# Business Intelligence

## Tudo o que você precisa saber para ingressar na área de BI

Entendendo BI - Alan Duarte

# Índice

# Seja bem vindo!

## Alan Duarte

Primeiramente gostaria de me apresentar,  meu nome é Alan Duarte Moreira, tenho 26 anos, atuo na área de BI como consultor há 7 anos e atualmente estou trabalhando como home office para uma empresa americana. Iniciei na área de BI com um estágio de ETL, na época não tinha conhecimento do que era ETL e muito menos o que era o BI, porém aceitei o estágio pela sede de mudança do meu emprego antigo, no qual estava estagnado.

No meu estágio, aprendi sobre SQL, banco de dados, modelagem, e o mais importante: o conceito do ETL e as ferramentas no qual a empresa trabalhava (na época PowerCenter e Data Quality). Era tudo muito interessante e desafiador. Logo então, fui contratado nessa empresa, e com o passar dos anos, consegui outras oportunidades na área. No inicio foi bem difícil, não tinha ninguém que me ajudasse. Só comecei a realmente entender melhor o conceito do BI, como funcionava o processo desde o início extraindo os dados, até o fim construindo os painéis, relatórios e as análises, nas ultimas duas empresas que trabalhei.

Durante esse tempo todo fui me especializando, fiz vários cursos e tirei as certificações nas ferramentas: PowerCenter, Data Quality, MicroStrategy, PowerBI, Alteryx e Tableau. Durante toda minha trajetória sempre tive  uma vontade muito grande de ensinar o pouco que sei e de ajudar as pessoas, para que o caminho delas fossem

mais fácil do que eu percorri para ter esse conhecimento. Esse foi o grande motivo para a criação da minha página **Entendendo BI.**

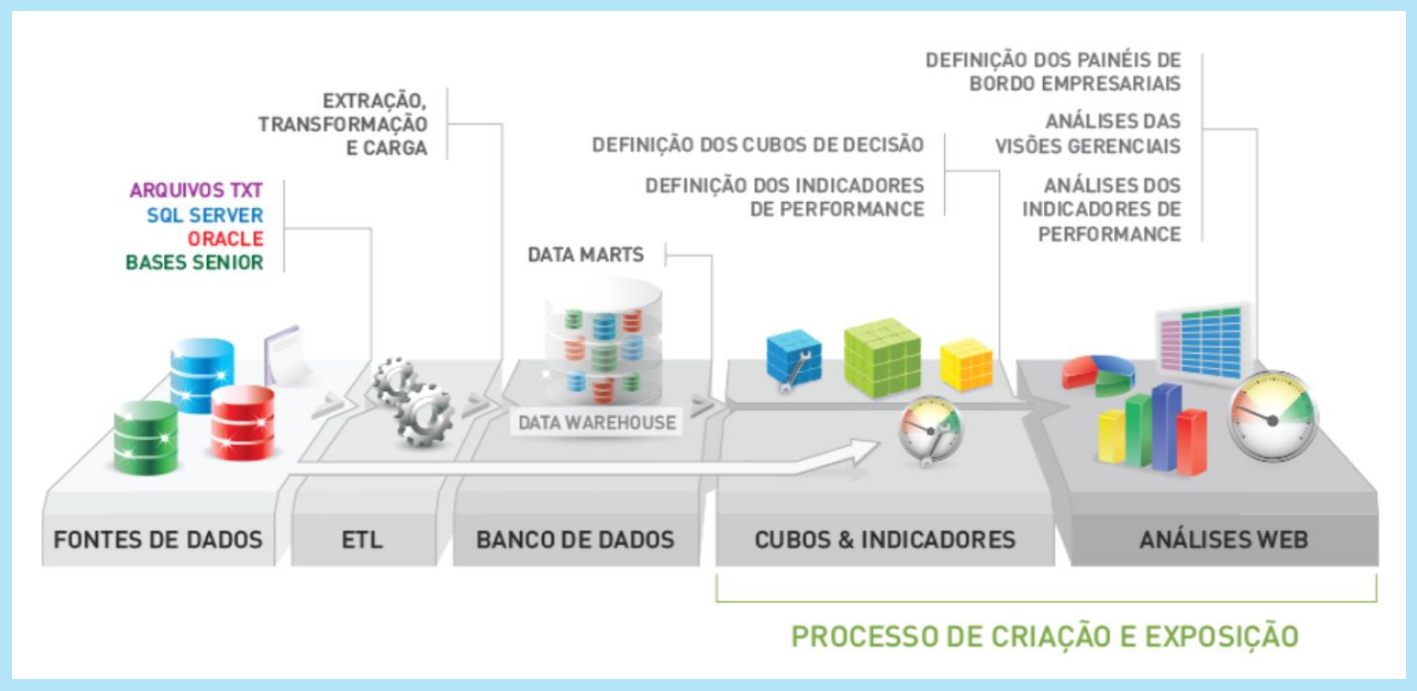
Neste e-book vou tentar resumir para você tudo aquilo que é **necessário saber para iniciar** na carreira de BI, seus **conceitos** e suas principais **ferramentas**. Com isto creio que você estará bem encaminhado para ingressar na carreira. Não se preocupe se durante a leitura você não conhecer algum termo, especialmente se tiver em negrito, pode ter certeza que mais pra frente ainda neste documento detalharei sobre ele!

Boa leitura e bons estudos!

# O que é Business Intelligence

Business Intelligence é basicamente um processo estratégico que tem como objetivo principal transformar dados brutos em insights de negócios, melhorando o desempenho de uma organização, auxiliando no processo de tomada de decisões da área de negócios.

Como isto é feito? Abaixo, demonstro um processo básico de um projeto padrão de BI:



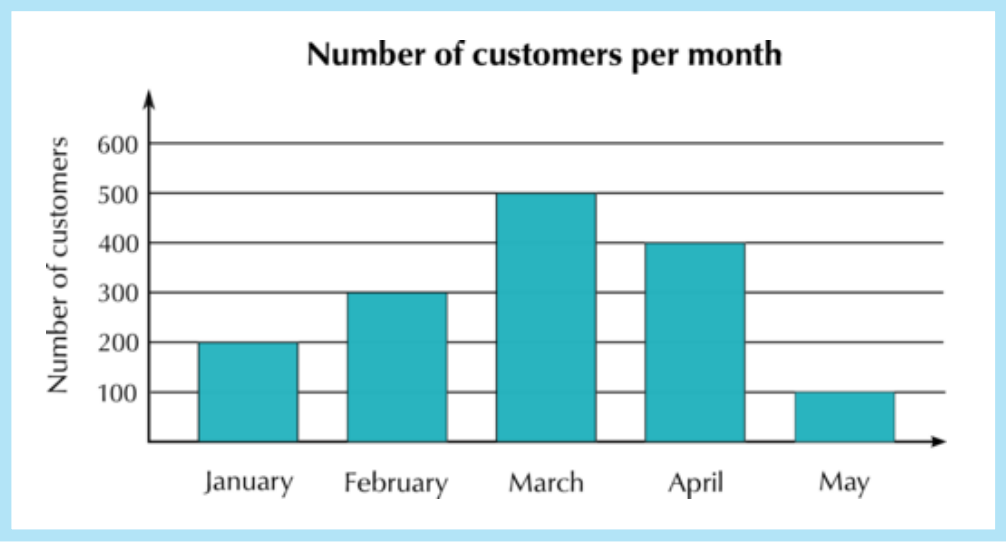

1. Os dados “crus” são extraídos de diversas **fontes** de dados de diferentes tipos, formatos e conexões;
2. Esta extração é feita pelo processo de **ETL** (Extract, Transform and Load) O ETL ajusta, trata, aplica as regras de negócio de dados e carrega-os no **Data Warehouse**.
3. No **DW** os dados estão unificados, tratados e modelados de forma **multidimensional**. O DW será basicamente o lugar onde os cubos e indicadores irão ler os dados, para as análises posteriores

4. Após o DW, são criados os **cubos** e **indicadores**, o que é basicamente a criação da **camada semântica**. A camada semântica é simplesmente o mapeamento das tabelas e suas respecitvas colunas do DW para dentro das ferramentas de BI;
5. Após tudo isto definido, é iniciada então a construção dos painéis, relatórios e dashboards, por meio das **ferramentas** de BI. São estas ferramentas que irão possibilitar a exibição de todo esse trabalho realizado de uma forma intuitiva e fácil de compreender.

Até o DW e a criação dos cubos, chamamos o processo de “Back-end”, que é o que fica por trás das cortinas. A parte “front-end” são então os relatórios e dashboards que são construídos e disponibilizados para os clientes, a fim de poder auxiliar nas decisões.

**Mas que tipos de decisões ? Um exemplo:**

Se fizermos um levantamento de dados de um parque de diversões, conseguiria identificar a situação abaixo, no qual  o mês de maio é o que menos tem pessoas frequentando o mesmo. Porque será que maio teve **1/5** das pessoas relacionadas ao mês de março ? Será que tem algo relacionado com férias escolares? Ou então maio neste país é um período chuvoso ? Somente neste exemplo poderíamos ligar a base de dados deste parque com a base climática, colégios e várias outras. Assim o dono deste parque poderia então decidir por ou abaixar o preço do ingresso em maio, ou até mesmo nem abrir neste mês já que a procura é tão baixa e evitaria certos prejuízos



# A ORIGEM DO BI

Não quero me prender muito à história do BI, a ideia deste e-book é mais ensinar sobre seus conceitos principais e essenciais. Mas é importante saber um pouco de como tudo começou. O BI desde o seu início até hoje, pode se dizer que passou por 3 fases. O BI **1.0**, **2.0** e atualmente estamos vivendo o que alguns chamam de **3.0**, outros que chamam de **2.5**.

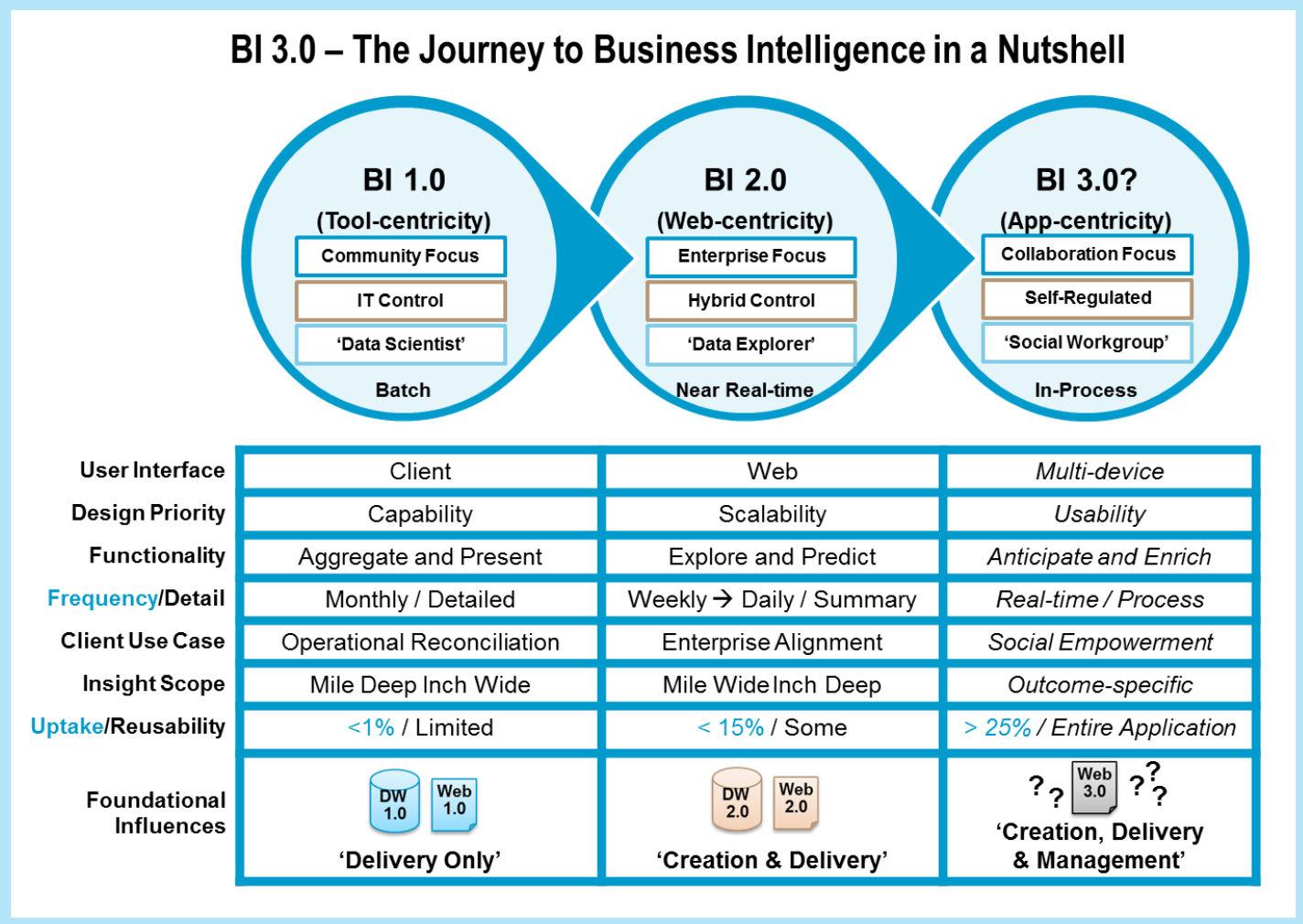
A primeira parte do BI, chamada de **BI 1.0**, se refere à uma era onde o BI existiu durante o final dos anos 90 e o início dos anos 2000, com o advento das tecnologias de **SQL**, **ETL**, **OLAP** e **DW.** Nesta época, os dados eram consolidados em um sistema único e as **queries** SQL eram usadas para extrair os dados de várias tabelas ao mesmo tempo. Neste período, o maior problema dos projetos de BI eram que eles ainda estavam com o controle centralizado nos departamentos de TI, os dados eram “sujos” e os relatórios demoravam bastante para ser entregues à gerência.

Do meio para o final dos anos 2000, houve um grande passo do BI em relação à sua maturidade, indo muito além de um simples gerador de relatórios, para um sistema de integração quase em tempo real ao processamento, com colaboração, “**self-service**” e descoberta de dados, podendo estar disponíveis off-line e on-line, sendo esta mudança chamada de **BI 2.0**. O 2.0 focou muito na parte da conectividade de seus “reports” para a web, promovendo flexibilidade e o dado certo para as pessoas certas. Muitas dessas mudanças foram devidas ao crescimento da internet e suas aplicações web.

Após toda a exposição dos dados evidenciada no BI 2.0, agora nos ambientes corporativos são necessárias ferramentas de visualização mais **intuitivas** e robustas, o que se refere à dashboards **interativos**, diversos tipos de gráficos e animações, sendo efetivos o suficiente para analisar a informação vinda de dentro e de fora da organização. Estamos vivendo agora o **BI 3.0**! Ele se diz respeito justamente a isso, ao fato do **BI estar tomando uma posição cada vez mais importante no comitê estratégico e negocial**, possibilitando com que o próprio usuário do negócio, sem muitos

conhecimentos técnicos, possa criar gráficos e explorar os dados de diferentes maneiras de dados, independentemente, sendo este conceito recém chamado de "**Self-Service BI**".

Abaixo temos uma imagem demonstrativa ilustrando exatamente o crescimento do BI até hoje:



# Dado e Informação

Dados e informações são muitos parecidos e ambos são bases para a formação do conhecimento, porém eles diferem em um conceito básico:

- O dado não possui nenhum significado relevante por si só. Olhando para um dado somente, não é possível extrair uma conclusão.
- A informação é justamente o resultado do processamento de dados. É o dado após uma certa organização e estruturação.

E o conhecimento? Nada mais é do que a informação processada e transformada em experiência pelo indivíduo. As informações são valiosas, porém o conhecimento demanda um saber, produz reflexão, ideias e experiências que as informações por si só não serão capazes de mostrar.

Pra facilitar, um exemplo prático e fácil:

| | |
|---|---|
| Dado | Chuva - São Paulo - 12/03/2019 |
| Informação | Choveu em São Paulo no dia 12 de Março de 2019. |
| Conhecimento | Sempre quando chove muito em São Paulo, as chances de ocorrer enchentes são grandes, assim como o trânsito fica ainda mais caótico. É sempre ideal checar a previsão do tempo antes de sair de casa |

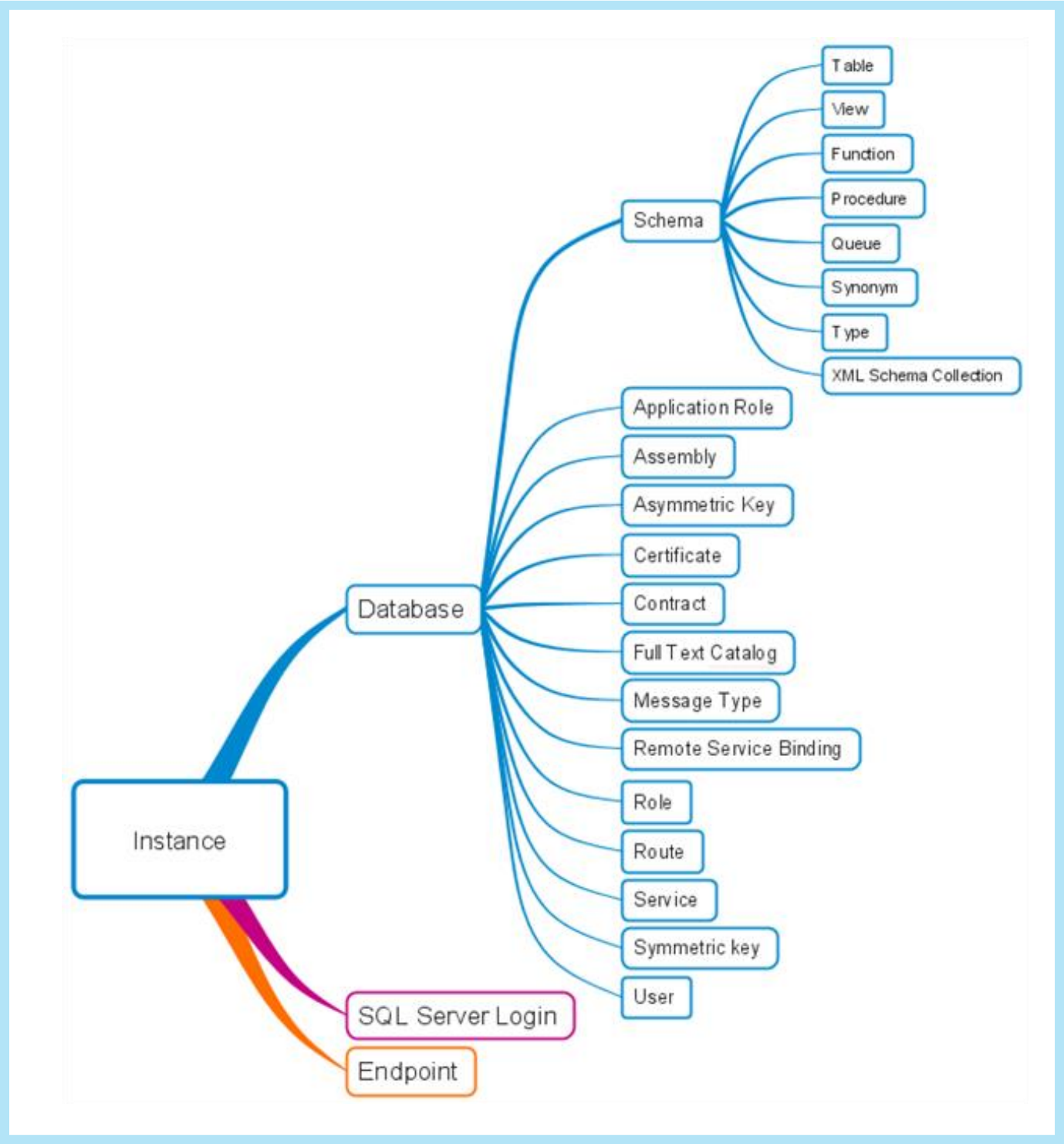
# Bancos de Dados

Banco de dados como o próprio nome ja diz, é um local que armazena uma coleção de dados organizados em um conjunto previamente definido, formalmente chamadas como **tabelas**, no qual os dados podem ser acessados e manipulados de várias formas, sem a necessidade de reorganização da estrutura das tabelas originais. Estes acessos são realizados por meio das ferramentas **SGBDs** (Sistemas Gerenciadores de Bancos de Dados) e com comandos da **linguagem SQL**. A imagem universal que representa um database hoje é:

Atualmente existem diversos banco de dados no mercado. Os mais comuns que temos são: **Oracle**, **Microsoft SQL Server**, **MySQL**, **IBM DB2 e PostgreSQL.** Cada banco é operado por sua ferramenta **SGBD** respectiva, e é importante saber que por mais que sejam muito parecidos, cada um tem sua especificidade.

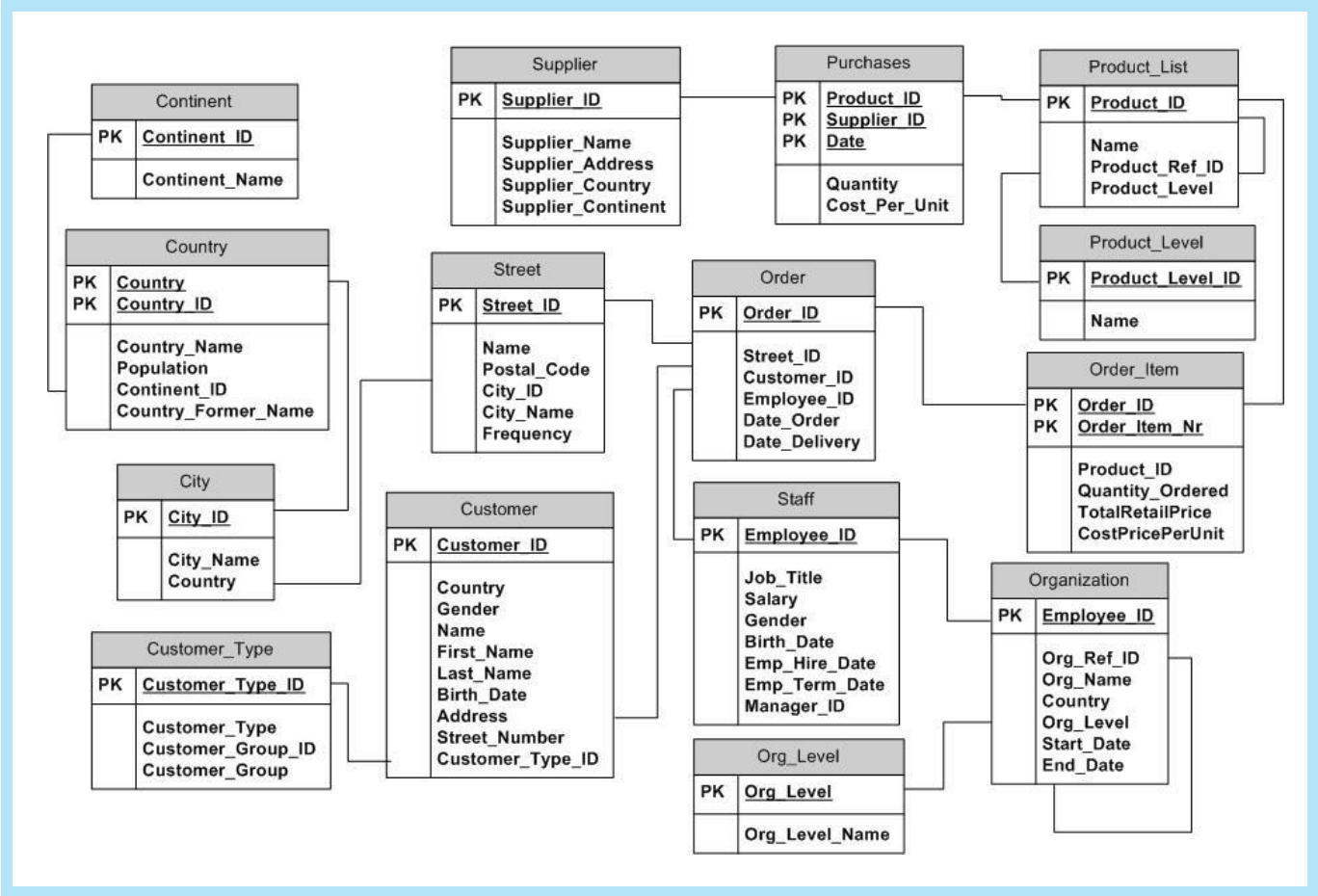
Além das tabelas, dentro dos banco de dados existe um mundo de objetos, como **esquemas**, **views**, **procedures**, **funções**, **pacotes** e vários outros. Não vou me aprofundar em todos eles, somente nos mais importantes para **este momento**. Embora as pequenas diferenças que temos de um banco pra outro, a maior parte se mantém. As tabelas geralmente se encontram dentro de um **Esquema**, que fica dentro de um **Database** ou **Banco,** que por sua vez está dentro de um **Servidor** ou **Instância.**

Para deixar um pouco mais claro o que estou falando, segue abaixo uma imagem de como é a estrutura de um banco de dados **SQL Server**:



# Modelagem de Dados

As tabelas do banco de dados são criadas e modeladas conforme a **modelagem relacional**. Essa modelagem basicamente define as regras de ligação que as tabelas possuem umas com as outras, como podemos ver no modelo abaixo:



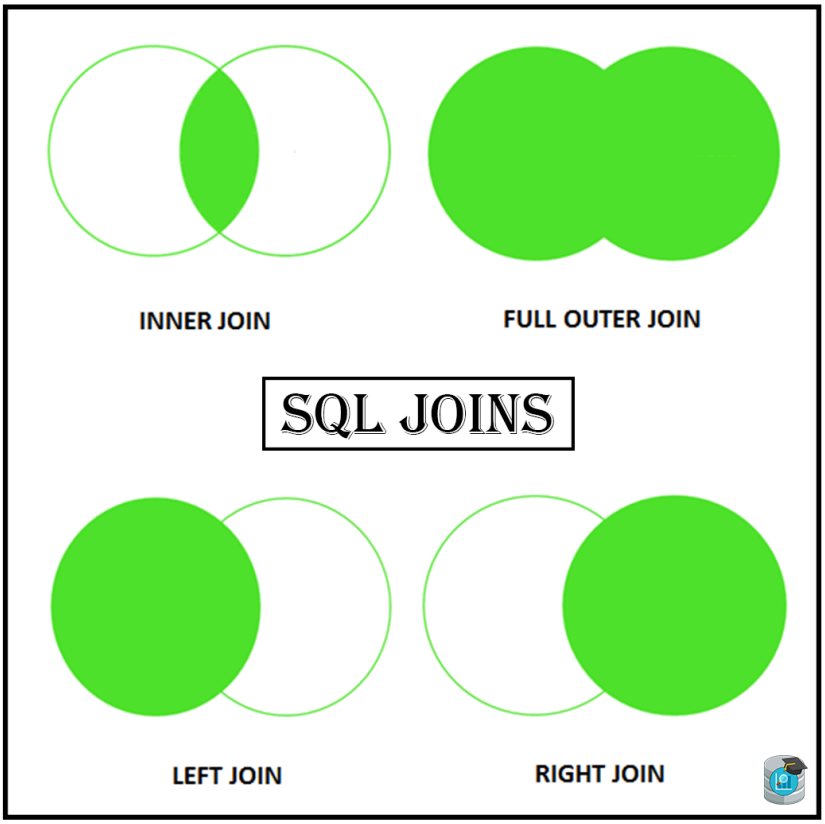
Existem basicamente 4 tipos de ligações possíveis a serem feitas para **cruzar** os dados de uma com a outra. Temos então os **Joins**:

**1) INNER JOIN**: O resultado traz somente os dados correspondentes das tabelas A e B

**2) LEFT JOIN**: O resultado traz todos os dados da tabela A + as correspondências com a tabela B

**3) RIGHT JOIN**: O resultado traz todos os dados da tabela B + as correspondências com a tabela A

**4) FULL OUTER JOIN**: O resultado traz todos os dados de todas as tabelas




Para exemplificar, logo abaixo temos duas tabelas, **Pessoas** e **Idades**. Iremos realizar os 4 tipos Joins entre as elas para demonstrar como são diferentes os resultados. Todo Join tem uma condição, no caso qual coluna deverá se corresponder com a outra para criar a **regra de ligação.** Neste exemplo iremos bater o ID_Pessoa com o ID_Idade.

Em SQL, o comando de um INNER JOIN entre as tabelas fica assim por exemplo:

```
SELECT * FROM Pessoas INNER JOIN Idades
ON Pessoas.ID_Pessoa = Idades.ID_Idade
```

OBS: O **asterisco** simboliza o retorno todas as colunas da query sendo executada, no caso as 4 colunas (2 de cada tabela).

Repare na imagem abaixo que quando não há correspondências e o join traz os dados das duas tabelas, os valores ficam vazios (**nulos**). Além disso, é bom ressaltar que não existe join melhor que outro. A escolha do join irá variar conforme cada tipo de resultado que você precisa:

| Pessoas | |
|---|---|
| ID_Pessoa | Nome |
| 1 | João |
| 2 | Carlos |
| 3 | Maria |

| Idades | |
|---|---|
| ID_Idade | Idade |
| 1 | 24 |
| 2 | 35 |
| 4 | 29 |

| INNER JOIN | | | |
|---|---|---|---|
| ID_Pessoa | Nome | ID_Idade | Idade |
| 1 | João | 1 | 24 |
| 2 | Carlos | 2 | 35 |

| FULL OUTER JOIN | | | |
|---|---|---|---|
| ID_Pessoa | Nome | ID_Idade | Idade |
| 1 | João | 1 | 24 |
| 2 | Carlos | 2 | 35 |
| 3 | Maria | | |
| | | 4 | 29 |

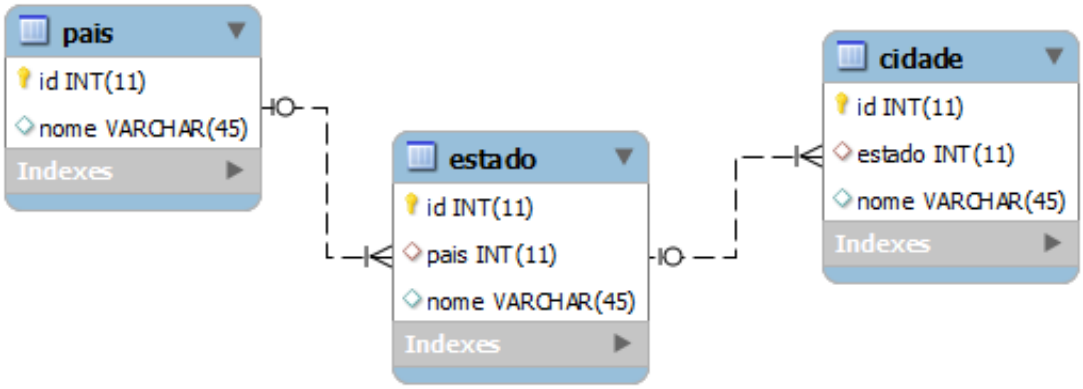
| LEFT JOIN | | | |
|---|---|---|---|
| ID_Pessoa | Nome | ID_Idade | Idade |
| 1 | João | 1 | 24 |
| 2 | Carlos | 2 | 35 |
| 3 | Maria | | |

| RIGHT JOIN | | | |
|---|---|---|---|
| ID_Pessoa | Nome | ID_Idade | Idade |
| 1 | João | 1 | 24 |
| 2 | Carlos | 2 | 35 |
| | | 4 | 29 |

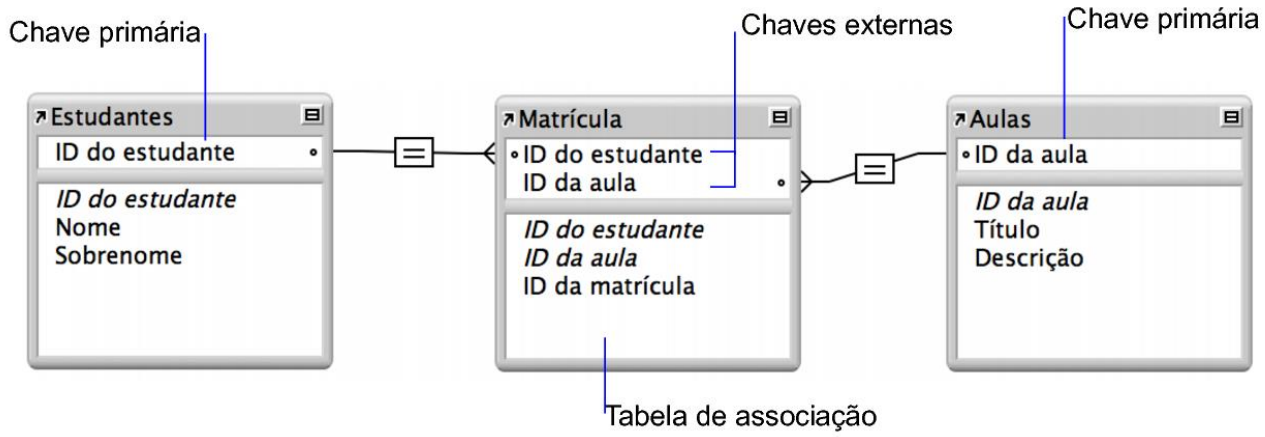
À respeito de modelagem de dados, existe um conceito muito importante chamado **cardinalidade.** A cardinalidade é simplesmente a **relação** que uma tabela possui com uma outra, a forma como elas se **comunicam**. No modelo de dados ela é demonstrada pelas linhas conectando as tabelas: Existem três tipos de relações entre tabelas (**1:1, 1:N e N:N**).

A **1:1** é uma relação de **Um para Um,** ou seja, para cada 1 registro em uma tabela A, só é possível conectar-se com 1 na tabela B. Exemplo simples: Em uma organização, a tabela **Gerente** é 1:1 com a tabela **Departamento**, pois pra cada departamento só se pode ter 1 gerente, e 1 gerente só pode gerenciar 1 departamento.

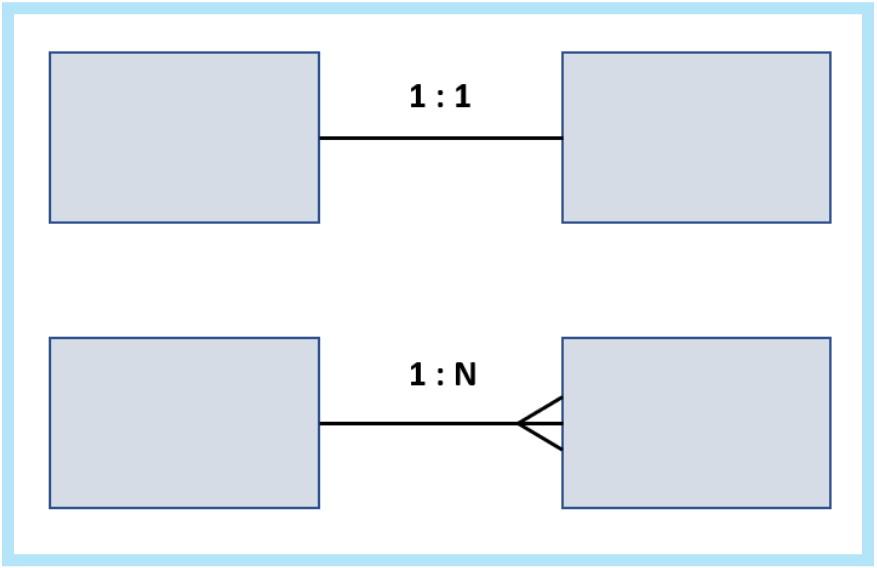
A **1:N** é uma relação de **1 para Muitos**, ou seja, para cada 1 registro em uma tabela A, é possível conectar-se com 1 ou vários na tabela B. Exemplo simples: As tabelas de **País**, **Estado** e **Cidade**. É fato que 1 País possui 1 ou vários Estados, assim como 1 Estado possui 1 ou várias Cidades.




A **N:N** é uma relação de **Muitos para Muitos,** ou seja, para cada 1 registro em uma tabela A, é possível se conectar com 1 ou vários outros na tabela B, **assim como o contrário**. Exemplo simples: As tabelas **Estudante** e **Aula,** 1 Estudante pode ter várias Aulas, assim como 1 Aula pode ter vários Estudantes. Esse tipo de relação é mais complicada, e para esses casos é preciso que se crie uma tabela **associativa** entre elas:



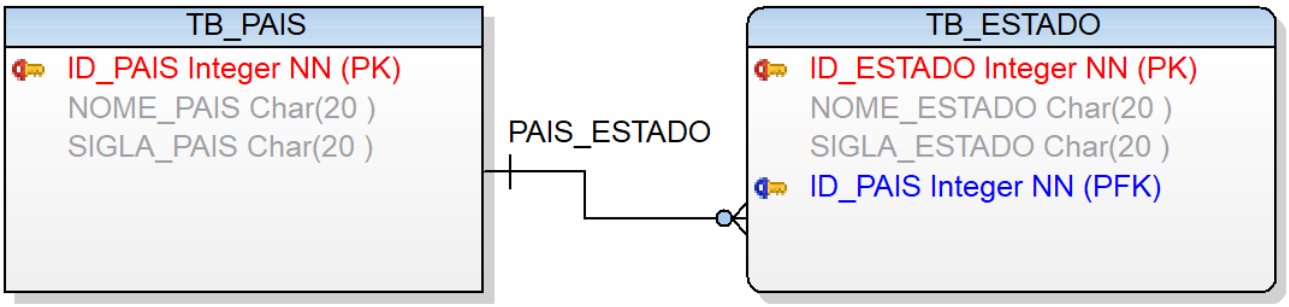

Se você reparou bem, existe uma forma de representar os tipos de ligações. Quando é **1** se usa uma linha **contínua**, e quando é **Muitos** se usa um "**pé de galinha"**.




A **chave primária** de uma tabela é uma **coluna eleita** em que os **dados nela não se repetem e não são nulos**. Se for necessário, é comum a criação de **chaves primárias compostas**, que são uma ou mais colunas formando a **PK** (Primary Key) da tabela. Por exemplo, se temos uma tabela com os dados de **Pessoas**, de cara a PK mais indicada no caso é o **CPF.**

As **chaves estrangeiras** são criadas quando se é feita a relação de uma tabela **pai** com uma tabela **filha.** No caso exemplo dos países, estados e cidades, **relacionalmente** falando, País é **pai** de Estado, assim como Estado é **pai** de Cidade.

No modelo de dados, País tem uma **PK** própria. Quando é feita a ligação de País com Estado de 1:N, **automaticamente** uma **FK** (Foreign Key) é criada em Estado, relacionando os dados de lá com a tabela **pai.** Da mesma forma é feito o processo com a tabela de **Cidade**.




Para esclarecer melhor, à nivel de dados ficaria assim então:

| País | |
|---|---|
| **ID** | **Nome** |
| 1 | Brasil |
| 2 | EUA |

| Estado | | | |
|---|---|---|---|
| **ID** | **Nome** | **UF** | **FK_País** |
| 1 | Distrito Federal | DF | 1 |
| 2 | São Paulo | SP | 1 |
| 3 | Rio de Janeiro | RJ | 1 |
| 4 | Florida | FL | 2 |

| Cidade | | |
|---|---|---|
| **ID** | **Nome** | **FK_Estado** |
| 1 | Brasília | 1 |
| 2 | São Paulo | 2 |
| 3 | Rio de Janeiro | 3 |
| 4 | Miami | 4 |
| 5 | Orlando | 4 |

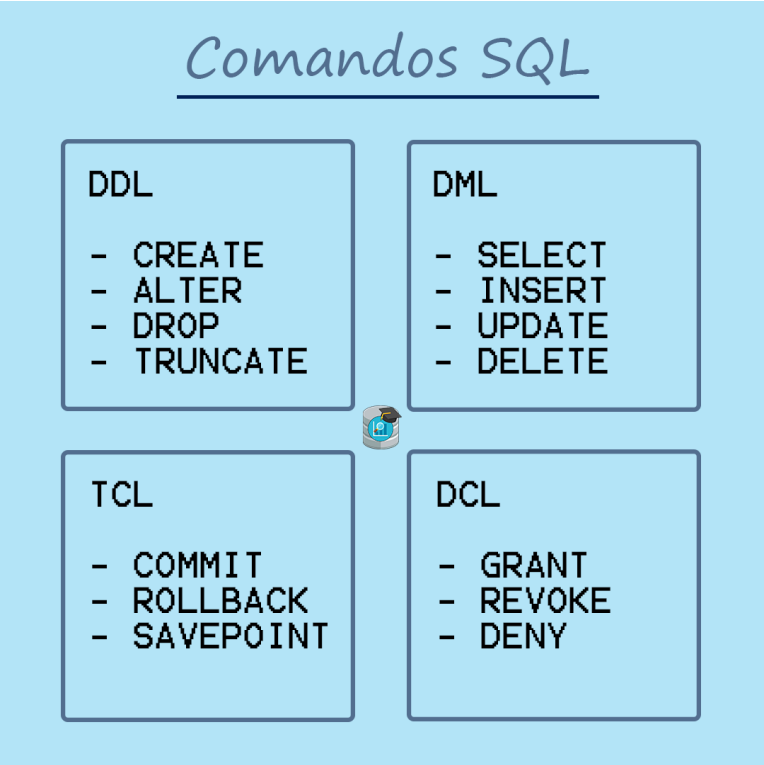
Mas para que servem estas chaves, afinal? A resposta é simples: Deixar o banco de dados **amarrado** e **protegido**. Com essas chaves criadas, juntamente com elas são criadas as **constraints** no banco de dados. Constraints nada mais são do que **restrições de dados**.

Ao tentar realizar um **INSERT** na tabela de Estado, devemos fornecer **valores** para as coluna de ID, Nome, UF e principalmente **FK_País.** Se deixarmos **NULO** (Vazio) na coluna de FK_País, ou então um **valor de ID que não existe na tabela de País**, a constraint irá verificar isso e retornar um erro dizendo que houve violação de integridade, não sendo possível encontrar o registro pai.

# Linguagem SQL

**SQL** (Structured Query Language), ou seja Linguagem de Query Estruturada, é a linguagem que permitirá manipular os objetos do banco de dados, tais como as **tabelas**, **views**, **colunas**, **pacotes**, **procedures** e vários outros.

Os comandos SQL podem ser divididos em 4 grandes categorias:




**1) DDL** (Data Definition Language) - São os comandos que vão definir a estrutura do banco de dados e seus objetos;

**2) DML** (Data Manipulation Language) - Comandos de manipulação de dados, os mais simples de serem usados;

**3) TCL** (Transaction Control Language) - Definição das transações do banco de dados;

**4) DCL** (Data Control Language) - Estes comandos lidam com direitos, controles e permissões dos objetos do banco

No caso das tabelas, pode-se usar os comandos **DDL** para organizar sua estrutura e os comandos **DML** para manipular seus dados, como no exemplo abaixo:

```
CREATE TABLE dbo.Tabela(
    Coluna1     INT,
    Coluna2     Char,
    Coluna3     Date,
    Coluna4     Boolean
);

INSERT INTO dbo.Tabela
(Coluna1, Coluna2, Coluna3, Coluna4)
VALUES
(1, 'A', '23/03/2019', TRUE);

TRUNCATE TABLE dbo.Tabela;

DROP TABLE dbo.Tabela;
```

Neste exemplo estamos criando uma tabela com **CREATE**, repare que para cada coluna criada é necessário definir seu **datatype.** Após criar a tabela, inserimos um registro e logo após limpamos todos os dados da tabela com o **TRUNCATE.** Em seguida deletamos a tabela do esquema **dbo** com o comando **DROP**.

# Tipos de Dados

No capítulo anterior foi mostrado como que se cria uma tabela com SQL, e ao criá-la, foi necessário o **datatype** (tipo de dados) de cada coluna. Atualmente existem diversos datatypes, como mencionei anteriormente, alguns datatypes podem variar de banco para banco, porém a grande maioria está presente em todos eles. Abaixo temos uma lista de alguns dos datatypes mais comuns hoje em dia:

**Datatypes de Texto:**

- CHAR (*necessário definir um tamanho*)
- VARCHAR (*necessário definir um tamanho*)
- VARCHAR2 (*necessário definir um tamanho*)
- TEXT
- BLOB

**Datatypes de Número:**

- INT
- INTEGER
- SMALLINT
- BIGINT
- DECIMAL (*necessário definir um tamanho e a quantidade de casas decimais*)
- NUMERIC (*necessário definir um tamanho e a quantidade de casas decimais*)

**Datatypes de Data:**

- DATE
- DATETIME
- TIMESTAMP
- YEAR
- TIME

# ETL e Qualidade de Dados

O Extract, Transform and Load (**ETL**) é o nome dado ao processo de **automação** da **extração** de dados de uma ou mais **origens**, podendo ter um ou mais sub-processos, **qualificando**, **consolidando**, **integrando** e por fim realizando a **inserção** dos dados em um ou mais destinos definidos no processo. A prática do ETL veio para revolucionar principalmente o processo de **integração de bases de dados heterogêneas**, o que antes era oneroso ou **impossível** de se realizar, demandando grandes trabalhos manuais e demorados. O ETL pode ser dividido em 3 etapas:

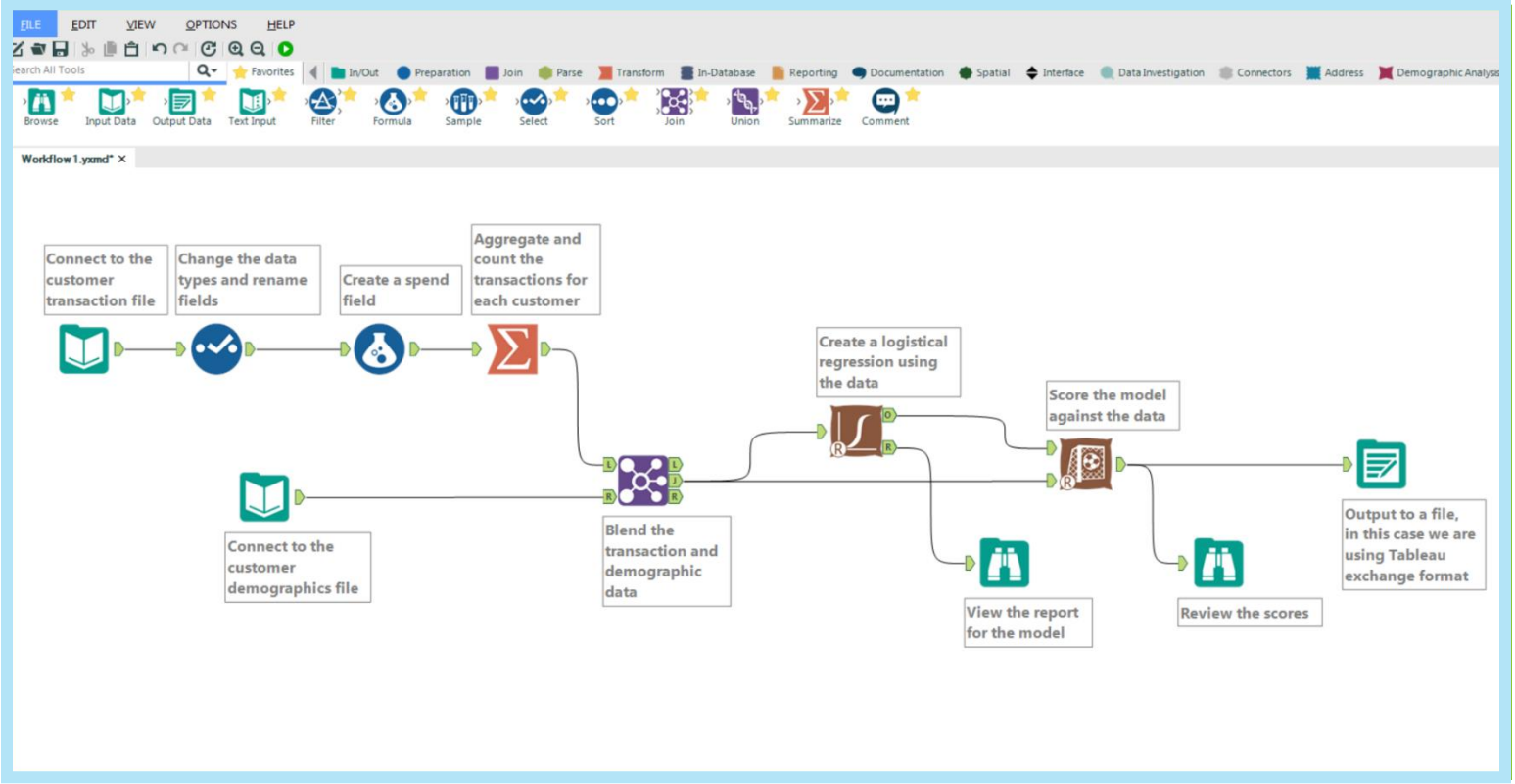
**1) Extração de dados**: Passo inicial no ETL que se dá pela conexão e extração dos dados das origens para dentro de seu ambiente.

**2) Transformação e Qualificação de dados:** Fator crítico no processo de ETL de projetos de DW. A qualificação de dados é responsável por **realizar todo o tratamento necessário a ser feito nos dados**, a fim de **melhorar a qualidade** geral da informação. A qualificação de dados possui 6 pilares:

- **Completude**: Qual dado está faltando ou sem uso?
- **Conformidade**: Qual dado está armazenado em um formato fora do padrão?
- **Consistência**: Quais valores de dados estão com informações conflitantes?
- **Precisão**: Qual dado está incorreto ou fora da data?
- **Duplicidade**: Quais registros estão repetidos?
- **Integridade**: Qual dado não está sendo referenciado corretamente?

**3) Inserção de dados**: Passo final do ETL onde os dados extraídos, tratados e qualificados são **inseridos**. Este passo é importante pois é nessa etapa em que é definido o modo em que os dados devem ser **aceitos** ou **rejeitados** no destino.

Abaixo segue um exemplo de um **mapa ETL**, este no caso feito na ferramenta **Alteryx**:




**O mapa ETL é o que vai definir toda a lógica de carga de uma ou mais tabelas.**

É neste momento em que a **maioria** das **regras de negócio** da organização são aplicadas. Dentro do mapa é possível se **conectar** com bancos de dados, arquivos, web services e vários outros tipos de fontes de dados. Após se conectar com uma fonte, inúmeras **transformações** podem ser realizadas até a carga final dos dados em uma tabela ou vários outros tipos de objetos. Um exemplo de um ETL bem simples é:

1. Ler os dados da Tabela A;
2. Fazer um **uppercase** (colocar em maíusculo) em todas as colunas de texto;
3. Se tiver dados nulos (vazios), **transformar** para “Sem Informação”
4. Verificar se os dados **já existem** na Tabela B. Se existir, realizar um **UPDATE**, se não existir, realizar um **INSERT**.

# DataWarehouse

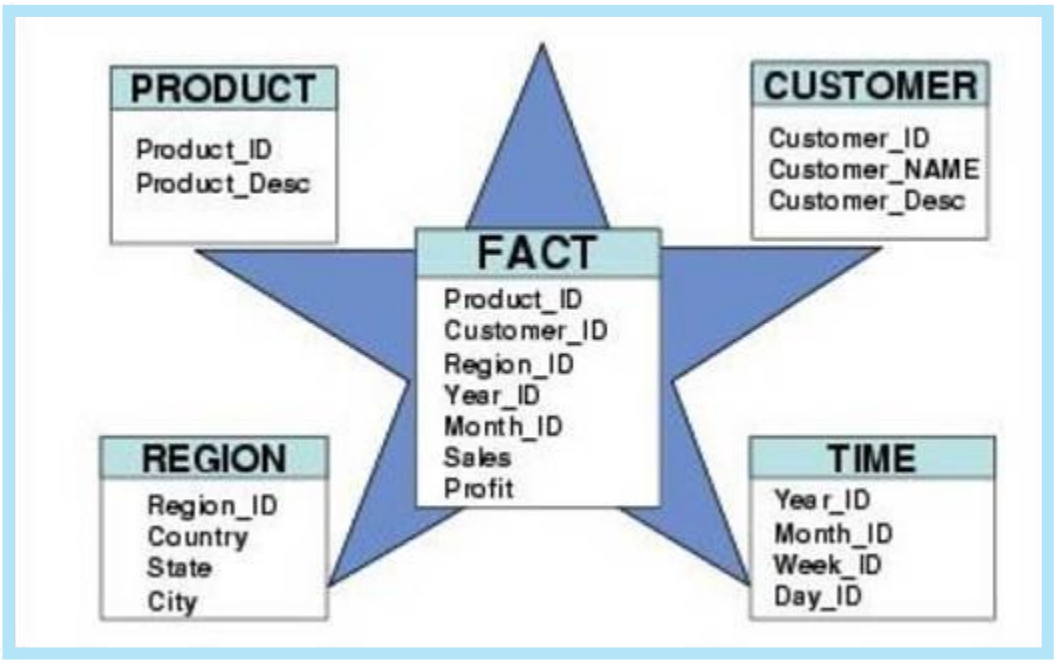
**DataWarehouse** (DW) como o próprio nome diz também, é um armazém virtual que tem como objetivo principal manter os dados provenientes do **ETL** e de suas diversas fontes de dados. No DW estes dados são **estruturados** e **organizados**, para posteriormente poderem servir como base para os relatórios, painéis e dashboards do **BI**.

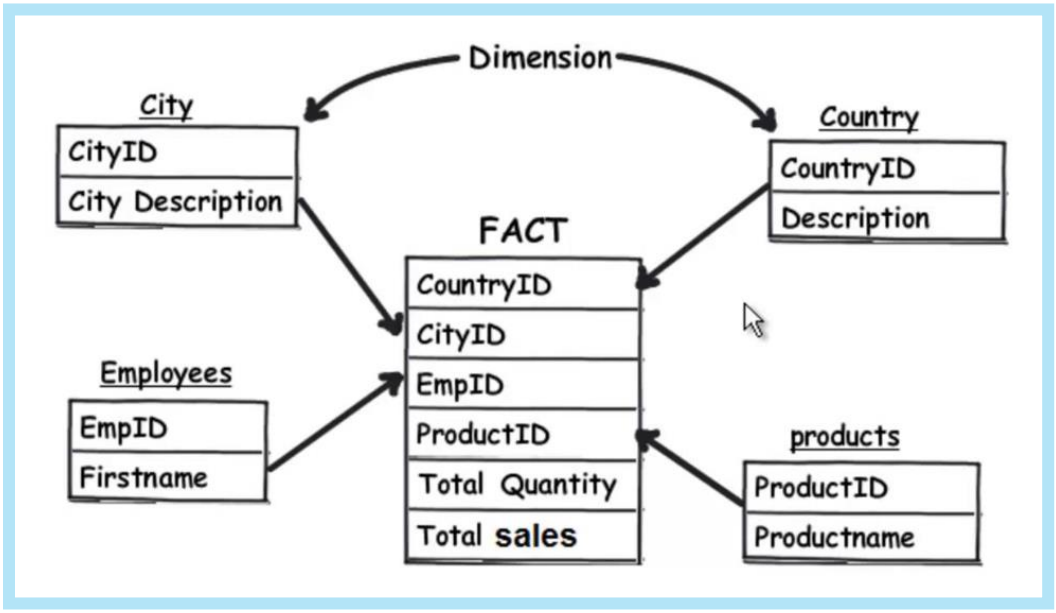
O DW usa um **design diferente** dos demais bancos de dados tradicionais (**modelagem relacional),** ele possui uma modelagem **específica** e **otimizada** para manter a precisão do dado ao longo de suas mudanças. O DW é modelado para manter um **longo período histórico** dos dados conforme suas alterações **sobre o tempo**, ao **contrário do banco transacional**. Essa modelagem é chamada de modelagem **multidimensional.**

No DW geralmente temos **a ausência das chaves PKs FKs e constraints (Neste caso é importantíssimo garantir as regras de integridade das constraints e FKs durante a fase de ETL)**, gerando um enorme ganho de **performance** nas consultas de dados, com um tempo de resposta muito **superior** em relação à uma consulta direta à base de dados transacionais, onde existem transações e atualizações constantemente, onerando mais ainda o tempo de resposta.

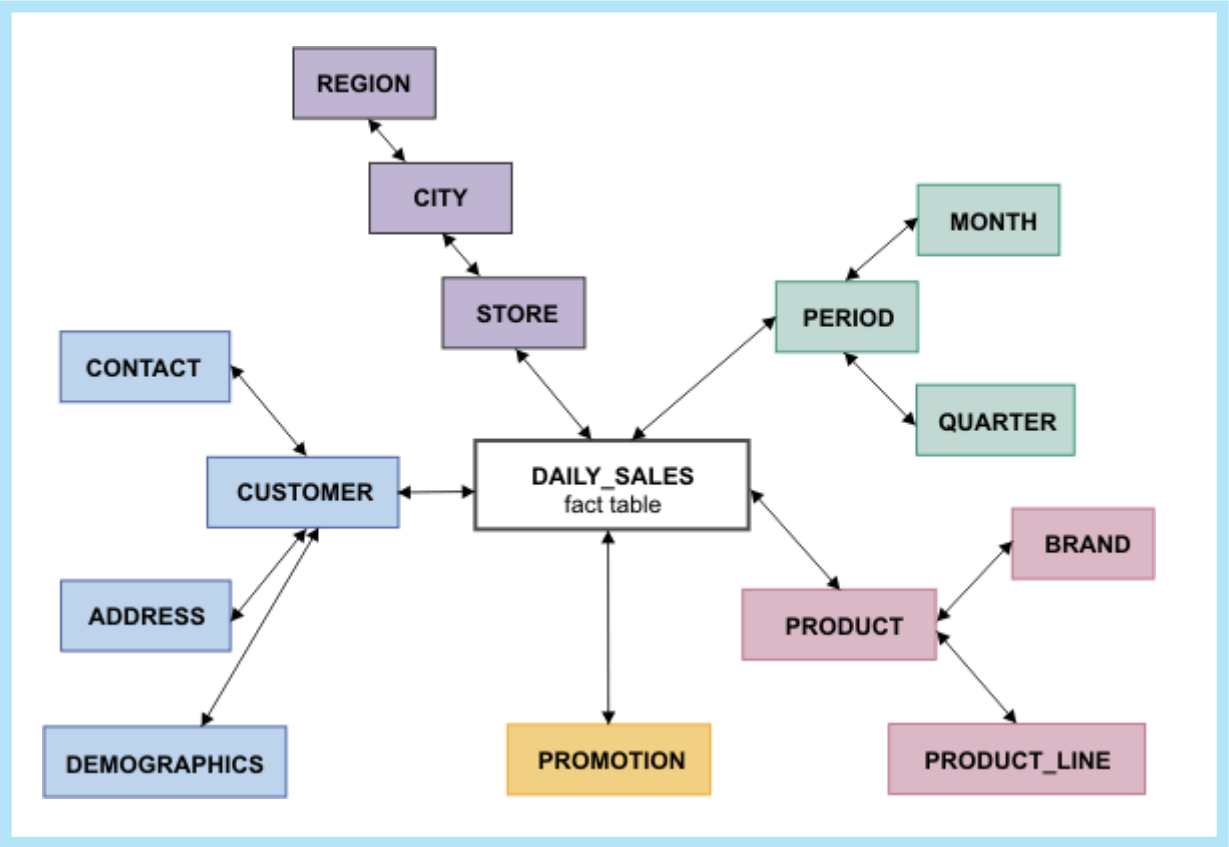
Os dois grandes idealizadores do DataWarehouse foram **Inmon** e **Kimball**. Uma leitura excelente que recomendo é o livro "**The DataWarehouse Toolkit**". O livro mostra sobre os vários tipos de **modelagens** que servem para abrigar os dados do BI, e as melhores **abordagens de desenvolvimento** a serem feitas ao modelar um DW. Dentre as modelagens, resumidamente as duas mais famosas são:

**Modelo Star Schema**: O modelo estrela é formado por **fatos** e **dimensões**, em que todas as tabelas de dimensões se relacionam **diretamente** com as tabelas de fatos, e basicamente nas dimensões estarão presentes as **informações descritivas** de cada assunto, cada uma com sua respectiva **chave gerada** que será usada na fato como **apontamento** desta informação



**Modelo SnowFlake**: No modelo de floco de neve as tabelas dimensionais relacionam-se com as tabelas de fatos, mas **algumas** dimensões relacionam-se apenas entre elas, isto ocorre para fins de **normalização das tabelas dimensionais**, visando diminuir o espaço ocupado por estas tabelas, com estas então se tornando dimensões apenas auxiliares.

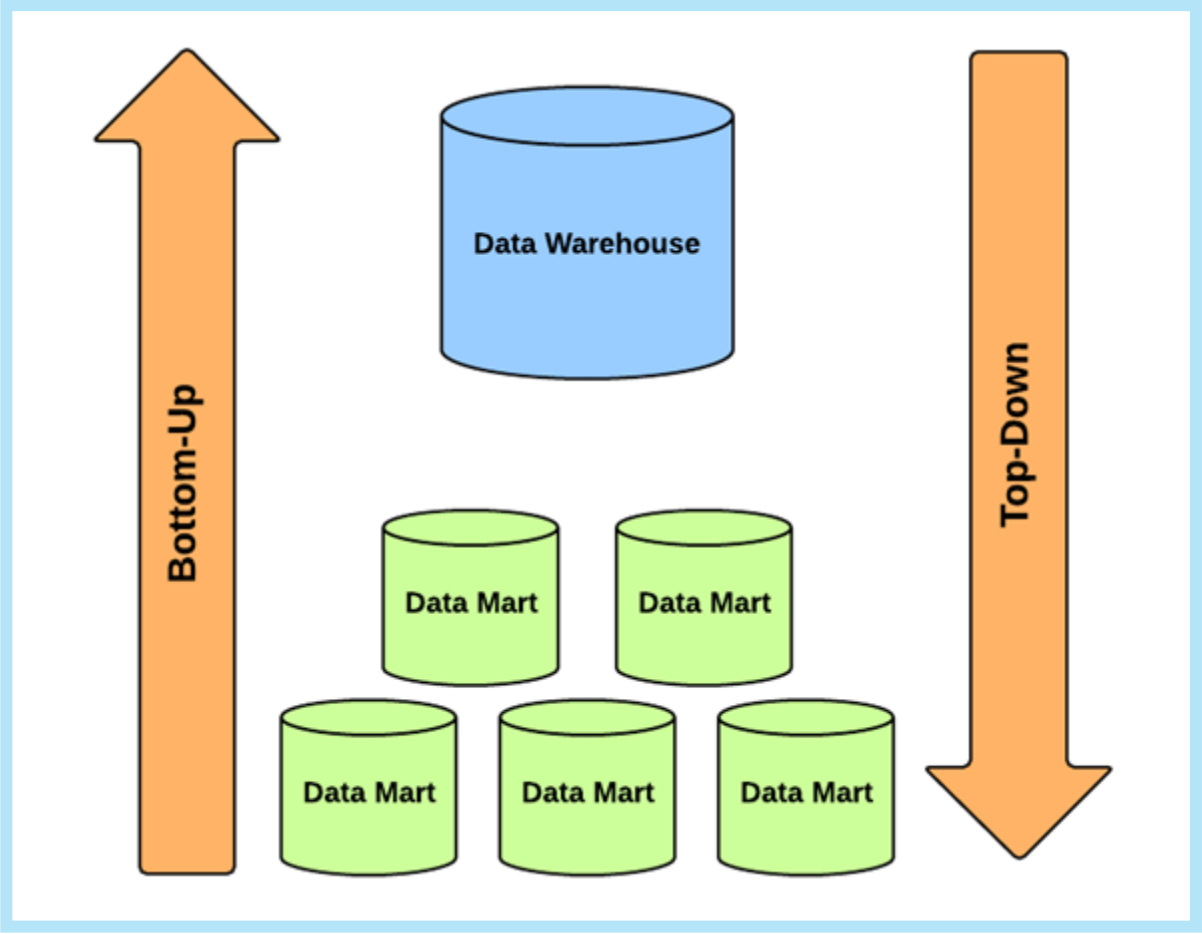



Como citado anteriormente, para o desenvolvimento de um DW existem dois tipos de abordagens famosas, sendo elas as técnicas de **Top-Down** e **Bottom-Up**. Na perspectiva do DW, as diferenças entre as abordagens se deve ao posicionamento que o **Data Mart** desempenha no processo de construção. O **Data Mart** é basicamente um mini DW, ele se faz necessário quando o DW acaba ficando muito grande, e logisticamente fica melhor organizar os dados em subconjuntos menores, como por exemplo um DW grande com 3 DataMarts (Vendas, Compras e Finanças):

• **Top-Down**: Está técnica é defendida por **Inmon**, em que o processo se começa por cima, com o DW sendo derivado dos Data Marts. Esté método vai do mais alto nível ao mais detalhado. Essa abordagem requer uma visão geral da solução e possui grande abrangência, onde requer a construção completa do DW para a disponibilização posterior dos Data Marts.

• **Bottom-Up**: **Kimball** defende esta técnica, em que os Data Marts são derivados do DW, indo do menor nível ao maior nível. Está técnica visa o famoso “dividir para conquistar”. Em projetos onde é difícil enxergar os limites e complexidades, se torna importante a construção incremental dos **Data Marts** até se chegar ao DW.

OBS: É importante citar que a escolha da abordagem de desenvolvimento irá variar bastante conforme o projeto. Existem projetos que se adaptam facilmente à uma abordagem, e outros não



# Tipos de Cargas de Dados

Como citado no processo de ETL, existem algumas formas diferentes para se realizar as cargas nas tabelas (**dimensões**) do **DW**. Por exemplo, se estamos com um ETL básico carregando os dados de uma tabela A para uma tabela B, temos em princípio 2 opções:

1. Inserir os registros novos
2. Atualizar os registros antigos

| Tabela A | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 1.800 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

| Tabela B | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| | | |
| | | |
| | | |

Isto é bem simples de ser feito. Basta criar uma lógica no mapa correspondendo pelo **ID** das duas tabelas, se **não houver** correspondência, indica que o registro é novo e deve ser **inserido**. Se **houver** correspondência, indica que o registro já existe e deve ser **atualizado**, por exemplo:

Na primeira carga, 3 registros foram inseridos:.

| Tabela A | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 1.800 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

| Tabela B | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 1.800 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

A complicação aparece justamente na hora de realizar a atualização dos dados. Como a organização irá decidir atualizar um registro existente? Para isto existem basicamente 3 **tipos de cargas de dados** de **SCD** (Slowly Changing Dimensions). SCD nada mais é do que uma **forma de carregar os dados no DW vagarosamente** no caso de suas alterações, não significa que o tempo de execução é devagar, mas sim a forma com que a **alteração é proporcionada no destino**. Abaixo os exemplos dos 3 tipos:

**SCD do Tipo 1:**

Este é o mais simples. Tudo o que é alterado no registro original é atualizado **diretamente** na tabela destino:

| Tabela A | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 2.200 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

| Tabela B | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 2.200 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

**SCD do Tipo 2:**

Este é um pouco mais complexo. A ideia desse tipo de carga é justamente obter o **controle histórico dos dados** no DW. Com isto os registros não são atualizados, e sim "**inseridos**" novamente com uma coluna indicando qual é o registro atual a ser considerado. O Comando de **UPDATE** mesmo só é feito na **Flag Atual**:

| Tabela A | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 2.200 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

| Tabela B | | | |
|---|---|---|---|
| **ID** | **Nome** | **Salário** | **Flag Atual** |
| 1 | João | R$ 1.800 | 0 |
| 1 | João | R$ 2.200 | 1 |
| 2 | Carlos | R$ 2.500 | 1 |
| 3 | Maria | R$ 3.200 | 1 |

**SCD do Tipo 3:**

O tipo 3 é bem similar ao tipo 2. A diferença é que o registro não é inserido novamente, e não existe mais uma coluna indicando o registro atual, o que ocorre é a **duplicação do atributo original**, resultando então em **Salário Anterior** e **Salário Atual**:

| Tabela A | | |
|---|---|---|
| **ID** | **Nome** | **Salário** |
| 1 | João | R$ 2.200 |
| 2 | Carlos | R$ 2.500 |
| 3 | Maria | R$ 3.200 |

| Tabela B | | | |
|---|---|---|---|
| **ID** | **Nome** | **Salário Anterior** | **Salário Atual** |
| 1 | João | R$ 1.800 | R$ 2.200 |
| 2 | Carlos | **NULO** | R$ 2.500 |
| 3 | Maria | **NULO** | R$ 3.200 |

Como Carlos e Maria não tiveram atualizações no salário, a coluna de Salário Anterior se mantém NULA.

Cada tipo de SCD tem suas **vantagens** e **desvantagens**. Enquanto o tipo 1 é mais **rápido** e **fácil**, ele **não mantém controle histórico** dos dados. Os tipos 2 e 3 são **parecidos**, mas à medida em que tipo de informações se deseja exibir, podem haver grandes **diferenças**. Os tipos de cargas devem ser muito bem definidos para cada tabela (dimensão) no DW.

**Importante:** Você reparou que no SCD tipo 2 a coluna de ID ficou duplicada com dois registros com o valor 1 ? Neste caso, não poderíamos mais deixar a coluna ID como PK. Ou deixamos a tabela sem PK definida, ou criamos as famosas **Surrogate Keys (SKs).** As SKs nada mais são do que colunas criadas para servirem como PK da tabela, visto que a coluna natural não pode mais ser. A grande maioria das SKs são colunas de auto-increment, as quais chamamos de **chaves artificiais.**

# Front-End: BI e Dashboards

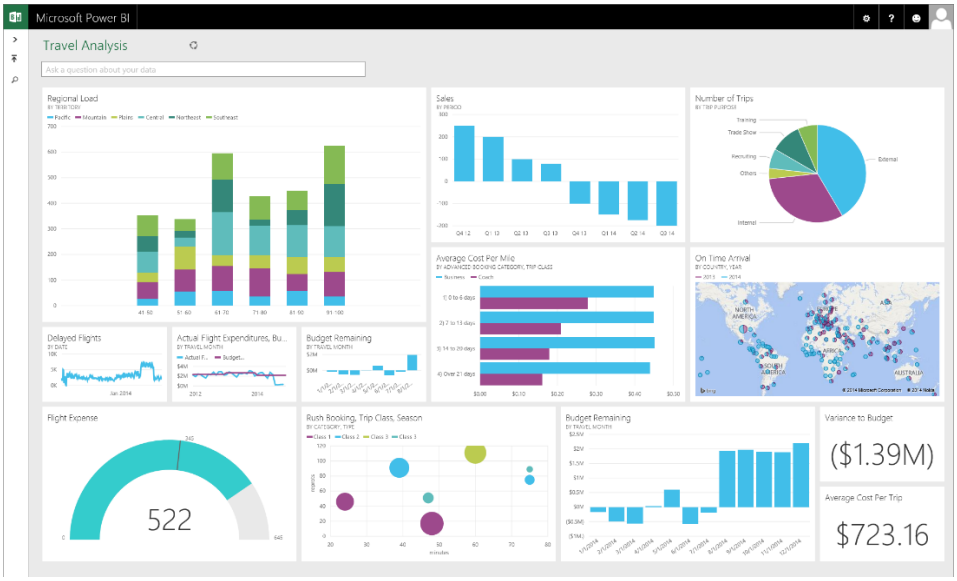
Como já citado bem no começo deste e-book, todo esse processo citado até agora faz parte do **back-end** de um projeto de BI, que é o que fica literalmente “por trás das cortinas”. Todo o desenvolvimento feito no **ETL** e no **DW** servirão como base para a parte **front-end** do BI. O que seria essa parte então?

O front-end do BI é o **mapeamento** de todas as tabelas **fatos** e **dimensões** do DW para dentro das **ferramentas de BI**, respeitando suas hierarquias, assim como a criação da **camada semântica** junto com suas **métricas** e **atributos**. Após isto é iniciada a construção dos relatórios, painéis e dashboards em que são exibidas todas essas informações previamente coletadas.

Na hora que formos mapear as tabelas e colunas no BI, precisamos respeitar essa hierarquia. Toda coluna que serve para **descrever** um dado, é um **atributo**, e todas as colunas que servem para **valorar** um dado, são **métricas**. Geralmente os atributos são do tipo **texto e data**, e as métricas do tipo **número, mas isto não é uma regra**. Uma coluna de salário é uma métrica não só porque é número, mas porque os valores de um salário fazem sentido com operações de **somas**, **multiplicações** e etc. Por outro lado, uma coluna de CEP também é numérico mas é um **atributo**, porque não faz sentido somar um CEP com outro.

Após criar a **camada semântica**, **mapear** as tabelas, **métricas** e **atributos**, e definir as regras de negócios restantes, tudo está pronto para o desenvolvimento dos painéis.

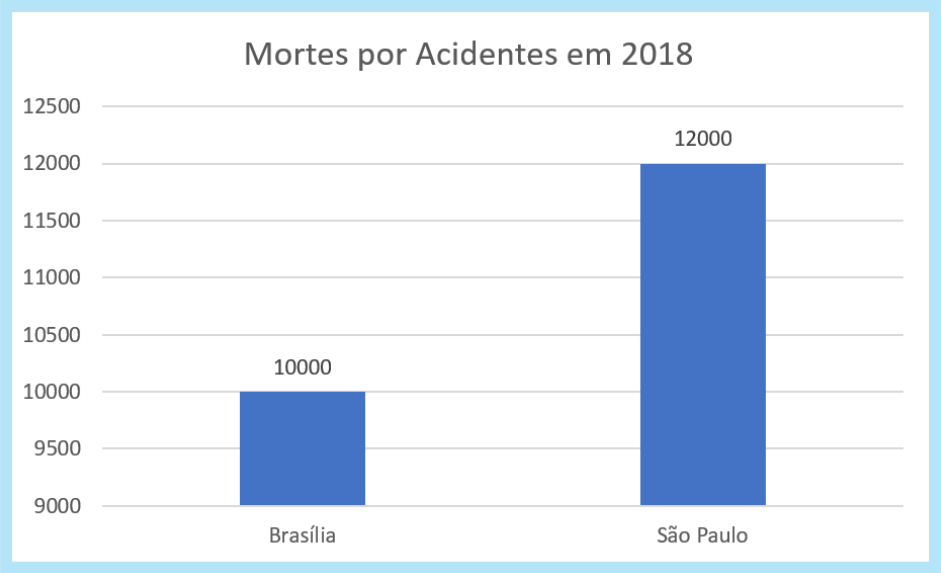

Somente essas telas interessam a alta gestão e a área negocial. Uma tela **interativa**, **objetiva**, **bonita** e com **informações relevantes** para auxiliar em uma decisão a ser tomada.



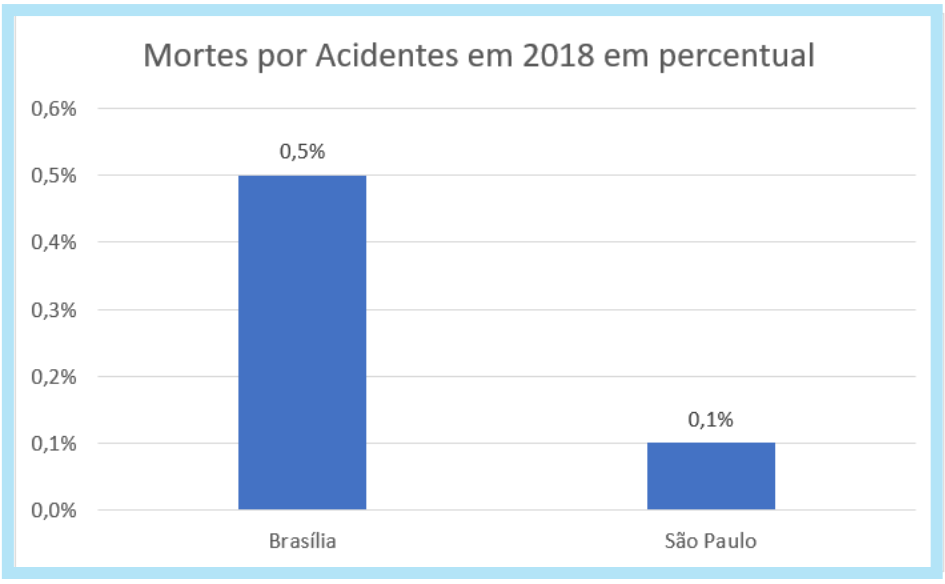

Nesta etapa final de BI Front-End, não é só criar qualquer gráfico arrastando as métricas e os atributos e montar um dashboard. É **necessário estudar** e **analisar** a **melhor forma** de **exibir** as **informações**, respeitando todo o **contexto** em que elas estão **inseridas**.

Aqui temos alguns exemplos **comuns** de erros de gráficos e visualizações:

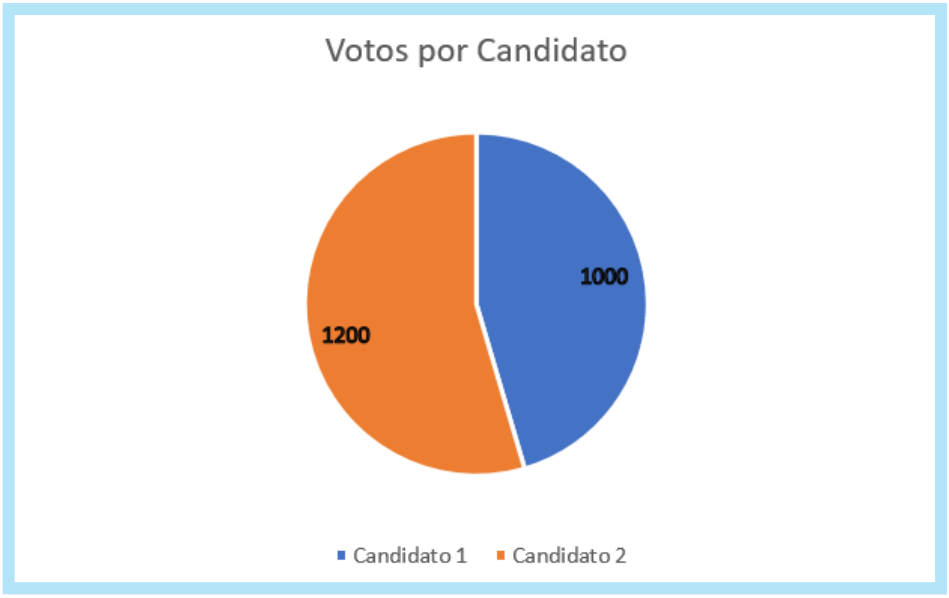
**1 – Cenário geral da informação**. Temos o gráfico abaixo mostrando a quantidade de mortes por acidentes de trânsito em **Brasília** e **São Paulo**, a princípio entende-se que São Paulo está pior que Brasília, pois tem 2 mil acidentes a mais por ano:



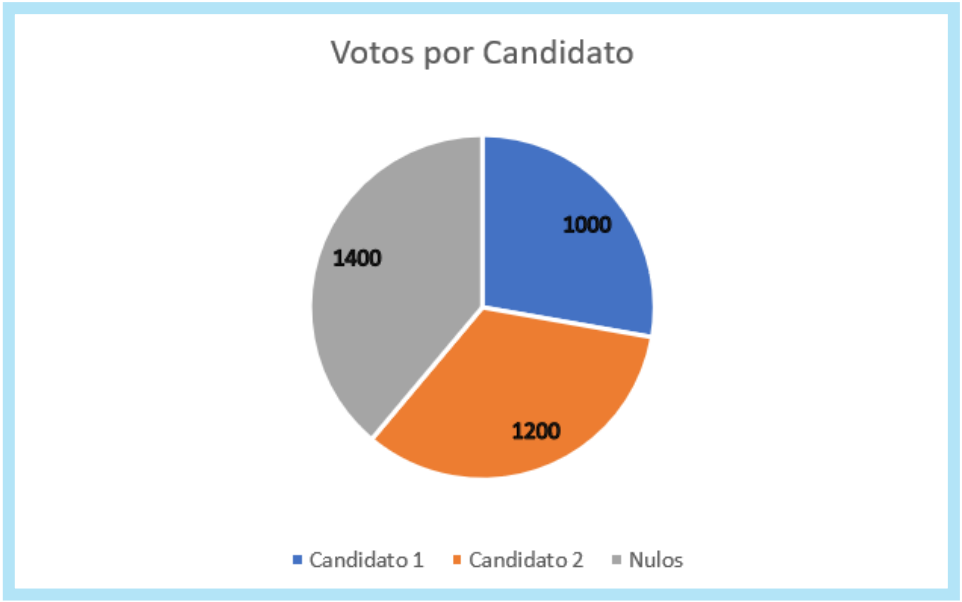

Brasília possui **2 milhões** de habitantes, e São Paulo **12 milhões.** São Paulo é **6 vezes** maior que Brasília e possui somente 2 mil acidentes a mais por ano. Uma forma melhor de se exibir essas informações seriam de forma percentual por exemplo, em que a quantidade de habitantes de cada cidade seria considerada:



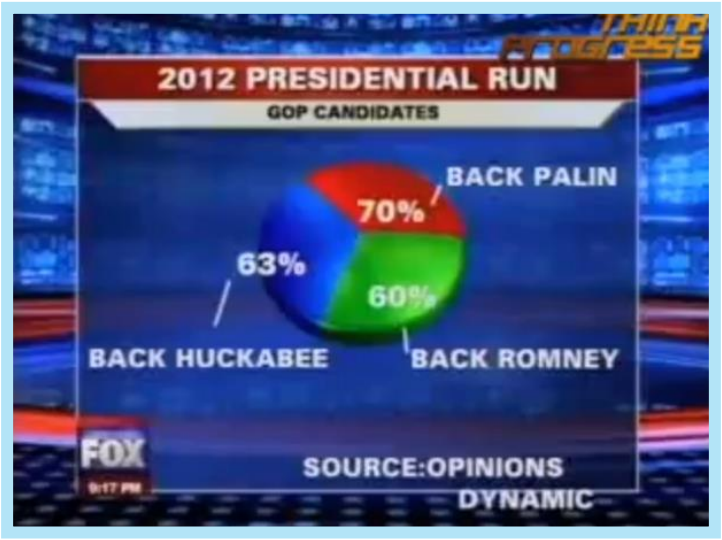
**2 – Dados ignorados**. Repare neste gráfico abaixo representando uma eleição qualquer entre 2 candidatos:




Analisando rapidamente, o Candidato 2 venceu com 200 votos a mais. Mas será que o total de votos foi 2.200 ? Realizando a análise novamente considerando os dados **nulos** da base de dados, percebe-se que tiveram **mais votos** em "**nenhum**" do que pro próprio candidato "vencedor". A análise é outra agora, certo?

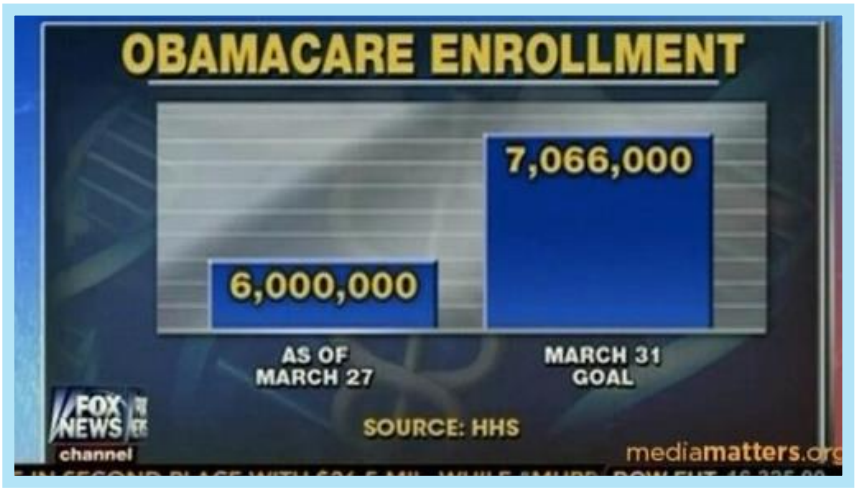


**3 – O somatório total não condiz**. Se temos um gráfico de pizza em percentuais, o total deverá **sempre somar 100%:**

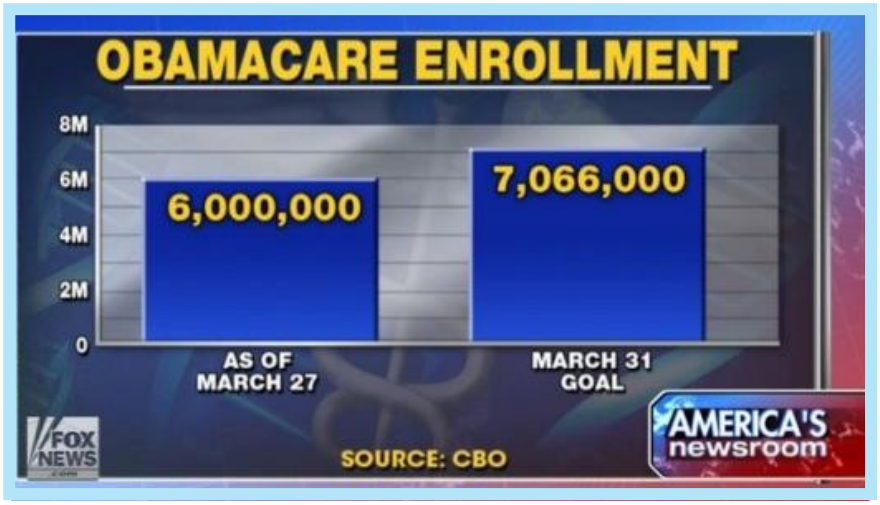



**4 – Informações Incompletas.** Você consegue me dizer qual estado teve o maior Market Share? O ideal seria exibir alguns números ou um gradiente de cor, quanto maior, mais escuro por exemplo:

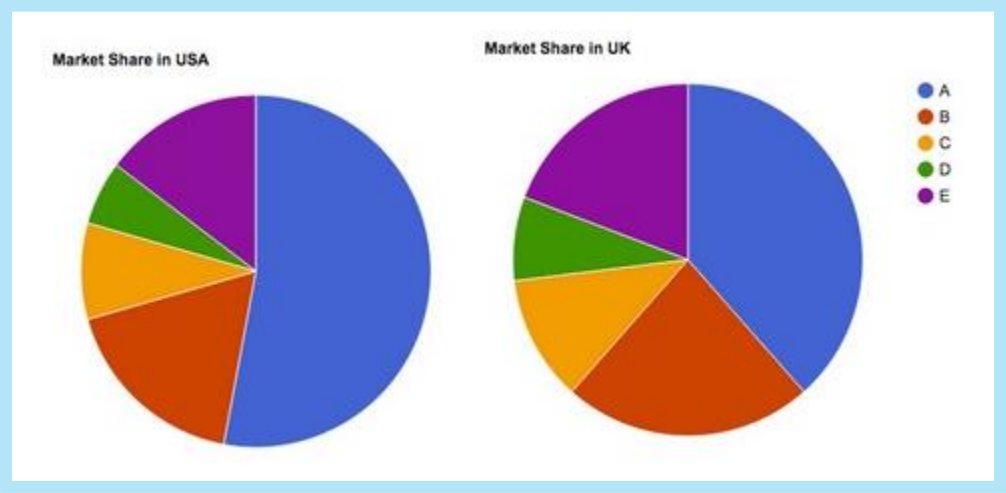
**5 – Proporção**. Repare no gráfico abaixo como que 7 milhões aparenta ser 3 vezes maior que 6 milhões:



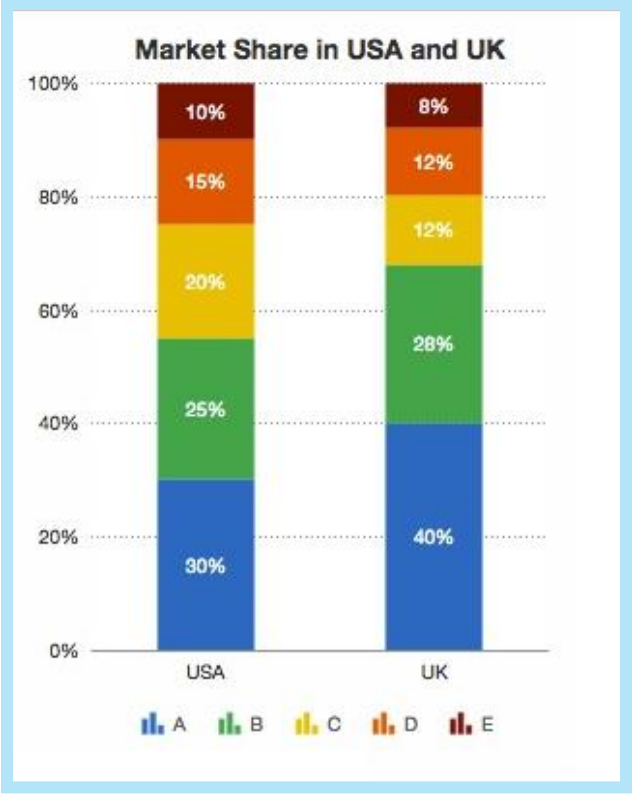

Não seria **melhor** exibir este gráfico assim ?



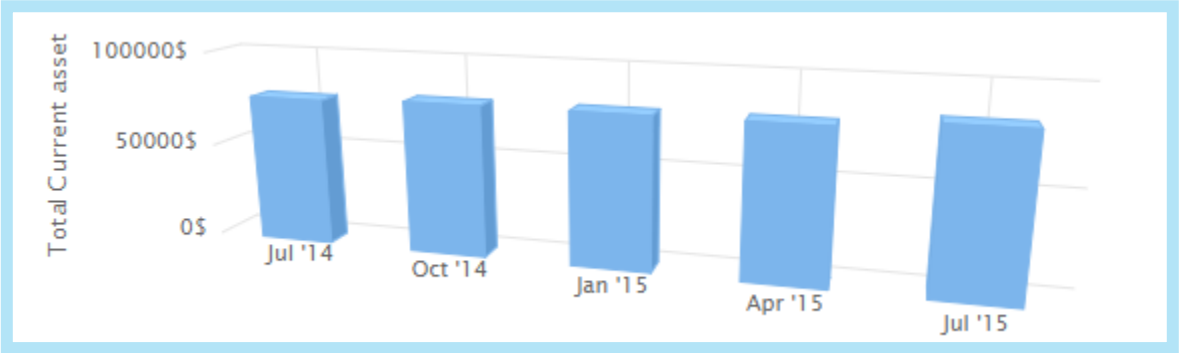
**6 – Difícil comparação**. Você consegue comparar a diferença de Market Share nos Estados Unidos e no Reino Unido, baseado nos gráficos abaixo?



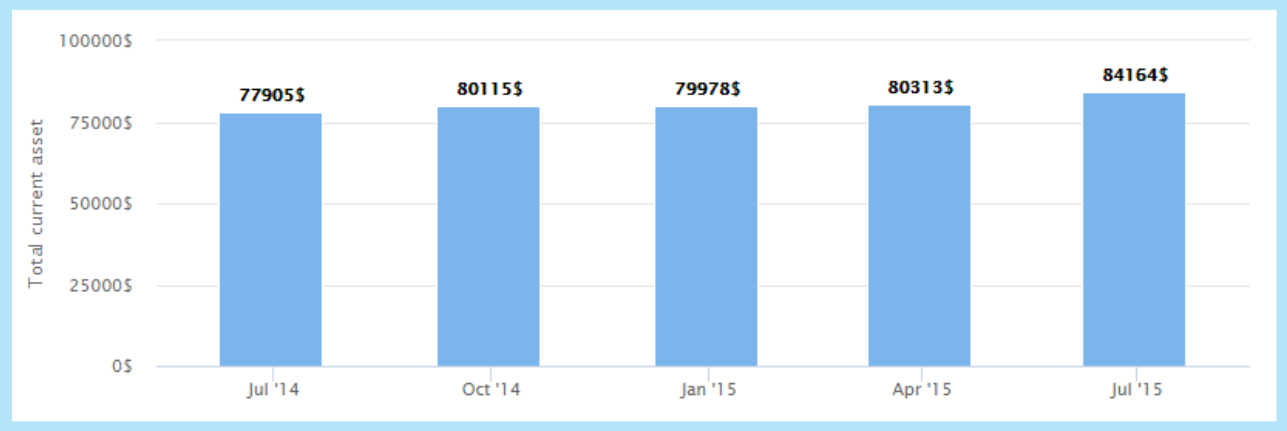

Uma análise muito mais **simples** e **eficiente**:



**7 – Simplicidade**. Na tentativa de criar um gráfico diferente e chamativo, foi optado pelo 3D. O problema é a **dificuldade** em distinguir se Outubro de 2014 teve mais ativos do que Janeiro de 2015:




Mesmo com a dificuldade ainda de enxergar a diferença no 2D, ele é mais **simples**. E neste caso basta colocar um **label** para exibir os valores de cada mês:



# Ferramentas de Mercado

Abaixo temos listadas as ferramentas de mercado mais usadas:

**Ferramentas de Query e Banco de Dados:**

- Oracle SQL Developer
- MySQL WorkBench
- SQL Server Management Studio
- DB Visualizer

**Ferramentas de Modelagem de Dados:**

- PowerDesigner
- ERWin
- Toad Data Modeler

**Ferramentas de ETL:**

- Informatica PowerCenter
- IBM InfoSphere DataStage
- SSIS - SQL Server Integrated Services
- Talend Data Integration
- Pentaho Data Integration
- Matillion ETL
- Alteryx

**Ferramentas de BI Front-End:**

- MicroStrategy
- SAP BO
- PowerBI
- Tableau
- QlikView

# Big Data e Data Science

O mercado de TI cresce rapidamente, de tal forma que em apenas 1 ano muitos conceitos e novidades surgem desestabilizando o mercado, forçando os players a se adequarem ou, então, irão correr os riscos de perder grande parte do mercado, que por sua vez é exigente e se atualiza de forma rápida.

Existem dois conceitos relacionados com o BI que estão crescendo muito nos últimos anos: o **Big Data** e o **Data Science**. Embora o nome, o conceito de Big Data não se diz respeito somente à **interpretação** de **grandes volumes** de dados, mas também sobre a sua vasta **variedade**. Para explicar melhor, atualmente existem dois tipos de dados, os **estruturados** e os **não-estruturados**.

Os **estruturados** são os dados **tabulares** que podemos facilmente colocar em um **excel** por exemplo. Os **não-estruturados** dizem respeito aos dados que **teoricamente** só poderiam ser analisados por **seres humanos**, porque em si não tem nenhuma estrutura definida, como por exemplo **vídeos**, **fotos**, **posts** no facebook ou **likes** no instagram.

O desafio do Big Data é justamente permitir analisar dados de fontes **muito grandes**, complexas ou não, estruturadas ou não, de uma forma **rápida e automática**, e recentemente, integrando-se com as ferramentas de apresentações (BI), gerando então o novo conceito de **Big Data Analytics**. Dentro do Big Data, existem outras fortes tecnologias que estão sendo inseridas também no mercado de BI, sendo elas o conceito de **Análise Preditiva**, e as três grandes linguagens de programação: **R** e **Python** e **SCALA**.

Já o **Data Science**, é basicamente um BI só que focado em análise preditivas. **Digamos que o BI tradicional olha para o passado e o Data Science olha para o futuro.** A ideia central do Data Science é poder analisar e estudar os dados da mesma forma que o BI padrão, só que identificando possíveis **comportamentos padrões** para tentar **prever** um futuro próximo.

O pilar principal do Data Science é a **Análise Preditiva**, cuja capacidade é de poder prever um acontecimento futuro incerto, baseado em informações presentes, sendo elas corretamente **modeladas e mineiradas.** Com o uso de **técnicas analíticas** a fim de juntar as **estatísticas** e o modelo de négocios, **é possível auxiliar na tomada de decisão futura**. As tecnologias mais importantes inseridas na análise preditiva são:

• **Data Mining**: Mineiração de dados é o processo de **descoberta de informações** presentes em grandes conjuntos de dados. Normalmente, esses padrões não podem ser descobertos com a exploração de dados tradicional pelo fato de as relações serem muito complexas ou por haver muitos dados

• **Machine Learning**: Nome que se dá à capacidade da máquina / computador **literalmente aprender conforme uma série de fatos e acontecimentos**, de uma forma em que com o tempo a máquina já saberá o evento futuro antes mesmo dele ocorrer, baseado em suas **variáveis** e no aprendizado do **comportamento padrão** das mesmas;

• **Inteligência Artificial**: Juntamente com o Machine Learning, a **AI** traz o conceito da inteligência das máquinas em si. Está muito ligada à **robótica**, como por exemplo robôs com comportamentos similares a humanos, demonstrando opiniões e comportamentos conforme uma variável externa.

# Conclusão

Todos estes conceitos listados aqui neste e-book fazem parte do que eu considero essencial para quem deseja ingressar na área de BI. É importante ressaltar que ainda existe muita coisa dentro do mundo do BI, o que temos aqui é somente o inicial!

Você gostou, achou útil, muito bom, ou esperava mais? Conte-me sua opinião sobre este e-book, pois eu planejo lançar futuramente outras versões mais avançadas.

**Importante**: não há melhor forma de aprender tudo o que falei senão colocando na prática. Caso tenha alguma dúvida, não hesite em entrar em contato comigo. Estou à disposição para ajudá-lo no que precisar.

Obrigado pela sua leitura, e siga-me nas redes sociais!

**Instagram - @entendendobi**

www.instagram.com/entendendobi/

**Facebook - @entendendobi**

www.facebook.com/entendendobi/

**Site - EntendendoBI**

www.entendendobi.com