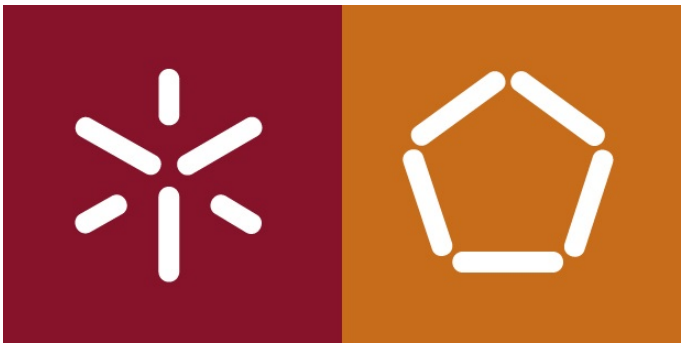


UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

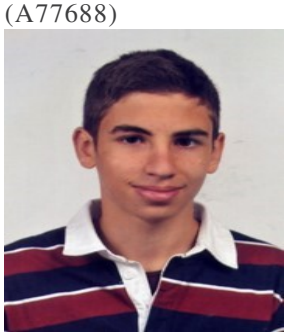
ANÁLISE DE DADOS

Data Warehouse NorthWind

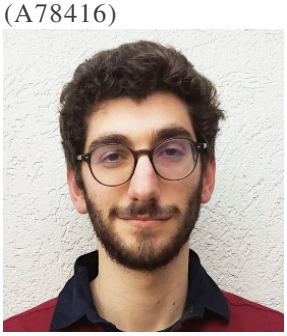
20 de Janeiro de 2019

GRUPO 2:

Francisco Matos
(A77688)



Francisco Oliveira
(A78416)



Gil Cunha
(A77249)

Luís Costa
(A74819)

Conteúdo

1 Introdução

Neste relatório é apresentado de forma clara e detalhada o trabalho efetuado, isto é, a implementação, planeamento e análise do nosso Sistema de Bases de Dados Multi-Dimensionais (SBDMD), tendo por base o modelo da base de dados ***NorthWind*** e os respetivos *scripts*, disponíveis para *MySQL*.

São expostas, neste documento, a projeção da arquitetura e a implementação de um *data warehouse* para suporte de decisões relativas à empresa *NorthWind* e ao seu comércio, com base na informação contida na sua base de dados operacional, assim como o seu povoamento e refrescamento.

Por fim, foram criados indicadores de *Business Intelligence em dashboards*, a partir da base de dados implementada (DW), através do *Microsoft Power BI Desktop* e procedemos à análise desses dados.

2 Sistema de Base de Dados

A primeira fase deste projeto consiste na criação de uma base de dados multi-dimensional - um *data warehouse* - desenvolvida com o objetivo de armazenar certa informação da base de dados *NorthWind* já existente, para posterior análise dos seus dados e recolha de informação.

2.1 Base de Dados Produção

A base de dados *NorthWind* suporta a operacionalidade de uma empresa fictícia que se dedica ao comércio internacional de produtos alimentares. Ao analisarmos o esquema desta base de dados transacional, identificamos várias vistas de dados correspondendo a vários cenários aplicacionais dentro do sistema de negócio da empresa referida.

Sendo assim, o primeiro passo efetuado passou pela análise e exploração do conteúdo da sua base de dados operacional **(produção)**, com o intuito de identificar quais atributos e respetivas tabelas (**fontes de dados**) que poderiam ser úteis armazenar no nosso *data warehouse* a desenvolver e, consequentemente, proporcionar informações úteis de modo a promover a melhoria de comércio por parte da empresa.

Após uma análise da base de dados relacional da *NorthWind*, orientada ao tema de **Vendas de Produtos** da empresa, foram selecionadas as seguintes tabelas, como **fonte de dados**, em que *apenas alguns atributos* foram escolhidos para serem utilizados como informação:

- ***Customers:*** clientes da empresa que solicitaram alguma encomenda;
- ***Suppliers:*** fornecedores da empresa que providenciam os produtos;
- ***Shippers:*** carregadores que transportam as encomendas da empresa;
- ***Products:*** produtos que compõem as encomendas solicitadas à empresa;
- ***Employees:*** funcionários da empresa;
- ***Orders:*** encomendas que foram solicitadas à empresa;
- ***Order_details:*** detalhes/características das encomendas realizadas;

Os atributos destas tabelas irão, mais tarde, preencher as tabelas de dimensão no esquema da nossa base de dados multi-dimensional (*data warehouse*).

2.2 Esquema Lógico

De seguida, com o estudo da base de dados *NorthWind* e análise da informação que esta contém, o grupo decidiu desenvolver um modelo dimensional, segundo um **esquema em estrela**, sempre com a perspetiva das **vendas de produtos** como se pode observar na imagem seguinte:

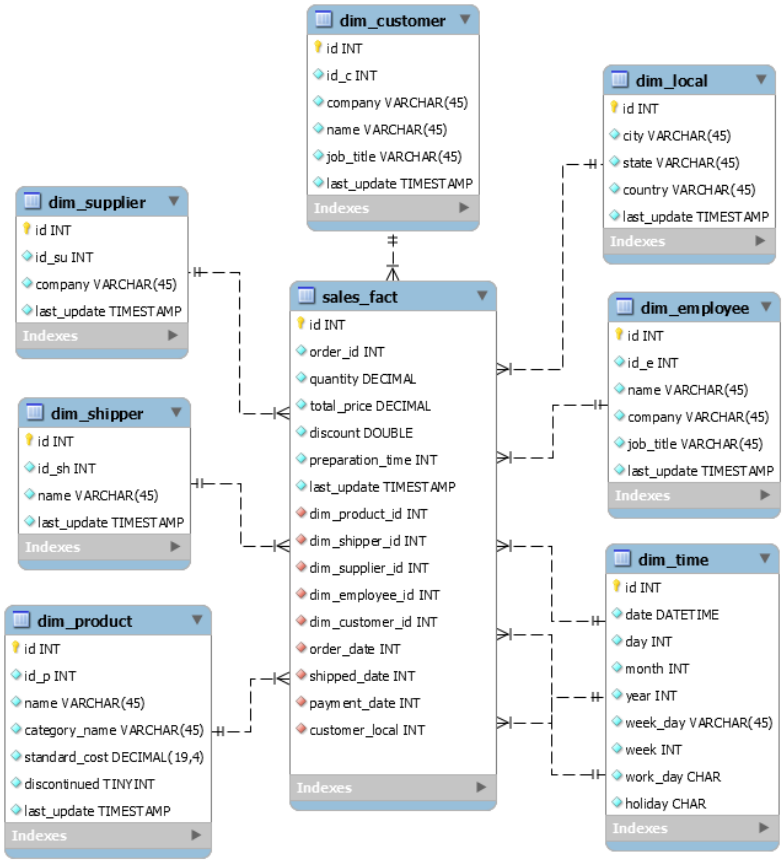


Figura 1: Modelo Lógico: Esquema multi-dimensional em Estrela

É possível observar que o modelo foi desenvolvido segundo a perspetiva de *Vendas*, como referido anteriormente. Significa que a informação-alvo que se pretende analisar, e guardar o *data warehouse* final, é referente às vendas da empresa e a todos os dados que estão relacionados a estas: clientes, funcionários, fornecedores, preços, quantidades, etc. O modelo desenvolvido consiste nas seguintes tabelas:

- ***dim_product:*** dimensão que representa os produtos vendidos na empresa, constituindo assim as diversas vendas. Contém o nome, categoria, preço e estado de cada produto;
- ***dim_shipper:*** dimensão que representa os remetentes dos produtos vendidos na empresa. Contém o nome deste;
- ***dim_supplier:*** dimensão que representa os fornecedores dos produtos vendidos na empresa. Contém o nome (empresa) do fornecedor;
- ***dim_customer:*** dimensão que representa os clientes da empresa. Contém o nome, a profissão e a companhia do cliente;
- ***dim_local:*** dimensão que representa os endereços locais dos clientes. Contém a cidade, estado e país da região a que se refere.
- ***dim_employee:*** dimensão que representa os funcionário da empresa. Contém o nome, companhia e função deste;
- ***dim_time:*** dimensão que representa um calendário. Contém a data, dia, mês, ano, dia da semana, semana e se corresponde a um dia útil/feriado. De notar que esta dimensão foi produzida por uma ferramenta externa - *folha de cálculo Excel* - e não retirada a partir da base de dados de produção.

- ***sales_fact*:** Tabela de factos das vendas, visto serem a componente a analisar. Contém a quantidade, preço total, desconto, tempo de preparação (diferença de dias entre order date e shipped date) e os identificadores do produto, remetente, fornecedor, funcionário , data de pedido, data de envio, data de pagamento e endereço do cliente correspondentes a cada venda.

Todas estas tabelas obtêm o id da sua **chave primária** com a opção *auto-increment*, ativada em cada uma delas, no *MySQL*.

As tabelas ***dim_product, dim_shipper, dim_supplier, dim_customer e dim_employee*** possuem ainda o id da entidade que representam, correspondente da base de dados de produção *NorthWind*. O histórico dos dados do *data warehouse* desenvolvido baseia-se, então, na relação entre os novos e os velhos registos, estabelecendo uma ligação entre eles pelo id de produção que têm em comum.

O grupo concluiu que um esquema em estrela seria suficiente para desenvolver um *data warehouse* eficaz, para além de que outros esquemas como *floco de neve* ou *constelação de estrelas* seriam difíceis de desenvolver a tempo, tendo em conta a sua complexidade e o prazo de entrega do projeto.

Para além disso, todas as tabelas (dimensões) têm um *timestamp* referente à última alteração sofrida em cada um dos registos, no momento da atualização.

2.3 Mapa Lógico de Dados (*Source to Target Map)*

De forma a compreender melhor como os dados são tratados desde a base de dados operacional *NorthWind* para o *data warehouse*, foi realizado um *Source-to-Target Map* que se encontra na secção **7 - Anexos**

Este mapa mostra a relação dos campos entre as tabelas da base de dados relacional (*source*) e as dimensões do esquema multi-dimensional do *data warehouse* (*target*). Neste estão descritos as informações das tabelas origem e destino e respetivos atributos, como: nome, tipo, tamanho e transformação efetuada na transição de dados entre as tabelas, isto é, a forma como foi feita a manipulação dos dados da fonte de modo a corresponder ao formato destino que é esperado.

2.4 Arquitetura do sistema

De forma a desenvolver um *data warehouse*, recorremos a uma arquitetura típica destes projetos. Partindo da base de dados produção fornecida, que se encontra inalterada, passamos para uma área intermédia, **Área de Retenção**, onde serão tratados os dados, e por fim o *data warehouse* em si, com os dados devidamente formatados e carregados.

O primeiro elemento, base de dados produção, já foi descrito no inicio deste documento, pelo que nesta secção serão descritos os restantes: área de retenção e *data warehouse*.

O esquema geral da implementação de um *data warehouse* é o seguinte:

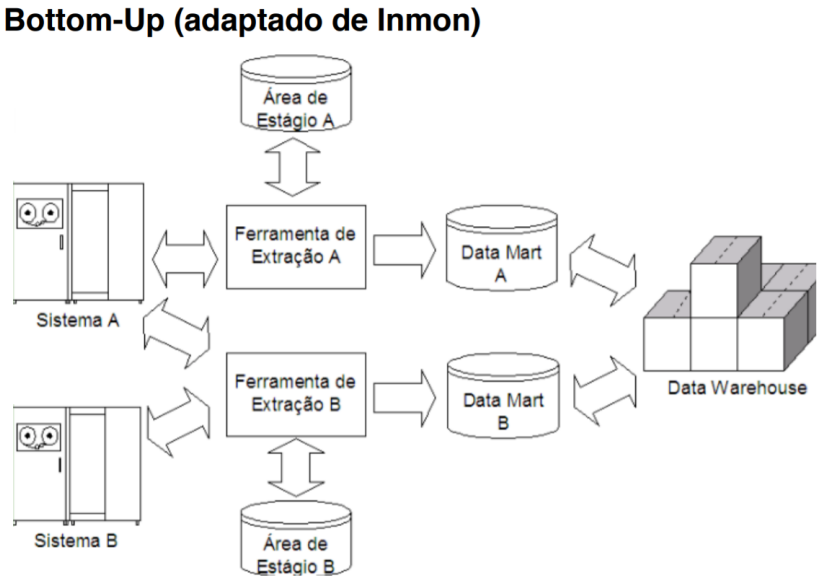


Figura 2: Esquema geral da implementação de um *data warehouse*, segundo uma perspetiva *Bottom-Up*

De uma forma resumida, em *Bottom-Up*, começa-se pela seleção da área de suporte à decisão a implementar, seguida da definição do detalhe dos factos (o grão) do processo selecionado. Posteriormente, faz-se a seleção das dimensões de análise sobre as quais se pretende analisar os factos e , por fim, a definição das medidas a integrar na estrutura de cada facto.

Executa-se o processo ETL, com áreas de estágio para processamento intermédio de dados, e por fim forma-se um *data warehouse*.

2.4.1 Área de retenção & *Data Warehouse*

Para garantir que os dados são carregados para o *data warehouse* corretamente, foi criada um terceiro *schema* auxiliar, para onde se irá extrair e transformar os dados antes do carregamento. Esta é a **Área de Retenção**, que agrega, sumariza e armazena a informação extraída das bases de dados **fontes** diferentes (neste caso apenas uma fonte, *NorthWind*), e onde esses dados heterogéneos são processados de maneira a assegurar a sua consolidação e consistência do *data warehouse*, e reconciliar a estrutura certificando que os dados possuem o mesmo formato. Assim, toda a informação encontra-se devidamente organizada para o futuro carregamento.

Por sua vez, o *data warehouse* desenvolvido é uma estrutura de dados orientada, integrada e variável no tempo, que tem por objetivo dar suporte aos processos de tomada de decisão. É um base de dados que contem dados extraídos do ambiente de produção da organização *NorthWind*, que foram selecionados, depurados e otimizados para processamento de consultas e não para processamento de transações.

O *data warehouse* recebe os dados depois de estes serem tratados na área de retenção, permitindo então realizar a análise dos dados para produzir depois estatísticas e indicadores. Os modelos lógicos da *área de retenção* e o *data warehouse* são semelhantes:

Modelo Lógico

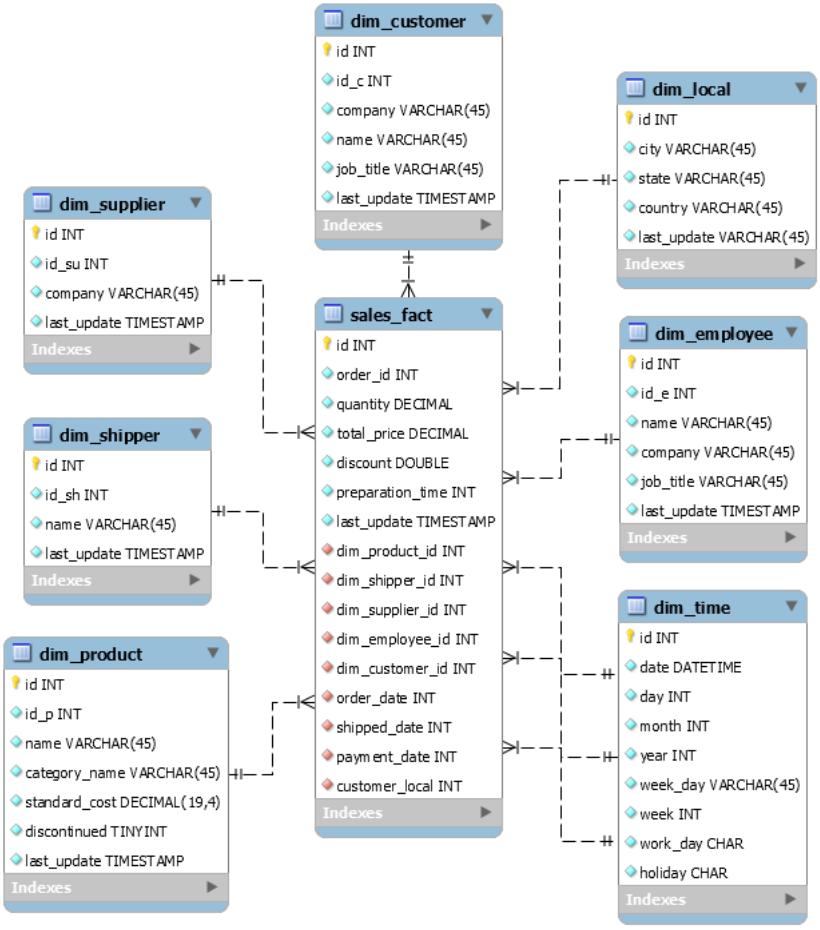


Figura 3: Modelo Lógico da Área de Retenção / *Data Warehouse*

Script de Criação

Posteriormente, gerou-se o *script* de criação, com a opção *Forward Engineer* do *MySQL*, para criar as diferentes tabelas dos modelos da área de retenção e *data warehouse*. De seguida apresenta-se, a título de exemplo, o código de criação da tabela dimensão de ***customers***, no *data warehouse*. De realçar, novamente, a existência de uma etiqueta *timestamp*, para permitir a capacidade de histórico no *data warehouse* e nas chaves ids **substituta e natural**. A chave primária (***surrogate key***) tem *auto-increment*, para ser incrementada automaticamente, e a chave de produção (***natural key***) permite estabelecer uma relação entre as várias instância de uma mesma entidade, durante o processo de histórico.

```
-- CREATE CUSTOMER

CREATE TABLE IF NOT EXISTS `datawarehouse`.`dim_customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `id_c` INT NOT NULL,
  `company` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `job_title` VARCHAR(45) NOT NULL,
  `last_update` TIMESTAMP NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

2.5 ETL

O processo **ETL (*Extract, Transform, Load*)** é um conjunto de processos que inclui a **extração** de dados de fontes de informação internas e externas, podendo estar em diferentes formatos, a **transformação** dos dados de acordo com as necessidades da organização e, finalmente, o **carregamento** dos mesmos numa estrutura de dados, como por exemplo um *data mart* ou um *data warehouse*.

O seguinte diagrama representa este processo no caso deste trabalho prático:

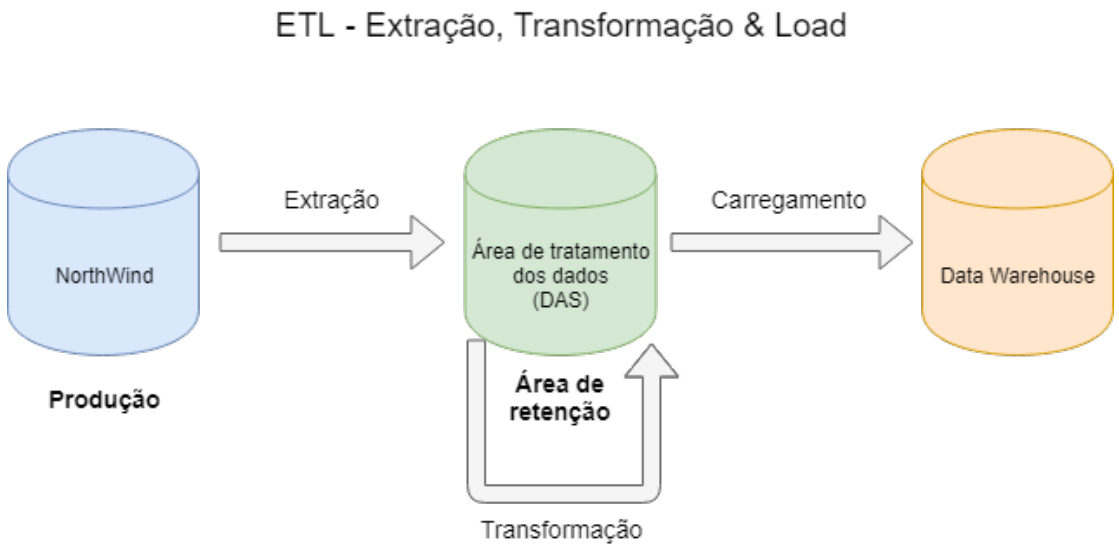


Figura 4: Processo geral de ETL

2.5.1 Extração & Transformação

A primeira etapa do *ETL*, é a extração, e este baseia-se no carregamento dos dados de todas as fontes de informação disponíveis, neste caso apenas uma, para a área de retenção. Este processo foi feito com *queries* de inserção a partir dos dados contidos na base de dados, *NorthWind*. No entanto, a dimensão do tempo (*dim_time*) teve de ser gerada manualmente, visto que esta informação, apesar de extremamente importante para qualquer sistema de *Data Warehousing*, não se encontra presente na base de dados *NorthWind* e como é estática apenas precisa de ser executada uma vez.

Foi necessário ter em atenção os imensos valores a *NULL* na área de produção, e de acordo com os nossos conhecimentos de *data warehouse* - não devem existir dados nulos num *DW* -, foi possível perceber que esses dados deviam ser devidamente tratados. Usando, como exemplo, a entidade/tabela *supplier*, podemos observar que caso o nome da empresa seja *NULL*, é colocado no *data warehouse* o valor **"Desconhecido"**, evitando assim a utilização de valores nulos, respeitando assim o conceito de *data warehouse*:

```
-- DIM_SUPPLIER

INSERT INTO dim_supplier (id_su, company, last_update) SELECT id, COALESCE(company,'
    Desconhecido'), now() FROM northwind.suppliers;
```

Inseriu-se também uma nova entrada, para representar uma entidade desconhecida, neste caso um *supplier*. Por exemplo, quando na base de dados um registo de um certo produto não tem a informação do fornecedor correspondente a esse mesmo produto, estando com valor *NULL*, ao realizar o processo ETL

para o *data warehouse*, esse produto será associado ao registo do fornecedor **desconhecido**. A chave primária deste será **-1**, para não interferir com os restantes registos do *data warehouse*, onde o identificador é incrementado automáticamente, por ordem crescente.

```
-- Fornecedor Desconhecido

INSERT INTO dim_supplier (id, id_su, company, last_update) VALUES (-1, -1, 'Desconhecido
    ', '1975-01-01 00:00:00');
```

id	company	last_name	first_name	email_address	job_title	business_phone
3	Supplier C	Kellev	Madeleine	NULL	Sales Representative	NULL
4	Supplier D	Sato	Naoki	NULL	Marketing Manager	NULL
5	Supplier E	Hernandez-Echevarria	Amava	NULL	Sales Manager	NULL
6	Supplier F	Havakawa	Satomi	NULL	Marketing Assistant	NULL
7	Supplier G	Glasson	Stuart	NULL	Marketing Manager	NULL
8	Supplier H	Dunton	Brvn Paul	NULL	Sales Representative	NULL
9	Supplier I	Sandberg	Mikael	NULL	Sales Manager	NULL
10	Supplier J	Sousa	Luis	NULL	Sales Manager	NULL
600	NULL	NULL	NULL	NULL	Sales Manager	NULL

Figura 5: **Base de Dados *NorthWind* -** fornecedor com id 600 com valores NULL

id	id_su	company	last_update
1	600	Desconhecido	2019-01-18 11:14:08
2	1	Supplier A	2019-01-18 11:14:08
3	2	Supplier B	2019-01-18 11:14:08
4	3	Supplier C	2019-01-18 11:14:08
5	4	Supplier D	2019-01-18 11:14:08
6	5	Supplier E	2019-01-18 11:14:08
7	6	Supplier F	2019-01-18 11:14:08

Figura 6: ***Data Warehouse* -** Valor NULL transformado em ”Desconhecido”

Este raciocínio foi utilizado para resolver todos os possíveis valores *NULL* das diferentes tabelas inseridos na base de dados de produção.

2.5.2 Carregamento

Depois de todo o tratamento dos dados realizado, surge a fase final do processo ETL. Visto que todos os dados na área de retenção já se encontram prontos para serem carregados, é apenas necessário realizar inserções simples, desta vez para o *Data Warehouse*

```
-- Carregamento de AR para DW da dimensao supplier

INSERT INTO dim_supplier (id, id_su, company, last_update) SELECT id, id_su, company,
    now() FROM area_ar.dim_supplier;
```

3 Refrescamento

Para mantermos os dados do *data warehouse* atualizados, necessitamos de executar processos de refrescamento. Podemos classificar o refrescamento aplicado neste projeto com um **refrescamento diferencial**, isto é, apenas realizamos a cópia dos dados que foram adicionados ou modificados ,a partir da **última atualização completa** da base de dados *NorthWind*. O principal benefício é a existência de menos arquivos para restaurar, já que não se aplicam vários arquivos na restauração, apenas o *full* e o último diferencial. Porém o tempo de recuperação pode ser um pouco extenso.

No carregamento dos dados da área de retenção para o *data warehouse*, no processo de **refrescamento**, precisamos de garantir que apenas são carregados os dados que foram inseridos ou sofreram atualização na base de dados *NorthWind*, após o último refrescamento feito ao *data warehouse*.

Visto isso, o primeiro passo do refrescamento consiste em carregar a área de retenção com todos os dados da base de dados *NorthWind*, de modo a área possuir a informação mais recente. Posteriormente, são executadas várias *queries*, cada uma referente a uma dimensão/tabela do *data warehouse*, que irão comparar as dimensões de uma mesma entidade entre a área de retenção e o *data warehouse*, e inserir os dados no *data warehouse* que se encontram somente na área de retenção (e não no *data warehouse*). Numa tentativa de compreender melhor este processo, foi desenvolvido o seguinte diagrama:

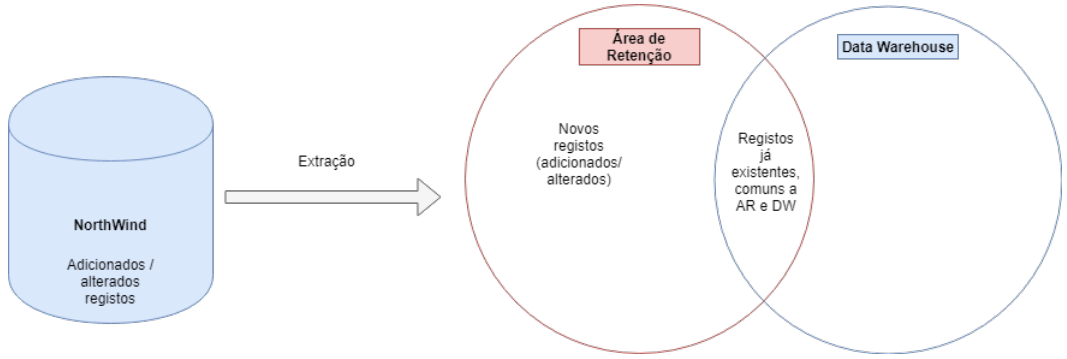


Figura 7: **Processo de refrescamento:** adicionar novos registos ao *data warehouse*

INSERTS

Para saber quais os dados a inserir no *data warehouse*, na fase de refrescamento, é realizado um `LEFT JOIN` em cada uma das *queries*, seguido de uma condição para garantir apenas a seleção dos casos diferentes, ao excluir os dados em comum entre a área de retenção e o *data warehouse*. Assim, pela figura seguinte, pode-se perceber que apenas são inseridos no *data warehouse* os dados novos que se encontram na área de retenção e este não possui (zona verde):

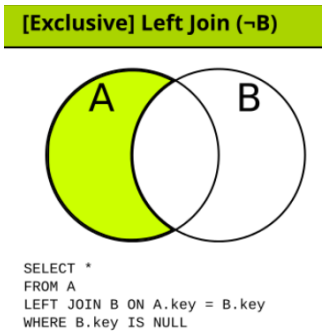


Figura 8: A - Área de retenção; B - *Data Warehouse*

A título de exemplo, apresenta-se de seguida a *query* de refrescamento da dimensão *supplier*, que insere no *data warehouse* apenas os registos novos/alterados dos fornecedores da base de dados *NorthWind*:

```
INSERT INTO datawarehouse.dim_supplier (id_su, company, last_update)
        SELECT area_ar.dim_supplier.id_su, area_ar.dim_supplier.company, now()
    FROM area_ar.dim_supplier
                LEFT JOIN datawarehouse.dim_supplier ON
                area_ar.dim_supplier.id_su = datawarehouse.dim_supplier.id_su and
                area_ar.dim_supplier.company = datawarehouse.dim_supplier.company
                        WHERE datawarehouse.dim_supplier.id IS NULL ;
```

Quando é atualizada uma instância na base de dados *Northwind* relacionado com uma das dimensões existentes no *data warehouse*, é realizado um `INSERT` na respetiva dimensão. Isto resulta numa nova entrada nessa tabela no *data warehouse*, com uma nova ***Surrogate Key*** constituída pelo identificador da dimensão (obtida pelo *auto-increment*), mas com a mesma ***Natural Key*** da antiga entrada (desatualizada) a que corresponde - o identificador da produção.

TRIGGERS

A cada uma das *queries* de refrescamento foi associado um `TRIGGER`, para atualizar as chaves estrangeiras (*foreign keys*) das vendas da tabela de factos (*sales_fact*), de acordo com os identificadores das novas entradas adicionadas. Quando uma *query* é executada para inserir no *data warehouse* os novos registos de uma certa dimensão, o *TRIGGER* respetivo é acionado e, após a inserção dos dados (`AFTER INSERT`), este atualiza as *foreign keys* na tabela *sales_fact*.

Por exemplo, se o registo de um *supplier* na base de dados *NorthWind* é atualizado, será inserido um novo registo deste no *data warehouse*, possuindo uma nova ***Natural Key***, diferente ao registo desatualizado correspondente, mas ambos ficam a possuir a mesma ***Surrogate Key***. É através desta ligação que o *TRIGGER* irá procurar atualizar todas as *foreign keys* das vendas que se encontram na tabela de factos e que estão associadas ao registo desatualizado, e alteram o valor dessas chaves para a ***Natural Key*** do registo do *supplier* atualizado.

```
DELIMITER //

CREATE TRIGGER supplier_after_insert
AFTER INSERT
   ON datawarehouse.dim_supplier FOR EACH ROW

BEGIN

   UPDATE datawarehouse.sales_fact
        INNER JOIN datawarehouse.dim_supplier ON
    datawarehouse.sales_fact.dim_supplier_id = datawarehouse.dim_supplier.id and
    datawarehouse.dim_supplier.id_su = new.id_su and datawarehouse.dim_supplier.id_su =
    new.id_su

        SET datawarehouse.sales_fact.dim_supplier_id = new.id;

END; //

DELIMITER ;
```

Após a realização de um refrescamento é necessário limpar as tabelas existentes na área de retenção à exceção da tabela *dim_time*.

Existe também a possibilidade de ser necessário realizar um `INSERT` na tabela de factos. Este processo de inserção difere dos restantes, pois é necessário garantir que as *foreign keys* das novas vendas inseridas na tabela encontram-se atualizadas.

Para isso, é criado um `TRIGGER` específico, associado à *query* `INSERT` na *sales_fact*, que procura a **Surrogate Key** das entradas mais recente de cada uma das entidades das dimensões relativas a uma determinada venda a ser inserida no momento. Assim, antes de inserir essa venda (`BEFORE INSERT`), os valores das suas *foreign keys* são alteradas para os mais atualizados. A procura da *Surrogate Key* é feita através do *id* da base de dados de produção e o correspondente *id* da dimensão, ou seja as **Naturak Keys**, e assim realizar o *match*. O *trigger* associado a *sales_facts* apresenta-se de seguida:

```
DELIMITER //

CREATE TRIGGER sales_before_insert
BEFORE INSERT
   ON datawarehouse.sales_fact FOR EACH ROW

BEGIN

    SET new.dim_customer_id = (select datawarehouse.dim_customer.id from datawarehouse.
    dim_customer inner join area_ar.dim_customer on datawarehouse.dim_customer.id_c =
    area_ar.dim_customer.id_c and area_ar.dim_customer.id = new.dim_customer_id ORDER BY
     id DESC LIMIT 1);

        SET new.dim_product_id = (select datawarehouse.dim_product.id from datawarehouse
    .dim_product inner join area_ar.dim_product on datawarehouse.dim_product.id_p =
    area_ar.dim_product.id_p and area_ar.dim_product.id = new.dim_product_id ORDER BY id
     DESC LIMIT 1);

        SET new.dim_shipper_id = (select datawarehouse.dim_shipper.id from datawarehouse
    .dim_shipper inner join area_ar.dim_shipper on datawarehouse.dim_shipper.id_sh =
    area_ar.dim_shipper.id_sh and area_ar.dim_shipper.id = new.dim_shipper_id ORDER BY
    id DESC LIMIT 1);

        SET new.dim_supplier_id = (select datawarehouse.dim_supplier.id from
    datawarehouse.dim_supplier inner join area_ar.dim_supplier on datawarehouse.
    dim_supplier.id_su = area_ar.dim_supplier.id_su and area_ar.dim_supplier.id = new.
    dim_supplier_id ORDER BY id DESC LIMIT 1);

        SET new.dim_employee_id = (select datawarehouse.dim_employee.id from
    datawarehouse.dim_employee inner join area_ar.dim_employee on datawarehouse.
    dim_employee.id_e = area_ar.dim_employee.id_e and area_ar.dim_employee.id = new.
    dim_employee_id ORDER BY id DESC LIMIT 1);

END; //

DELIMITER ;
```

De notar que antes de cada inserção do novo registo ou alteração, no *data warehouse*, o atributo *last_update (timestamp)*, é atualizado para o instante atual da inserção.

3.1 Testes de avaliação

Um exemplo de uma situação de refrescamento poderá ser a de um `INSERT` de uma instância na tabela de factos *sales_fact*, em que refere valores de *foreign keys* desatualizados.

id	order_id	quantity	total_price	discount	preparation_time	last_update	dim_product_id	dim_shipper_id	dim_supplier_id	dim_employee_id	dim_customer_id
10	31	10	300	0	2	2019-01-18 12:24:26	6	1	2	3	4
11	31	10	530	0	2	2019-01-18 12:24:26	18	1	2	3	4
12	31	10	35	0	2	2019-01-18 12:24:26	27	1	2	3	4

Figura 9: Valores iniciais relativos às vendas de produtos 10, 11 e 12, da encomenda nº 31

Para criarmos este situação de teste, é necessário atualizar os dados da base de dados de produção para provocar uma mudança das *surrogate keys*, ao adicionar novos registos atualizados respetivos às mesmas entidades associadas às vendas, no *data warehouse*.

```
update northwind.products set product_name = "new_product_name" where id = 6;
update northwind.shippers set company = "new_company" where id = 1;
update northwind.suppliers set company = "new_company" where id = 2;
update northwind.employees set company = "new_company" where id =3;
update northwind.customers set company = "new_company" where id = 4;
```

Figura 10: Atualizações nos campos das entidades *product, shipper, supplier, employee e customer* associados às vendas anteriores

Após a atualização dos dados da base de produção é necessário então realizar os processos de refrescamento necessários. Começamos por fazer a extração dos dados para a área de retenção e, posteriormente, a inserção no *data warehouse*.

id	order_id	quantity	total_price	discount	preparation_time	last_update	dim_product_id	dim_shipper_id	dim_supplier_id	dim_employee_id	dim_customer_id
10	31	10	300	0	2	2019-01-18 12:40:44	6	4	11	10	30
11	31	10	530	0	2	2019-01-18 12:40:44	18	4	11	10	30
12	31	10	35	0	2	2019-01-18 12:40:44	27	4	11	10	30

Figura 11: Valores intermédios, na área de retenção

De notar que as *foreigns keys* das vendas foram atualizadas, de acordo com a *surrogate key* dos registos anteriormente adicionados, pois são mais recentes! Isto foi possível através dos *triggers* implementados.

É feito, de seguida, um *update* na base de dados produção sobre a quantidade da respetiva encomenda (*order*) e assim forçar um *update* na tabela *sales_fact* do *data warehouse*.

```
update northwind.order_details set quantity = 6666 where order_id = 31;
```

Figura 12: Quantidade da encomenda nº31 é atualizada para 6666

Após este *update* é necessário fazer novamente o processo de refrescamento e obtêm-se assim, finalmente, o seguinte resultado:

id	order_id	quantity	total_price	discount	preparation_time	last_update	dim_product_id	dim_shipper_id	dim_supplier_id	dim_employee_id	dim_customer_id
59	31	6666	199980	0	2	2019-01-18 12:46:46	6	4	11	10	30
60	31	6666	353298	0	2	2019-01-18 12:46:46	18	4	11	10	30
61	31	6666	23331	0	2	2019-01-18 12:46:46	27	4	11	10	30

Figura 13: Valores finais das vendas de produtos relativos à encomenda nº31. Foram criados novos registos para as vendas, ganhando novas *surrogate keys* e mantendo as *natural keys*, possuindo valores de quantidade e *foreign keys* atualizados, de acordo com os *updates* feitos anterioremente.

4 Business Intelligence

O conceito de ***Business Intelligence*** refere-se ao processo de recolha, transformação, organização, análise e distribuição de dados de várias fontes de informação para melhorar o processo de tomada de decisão de negócios. Deste modo, transforma-se uma grande quantidade de dados brutos em informação útil para tomadas de decisão estratégicas, baseando-se em experiências passadas.

4.1 Power BI

De forma a ser possível realizar uma análise de dados mais eficiente e prática, foram desenvolvidos vários indicadores que permitissem ao utilizador relacionar os dados do *Data Warehousing* para futuras tomadas de decisão.

Todos os indicadores desenvolvidos foram agrupados na ***dashboard*** denominada AD_TP, permitindo a sua visualização em computador e no telemóvel. Estes indicadores são apresentados na seguinte secção, em que a sua descrição encontra-se na legenda correspondente.

4.2 Indicadores

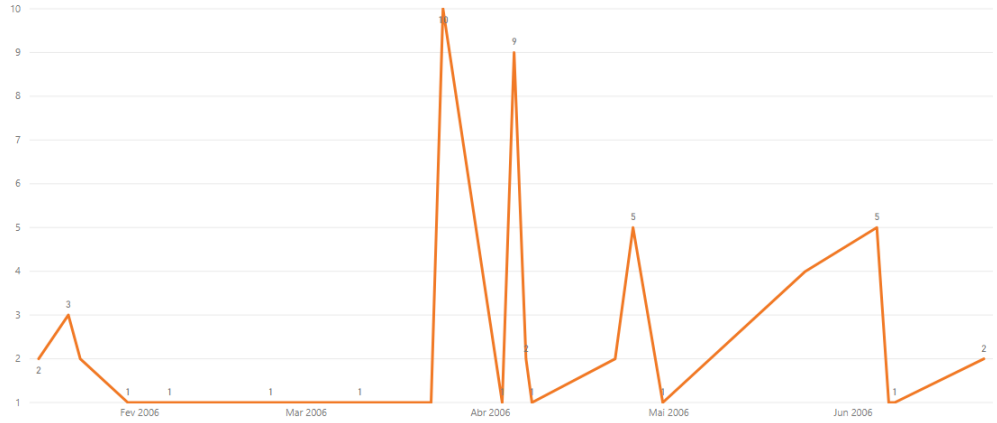


Figura 14: Indicador 1 - Relação das datas com o número de vendas ocorridas em cada dia

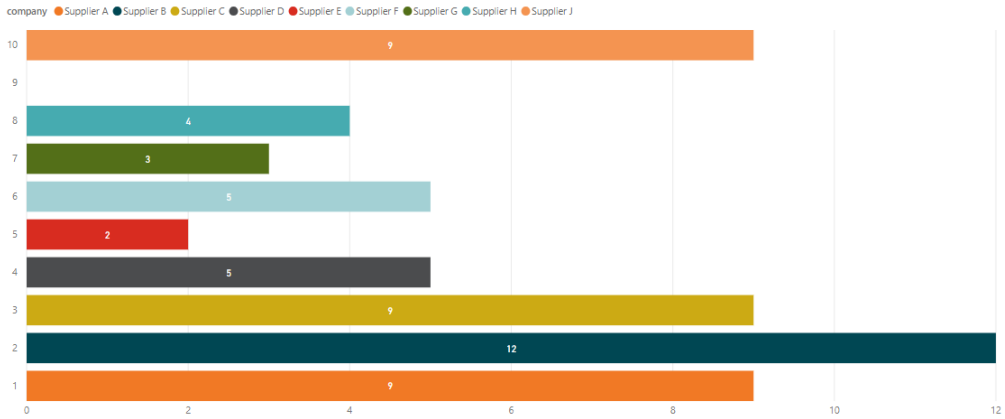


Figura 15: Indicador 2 - Relação da quantidade de produtos fornecidos por fornecedor

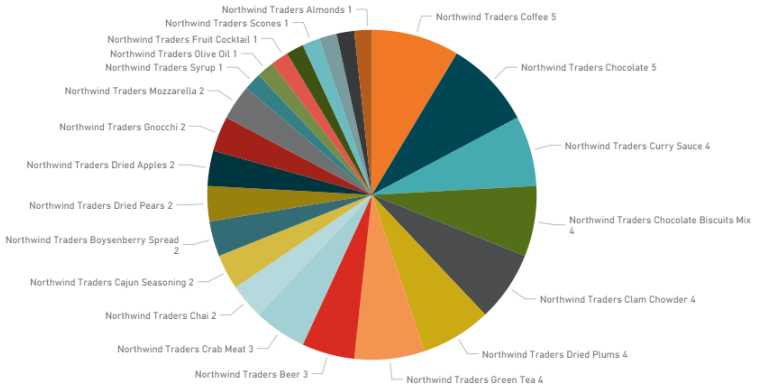


Figura 16: Indicador 3 - Relação da quantidade de vendas de cada produto

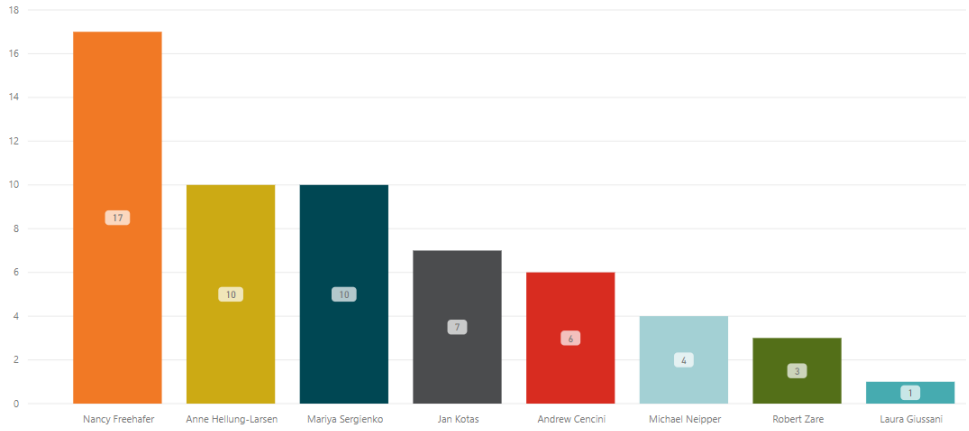


Figura 17: Indicador 4 - Relação da quantidade de vendas feitas por cada empregado

Figura 18: Indicador 5 - Relação da quantidade de vendas por cidade

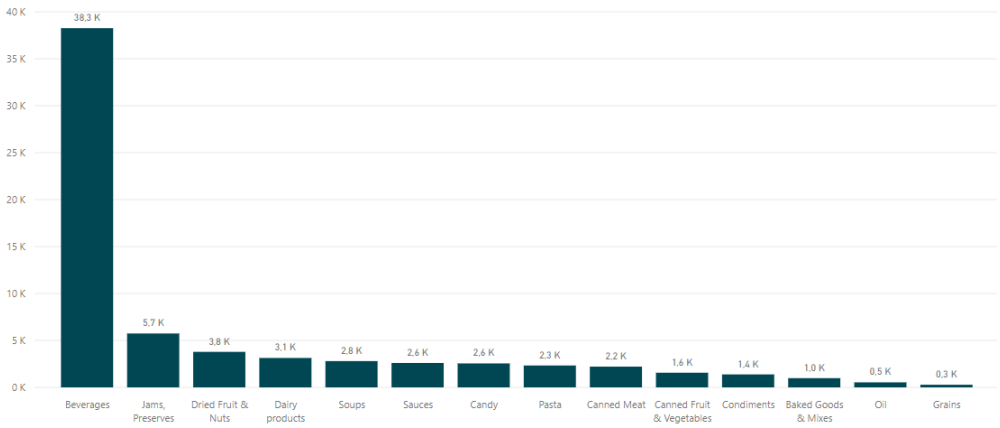


Figura 19: Indicador 6 - Relação do preço total das vendas por categoria dos produtos

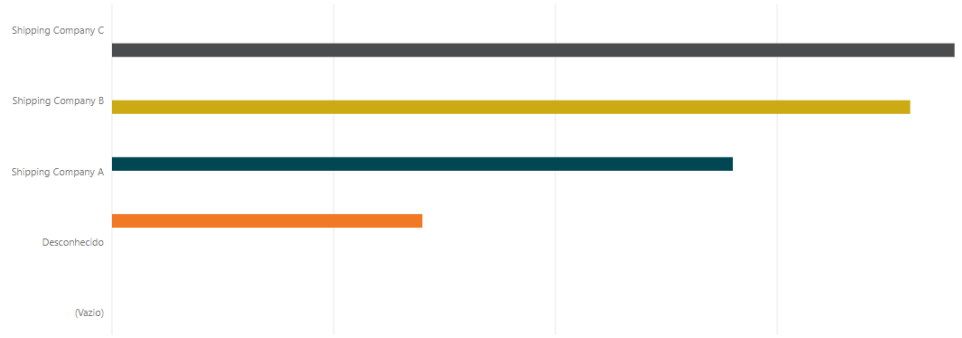


Figura 20: Indicador 7 - Relação da quantidade de vendas enviadas por empresa de transporte

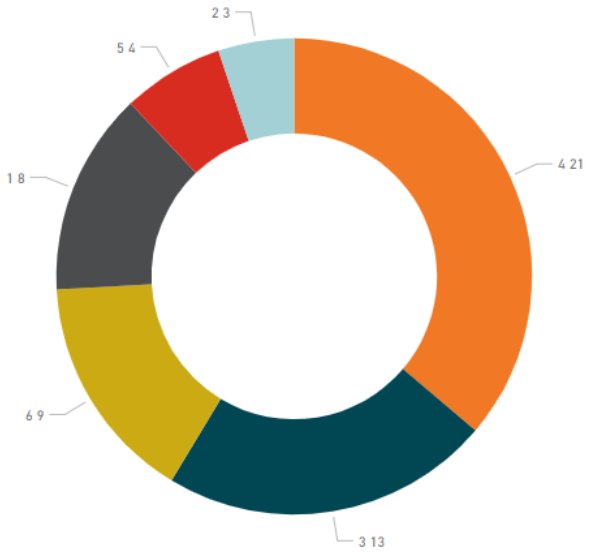


Figura 21: Indicador 8 - Relação da quantidade de vendas por mês

5 Análise de Informação

Através destes indicadores é possível realizar uma análise mais aprofundada aos dados fornecidos. Por exemplo, a altura do ano com mais vendas é durante o mês de abril, a empregada que realizou mais vendas foi *Nancy Freehafer*, o tipo de produto mais vendido, neste caso, as bebidas etc. Com estas informações disponíveis, o responsável por esta empresa pode tomar decisões certas mais facilmente e com um maior grau de apoio.

Eis algumas informações obtidas pelos indicadores criados:

- Na figura **14 - quantidade de vendas por data**, podemos concluir que próximo de Abril foi uma zona temporal com o maior número de vendas, bem como num período um pouco antes do inicio de Julho houve igualmente uma grande quantidade de vendas realizadas. A empresa pode assim decidir apostar mais no preenchimento e manutenção do seu *stock* nestes períodos, já que são requisitadas grandes quantidades de produtos.

- Na figura **15 - quantidade de vendas por fornecedor**, os fornecedores aparecem de baixo para cima, de acordo com a legenda. Facilmente conseguimos concluir então que o fornecedor I (nem aparece na legenda) não realizou vendas. Para além disso, o fornecedor B destacou-se com 12 vendas, seguido pelos fornecedores A, C e J, todos com 9 vendas cada. A empresa pode optar por acabar o contrato com o fornecedor I já que não obtém produtos deste, ou então procurar alguma forma de beneficiar com este contrato. Por outro lado, pode procurar fortalecer os contratos que tem com os fornecedores A, C e J.

- Na figura **16 - quantidade de vendas por produto**, a legenda mostra primeiro o nome do produto seguido do número de vendas. Com esta informação concluímos que em geral as vendas são bastante uniformes, porém destaca-se o café e o chocolate como produtos mais vendidos, pelo que a empresa pode apostar mais na existência de quantidade e variedade destes produtos, aumentando assim o número de vendas e, consequentemente, o número de receitas/lucro.

- Na figura **17 - quantidade de vendas por empregado**, relacionar o número de vendas realizadas por cada empregado. Olhando para a figura conseguimos concluir que a funcionária Nancy Freehafer teve maior número de vendas, seguido por Mariya Sergienko e Anne Hellung-Larsen. A empresa assim pode ter esta informação em conta na gestão do seu *staff*, nomeadamente controlar as promoções e despedimentos da equipa de empregados. Por exemplo aumentar a funcionária Nancy Freehafer e despedir o funcionário funcionário com menos vendas.

- Na figura **18 - quantidade de vendas por cidade**. Nesta imagem constatamos que New York tem o maior número de vendas, seguido de Portland, Chicago e Milwaukee. Com esta informação, a empresa pode optar por instalar mais lojas nessas localizações onde são realizadas mais vendas e apostar nessas infraestruturas, de forma a atrair mais clientela.

- Na figura **19 - preço total por categoria**. Aqui constatamos imediatamente, que o maior lucro vem de bebidas e refrigerantes vendidos, e todas as restantes categorias permanecem aproximadamente equilibradas. Assim a empresa pode apostar mais nos produtos que se encontram na categoria de bebidas (*Beverages*), numa tentativa de aumentar o número de vendas e lucro.

- Na figura **20 - quantidade de vendas por empresa de transporte**. Nesta imagem conseguimos concluir que todas as empresas se mantém proximamente equilibradas nas vendas, referindo-se no entanto a empresa C como sendo a empresa com mais vendas.

- Na figura **21 - quantidade de vendas por mês**, a legenda mostra a sequência número do mês seguido da quantidade de vendas. Sabendo isto, conseguimos extrair que o mês Abril (ID 4) é o mês com maior número de vendas, com 21 vendas, proximamente seguido por Março e Julho, com 13 e 9 vendas, respetivamente.

6 Conclusão

Na realização deste trabalho o grupo pode concluir sobre a importância sobre uma Análise de Dados e a sua relação com o conceito de *Business Intelligence*. Qualquer organização precisa de utilizar toda informação disponível para criar e manter vantagem competitiva. Através de uma análise dos seus dados - *business intelligence*, consegue tomar decisões corretas e rápidas, para aumentar a sua produtividade, ganhar vendas e inteligência de mercado, garantir resultados próximos ao estabelecidos, aumentar os lucros e reunir bastante informação útil.

Um *data warehouse* integra os dados internos e externos da organização numa estrutura única, permitindo uma melhor utilização dos mesmos, aumentando a sua capacidade de resposta e adaptação. Assim pode ajudar as organizações a descobrir novas formas de competir,numa economia globalizada, potenciando novos e melhores produtos e/ou serviços, mais rápida do que a concorrência e sem aumentar o custo dos produtos e/ou serviços.

Por fim, este trabalho prático proporcionou um melhoramento na perspetiva e capacidade de planeamento e gestão de bases de dados, por parte dos elementos do grupo, e promoveu o conhecimento quanto a importância da informação como poder competitivo entre organizações.

7 Anexos

Target				Source				Transformação
Nome da Tabela	Nome do Atributo	Meta Dados	Tipo de Tabela	Base de Dados	Nome da Tabela	Nome do Atributo	Meta Dados	
dim_customer	id	Int	Dimensional				Int	Chave substituta do customer
dim_customer	id_c	Int	Dimensional	NorthWind	customers	id	Int	Chave natural de customer na NorthWind
dim_customer	company	Varchar(45)	Dimensional	NorthWind	customers	company	Varchar(45)	SELECT company FROM customer
dim_customer	name	Varchar(45)	Dimensional	NorthWind	customers	first_name	Varchar(45)	SELECT concat(first_name," ", last_name) FROM customer
dim_customer	name	Varchar(45)	Dimensional	NorthWind	customers	last_name	Varchar(45)	
dim_customer	job_title	Varchar(45)	Dimensional	NorthWind	customers	job_title	Varchar(45)	SELECT job_title FROM customer
dim_supplier	id	Int	Dimensional				Int	Chave substituta do supplier
dim_supplier	id_su	Int	Dimensional	NorthWind	suppliers	id	Int	Chave natural do supplier na NorthWind
dim_supplier	company	Varchar(45)	Dimensional	NorthWind	suppliers	company	Varchar(45)	SELECT company FROM suppliers
dim_shipper	id	Int	Dimensional				Int	Chave substituta do shipper
dim_shipper	id_sh	Int	Dimensional	NorthWind	shippers	id	Int	Chave natural de shipper na NorthWind
dim_shipper	name	Varchar(45)	Dimensional	NorthWind	shippers	first_name	Varchar(45)	SELECT concat(first_name," ", last_name) FROM shippers
dim_shipper	name	Varchar(45)	Dimensional	NorthWind	shippers	last_name	Varchar(45)	
dim_product	id	Int	Dimensional				Int	Chave substituta do product
dim_product	id_p	Int	Dimensional	NorthWind	products	id	Int	Chave natural do product na NorthWind
dim_product	name	Varchar(45)	Dimensional	NorthWind	products	product_name	Varchar(45)	SELECT products_name FROM product
dim_product	category_name	Varchar(45)	Dimensional	NorthWind	products	category	Varchar(45)	SELECT category FROM products
dim_product	standard_cost	Decimal(19,4)	Dimensional	NorthWind	products	standard_cost	Decimal(19,4)	SELECT standard_cost FROM products
dim_product	discontinued	TinyInt	Dimensional	NorthWind	products	discontinued	TinyInt	SELECT discontinued FROM products
dim_employee	id	Int	Dimensional				Int	Chave substituta do employee
dim_employee	id_e	Int	Dimensional	NorthWind	employees	id	Int	Chave natural do employee na NorthWind
dim_employee	name	Varchar(45)	Dimensional	NorthWind	employees	first_name	Varchar(45)	SELECT concat(first_name," ", last_name) FROM employees
dim_employee	name	Varchar(45)	Dimensional	NorthWind	employees	last_name	Varchar(45)	
dim_employee	company	Varchar(45)	Dimensional	NorthWind	employees	company	Varchar(45)	SELECT company FROM employees
dim_employee	job_title	Varchar(45)	Dimensional	NorthWind	employees	job_title	Varchar(45)	SELECT job_title FROM employees
dim_local	id	Int	Dimensional	NorthWind	customers	id	Int	Chave natural de Local, especifica para o DW
dim_local	city	Varchar(45)	Dimensional	NorthWind	customers	city	Varchar(45)	SELECT DISTINCT city FROM customer

Figura 22: Mapa Lógico

dim_local	state	Varchar(45)	Dimensional	NorthWind	customers	state_province	Varchar(45)	SELECT DISTINCT state_province FROM customer
dim_local	country	Varchar(45)	Dimensional	NorthWind	customers	country_region	Varchar(45)	SELECT DISTINCT country_region FROM customer
sales_fact	id	Int	Factos	NorthWind				Chave natural da Tabela de Factos, especifica para o DW
sales_fact	order_id	Int	Factos	NorthWind	order_details	order_id	Int	SELECT order_id FROM order_details INNER JOIN order ON order_details.order_id = orders.id
sales_fact	quantity	Decimal(19,4)	Factos	NorthWind	order_details	quantity	Decimal(19,4)	SELECT quantity FROM order_details INNER JOIN order ON order_details.order_id = orders.id
sales_fact	total_price	Decimal(19,4)	Factos	NorthWind	order_details	quantity	Decimal(19,4)	SELECT quantity * unit_price FROM order_details INNER JOIN orders ON order_details.order_id = orders.id
sales_fact	total_price	Decimal(19,4)	Factos	NorthWind	order_details	unit_price	Decimal(19,4)	
sales_fact	discount	Double	Factos	NorthWind	order_details	discount	Double	SELECT discount FROM order_details INNER JOIN orders ON order_details.order_id = orders.id
sales_fact	preparation_time	Int	Factos	NorthWind	orders	shipped_date	DateTime	SELECT COALESCE(DATEDIFF(shipped_date,order_date),-1) FROM orders
sales_fact	preparation_time	Int	Factos	NorthWind	orders	order_date	DateTime	
sales_fact	dim_product_id	Int	Factos	NorthWind	products	id	Int	SELECT id FROM products INNER JOIN products on order_details.product_id = products_id
sales_fact	dim_shipper_id	Int	Factos	NorthWind	orders	shipper_id	Int	SELECT COALESCE(shipper_id,-1) FROM order_details INNER JOIN orders ON order_details.order_id = orders.id
sales_fact	dim_supplier_id	Int	Factos	NorthWind	suppliers	supplier_ids	LongText	SELECT SUBSTRING_INDEX(supplier_ids, ";", 1) FROM products INNER JOIN products ON order_details.product_id =product.products_id
sales_fact	dim_employee_id	Int	Factos	NorthWind	orders	employee_id	Int	SELECT employee_id FROM order_details INNER JOIN orders ON order_details.product_id = products_id
sales_fact	dim_customer_id	Int	Factos	NorthWind	orders	customer_id	Int	SELECT customer_id FROM order_details INNER JOIN orders ON order_details.product_id = products_id
sales_fact	order_date	Int	Factos	NorthWind	orders	order_date	DateTime	SELECT order_date FROM order_details INNER JOIN orders ON order_details.product_id = products_id
sales_fact	shipped_date	Int	Factos	NorthWind	orders	shipped_date	DateTime	SELECT COALESCE((select dim_time.id from dim_time where date = date(shipped_date)),-1) FROM order_details INNER JOIN orders ON order_details.product_id = products_id
sales_fact	payment_date	Int	Factos	NorthWind	orders	paid_date	DateTime	SELECT COALESCE((select dim_time.id from dim_time where date = date(paid_date)),-1) FROM order_details INNER JOIN orders ON order_details.product_id = products_id
sales_fact	customer_local	Int	Factos	NorthWind	customers	city	Varchar(45)	SELECT COALESCE((select dim_local.id where dim_local.id = northwind.customers.id),-1) FROM order_details inner join dim_local on dim_local.city = northwind.customers.city

Figura 23: Mapa Lógico