Databases Project – Spring 2021

Team No:

Members:

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 3

# Changes

Based on the feedback, we have made some changes including ER model, DDL, data processing and query implementation of Deliverable 2.

## **ER MODEL**

Removed the 'contition' entity and moved all its attributes to the 'case' entity. Changed the attributes of 'road\_en' entity into 'case\_id' and 'road\_con', the pairs of them serve as primary keys. Changed the attributes of 'other\_fac\_en' entity into 'party\_id' and 'other\_fac', both of them serve as primary keys. Replaced the attribute 've\_num' of 'vehicle' entity by 'party\_id'. Changed the key of the 'vehicle' entity to all its attributes. Separated the 'safety\_equip\_en' entity into two entities 'safety\_equip\_en\_par' (the safety equipment for party) and 'safety\_equip\_en\_vic' (the safety equipment for victim'). For 'safety\_equip\_en\_par', its attributes are changed to 'safety\_equip' and 'party\_id', both of them serve as primary keys. For 'safety\_equip\_en\_vic', its attributes are changed to 'safety\_equip' and 'vic\_id' both of them serve as primary keys.

**(safety\_equipment助教建议我们设两个表，但貌似肯定了只是一个entity: "It is indeed correct to have a single "safety\_equipment" entity in the ER, however, during the translation I think you would need two separate tables: safetey\_equipment for party and victim.")**

**[ER MODEL] 还没放上来！**

## **DDL**

We deleted the independent tables for weather, other\_fac\_en, safety\_equipment, road\_en, and merged them to the relational table such as under\_w, under\_r, have, have\_ps, have\_vs. Now in the relational table, such conditions (weather, road condition, other\_fac, safety\_equip, road\_con) will be like an attribute and the case\_id/party\_id/victim\_id will act as a foreign key referred from the corresponding table. We use the pairs of (xx\_id, condition) as the primary key to ensure that they are unique. In these tables (other\_fac\_en, road\_en, weather\_en, safety\_p, safety\_v), we add “on delete cascade” constraint on the foreign key, because once the corresponding party/case/victim is deleted from the database, the record related to them would also be deleted.

We deleted the relational table between party\_involve-take-vehicle, but just added a column “ve\_num” in the party\_involve table referring to the “ve\_num” of vehicle table. We also deleted the table between PCF-casue-case and added a column “pcf\_num” in the case table referring to the “pcf\_num” of PCF table.

We added “on delete cascade” constraint on the “party\_id” foreign key in the associate\_victim table, and the “case\_id” foreign key in the party\_involve table, because party is the weak entity of the case, and the victim if the weak entity of the party (by our assumption and data processing).

We changed the type of “case\_id” to VARCHAR2, because there may be several zeros in front of the number, which is also a part of the identity.

We have 11 tables now, and the DDL is attached below. Modified part is marked.

===========================================================================================

CREATE TABLE PCF(

pcf\_num INTEGER,

pcf\_violation\_code INTEGER,

pcf\_violation\_category VARCHAR2(50),

subsection VARCHAR2(3),

pcf\_type VARCHAR2(50),

PRIMARY KEY (pcf\_num)

);

CREATE TABLE Vehicle(

ve\_num INTEGER,

ve\_type VARCHAR2(50),

ve\_make VARCHAR2(30),

ve\_year INTEGER,

school\_bus\_rel VARCHAR2(5),

PRIMARY KEY (ve\_num)

);

CREATE TABLE Location(

loc\_num INTEGER,

population INTEGER,

county\_city INTEGER,

loc\_type VARCHAR2(20),

ramp\_int VARCHAR2(10),

PRIMARY KEY (loc\_num)

);

CREATE TABLE Case(

case\_id VARCHAR2(30),

loc\_num INTEGER NOT NULL,

pcf\_num INTEGER NOT NULL,

col\_severity VARCHAR2(30),

col\_time DATE,

col\_date DATE,

hit\_run VARCHAR2(30),

jurisdiction INTEGER,

officer\_id VARCHAR2(10),

process\_date DATE,

tow\_away INTEGER,

col\_type VARCHAR2(30),

lighting VARCHAR2(50),

road\_surf VARCHAR2(10),

PRIMARY KEY (case\_id),

FOREIGN KEY (loc\_num) REFERENCES Location(loc\_num),

FOREIGN KEY(pcf\_num) REFERENCES PCF(pcf\_num)

);

CREATE TABLE Party\_involve(

party\_id INTEGER,

case\_id VARCHAR2(30) NOT NULL,

at\_fault INTEGER,

phone VARCHAR2(3),

fin\_resp VARCHAR2(3),

haz\_mat VARCHAR2(3),

move\_pre VARCHAR2(30),

age INTEGER,

drug\_phy VARCHAR2(3),

sobriety VARCHAR2(3),

party\_type VARCHAR2(15),

party\_num INTEGER,

sex VARCHAR2(6),

ve\_num INTEGER,

PRIMARY KEY (party\_id),

FOREIGN KEY (case\_id) REFERENCES Case(case\_id)

ON DELETE CASCADE,

FOREIGN KEY (ve\_num) REFERENCES Vehicle(ve\_num)

);

CREATE TABLE Associate\_victim(

vic\_id INTEGER,

party\_id INTEGER NOT NULL,

vic\_age INTEGER,

ejected INTEGER,

vic\_role INTEGER,

deg\_injury VARCHAR2(50),

vic\_seat INTEGER,

vic\_sex VARCHAR2(6),

PRIMARY KEY (vic\_id),

FOREIGN KEY (party\_id) REFERENCES Party\_involve(party\_id)

ON DELETE CASCADE

);

CREATE TABLE Other\_fac\_en(

party\_id INTEGER,

other\_fac VARCHAR2(3),

PRIMARY KEY (party\_id, other\_fac),

FOREIGN KEY (party\_id) REFERENCES Party\_involve(party\_id)

ON DELETE CASCADE

);

CREATE TABLE road\_en(

case\_id VARCHAR2(30),

road\_con VARCHAR2(20),

PRIMARY KEY (case\_id, road\_con),

FOREIGN KEY (case\_id) REFERENCES Case(case\_id)

ON DELETE CASCADE

);

CREATE TABLE Weather\_en(

case\_id VARCHAR2(30),

weather VARCHAR2(20),

PRIMARY KEY (case\_id, weather),

FOREIGN KEY (case\_id) REFERENCES Case(case\_id)

ON DELETE CASCADE

);

CREATE TABLE safety\_p(

party\_id INTEGER,

safety\_equip VARCHAR2(3),

PRIMARY KEY (party\_id, safety\_equip),

FOREIGN KEY (party\_id) REFERENCES Party\_involve(party\_id)

ON DELETE CASCADE

);

CREATE TABLE safety\_v(

vic\_id INTEGER,

safety\_equip VARCHAR2(3),

PRIMARY KEY (vic\_id, safety\_equip),

FOREIGN KEY (vic\_id) REFERENCES Associate\_victim(vic\_id)

ON DELETE CASCADE

);

===========================================================================================

## **Query Implementation of Deliverable 2**

**Query 1:**

***Description of logic:***

List the year-number of collisions per year. We use “group by” to group case by year (extracted from col\_date) and count the number of cases of each year.

***SQL statement***

SELECT EXTRACT (YEAR FROM col\_date) AS YEAR, count(\*) AS N\_collisions

FROM case

GROUP BY EXTRACT (YEAR FROM col\_date)

ORDER BY YEAR ASC

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

|  |  |
| --- | --- |
| YEAR | N\_COLLISIONS |
| 2001 | 522562 |
| 2002 | 544741 |
| 2003 | 538954 |
| 2004 | 538295 |
| 2005 | 532725 |
| 2006 | 498850 |
| 2007 | 501908 |
| 2017 | 7 |
| 2018 | 21 |

**Query 2:**

***Description of logic:***

In the “take” table, group entries by “ve\_make” and count the number of parties of each “ve\_make”, then find the max count and the corresponding “ve\_make”. Before that we need to use “ve\_number” to know the “ve\_make”, so we first join table “vehicle” and “party\_involve”. To illustrate the whole row of the most popular, we fetch first 1 row only.

***SQL statement***

SELECT VE\_MAKE, COUNT(\*) AS N\_COLLISION

FROM (VEHICLE INNER JOIN PARTY\_INVOLVE ON VEHICLE.VE\_NUM = PARTY\_INVOLVE.VE\_NUM)

GROUP BY VE\_MAKE

ORDER BY N\_COLLISION DESC

FETCH FIRST 1 ROWS ONLY

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

|  |  |
| --- | --- |
| VE\_MAKE | N\_VEHICLE |
| FORD | 1129701 |

**Query 3:**

***Description of logic:***

In the lighting attribute of condition, find the description that contains “dark”, and count the fraction of cases that occur in such condition, and and keep 2 significant digits..

***SQL statement***

SELECT ROUND(NOM/(SELECT COUNT(\*) FROM CASE),2)

FROM

(SELECT COUNT(\*) AS NOM

FROM CASE

WHERE CASE.LIGHTING LIKE '%dark%')

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

|  |
| --- |
| FRACTION |
| 0.28 |

**Query 4:**

***Description of logic:***

Find the number of collisions that have occurred under snowy weather. We count the number of entries that have weather\_con = ‘snowing’ in the table “weather\_en”

***SQL statement***

SELECT COUNT(\*)

FROM WEATHER\_EN

WHERE WEATHER LIKE '%snowing%'

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

|  |
| --- |
| N\_COLLISIONS |
| 8530 |

**Query 5:**

***Description of logic:***

Group by collisions by which day they are during a week, and count the total number of collisions of that day, then find the row of highest number of cases. We use TO\_CHAR (COL\_DATE, 'D') to extract the day of the week.

***SQL statement***

SELECT TO\_CHAR(COL\_DATE, 'D') AS WEEK\_DAY, COUNT(\*) AS N\_COLLISONS

FROM CASE

GROUP BY TO\_CHAR(COL\_DATE, 'D')

ORDER BY N\_COLLISONS DESC

FETCH FIRST 1 ROWS ONLY

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

|  |  |
| --- | --- |
| WEEK\_DAY | N\_COLLISIONS |
| 6 | 614853 |

**Query 6:**

***Description of logic:***

List all weather types and their corresponding number of collisions in ascending order of the collisions.We group cases by “weather” and list “weather” and the count number.

***SQL statement***

SELECT WEATHER, COUNT(\*) AS N\_COLLISION

FROM WEATHER\_EN

GROUP BY WEATHER

ORDER BY N\_COLLISION

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

|  |  |
| --- | --- |
| WEATHER | N\_COLLISION |
| other | 6960 |
| snowing | 8530 |
| wind | 13952 |
| fog | 21259 |
| raining | 223752 |
| cloudy | 548250 |
| clear | 2941042 |

**Query 7:**

***Description of logic:***

Count the number of parties that are at-fault, with financial responsibility and loose material. We first extract the “road\_num” of “road\_loose”, and find which parties are associated with such road condition. We filter the “party\_id” table who is at fault and with financial responsibility. Finally we count the number of the selected parties.

***SQL statement***

SELECT COUNT(\*) AS N\_PARTIES

FROM PARTY\_INVOLVE P, ROAD\_EN R

WHERE P.CASE\_ID = R.CASE\_ID AND P.AT\_FAULT = 1 AND P.FIN\_RESP = 'Y' AND R.ROAD\_CON LIKE '%loose material%'

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

|  |
| --- |
| N\_PARTIES |
| 4803 |

**Query 8:**

***Description of logic:***

Find the median victim age: we directly use the “MEDIAN” function of SQL from the associate\_victim table.

Find the most common victim seating position. We group the victims with seating position, and count the number of victims of each vic\_seat, order them in the descending order of this number and find the max.

***SQL statement***

***8.a***

SELECT MEDIAN(vic\_age) AS MEDIAN\_VIC\_AGE

FROM ASSOCIATE\_VICTIM v2;

***8.b***

SELECT VIC\_SEAT AS MOST\_COMMON\_SEAT\_POSITION

FROM

(SELECT COUNT(vic\_seat) AS count, vic\_seat

FROM associate\_victim v2

GROUP BY vic\_seat

ORDER BY count DESC)

FETCH FIRST 1 ROWS ONLY;

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

***8.a***

|  |
| --- |
| MEDIAN\_VIC\_AGE |
| 25 |

***8.b***

|  |
| --- |
| MOST\_COMMON\_SEAT\_POSITION |
| 3 |

**Query 9:**

***Description of logic:***

Fraction of all participants (victims + parties) that have been victims using a belt. All participants refer to both parties and victims, so our denominator is the sum of number of all victims and parties. We first extract the ‘vic\_id’s who use belt using table have\_vs and safety\_equip\_en. Then we count the unique ‘vic\_id’s and use this number as the numerator. Finally we get the fraction and keep 2 significant digits.

***SQL statement***

SELECT ROUND(A.FRACTION,3) AS fraction

FROM(SELECT DISTINCT

(SELECT COUNT(vic\_id) AS count

FROM

(SELECT h1.vic\_id as vic\_id

FROM SAFETY\_V H1

WHERE H1.SAFETY\_EQUIP like '%C%') v\_belt)/

((SELECT COUNT(party\_id) FROM party\_involve)

+(SELECT COUNT(vic\_id) FROM associate\_victim)) as fraction

FROM party\_involve) a

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

|  |
| --- |
| FRACTION |
| 0.011 |

**Query 10:**

***Description of logic:***

Compute the fraction of collisions happening for each hour of the day, and display as ratio as fraction for all the hours of the day. We first use cast(col\_time as timestamp) to extract the hour in which the case occurred. Then we group the cases by the specific hour and count the number of the cases, then order them by the number. We also calculate the total number of the cases. Then we divide the count number of each hour by the total number to get each fraction.

***SQL statement***

SELECT DISTINCT

EXTRACT(hour from cast(col\_time as timestamp)) as hour, ROUND((COUNT(\*)/(SELECT COUNT(\*) FROM CASE)),3) as FRACTION

FROM CASE

GROUP BY EXTRACT(hour from cast(col\_time as timestamp))

ORDER BY hour ASC

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



## **Data processing**

For identifiers of pcf\_num, ve\_num and loc\_num (now there is no con\_num), we used Pandas of Python to add index on the dataframes (reset\_index(drop=True)). Thus the indexes are all natural numbers and unique.

Case\_id, party\_id and victim\_id do not have null value. For road\_en, we first extracted all distinct road\_en values and dropped the null value, then join with the 2 columns of road condition in the collision2018.csv to get the not null (case\_id, road\_con) pairs, and dropped the duplicates. In this way if a collision’s road condition is unavailable, it will not appear in the road\_en table. The same method also works for weather\_en, other\_fac\_en, safety\_p and safety\_v. For vehicle, we first extracted all the distinct tuples of their attributes, and we did not drop the null values, so there will not be null value in the party\_involve table for ve\_num, but there is a row of “ve\_num|null|null|null|null” is the vehicle table. It is the same for location and PCF table.

## Assumptions

The case\_id is of the type String, because there are zeros in front of the number. If we use it as String, there will be no problem of duplicated case\_id. And it shows that “case\_id+party\_num” = unique party\_id

Each case happens under exactly one location and PCF. Each party drive one kind of

## 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>