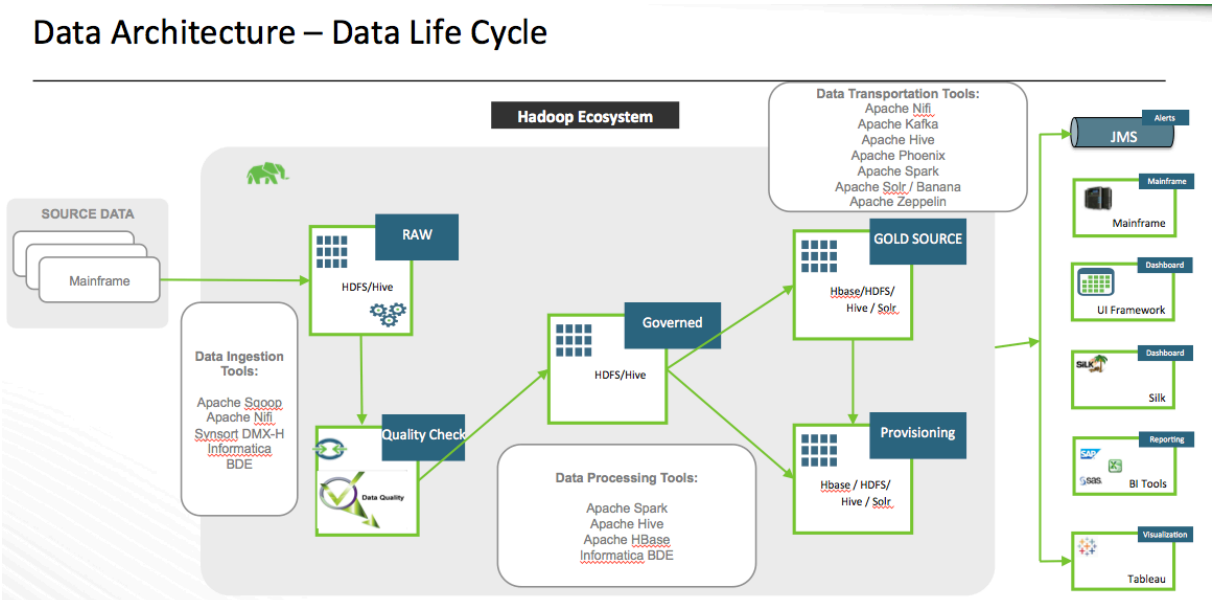

Figura 5 - Arquitetura de dados de um Data Lake - Fonte da imagem: https://hortonworks.com/apache/spark/

O Scoop funciona como uma espécie de conector que conversa com a estrutura do banco de dados relacional (RDBMS, neste caso do exercício, o mySQL), captura essa informação para um cluster de tratamento, trabalha o dado de modo que ele seja estruturado para no ambiente Big data e exporta esse dados como saída para o Sistema Hadoop, por exemplo, para armazenamento.

No caso do exercício proposto, podemos criar uma Job Scoop que capta informações a cada uma hora da base de dados Yara, onde aplicativo Lavouras Demonstrativas insere os dados. O Scoop irá trata-los por meio de manipulação de tipo e estrutura para então salva-lo em um cluster Hadoop (seja em uma base de dados não estruturada como o Hbase, em .json files ou em um banco de dados RDBMS dentro do ambiente do Data Lake.

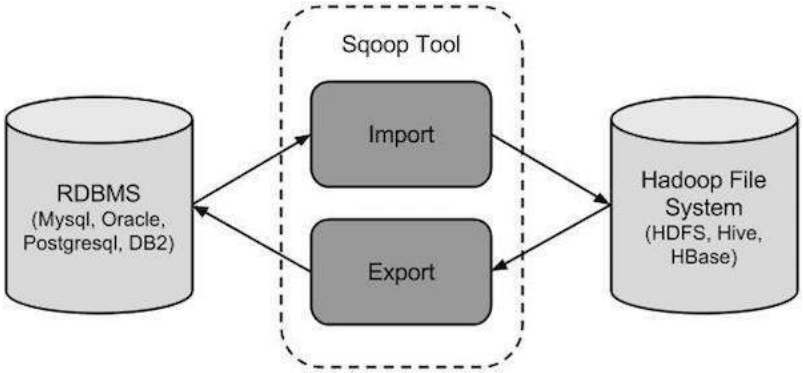

Figura 6 - Fluxo de transmissão de dados com Scoop

4. Exercício

Caso a aplicação evolua e inclua dados de sensores, como você lidaria com a presença de estruturas de dados heterogêneas (dados relacionais, dados contínuos no tempo)?

Solução:

Captação de dados de sensores faz parte do que chamamos de IoT ou Internet of Things, que é o desenvolvimento de micro sensores, sistemas em chipboards, entre outros equipamentos que vão gerar dados em tempo real para que seja consumido. Isso também é bastante considerado quando é implantado uma arquitetura Big data.

Para esse processo, eu usaria uma aplicação bastante comum no pacote Open Source da apache que também já é bastante utilizada pelas companhias de BigData: apache Kafka.

Apache Kafka tem como função a criação de pipelines para captação de dados em tempo real. Desta maneira, a aplicação consegue obter dados entre sistemas e aplicações, bem como reagir a infraestruturas como sensores e microchips e os transformam em pacotes de dados.

Um exemplo seria um conjunto de sensores que identifica temperatura de um determinado terreno. Em um determinado espaço de tempo, o equipamento emitirá um sinal que será captado pelo Gateway IoT e transmitido para um cluster de dados do Kafka. O Apache Kafka se encarregará neste momento de obter o pacote de transmissão de dados recebido no gateway e decidir como distribuir esses dados dentro do ambiente Big data para que seja utilizado, por exemplo, por um framework de processamento (como Spark) para uma análise ou processados para ser armazenado em um Cluster Hadoop. Abaixo uma ilustração de como seria o fluxo utilizando o Kafka:

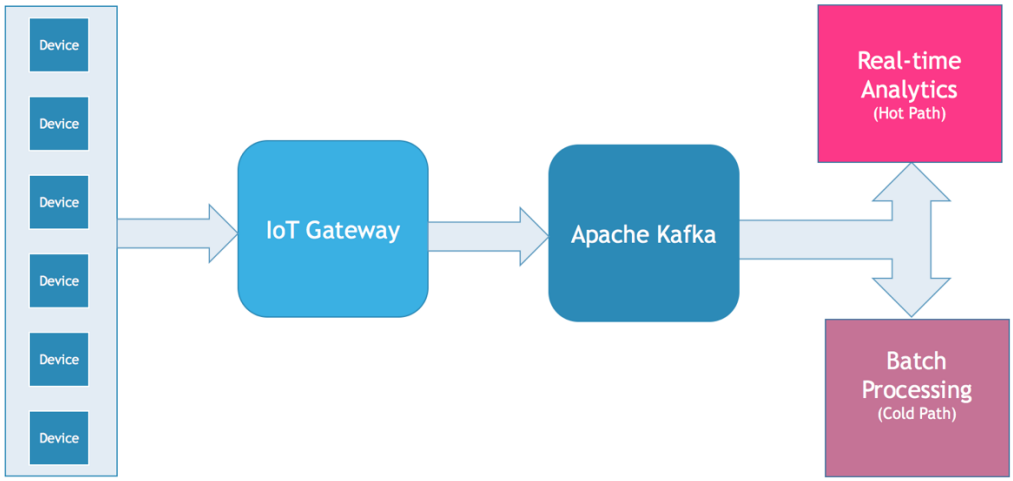

Figura 7 - Modelo simples de captação de dados Streaming no Apache Kafka

5. Definições de negócio:

Para o desenvolvimento do Modelo e responder as perguntas, foi importante também entender termos específicos do negócio, definições e abordagens para uma melhor solução. Desta maneira, listei algumas definições a seguir:

- Pousio (ou poisio entre outro milhões de espec.), em agricultura, é nome que se dá ao descanso ou repouso proporcionado às terras cultiváveis, interrompendo lhe as culturas para tornar o solo mais fértil.

- Talhão nada mais é do que a unidade mínima de cultivo de uma propriedade que é construído com base em relevo e planejamento de mecanização.

Fonte: http://inteliagro.com.br/o-que-e-talhao/
https://pt.wikipedia.org/wiki/Pousio/