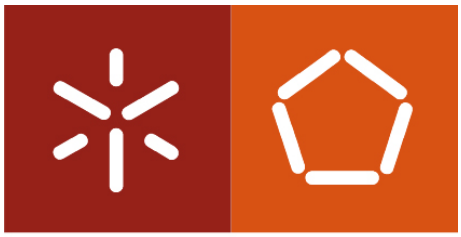

Universidade do Minho

Aplicações Informáticas na Biomedicina

Trabalho Prático

Urgências Gerais

César Silva
PG41842

Filipe Cunha
A83099

Hugo Gião
PG41073

José Alves
A82885

Tiago Ramires
PG41101

Novembro 2019

Conteúdo

1 Introdução **1**

2 Implementação **1**
2.1 Modelo dimensional em Estrela 1
2.2 Povoamento em SQL 2
2.2.1 Povoamento dim_local 2
2.2.2 Povoamento dim_idade 2
2.2.3 Povoamento dim_date 3
2.2.4 Povoamento dim_time 4
2.2.5 Povoamento factos 4

3 Talend Open Studio **5**
3.1 Povoamento 5
3.1.1 Povoamento dim_local 5
3.1.2 Povoamento dim_idade 6
3.1.3 Povoamento dim_date 6
3.1.4 Factos 8
3.2 Talend vs SQL 10

4 Criação de indicadores clínicos usando Power BI **11**
4.1 Indicadores Criados 11
4.1.1 Indicadores relativos a idades e distribuição de consultas por grupos etários 11
4.1.2 Indicadores relativos a consultas, a especialidade na qual ela se enquadra e a sua causa 12
4.1.3 Indicadores relativos a género e localidade dos pacientes . 13
4.1.4 Indicadores relativos ao género e residência dos pacientes das consultas 14
4.1.5 Aplicação que faça uso dos nossos Indicadores e Dashboards 15

5 Conclusão **16**

1. Introdução

O presente trabalho foi realizado no âmbito da disciplina de Aplicações Informáticas na Biomedicina onde se irá realizar a demonstração/resposta das questões colocadas no enunciado do mesmo.

Era-nos pedido para efectuar a análise de dados referentes há realização de urgências gerais num determinado hospital nacional.

Os dados disponibilizados contém dados reais da lista de realização de urgências gerais num determinado hospital nacional. Os dados estavam armazenados numa base de dados *Oracle* e desses dados foram extraídos 1000 registos de uma tabela da base de dados para um ficheiro no formato *CSV*. A informação representada inclui o número de episódio da urgência (identificador único), a data e a hora da admissão do paciente nas urgências, a data e a hora da alta, a descrição da especialidade da alta, a descrição do local, a descrição da proveniência, a descrição da causa da entrada nas urgências, o género do paciente, bem como a sua data de nascimento.

2. Implementação

Por forma a criar o *Data Warehouse*, foi inicialmente criada uma base de dados *SQL* para a qual foram importados os dados contidos no ficheiro *CSV* disponibilizado. De seguida, e a partir da informação obtida, foi criado o respectivo modelo dimensional em Estrela.

2.1. Modelo dimensional em Estrela

Como esquema de modelação do *Data Warehouse* a desenvolver, optou-se por a implementação de um modelo dimensional em Estrela, em que consiste em uma tabela de factos ligada a um conjunto de dimensões. Com este modelo, a compreensão e navegação dos dados torna-se mais fácil e o desempenho nas consultas de dados torna-se consideravelmente mais eficiente.

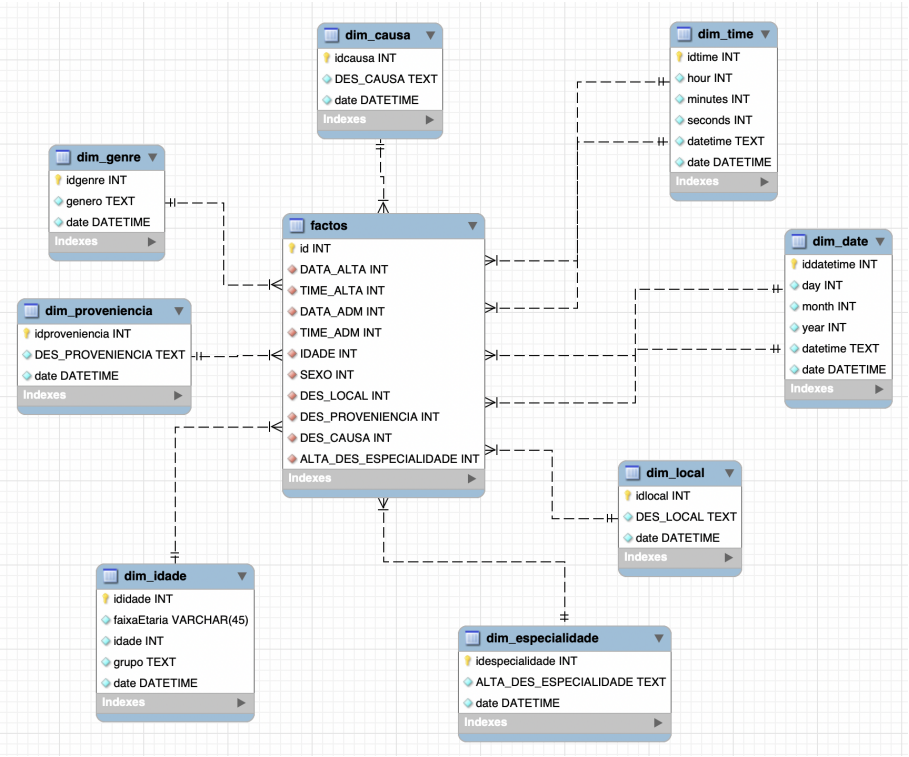

Figura 1: Esquema em estrela

2.2. Povoamento em SQL

Efectivamente, depois de definir o modelo conceptual, e gerar o modelo físico, iniciou-se o povoamento da base de dados. Numa primeira abordagem, utilizou-se a linguagem SQL para povoar as tabelas de dimensão seguidas da tabela de factos.

2.2.1. Povoamento dim_local

O povoamento da tabela dim_local, foi bastante simples, visto que apenas foi necessário mapear os valores únicos da coluna DES_LOCAL (descrição do local) do *dataset* original para a dimensão.

```
insert into dim_local (DES_LOCAL, date)
        select DES_LOCAL,now() from urg_inform_geral
        group by DES_LOCAL;
```

2.2.2. Povoamento dim_idade

O povoamento da tabela dim_idade teve em conta a faixa etária e o grupo a que cada idade pertence. Posto isto, foi utilizado um *case* para adicionar cada idade na categoria correspondente.

```
insert into dim_idade (faixaEtaria, idade, grupo, date)(
    select distinct
                CASE WHEN j.t < 10 THEN "00-10"
                        when j.t < 20 then "10-20"
                        when j.t < 30 then "20-30"
                        when j.t < 40 then "30-40"
                        when j.t < 50 then "40-50"
                        when j.t < 60 then "50-60"
                        when j.t < 70 then "60-70"
                        when j.t < 80 then "70-80"
                        when j.t < 90 then "80-90"
                        when j.t < 100 then "90-100"
                        else "Undef"
                END AS faixaEtaria,
        j.t as idade,
                CASE WHEN j.t < 10 THEN "Criança"
```

```
                    WHEN j.t < 18 THEN "Adolescente"
                    WHEN j.t < 40 THEN "Adulto"
                    WHEN j.t < 60 THEN "Meia idade"
                    else "Idoso"
                END AS grupo,
        now()
    from
        (SELECT TIMESTAMPDIFF(YEAR,idade.dta_nascimento,now()) as
            t,idade.dta_nascimento as n
          from ( select u.dta_nascimento from urg_inform_geral as u) as idade
            group by n) as j) ;
```

2.2.3. Povoamento dim_date

Para povoar a tabela dim_date foi necessário realizar um *parse* das datas do *CSV* de modo a separar um datetime em date e time. Isto permite uma maior normalização dos dados e melhor desempenho/facilidade em consultas agregadas por tempo ou data.

```
insert into dim_date ( day, month, year, datetime, date)
    select  distinct
                day(STR_TO_DATE(DATAHORA_ADM, "%Y-%m-%d %H:%i:%s")),
                month(STR_TO_DATE(DATAHORA_ADM, "%Y-%m-%d %H:%i:%s")),
                year(STR_TO_DATE(DATAHORA_ADM, "%Y-%m-%d %H:%i:%s")),
        date(DATAHORA_ADM),
        now() from urg_inform_geral
        UNION
    select  distinct
                day(STR_TO_DATE(DATAHORA_ALTA, "%Y-%m-%d %H:%i:%s")),
                month(STR_TO_DATE(DATAHORA_ALTA, "%Y-%m-%d %H:%i:%s")),
                year(STR_TO_DATE(DATAHORA_ALTA, "%Y-%m-%d %H:%i:%s")),
        date(DATAHORA_ALTA),
        now() from
        urg_inform_geral;
```

2.2.4. Povoamento dim_time

De forma análoga ao povoamento da tabela dim_date, na tabela dim_time foi feito um parse das datas de forma a promover a normalização.

```
insert into dim_time (hour, minutes, seconds, datetime, date)
    select  distinct
        hour(STR_TO_DATE(DATAHORA_ADM, "%Y-%m-%d %H:%i:%s")),
        minute(STR_TO_DATE(DATAHORA_ADM, "%Y-%m-%d %H:%i:%s")),
        second(STR_TO_DATE(DATAHORA_ADM, "%Y-%m-%d %H:%i:%s")),
        time(DATAHORA_ADM),
        now() from urg_inform_geral
        UNION
        select  distinct
        hour(STR_TO_DATE(DATAHORA_ALTA, "%Y-%m-%d %H:%i:%s")),
        minute(STR_TO_DATE(DATAHORA_ALTA, "%Y-%m-%d %H:%i:%s")),
        second(STR_TO_DATE(DATAHORA_ALTA, "%Y-%m-%d %H:%i:%s")),
        time(DATAHORA_ALTA),
        now() from urg_inform_geral;
```

2.2.5. Povoamento factos

Finalmente, depois de termos todas as tabelas dimensionais povoadas, procedemos à povoação da tabela de factos. É nesta fase onde validamos a povoação da base de dados na medida em que, a tabela de factos representa todas as relações dos dados do sistema, pelo que se alguma tabela de dimensão estiver mal povoada, não é possível povoar a tabela de factos.

```
insert into factos (id, DATA_ALTA, TIME_ALTA, DATA_ADM, TIME_ADM, IDADE, SEXO,
DES_LOCAL, DES_PROVENIENCIA, DES_CAUSA, ALTA_DES_ESPECIALIDADE)
    select u.URG_EPISODIO,talta.iddatetime,ttalta.idtime,tadmi.iddatetime,
    ttadmi.dtime,i.ididade,g.idgenre,l.idlocal,p.idproveniencia,c.idcausa,
    e.idespecialidade from urg_inform_geral as u
        inner join dim_date as talta on date(u.DATAHORA_ALTA) = talta.datetime
        inner join dim_date as tadmi on date(u.DATAHORA_ADM) = tadmi.datetime
        inner join dim_time as ttalta on time(u.DATAHORA_ALTA) = ttalta.datetime
        inner join dim_time as ttadmi on time(u.DATAHORA_ADM) = ttadmi.datetime
        inner join dim_idade as i
            on TIMESTAMPDIFF(YEAR,u.DTA_NASCIMENTO,now()) = i.idade
        inner join dim_genre as g on u.SEXO = g.genero
```

```
inner join dim_local as l on u.DES_LOCAL = l.DES_LOCAL
inner join dim_proveniencia as p on u.DES_PROVENIENCIA = p.DES_PROVENIENCIA
inner join dim_causa as c on u.DES_CAUSA = c.DES_CAUSA
inner join dim_especialidade as e
    on u.ALTA_DES_ESPECIALIDADE = e.ALTA_DES_ESPECIALIDADE;
```

3. Talend Open Studio

Apresentámos de seguida a povoação da *Data Warehouse* via *jobs* de *Talend*

3.1. Povoamento

Não apresentaremos todos os *jobs* nesta secção devido a alguns serem muito similares. Dos métodos disponíveis no *Talend* destacámos os seguintes:

1. Numeric.sequence - Utilizado para criação dos *id*'s das várias dimensões consideradas.
2. TalendDate.formatDate - Para conversão de datas para caracteres em determinados formatos.

3.1.1. Povoamento dim_local

No povoamento da dimensão local, começamos por filtrar as linhas repetidas usando a ferramenta *tUniqRow* para remover entradas repetidas tendo em conta o atributo *_DES_LOCAL_*. De seguida, usamos um *tMap* para construir os valores a ser guardados. Finalmente, guardámos os valores na tabela alvo.

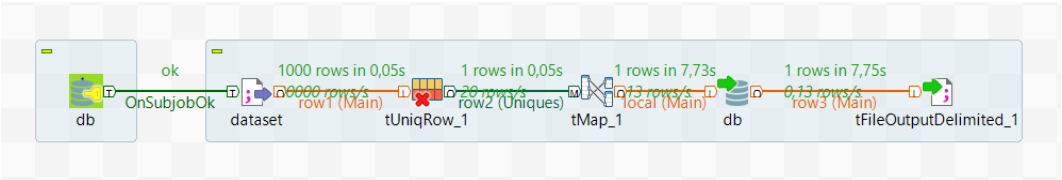

Figura 2: Job povoamento_local1

3.1.2. Povoamento dim_idade

Inicialmente, começámos por calcular as faixas etárias, os grupos e a idade para cada uma das entradas relativas a coluna *_DATA_NASCIMENTO_* usando um *tMap*. De seguida, removes as idades repetidas via *tUniqRow*. Finalmente, construímos os id's de cada umas das entradas via outro *tMap* e inserimos os valores na base de dados.

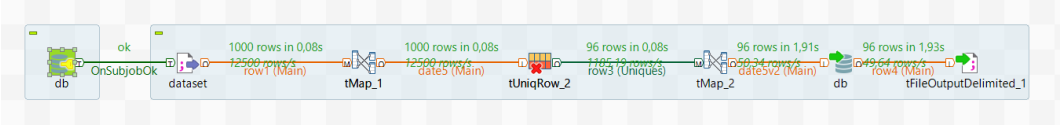

Figura 3: Job povoamento_idade

3.1.3. Povoamento dim_date

Inicialmente extraímos as datas das entradas *_DATA_ADM_* e *_DATA_ALTA_* do *dataset* e exportamos estas para ficheiros separadamente via os *jobs date1* e *date2*.

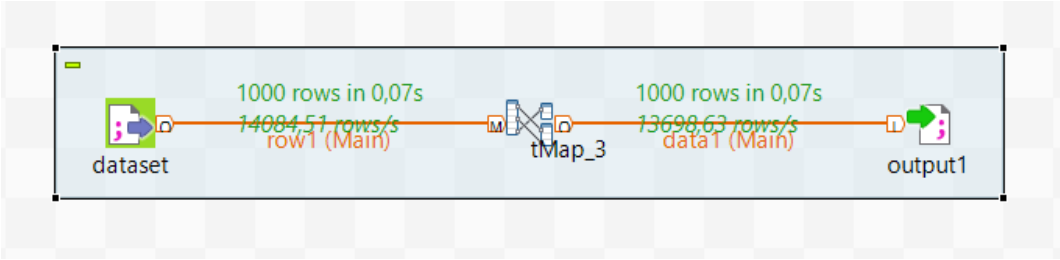

Figura 4: Job povoamento_date1

De seguida, combinamos a informação extraída via um *tUnite*, extraímos os valores repetidos via *tUniqRow* e criamos a informação para a *data warehouse* via um *tMap* e guardámos a informação.

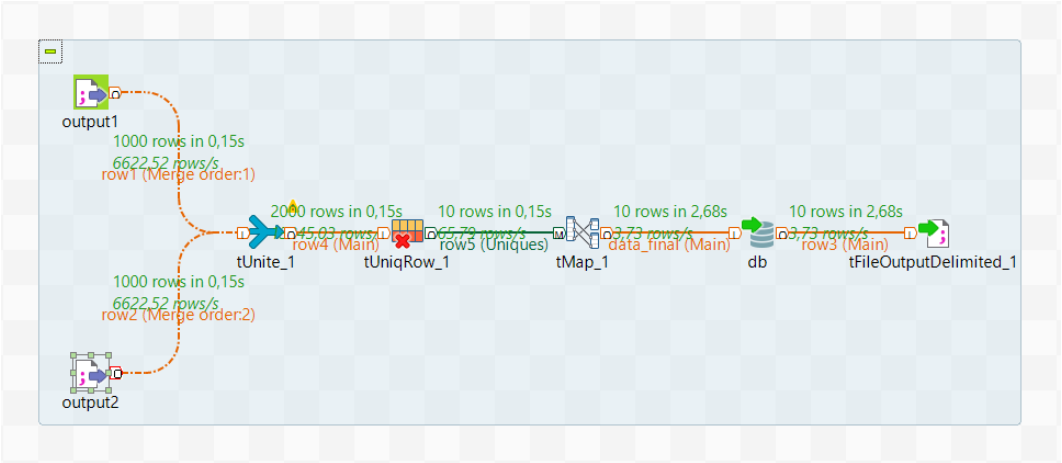

Figura 5: Job povoamento_date3

3.1.4. Factos

Para o povoamento da tabela de factos, realizamos um *inner-join* via um *tMap*, das várias tabelas exportadas via *csv*. Os *inner-join*'s são dados usando um esquema similar ao do povoamento via *SQL*.

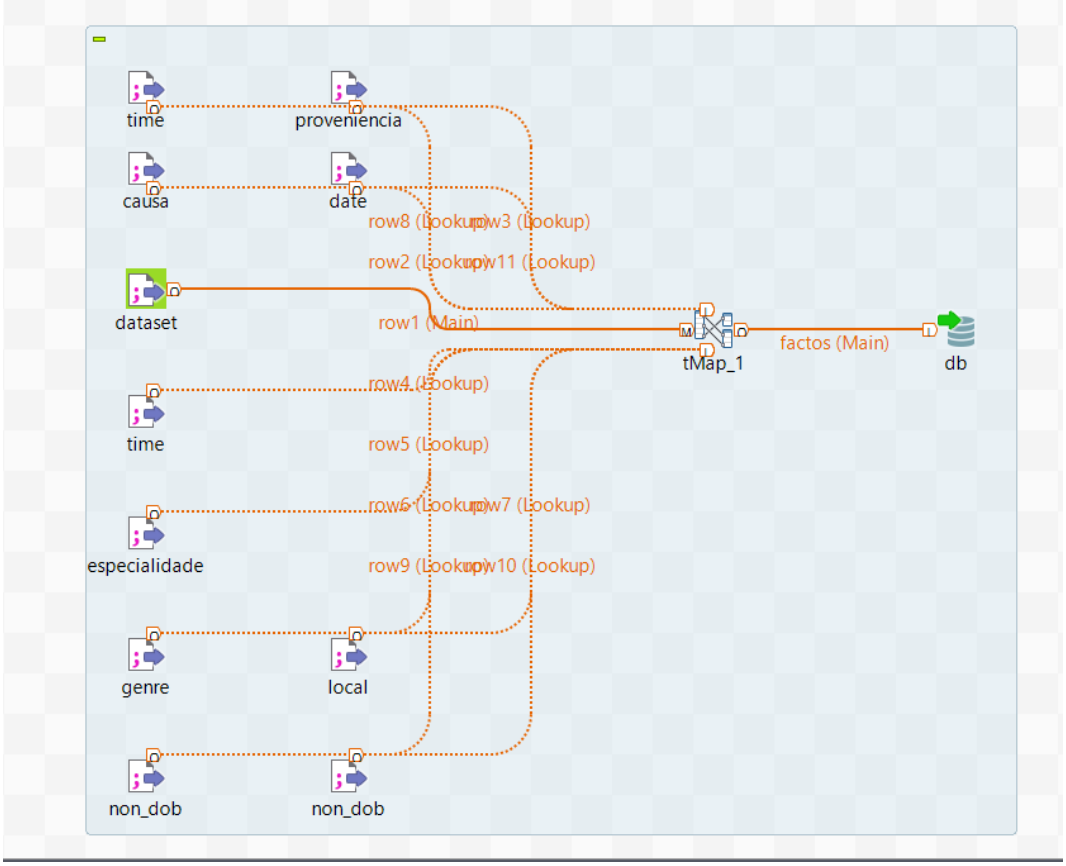

Figura 6: Job povoamento_factos

Apresentámos um excerto do esquema de *inner-join*'s do *tMap* considerado no *job*.

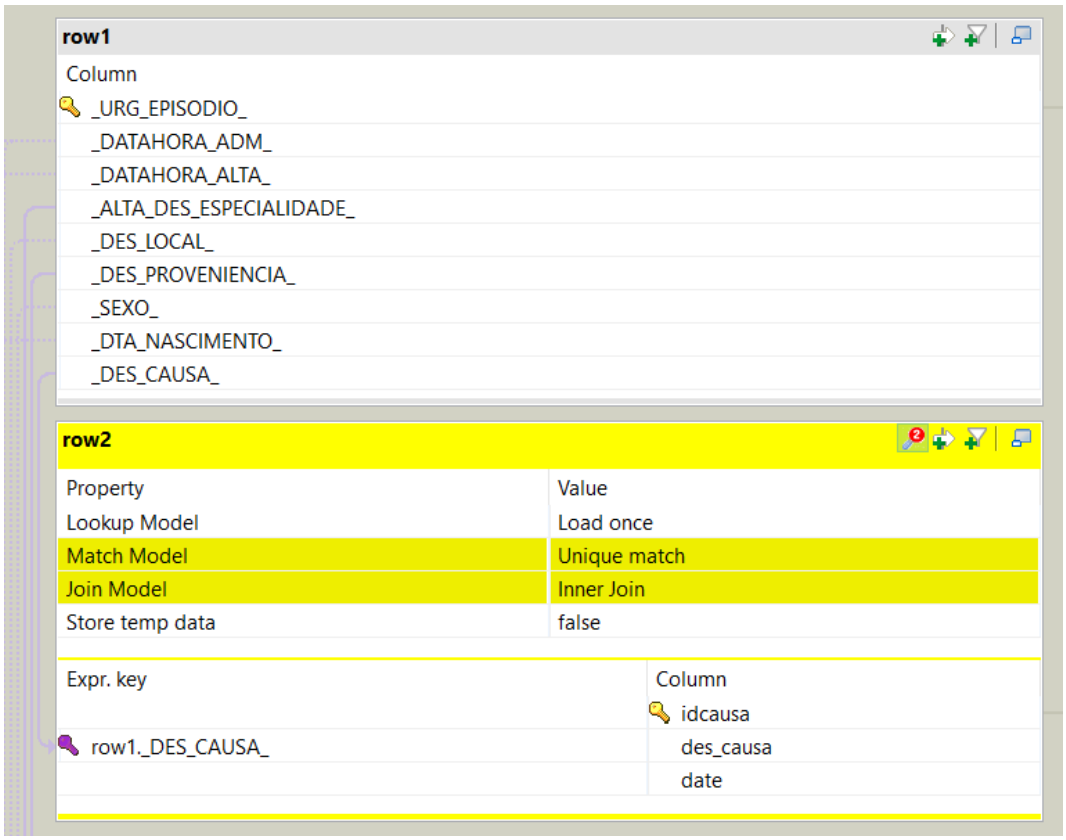

Figura 7: Excerto to *tMap*

3.2. Talend vs SQL

Discutimos agora as diferenças principais de ambas as metodologias de povoação abordadas neste trabalho.

Destas destacámos as seguintes.

1. Portabilidade - Os scripts *SQL* usados para povoação dos dados são facilmente usados noutros sistemas, ao contrário do Talend.
2. Compatibilidade - O *Talend* permite a aplicação de processos *ETL* a diferentes fontes de dados no mesmo fluxo de ETL.
3. Complexidade - Talend permite criar processos *ETL* de forma mais simples e intuitiva, mas em contrapartida não fornece a mesma liberdade que os scripts *SQL* fornecem.
4. Abstração - Um utilizador de *Talend* não precisa de ter conhecimento prévio de bases de dados para utilizar a ferramenta.

4. Criação de indicadores clínicos usando Power BI

4.1. Indicadores Criados

Optámos por agrupar os indicadores criados por *dashboards* de acordo com o seu conteúdo e uso pretendido.

4.1.1. Indicadores relativos a idades e distribuição de consultas por grupos etários

Criámos 3 indicadores para esta *dashboard*, o primeiro indicador consiste num gráfico de *donut* utilizado para mostrar a distribuição das consultas por grupos etários.

O segundo indicador é por sua vez um indicador que mostra a proporção de consultas de cada faixa etária.

Por sua vez o terceiro indicador mostra a evolução do numero de consultas com o aumento da idade.

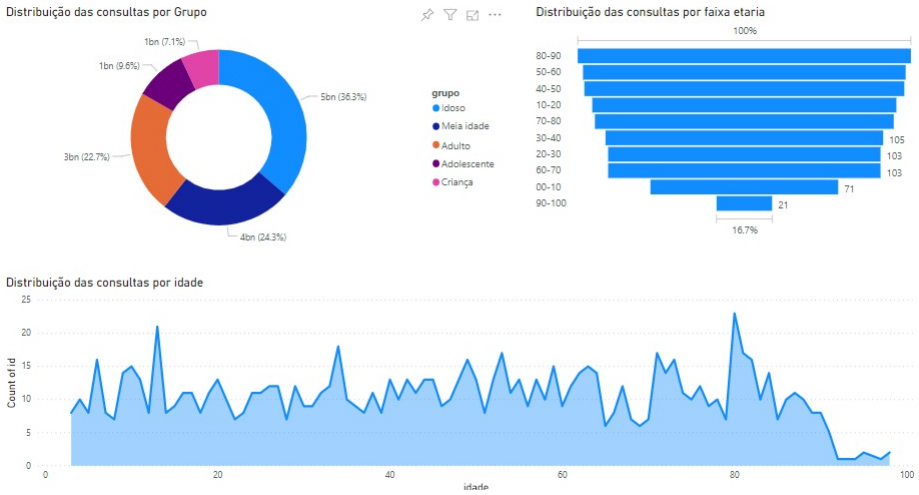

Figura 8: *Dashboard* relativa a distribuição de idades, grupos e faixa etárias.

Motivação para a criação destes indicadores A motivação para a criação destes indicadores assenta na necessidade de hospitais, clínicas e outras instituições de precisarem substancialmente de dados e informação relativa a distribuição das idades e faixas etárias dos seus pacientes sendo que os mesmo implicam directamente os inventários, alocação de recursos hospitalares e contratação de funcionários.

4.1.2. Indicadores relativos a consultas, a especialidade na qual ela se enquadra e a sua causa

O primeiro indicador mostra o numero de consultas já realizado.

O segundo indicador mostra as proporções das consultas pela sua causa.

O terceiro mostra a distribuição de consultas pela especialidade na qual se enquadram.

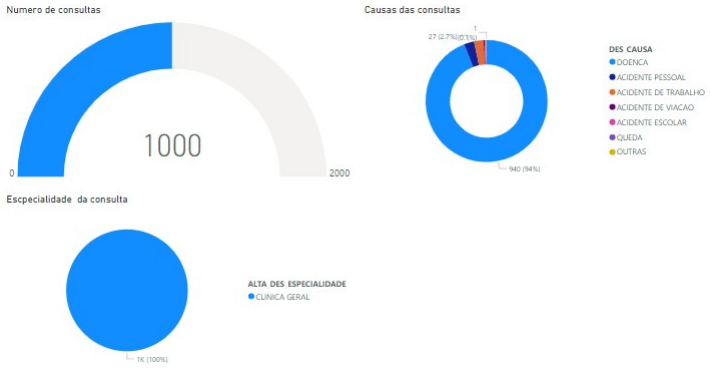

Figura 9: *Dashboard* relativa a consultas, a especialidade em que se enquadram e o seu motivo.

Motivação para a criação destes indicadores A motivação para a criação destes indicadores assenta no facto de conhecimento sobre o tipo de consultas realizado,o seu numero e as especialidades em que estas se enquadram pode conduzir a um melhor uso dos recursos por parte das instituições de saúde estes passam pela contratação de pessoal,compra de medicamentos,material e software.

4.1.3. Indicadores relativos a género e localidade dos pacientes

Criamos uma terceira *dashboard* com o intuito de mostrar informação variada sobre os pacientes.

O primeiro indicador mostra a distribuição de consultas por localidade dos utentes.O segundo pelo seu género.

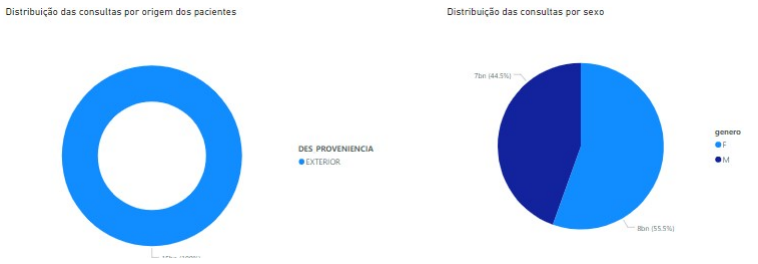

Figura 10: Indicadores relativos a distribuição de consultas por género e localidade

Motivação para a criação destes indicadores A motivação de indicadores surge pelo facto de conhecer as necessidades e o perfil dos utentes é um fator positivo no processo de tomada de decisão das instituições clínicas.

4.1.4. Indicadores relativos ao género e residência dos pacientes das consultas

Criamos uma quarta *dashboard* com o intuito de mostrar informação variada sobre a evolução e distribuição temporal das consultas.

O primeiro indicador mostra a evolução do numero de consultas por ano enquanto o segundo mostra a distribuição das consultas por mês.

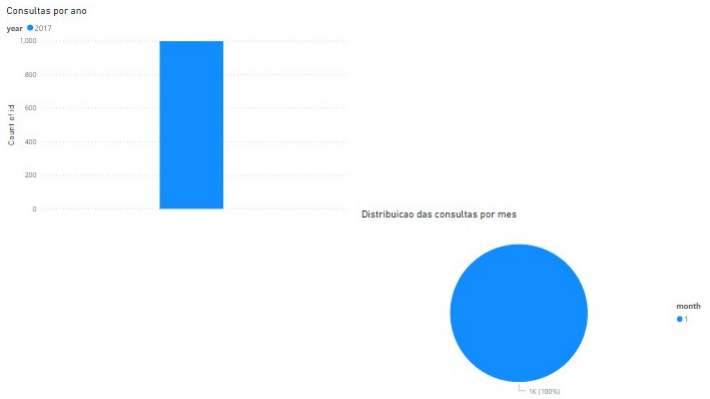

Figura 11: Indicadores relativos a evolução e distribuição temporal das consultas.

Motivação para a criação destes indicadores Este indicador motiva-se por possibilitar fazer predições sobre consultas futuras tanto pela evolução anual das mesmas e pela proporção mensal das mesmas no passado.

4.1.5. Aplicação que faça uso dos nossos Indicadores e Dashboards

Contexto As diferentes dashboards e indicadores clínicos que criamos poderiam ser utilizados como parte de uma aplicação informática.

A aplicação que nós contemplamos consiste numa aplicação web e mobile acessível a membros do hospital que lhes permita aceder a um numero de dashboards e indicadores dos quais os nossos, de forma conveniente e rápida, de forma a poder obter uma melhor visualização do estado do hospital.Esta aplicação seria de uso para auxiliar o processo de tomada de decisão no hospital.

Funcionamento da aplicação Esta aplicação teria um servidor com acesso aos dados presentes na base de dados do hospital,estes necessários à geração dos diversos indicadores e dasboards.

Para os utilizadores comunicarem com o servidor a nossa aplicação possuiria também uma interface Web e uma aplicação mobile.

Tecnologias utilizadas O servidor poderia ser implementado em **Python**, utilizando o conector de **Mysql** para **Python** assumindo que a base de dados a que temos acesso é deste tipo ou outro conector consoante a base de dados utilizada pelo hospital. Para gerar dashboards apesar de termos utilizado o **PowerBi** nas aulas e no trabalho prático provavelmente usaríamos uma biblioteca de criação de gráficos como o **Seaborn**, **Plotly** e ,**Mathplotlit** para gerar os indicadores de Business Inteligente apartir dos dados obtidos.

Para construir a interface Web usaríamos uma biblioteca de **Python** como **Flask** ou **Django**.Para a aplicação mobile utilizaríamos o **Android Studio** para **Android** e **Xcode** para **Ios**.

Para enviar os dasboards e indicadores bem como os restantes dados necessários para o funcionamento da aplicação usaríamos **Json**.

Alternativamente poderíamos utilizar a biblioteca **Dash** de **Python** .Esta biblioteca é open source e de uso gratuito e permite criar dashboards acessíveis em paginas web .O uso desta biblioteca proporcionaria um desenvolvimento mais rápido porem menos flexibilidade e funcionalidades.

5. Conclusão

Este trabalho permitiu explorar a criação de um *datawarehouse*, assim como a sua análise. A maior dificuldade que o grupo encontrou durante a realização do trabalho foi no povoamento da base de dados utilizando o *Talend Open Studio*. A utilização da ferramenta Microsoft Power BI foi relativamente simples e achamos que os *dashboards* criados são bastante úteis. De uma maneira geral achamos que o resultado final deste trabalho foi bastante positivo.

Futuramente pretendemos utilizar os conhecimentos adquiridos na realização deste trabalho em diferenets projetos que façam uso do processos de ETL,analise de dados