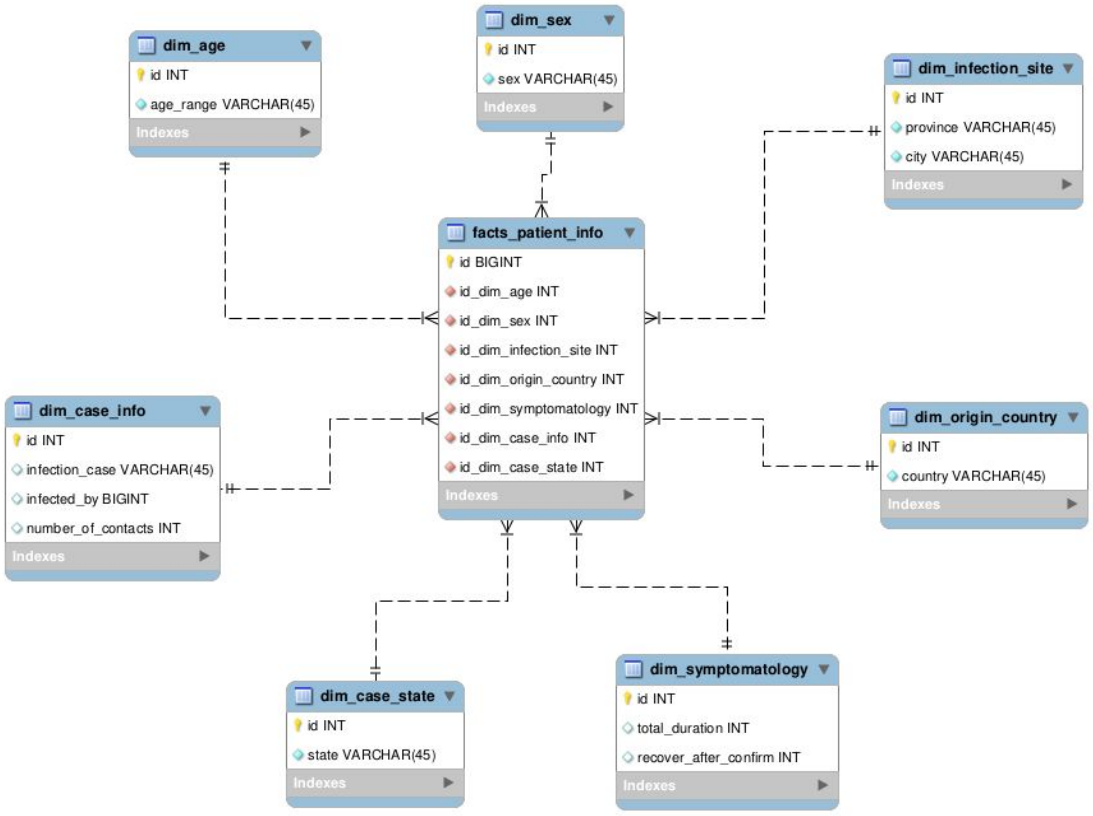
Análise de Dados - TP02

Vasco Ramos, PG42852

Criação do EER (Modelo Dimensional em Estrela)

Para criar um modelo lógico cujo modelo dimensional fosse um esquema de estrela, a partir do CSV fornecido, o primeiro passo foi perceber qual era a entidade da tabela de facto (neste caso, o paciente) e a partir daí tentar agrupar colunas do CSV representativas de conceitos correlacionáveis em tabelas de dimensão associadas à tabela de facto. O resultado final foi o seguinte modelo:



As tabelas dimensionais de **idade**, **sexo** e **país de origem** são autoexplicativas. No caso do **sítio de infeção** agrupei as colunas província e *cidade* para termos uma representação única de cada local de infeção registado. Com os dados relativos ao início dos sintomas, confirmação de infeção e data de alta médica ("cura"), criei uma **tabela de sintomatologia**, onde são representados os intervalos de tempo (em dias) desde o início dos sintomas até à alta e desde a confirmação até à alta. Na **tabela de estado**, decidi isolar este campo, pois, não achei que tivesse uma correlação direta com nenhuma das outras colunas. E,

por fim, decidi agrupar a informação relativa a: razão de infeção, pessoa por quem foi infectada e o *número de pessoas com que se esteve em contacto*, numa tentativa de conseguir criar uma vista relativa aos focos e razões de infeção geralmente úteis para informar e testar pessoas que estão possivelmente infetadas e ter uma perceção da origem dessas infeções.

Criação do Modelo Físico

O próximo passo foi a criação do modelo físico, através do *MySQL Workbench*, que ficou com uma estrutura geral semelhante à vista parcial possível de ver na seguinte imagem:

```
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `covid_korea` ;

-- -----------------------------------------------------
-- Schema covid_korea
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `covid_korea` ;
USE `covid_korea` ;

-- -----------------------------------------------------
-- Table `covid_korea`.`dim_age`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `covid_korea`.`dim_age` ;

CREATE TABLE IF NOT EXISTS `covid_korea`.`dim_age` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `age_range` VARCHAR(45) NOT NULL DEFAULT 'N/A',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `age_range_UNIQUE` (`age_range` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `covid_korea`.`dim_sex`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `covid_korea`.`dim_sex` ;

CREATE TABLE IF NOT EXISTS `covid_korea`.`dim_sex` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `sex` VARCHAR(45) NOT NULL DEFAULT 'N/A',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `sex_UNIQUE` (`sex` ASC) VISIBLE)
ENGINE = InnoDB;
```

Povoação das tabelas

Após a criação do modelo físico, passei para o povoamento das tabelas. Para isso desenvolvi um script em Python que gera um ficheiro SQL com todos os inserts necessários para a Base de Dados. Na imagem é possível ver a estrutura principal do programa e uma das funções desenvolvidas:

```python
def insert_dim_age(rows):
    global dim_age

    ages = set([row[2] for row in rows])

    for idx, age in enumerate(ages):
        for row in rows:
            if row[2] == age:
                patient_info_dict[row[0]][0] = idx + 1

        if age != "":
            dim_age += "insert into dim_age (age_range) values ('{}');\n".format(age)
        else:
            dim_age += "insert into dim_age values ();\n"


def main():
    csv_rows = read_csv_to_list("South_Korea_Covid19.csv")

    init_facts_patient_info(csv_rows)

    insert_dim_age(csv_rows)
    insert_dim_sex(csv_rows)
    insert_dim_infection_site(csv_rows)
    insert_dim_origin_country(csv_rows)
    insert_dim_symptomatology(csv_rows)
    insert_dim_case_state(csv_rows)
    insert_dim_case_info(csv_rows)

    insert_facts_patient_info()

    with open("populate_sript.sql", "w") as script_file:
        script_file.write("use covid_korea;")
        script_file.write(dim_age)
        script_file.write(dim_sex)
        script_file.write(dim_infection_site)
        script_file.write(dim_origin_country)
        script_file.write(dim_symptomatology)
        script_file.write(dim_case_state)
        script_file.write(dim_case_info)
        script_file.write(facts_patient_info)
```

Queries à BD

O passo final era interrogar a Base de Dados com um conjunto de *queries*, neste caso 2. A primeira *query* tem o objetivo de mostrar quais as 5 províncias de Coreia do Sul mais afetadas pelo Covid 19:

```
-- ------------------------------------------------------
-- Get the 5 provinces of Korea most affected by covid 19
-- ------------------------------------------------------
select province, count(facts.id) as `#cases`
from facts_patient_info as facts
    inner join dim_infection_site on id_dim_infection_site=dim_infection_site.id
group by
    province
order by
    `#cases` desc
limit 5;
```

Por outro lado, a segunda *query* tem o objetivo de listar os 10 perfis de pessoas (sexo e gama de idades) mais afetadas pelo Covid 19 na Coreia do Sul:

```
-- ------------------------------------------------------
-- Get the 10 profiles of people most affected by covid 19 in Korea
-- ------------------------------------------------------
select age_range, sex, count(facts.id) as `#cases`
from facts_patient_info as facts
    inner join dim_sex on id_dim_sex=dim_sex.id
    inner join dim_age on id_dim_age=dim_age.id
where
    age_range != 'N/A' and sex != 'N/A'
group by
    age_range,
    sex
order by
    `#cases` desc
limit 10;
```