Databases Project – Spring 2021

Team No: 65

Members: Zad ABI FADEL, Loïc Houmard, Jonas Blanc

Contents

Contents 1

Deliverable 1 2

Assumptions 2

Entity Relationship Schema 2

Schema 2

Description 2

Relational Schema 2

ER schema to Relational schema 2

DDL 2

General Comments 2

Deliverable 2 3

Assumptions 3

Data Loading/Cleaning 3

Query Implementation 3

General Comments 3

Deliverable 3 4

Assumptions 4

Query Implementation 4

Query Performance Analysis – Indexing 4

General Comments 4

# Deliverable 1

## Assumptions

On Identification:

Every party number should be unique within a collision. Every party\_id, victim\_id, case\_id should be unique by its own within the corresponding .csv files.

On data:

We assumed that in the .csv files every field would be represented by its key or that we would make it so during the data cleaning phase. We assumed that every description could fit in 150 char. We assumed based on data that party\_id, victim\_id and case\_id can be typed as integer.

On integrity:

Every victim should be associated with an unique party. Every party should be implicated in a unique collision.

## Entity Relationship Schema

### Schema

Diagram

Description automatically generated

### Description

For the ER diagram, we first decided to divide the attributes into 3 main entities called Victim, Party and Collision, because it seemed to us that they were the main actors in the model.

Then, we saw that it didn't make much sense to have only these 3 entities, because some attributes wouldn't be logically attributed to them. For example, it wouldn't make sense that a collision has an attribute population, because they are not directly correlated. Therefore, we tried to group attributes that logically belonged to a common idea together (star schema). For the collisions, we saw that there were many attributes related to the location of the collision, the conditions under which the collision happened and the legal part related to the collision. For the parties, many attributes were related to the vehicle. Hence, we wanted to add these 4 entities to our diagram (but finally modified it slightly, see below).

Also, after we spoke with some assistants, we realised that it would be a good idea to create entities for attributes that are lists with some finite non-logically predefined values (A:..., B:...). The reasons are the following: it would be easier to enforce the data we store to be cleaned and in the same format (it avoids to have one time 'a' and one time 'A' referencing to the same value) and it would make it more modulable and easier to change (if we realize that we would like to add/remove an option, we could simply add/remove one row in the table of the entity and add/invalidate these entries in the other table).

When there were many times the same attribute in the csv files (...\\_1 and ...\\_2), we also decided to create an entity. This has the advantage to be more modulable, since we could decide to add a third (...\\_3) attribute or even more of them in the future if we would like to slightly change the model. For that, we simply allowed the relation to have many of these new entities.

Finally, when we wanted to merge all our previous ideas together to construct the diagram, we found that creating the 4 entities mentioned above was not really practical because we would have to create these entities which now have no (or not many) attributes (since their corresponding attributes were often lists which we now model with an entity and bind through a relation), which makes them almost useless and increases the complexity of the diagram. Therefore, we decided to create N-ary relations directly to group the collision and all the attributes related to a given theme. This seems easier to understand and will create the same result in the database (since every attribute will finally be stored in the Collision table after the merging due to the many-to-one relation) when we translate it from the ER model to the SQL DDL commands.

## Relational Schema

### ER schema to Relational schema

Diagram

Description automatically generated

### DDL

|  |
| --- |
| *------------------------Design implementations--------------------*  *-- Boolean => char(1)*  *-- definition => varchar(150)*  *-- Table\_name (First letter upper case then underscores)*  *-- One-to-Many (Store key in one)*  *-- No state is null, set key to null*  *-- In an entity: id is id of current entity, create new attribute table\_id for referenced id*  *--Questions*  *--victim age/ pregnancy: age of 999 implies that person is not yet born, so that we don't lose information about the age of the mother*  *-- there would be 2 distinct victims (mother normal age, and yet to be born child age 99)*  *--in Victims: attribute victim\_seating\_position\_id || seating\_position\_id*  *-- merge state: Unknown with blank ? => key == null ?*  *-- Used in both victims and parties*  *--Check for line between collisions and is\_implied in*  *--Check if attributes of is\_implied\_in are done correctly*  *--Pcf\_violation\_subsection: which type?*  *--County\_city\_location: which type?*    *-------------------------Conditions start-------------------------*  **CREATE TABLE** Weather  (  **id         char**(1), *-- check if if is one of letter*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Road\_surface  (  **id         char**(1), *-- check if if is one of letter*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Road\_condition  (  **id         char**(1), *-- check if if is one of letter*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Lighting  (  **id         char**(1), *-- check if if is one of letter*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE Condition**  (  **id                int**,  **road\_surface\_id   char**(1) **references** Road\_surface (**id**),  **lighting\_id       char**(1) **references** Lighting (**id**),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Condition\_with\_weather  (  **condition\_id           int     references Condition** (**id**),  **wheather\_id            char**(1) **references** Weather (**id**),  **PRIMARY KEY** (**condition\_id**, **wheather\_id**)  );  **CREATE TABLE** Condition\_with\_road\_condition  (  **condition\_id           int     references Condition** (**id**),  **road\_condition\_id      char**(1) **references** Road\_condition (**id**),  **PRIMARY KEY** (**condition\_id**, **road\_condition\_id**)  );  *-------------------------Conditions end---------------------------*  *-------------------------Collisions start-------------------------*  **CREATE TABLE** Type\_of\_collision  (  **id         char**(1), *--check char between a & h*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Collision\_severity  (  **id         int CHECK** (0 <= **id and id** <= 4),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Hit\_and\_run  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Primary\_collision\_factor  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Pcf\_violation\_category  (  **id         int CHECK** ((0 <= **id and id** <= 24)),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Ramp\_intersection  (  **id         int CHECK** (1 <= **id and id** <= 8),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Location\_type  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Population  (  **id         int CHECK** (0 <= **id and id** <= 9),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Collisions  (  **case\_id                     int**,  **collision\_date              date**,  **collision\_time              timestamp**,  **tow\_away                    char**(1) **CHECK** (**tow\_away** = **'Y' or tow\_away** = **'N'**),  **type\_of\_collision\_id        char**(1) **references** Type\_of\_collision (**id**),  **collision\_severity\_id       int not null references** Collision\_severity (**id**),  *-- Relations is\_judged*  **jurisdiction                int CHECK** (0<=**jurisdiction and jurisdiction** <= 9999),  **officer\_id                  int**,  **pcf\_violation               int**,  **pcf\_violation\_subsection    varchar**(150),  **process\_date                date**,  **hit\_and\_run\_id              char**(1) **references** Hit\_and\_run (**id**),  **primary\_collision\_factor\_id char**(1) **references** Primary\_collision\_factor (**id**),  **pcf\_violation\_category\_id   int references** Pcf\_violation\_category (**id**),  *-- Relations happens\_in*  **county\_city\_location        int**,  **ramp\_intersection\_id        int references** Ramp\_intersection (**id**),  **location\_type\_id            char**(1) **references** Location\_type (**id**),  **population\_id               int references** Population (**id**),  *-- Relations happens\_under*  **condition\_id                int references Condition** (**id**),  **PRIMARY KEY** (**case\_id**)  );  *-------------------------Collisions end---------------------------*  **CREATE TABLE** Safety\_equipment  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  *----------------------------Victims start-------------------------*  **CREATE TABLE** Victim\_degree\_of\_injury  (  **id         int CHECK** (0 <= **id and id** <= 7), *-- can we make sure id and def are consistent*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Victim\_seating\_position  (  **id         char**(1), *--can we check if id is number or char?*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Victim\_role  (  **id         int CHECK** (1 <= **id and id** <= 6),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Victim\_ejected  (  **id         int CHECK** (0 <= **id and id** <= 3), *--make sure entity is still created if id is null*  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Victims  (  **id                         int**,  **pregnant                   char**(1) **not null**,  **victim\_age                 int**,  **victim\_sex                 char**(1),  *--- referenced ids--*  **party\_id                   int     not null**,  **victim\_degree\_of\_injury\_id int     not null references** Victim\_degree\_of\_injury (**id**),  **victim\_seating\_position\_id char**(1) **references** Victim\_seating\_position (**id**),  **victim\_role\_id             int     not null references** Victim\_role (**id**),  **victim\_ejected\_id          int references** Victim\_ejected (**id**),  *--     party\_id int not null REFERENCES PARTICIPANT (party\_id),*  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Victim\_equiped\_with\_safety\_equipment  (  **victim\_id           int     references** Victims (**id**),  **safety\_equipment\_id char**(1) **not null references** Safety\_equipment (**id**),  **PRIMARY KEY** (**victim\_id**, **safety\_equipment\_id**)  );  *----------------------------Victims end---------------------------*  *----------------------------Parties start-------------------------*  *-- Related entities with party: one to many*  **CREATE TABLE** Movement\_preceding\_collision  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Party\_drug\_physical  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Party\_sobriety  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Party\_type  (  **id         int**,  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Statewide\_vehicle\_type  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Cellphone\_use  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  *-- Relations with party: Many to many*  **CREATE TABLE** Other\_associated\_factor  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Financial\_responsibility  (  **id         char**(1),  **definition varchar**(150),  **PRIMARY KEY** (**id**)  );  *-- Parties*  **CREATE TABLE** Parties  (  **id                              int**,  *-- Atributes*  **hazardous\_materials             char**(1),  **party\_age                       int**,  **party\_sex                       char**(1),  *-- relation to collision*  **collision\_case\_id               int     not null references** Collisions (**case\_id**),  **financial\_responsibility\_id     char**(1) **references** Financial\_responsibility (**id**),  **school\_bus\_related              char**(1) **not null**,  **at\_fault                        char**(1) **not null**,  *-- referenced ids*  **movement\_preceding\_collision\_id char**(1) **references** Movement\_preceding\_collision (**id**),  **party\_drug\_physical\_id          char**(1) **references** Party\_drug\_physical (**id**),  **party\_sobriety\_id               char**(1) **references** Party\_sobriety (**id**),  **party\_type\_id                   int references** Party\_type (**id**),  **statewide\_vehicle\_type\_id       char**(1) **references** Statewide\_vehicle\_type (**id**),  **vehicle\_make                    varchar**(150),  **vehicle\_year                    int**,  **cellphone\_use\_id                char**(1) **references** Cellphone\_use (**id**),  *-- key*  **PRIMARY KEY** (**id**)  );  **CREATE TABLE** Party\_equiped\_with\_safety\_equipment  (  **party\_id            int     not null references** Parties (**id**),  **safety\_equipment\_id char**(1) **not null references** Safety\_equipment (**id**),  **PRIMARY KEY** (**party\_id**, **safety\_equipment\_id**)  );  **CREATE TABLE** Party\_associated\_with\_safety\_other\_associated\_factor  (  **party\_id                   int     not null references** Parties (**id**),  **other\_associated\_factor\_id char**(1) **not null references** Other\_associated\_factor (**id**),  **PRIMARY KEY** (**party\_id**, **other\_associated\_factor\_id**)  );  *------------------------------Parties end-------------------------* |

## General Comments

In general, we found it pretty hard to create the ER diagram at first because there were a lot of attributes to proceed and understand and also because we didn’t have much experience with this kind of work. But after having spent some time, we think that our implementation is now logical and should allow us to retrieve the information without having too many problems.

The allocation between the members was good, since we almost always worked together as a team. We first all took part in the elaboration of the ER diagram by concentrating us each on a CSV file and then talking with each other to see which attributes could belong together.  
We then all wrote some of the SQL DDL commands to create the tables and wrote the report together.

# Deliverable 2

## Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Data Loading/Cleaning

## Query Implementation

<For each query>

**Query a:**

***Description of logic:***

<What does the query do and how do I decide to solve it>

***SQL statement***

<The SQL statement>

***Query result (if the result is big, just a snippet)***

<The SQL statement result>

## General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>

# Deliverable 3

# Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Query Implementation

<For each query>

**Query a:**

***Description of logic:***

<What does the query do and how do I decide to solve it>

***SQL statement***

<The SQL statement>

***Query result (if the result is big, just a snippet)***

<The SQL statement result>

## Query Performance Analysis – Indexing

<In this section, for 6 selected queries explain in detail why do you see given improvements (or not). For example, why building an index on certain field changed the plan and IO.>

**Query 1**

<Initial Running time/IO:

Optimized Running time/IO:

Explain the improvement:

Initial plan

Improved plan>

**Query 2**

<Initial Running time/IO:

Optimized Running time/IO:

Explain the improvement:

Initial plan

Improved plan>

# General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>