**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

# Databases Project – Spring 2021

Team No: 71

Members: Culic Iana, Raita Omar, Zakariya Taha

## Contents

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

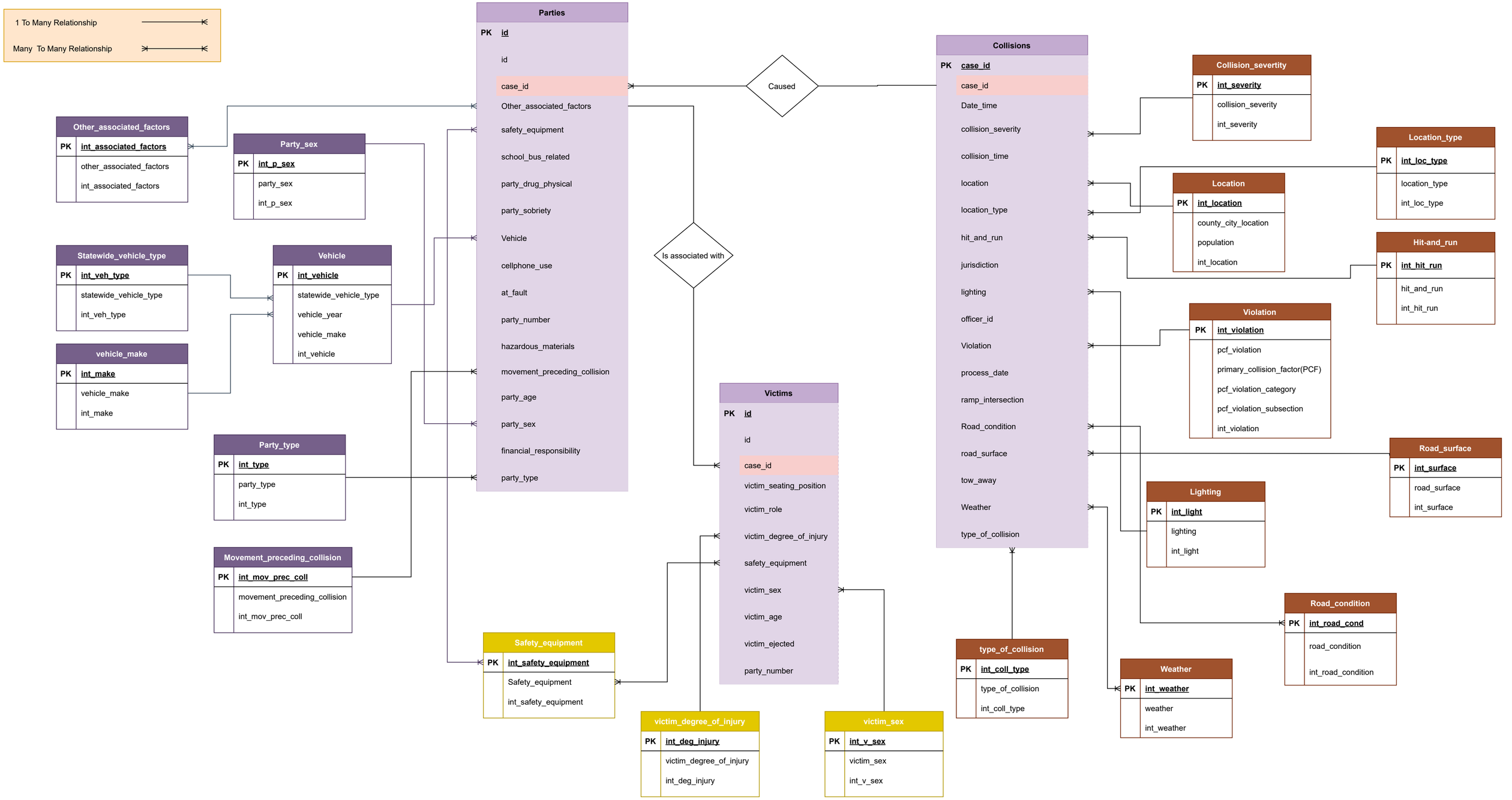
# Deliverable 1

## *Assumptions*

- A star schema was used as the main building block in the ER diagram.
- Both Victims and Parties share the same "safety_equipment" table as the initial tables "party_safety_equipment1/2" and "victims_safety_equipment1/2" contain the same data.
- A dictionary encoding was used for some attributes to enhance the performance.
- A 1 to many relationship was used in the dictionary encoding of simple attributes such as Lighting, Party_type, Vehicle_make.
- A many to many relationship was used in the dictionary encoding of composed attributes such as weather 1 /2, road_condtion 1 /2 and safety equipment 1 /2.

## *Entity Relationship Schema*

Schema


1 To Many Relationship
Many To Many Relationship
Parties
PK id
id
case_id
Other_associated_factors
safety_equipment
school_bus_related
party_drug_physical
party_sobriety
Vehicle
cellphone_use
at_fault
party_number
hazardous_materials
movement_preceding_collision
party_age
party_sex
financial_responsibility
party_type
Caused
Is associated with
Collisions
PK case_id
case_id
Date_time
collision_severity
collision_time
location
location_type
hit_and_run
jurisdiction
lighting
officer_id
Violation
process_date
ramp_intersection
Road_condition
road_surface
tow_away
Weather
type_of_collision
Collision_severtity
PK int_severity
collision_severity
int_severity
Location_type
PK int_loc_type
location_type
int_loc_type
Location
PK int_location
county_city_location
population
int_location
Hit-and-run
PK int_hit_run
hit_and_run
int_hit_run
Violation
PK int_violation
pcf_violation
primary_collision_factor(PCF)
pcf_violation_category
pcf_violation_subsection
int_violation
Road_surface
PK int_surface
road_surface
int_surface
Lighting
PK int_light
lighting
int_light
Road_condition
PK int_road_cond
road_condition
int_road_condition
Weather
PK int_weather
weather
int_weather
type_of_collision
PK int_coll_type
type_of_collision
int_coll_type
Other_associated_factors
PK int_associated_factors
other_associated_factors
int_associated_factors
Party_sex
PK int_p_sex
party_sex
int_p_sex
Statewide_vehicle_type
PK int_veh_type
statewide_vehicle_type
int_veh_type
Vehicle
PK int_vehicle
statewide_vehicle_type
vehicle_year
vehicle_make
int_vehicle
vehicle_make
PK int_make
vehicle_make
int_make
Party_type
PK int_type
party_type
int_type
Movement_preceding_collision
PK int_mov_prec_coll
movement_preceding_collision
int_mov_prec_coll
Victims
PK id
id
case_id
victim_seating_position
victim_role
victim_degree_of_injury
safety_equipment
victim_sex
victim_age
victim_ejected
party_number
Safety_equipment
PK int_safety_equipment
Safety_equipment
int_safety_equipment
victim_degree_of_injury
PK int_deg_injury
victim_degree_of_injury
int_deg_injury
victim_sex
PK int_v_sex
victim_sex
int_v_sex

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

## Description

We chose to keep the same raw tables that are provided in the project description as the 3 main entities in our ER diagram (parties, collisions, and victims). Following the idea of a star schema, we created new dimension tables from some attributes of the main entities.

For efficiency purposes, we decided to create new entities in the following situations:

- Attributes that contain a small number of unique combinations (e.g. “Vehicle”, “Location” and “Violation”)
- Redundant attributes such as weather_1 and weather_2, that have been merged into “weather”.
- Storing integers instead of strings for attributes that typically have a small number of unique instances (e.g., “Satetwide_vehicle_type”, “vehicle_make” and “primary_collision_factor”.

The relationship between a fact table and its corresponding dimension table is:

“One to many” for simple attributes, as we may have many instances in the fact table who share the same data in the dimension table. The relationship maps each unique tuple in the dimension table to its occurrences in the fact table.

“Many to many” for composed attributes such as weather 1 /2, road_condtion 1 /2 and safety equipment 1 /2. The idea is to create a new relationship table (e.g Weather_related) that maps each primary key in the fact table to a tuple (weather_1, weather_2) in the dictionary table.

Regarding the 3 fact tables, we chose to represent the relationship between Collisions and Parties as a one to many relationship since one collision can be caused by multiple parties. The same applies to the relationship between Parties and Victims since there might be multiple victims that are associated with each party.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

# *Relational Schema*

## ER schema to Relational schema

Using the ER schema proposed in the previous part we are proposing this Relational schema describing it directly by the DDL bellow.

## DDL

In alphabetical order our DDLs are :

```
create table "Collisions"
(
   CASE_ID VARCHAR2(30) not null
         constraint COLLISIONS_PK
               primary key,
   "Date_Time" DATE,
   JURISDICTION NUMBER,
   OFFICER_ID VARCHAR2(10),
   PROCESS_DATE DATE,
   RAMP_INTERSECTION NUMBER,
   TOW_AWAY NUMBER,
   LOCATION_TYPE NUMBER
         constraint COLLISIONS_LOCATION_TYPE__FK
               references "Location_type"
                     on delete set null,
   LOCATION NUMBER
         constraint COLLISIONS_LOCATION__FK
               references "Location"
                     on delete set null,
   HIT_AND_RUN NUMBER
         constraint COLLISIONS_HIT_AND_RUN__FK
               references "Hit_and_run"
                     on delete set null,
   VIOLATION NUMBER
         constraint COLLISIONS_VIOLATION__FK
               references "Violation"
                     on delete set null,
   LIGHTING NUMBER
         constraint COLLISIONS_LIGHTING__FK
               references "Lighting"
                     on delete set null,
   ROAD_SURFACE NUMBER
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
        constraint COLLISIONS_ROAD_SURFACE__FK
                references "Road_surface"
                        on delete set null,
    TYPE_OF_COLLISION NUMBER
        constraint COLLISIONS_TYPE_OF_COLLISION__FK
                references "Type_of_collision"
                        on delete set null,
    COLLISION_SEVERITY NUMBER
        constraint COLLISIONS_COLLISION_SEVERITY__FK
                references "Collision_severity"
)

create table "Collision_severity"
(
    INT_SEVERITY NUMBER not null
        constraint COLLISION_SEVERITY_PK
                primary key,
    COLLISION_SEVERITY VARCHAR2(25)
)

create table "Hit_and_run"
(
    INT_HIT_AND_RUN NUMBER not null
        constraint HIT_AND_RUN_PK
                primary key (INT_HIT_AND_RUN),
    HIT_AND_RUN VARCHAR2(15)
)

create table "Lighting"
(
    INT_LIGHT NUMBER not null
        constraint LIGHTING_PK
         primary key (INT_LIGHT),
    LIGHTING VARCHAR2(40)
)

create table "Location"
(
    INT_LOCATION NUMBER
        constraint LOCATION_PK
         primary key (INT_LOCATION)
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
        not null,
    COUNTY_CITY_LOCATION NUMBER,
    POPULATION NUMBER
)

create table "Location_type"
(
    INT_LOC_TYPE NUMBER not null
        constraint LOCATION_TYPE_PK
         primary key (INT_LOC_TYPE),
    LOCATION_TYPE VARCHAR2(12)
)

create table "Movement_preceding_collision"
(
    INT_MOV_PREC_COLL NUMBER not null
        add constraint "MOVEMENT_PRECEDING_COLLISION(1)_PK"
         primary key (INT_MOV_PREC_COLL),
    MOVEMENT_PRECEDING_COLLISION VARCHAR2(26)
)

create table "Parties"
(
    AT_FAULT NUMBER,
    CASE_ID VARCHAR2(30) not null
        constraint PARTIES_PK
         primary key (ID),
    CELLPHONE_USE VARCHAR2(1),
    FINANCIAL_RESPONSIBILITY VARCHAR2(1),
    HAZARDOUS_MATERIALS VARCHAR2(1),
    ID NUMBER not null,
    PARTY_AGE NUMBER,
    PARTY_DRUG_PHYSICAL VARCHAR2(1),
    PARTY_NUMBER NUMBER,
    PARTY_SOBRIETY VARCHAR2(1),
    SCHOOL_BUS_RELATED VARCHAR2(1),
    MOVEMENT_PRECEDING_COLLISION NUMBER
        constraint PARTIES_MOVEMENT_PRECEDING_COLLISION__FK
            references "Movement_preceding_collision"
                on delete set null,
    PARTY_SEX NUMBER
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
        constraint PARTIES_PARTY_SEX__FK
                references "Party_sex"
                        on delete set null,
    PARTY_TYPE NUMBER
        constraint PARTIES_PARTY_TYPE__FK
                references "Party_type"
                        on delete set null,
    VEHICLE NUMBER
        constraint PARTIES_VEHICLE__FK
                references "Vehicle"
                        on delete set null
)

create table "Parties_factors_related"
(
    ID NUMBER
        constraint PARTIES_FACTORS_RELATED_PARTIES__FK
                references "Parties",
    OTHER_ASSOCIATED_FACTORS NUMBER
)

create table "Parties_safety_related"
(
    ID NUMBER
        constraint PARTIES_SAFETY_RELATED_PARTIES__FK
                references "Parties",
    PARTY_SAFETY_EQUIPMENT NUMBER
        constraint PARTIES_SAFETY_RELATED_SAFETY_EQUIPMENT__FK
                references "Safety_equipment"
)

create table "Party_sex"
(
    INT_P_SEX NUMBER not null
        constraint "PARTY_SEX(1)_PK"
                primary key (INT_P_SEX),
    PARTY_SEX VARCHAR2(6)
)

alter table "Party_sex"
    add constraint "PARTY_SEX(1)_PK"
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
            primary key (INT_P_SEX)

create table "Party_type"
(
   INT_TYPE NUMBER not null
            constraint "PARTY_TYPE(1)_PK"
            primary key (INT_TYPE),
   PARTY_TYPE VARCHAR2(14)
)

create table "Pcf_violation_category"
(
   INT_CAT NUMBER not null
            constraint PCF_VIOLATION_CATEGORY_PK
            primary key (INT_CAT),
   PCF_VIOLATION_CATEGORY VARCHAR2(35)
)

create table "Primary_collision_factor"
(
   INT_FACT NUMBER not null
            constraint PRIMARY_COLLISION_FACTOR_PK
            primary key (INT_FACT),
   PRIMARY_COLLISION_FACTOR VARCHAR2(30)
)

create table "Road_condition"
(
   INT_ROAD_CONDITION NUMBER not null
            constraint ROAD_CONDITION_PK
            primary key (INT_ROAD_CONDITION),
   ROAD_CONDITION VARCHAR2(15)
)

create table "Road_condition_related"
(
   CASE_ID VARCHAR2(30) not null
            constraint ROAD_CONDITION_RELATED_PK
            primary key (CASE_ID),
   ROAD_CONDITION NUMBER
            constraint ROAD_CONDITION_RELATED_ROAD_CONDITION__FK
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
                    references "Road_condition"
            constraint ROAD_CONDITION_RELATED_COLLISION__FK
                    references "Collision"
)

create table "Road_surface"
(
   INT_SURFACE NUMBER not null
            constraint ROAD_SURFACE_PK
            primary key (INT_SURFACE),
   ROAD_SURFACE VARCHAR2(8)
)

create table "Safety_equipment"
(
   INT_SAFETY_EQUIPMENT NUMBER not null

           constraint "SAFETY_EQUIPMENT(1)_PK"
            primary key (INT_SAFETY_EQUIPMENT),
   VICTIM_SAFETY_EQUIPMENT VARCHAR2(1)
)

create table "Statewide_vehicle_type"
(
   INT_VEH_TYPE NUMBER not null
           constraint "STATEWIDE_VEHICLE_TYPE(1)_PK"
            primary key (INT_VEH_TYPE),
   STATEWIDE_VEHICLE_TYPE VARCHAR2(34)
)

create table "Type_of_collision"
(
   INT_COLL_TYPE NUMBER not null
           constraint TYPE_OF_COLLISION_PK
            primary key (INT_COLL_TYPE),
   TYPE_OF_COLLISION VARCHAR2(10)
)

create table "Vehicle"
(
   INT_VEHICLE NUMBER not null
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
        constraint VEHICLE_MAKE_VEHICLE__FK
          references "Vehicle",
   VEHICLE_YEAR NUMBER,
   VEHICLE_MAKE NUMBER,
   STATEWIDE_VEHICLE_TYPE NUMBER
        constraint VEHICLE_STATEWIDE_VEHICLE_TYPE__FK
              references "Statewide_vehicle_type"
)

create table "Victim_degree_of_injury"
(
   INT_DEG_INJURY NUMBER not null
        constraint VICTIM_DEGREE_OF_INJURY_PK
         primary key (INT_DEG_INJURY),
   VICTIM_DEGREE_OF_INJURY VARCHAR2(20)
)

create table "Victims"
(
   CASE_ID VARCHAR2(30),
   ID NUMBER not null
        constraint VICTIMS_1_PK
          primary key (ID),
   PARTY_NUMBER NUMBER,
   VICTIM_AGE NUMBER,
   VICTIM_EJECTED NUMBER,
   VICTIM_ROLE NUMBER,
   VICTIM_SEATING_POSITION NUMBER,
   VICTIM_SEX NUMBER
        constraint VICTIMS_PARTY_SEX__FK
              references "Party_sex",
   VICTIM_DEGREE_OF_INJURY NUMBER
        constraint VICTIMS_VICTIM_DEGREE_OF_INJURY__FK
              references "Victim_degree_of_injury"
)


create table "Victims_safety_related"
(
   ID NUMBER
        constraint VICTIMS_SAFETY_RELATED_VICTIMS__FK
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
                references "Victims",
   VICTIM_SAFETY_EQUIPMENT NUMBER
         constraint VICTIMS_SAFETY_RELATED_SAFETY_EQUIPMENT__FK
                references "Safety_equipment"
)

create table "Violation"
(
   primary key (INT_VIOLATION),
   PCF_VIOLATION NUMBER,
   PRIMARY_COLLISION_FACTOR VARCHAR2(30),
   PCF_VIOLATION_CATEGORY VARCHAR2(35),
   PCF_VIOLATION_SUBSECTION VARCHAR2(1)
)

create table "Weather"
(
   primary key (INT_WEATHER),
   WEATHER VARCHAR2(10)
)


create table "Weather_related"
(
   CASE_ID VARCHAR2(30),
   WEATHER NUMBER
         WEATHER_RELATED
          constraint WEATHER_RELATED_WEATHER_FK
                references "Weather"
         WEATHER_RELATED
          constraint WEATHER_RELATED_COLLISION_FK
                references "Collision"


)
```

## *General Comments*

There are many foreign keys relations as we decided to use dictionary encoding. There are also many foreign keys constraints.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

# Deliverable 2

## *Assumptions*

We kept the same assumptions as for Deliverable 1 except for Collision_date and Collision_time that were moved back to the Collision table as we couldn't upload them to DataGrip in the correct format. We also merged them into a single column Date_Time.

## *Data Loading/Cleaning*

The data cleaning and preparation has been done in Python.

**Cleaning:**

Hit_and_run contains a value D with 1 occurrence. We decided to remove the corresponding line as there are no indications on what it means in the project description.

Lines with Null value in Time have been automatically removed by oracle during the data loading since it does not satisfy the date format.

**Tables creation:**

After loading the raw tables (collisions2018, parties2018, victims2018) as data frames using the python pandas package, we created the dictionaries in the following way:

1. Select the specific columns (attributes of the new entity) from the raw table:

```
location = df_col[['county_city_location', 'population']]
```

2. Use the drop_duplicates() method to keep only the unique combinations of the attributes.
3. Use the reset_index() method to reset the indices (1 to n) instead of randomly shuffled indices:

```
location = location.drop_duplicates().reset_index(drop=True).reset_index()
```

4. Rename the index column to the primary key name of the corresponding entity:

```
.rename(columns = {'index':'int_location'})
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

5. Save the table to Entity.csv:

```
location.to_csv('Location.csv', index=False)
```

6. Merge the raw data frame with the entity data frame on corresponding columns. This method will add an extra column 'int_location' to the raw data frame:

```
df = pd.merge(df,location, left_on=['county_city_location', 'population'], right_on =
['county_city_location', 'population'])
```

7. Drop the columns that belong to the new entity from the raw table:

```
df = df.drop(columns=['county_city_location', 'population'], inplace=False)
```

8. Rename the added column:

```
df = df.rename(columns={"int_location": "location"}, inplace=False)
```

9. Repeat this process for all the tables that require a dictionary encoding.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

# *Query Implementation*

**Query 1:**

***Description of logic:***

We have a separate entity for Collisions and so the query extracts the year from Date_Time and then we order the results by year and order in Descending order.

***SQL statement***

```
SELECT  DISTINCT EXTRACT(YEAR FROM "Date_Time") Years, COUNT(*) Nb_collisions
FROM "Collisions" COL
GROUP BY  EXTRACT(YEAR FROM "Date_Time")
ORDER BY Years DESC;
```

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

| | YEARS | NB_COLLISIONS |
|---|---|---|
| 1 | 2018 | 21 |
| 2 | 2017 | 7 |
| 3 | 2007 | 492964 |
| 4 | 2006 | 490456 |
| 5 | 2005 | 523568 |
| 6 | 2004 | 530210 |
| 7 | 2003 | 532630 |
| 8 | 2002 | 537218 |
| 9 | 2001 | 513196 |

**Query 2:**

***Description of logic:***

As a vehicle is a separate entity, which works as a dictionary including all types of vehicles, and as wel we havel a dictionary for the Vehicle make. We need to query Vehicle, Parties and Vehicle_make. As Vehicle contains one line for each car type for example ( Ford F150 is one line) we need to query Parties as well as to count each instance.

***SQL statement***

```
SELECT  M.VEHICLE_MAKE, COUNT(*)
FROM "Vehicle" V, "Parties" P , "Vehicle_make" M
Where V.INT_VEHICLE = P.VEHICLE AND V.VEHICLE_MAKE=M.INT_MAKE
GROUP BY  M.VEHICLE_MAKE
ORDER BY 2 DESC
fetch first row only;
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

***Query result (if the result is big, just a snippet)***
FORD with 1129701 cars.

**Query 3:**
***Description of logic:***
In our conception Collisions have a dictionary encoding for Lightning. We need to query collisions with the result of L.LIGHTING LIKE '%dark%' in the nested query and then divide by total instances which we get with MAX(ROWNUM).
***SQL statement***

```
SELECT T.count/MAX(ROWNUM)
FROM (
      SELECT COUNT(*) count
      FROM "Collisions" C,
          "Lighting" L
      WHERE C.LIGHTING = L.INT_LIGHT
        and L.LIGHTING LIKE '%dark%'
   ) T, "Collisions" Col
GROUP BY T.count
```

***Query result (if the result is big, just a snippet)***
Fraction of total collisions: 0.248

**Query 4:**
***Description of logic:***
“Collisions” table is related to Weather by Weather_related table which contains CaseID and the possible set of weathers during the collision. This way we need to query the three tables.
***SQL statement***

```
SELECT COUNT(*)
FROM "Collisions" Col, "Weather" W, "Weather_related" Wr
WHERE Col.CASE_ID = Wr.CASE_ID and Wr.WEATHER = W.INT_WEATHER and
    W.WEATHER Like '%snow%'
```

***Query result (if the result is big, just a snippet)***
The number of collisions that have occurred under snowy weather is 8528.

**Query 5:**
***Description of logic:***
We extract Day from Date_Time and then group by.
***SQL statement***

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
SELECT  DISTINCT to_char("Date_Time", 'DAY') Days, COUNT(*) Nb_collisions
FROM "Collisions" COL
GROUP BY  to_char("Date_Time", 'DAY')
ORDER BY 2 DESC
fetch first row only;
```

***Query result (if the result is big, just a snippet)***
The busiest day is Friday with 604301 collisions.

**Query 6:**
***Description of logic:***
Using Collisions, Weather and the Weather_related table which contains the many to many relation between Collison and Weather.
***SQL statement***

```
SELECT W.weather, Count(*) Nb_Collisions
FROM "Collisions" Col, "Weather" W, "Weather_related" Wr
WHERE Col.case_id = Wr.case_id and Wr.weather=W.int_weather
GROUP BY  W.weather
ORDER BY 2 desc
```

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

| | WEATHER | NB_COLLISIONS |
|---|---|---|
| 1 | clear | 2890532 |
| 2 | cloudy | 541050 |
| 3 | raining | 223685 |
| 4 | fog | 21203 |
| 5 | wind | 13883 |
| 6 | snowing | 8528 |
| 7 | other | 6933 |

**Query 7:**
***Description of logic:***
Equivalent logic to Query 6.
***SQL statement***

```
SELECT  Count(*)
From "Parties" Party, "Road_condition_related" Road, "Road_condition" R
WHERE Party.AT_FAULT = 1 and TO_CHAR(Party.FINANCIAL_RESPONSIBILITY) = 'Y'and
    Party.CASE_ID = Road.CASE_ID and Road.ROAD_CONDITION = R.INT_ROAD_CONDITION and
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
    R.ROAD_CONDITION LIKE '%loose%'
```

***Query result (if the result is big, just a snippet)***
At-fault collision parties with financial responsibility and loose material road conditions are 4689.

**Query 8:**
***Description of logic:***
With a nested query on Victim table we output at the same time Median age, Victim seating postion and the of its number of instances.
***SQL statement***

```
SELECT  MEDIAN(V.VICTIM_AGE), T.VICTIM_SEATING_POSITION, T.Nb
FROM
(
  SELECT V.VICTIM_SEATING_POSITION, COUNT(V.VICTIM_SEATING_POSITION) Nb
  FROM "Victims" v
  GROUP BY V.VICTIM_SEATING_POSITION
  ORDER BY 2 DESC
  fetch first row only
) T, "Victims" V
GROUP BY T.VICTIM_SEATING_POSITION, T.Nb
```

***Query result (if the result is big, just a snippet)***
Median age = 25
Victim seating positon = 3 with 1332081 instances

**Query 9:**
***Description of logic:***
In this query we addition Victims and Parties as to get participats of the collision. We do two nested queries. First outputting Victims with a seatbelt using many to many Victims_safety_related and adding to it the equivalent with parties and then finally dividing this sum with the number of total Victims and Parties.
***SQL statement***

```
SELECT
    ((
      SELECT Count(*) v_using_belt
      FROM
     "Victims" V, "Collisions" Col, "Victims_safety_related" Vsr, "Safety_equipment" S
      Where V.CASE_ID = Col.CASE_ID and
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
      Vsr.ID=V.ID and
      S.INT_SAFETY_EQUIPMENT = Vsr.VICTIM_SAFETY_EQUIPMENT and
      S.VICTIM_SAFETY_EQUIPMENT = 'C'
    ) + (
      SELECT Count(*) p_using_belt
  FROM "Parties" P, "Collisions" Col, "Parties_safety_related" Psr, "Safety_equipment" S
  Where P.CASE_ID = Col.CASE_ID and
   Psr.ID=P.ID and
   S.INT_SAFETY_EQUIPMENT = Psr.PARTY_SAFETY_EQUIPMENT and
   S.VICTIM_SAFETY_EQUIPMENT = 'C'
    )) /
      ((
        Select Count(*)
        From "Victims"
        ) +
      (
        Select Count(*)
        From "Parties"
        ))
AS RATIO_USING_BELT
from DUAL
```

***Query result (if the result is big, just a snippet)***
The fraction is 0.737

**Query 10:**
***Description of logic:***
Here again we use MAX(ROWNUM) as to get total number of instances for the ratio. To get the hour in string we use DISTINCT TO_CHAR("Date_Time",'HH24').
***SQL statement***

```
SELECT  DISTINCT TO_CHAR("Date_Time",'HH24') Hour,
          COUNT(*)/MAX(ROWNUM) Ratio
FROM "Collisions" COL
GROUP BY  TO_CHAR("Date_Time",'HH24')
ORDER BY Hour DESC
```

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

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

| | HOUR | RATIO |
|---|---|---|
| 1 | 00 | 0.027570441190038047054156973516222894857 7 |
| 2 | 01 | 0.018589917469605623049508844988815664809 1 |
| 3 | 02 | 0.018367827794985032772661945620777389805 |
| 4 | 03 | 0.011724331668899017719499404598965430278 4 |
| 5 | 04 | 0.009968046564508483902296736042746006649231 |
| 6 | 05 | 0.014656279461237608843433125528455015064 |
| 7 | 06 | 0.026177996009077911373244027428197654504 6 |
| 8 | 07 | 0.051098074226635537253090750785325273275 5 |
| 9 | 08 | 0.051535291257844957128966675506054980995 3 |
| 10 | 09 | 0.040478022806714174669531363513566519951 3 |
| 11 | 10 | 0.042124073136418242503948052823386063352 8 |
| 12 | 11 | 0.048840901820978293116678848509257371672 2 |
| 13 | 12 | 0.057627881794626727143149159977238449341 7 |
| 14 | 13 | 0.057662628816784667864219298533960229945 6 |
| 15 | 14 | 0.065282876165731116889258547912201500875 8 |
| 16 | 15 | 0.076876691229198364265770349720508664286 7 |
| 17 | 16 | 0.072629702902470002690490304566817249412 9 |
| 18 | 17 | 0.078753794606479627210125211655484259461 3 |
| 19 | 18 | 0.063211653508333365559326303647301054673 9 |
| 20 | 19 | 0.044795739526635554677778222804747179411 4 |
| 21 | 20 | 0.035425840138310240653520859096856668306 5 |
| 22 | 21 | 0.033328803274459141354962707118987520792 7 |
| 23 | 22 | 0.029070215500560869924306729693290747179 3 |
| 24 | 23 | 0.024223307554767137976974143076981297395 6 |

## *General Comments*

Omar - Data Cleaning and table preparation.

Iana, Taha - Queries.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

# 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*

**Query 1:** For the drivers of age groups: underage (less and equal to 18 years), young I [19, 21], young II [22,24], adult [24,60], elder I [61,64], elder II [65 and over), find the ratio of cases where the driver was the party at fault. Show this ratio as percentage and display it for every age group – if you were an insurance company, based on the results would you change your policies?

***Description of logic:***

This query is longer than usual as we needed to specify different age groups and we haven't found a way how to do it differently.

***SQL statement***

```
SELECT u.Group_ages, 100*(u.count_1/v.count_2)

from
(SELECT (CASE
  WHEN P.Party_age <= 18 THEN 'underage'
  WHEN P.Party_age > 18 and P.Party_age < 22 THEN  'young I'
  WHEN P.Party_age > 21 and P.Party_age < 25 THEN  'young II'
  WHEN P.Party_age > 23 and P.Party_age < 61 THEN  'adult'
  WHEN P.Party_age > 60 and P.Party_age < 65 THEN  'elder I'
  ELSE 'elder II'
  END) as Group_ages, Count(*) count_1

FROM "Parties" P, "Party_type" Pt
Where P.at_fault = 1 and P.party_type = Pt.int_type and Pt.Party_type like '%driver%'
Group by (CASE
  WHEN P.Party_age <= 18 THEN 'underage'
  WHEN P.Party_age > 18 and P.Party_age < 22 THEN  'young I'
  WHEN P.Party_age > 21 and P.Party_age < 25 THEN  'young II'
  WHEN P.Party_age > 23 and P.Party_age < 61 THEN  'adult'
  WHEN P.Party_age > 60 and P.Party_age < 65 THEN  'elder I'
  ELSE 'elder II'
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
  END)) u, (SELECT (CASE
  WHEN P.Party_age <= 18 THEN 'underage'
  WHEN P.Party_age > 18 and P.Party_age < 22 THEN  'young I'
  WHEN P.Party_age > 21 and P.Party_age < 25 THEN  'young II'
  WHEN P.Party_age > 23 and P.Party_age < 61 THEN  'adult'
  WHEN P.Party_age > 60 and P.Party_age < 65 THEN  'elder I'
  ELSE 'elder II'
  END) as Group_ages, Count(*) count_2

FROM "Parties" P, "Party_type" Pt
Where P.party_type = Pt.int_type and Pt.Party_type like '%driver%'
Group by (CASE
  WHEN P.Party_age <= 18 THEN 'underage'
  WHEN P.Party_age > 18 and P.Party_age < 22 THEN  'young I'
  WHEN P.Party_age > 21 and P.Party_age < 25 THEN  'young II'
  WHEN P.Party_age > 23 and P.Party_age < 61 THEN  'adult'
  WHEN P.Party_age > 60 and P.Party_age < 65 THEN  'elder I'
  ELSE 'elder II'
  END)) v
Where u.Group_ages = v.Group_ages
```

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

As an insurance company we would change the prices so they correspond to the number of accidents of each group. We can clearly see that elder II and underage groups cause many accidents and so their insurence should be more expensive. On the other hand adults and elder I cause less accidents and so their insurance should be cheaper.

| | GROUP_AGES | 100*(U.COUNT_1/V.COUNT_2) |
|---|---|---|
| 1 | elder II | 64.97303845855851595304194006377393157534 |
| 2 | underage | 64.71121266135821926135959569543184471988 |
| 3 | young I | 57.45227673897033651349243591948792368954 |
| 4 | young II | 51.92481990217680058756426484146703545 7 |
| 5 | adult | 40.97509123437273260086032334169317253 59 |
| 6 | elder I | 40.07045843736519431626557209898168603511 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 2:** Find the top-5 vehicle types based on the number of collisions on roads with holes. List both the vehicle type and their corresponding number of collisions.

***Description of logic:***
In this query we need to use Vehicle, Statewide, Statewide_vehicle_type,Collisions, Road_condition,Road_condition_related and Parties tables as the informations needed are in different tables. Then it's a simple comparaison and making sure that STATEWIDE_VEHICLE_TYPE IS NOT NULL and outputting top 5 results.

***SQL statement***

```
SELECT Svt.STATEWIDE_VEHICLE_TYPE, Count(*) Nb_Collisions
FROM "Vehicle" V, "Statewide_vehicle_type" Svt, "Collisions" Col,
  "Road_condition" RC, "Road_condition_related" Rcr, "Parties" P

Where Svt.INT_VEH_TYPE=V.STATEWIDE_VEHICLE_TYPE and
   V.INT_VEHICLE = P.VEHICLE and
   P.CASE_ID = Col.CASE_ID and
   Col.CASE_ID = Rcr.CASE_ID and
   Rcr.ROAD_CONDITION = Rc.INT_ROAD_CONDITION and
   Rc.ROAD_CONDITION like '%holes%' and
   Svt.STATEWIDE_VEHICLE_TYPE IS NOT NULL
GROUP BY Svt.STATEWIDE_VEHICLE_TYPE
ORDER BY 2 DESC
FETCH first 5 Rows Only;
```

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

| | STATEWIDE_VEHICLE_TYPE | NB_COLLISIONS |
|---|---|---|
| 1 | passenger car | 10321 |
| 2 | pickup or panel truck | 2198 |
| 3 | motorcycle or scooter | 449 |
| 4 | bicycle | 428 |
| 5 | truck or truck tractor with traile | 365 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 3:** Find the top-10 vehicle makes based on the number of victims who suffered either a severe injury or were killed. List both the vehicle make and their corresponding number of victims.

***Description of logic:***

Similar to the last query, many tables are queried. We use like to check severe injury or killed and then we output the top ten.

***SQL statement***

```
SELECT VehM.VEHICLE_MAKE, COUNT(*)
FROM "Parties" P, "Victims" V, "Vehicle_make" VehM, "Vehicle" Veh, "Victim_degree_of_injury" Vdi
WHERE P.CASE_ID = V.CASE_ID AND P.VEHICLE = Veh.INT_VEHICLE AND Veh.VEHICLE_MAKE = VehM.INT_MAKE
and
    Vdi.INT_DEG_INJURY = V.VICTIM_DEGREE_OF_INJURY
AND (Vdi.VICTIM_DEGREE_OF_INJURY LIKE '%severe injury%' OR Vdi.VICTIM_DEGREE_OF_INJURY LIKE
'%killed%') and VehM.VEHICLE_MAKE IS NOT NULL
GROUP BY VehM.VEHICLE_MAKE
ORDER BY 2 DESC
FETCH NEXT 10 ROWS ONLY;
```

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

| | VEHICLE_MAKE | COUNT(*) |
|---|---|---|
| 1 | FORD | 31727 |
| 2 | CHEVROLET | 22835 |
| 3 | TOYOTA | 22292 |
| 4 | HONDA | 20212 |
| 5 | DODGE | 9028 |
| 6 | NISSAN | 8070 |
| 7 | GMC | 4696 |
| 8 | NOT STATED | 4172 |
| 9 | HARLEY-DAVIDSON | 3810 |
| 10 | MISCELLANEOUS | 3753 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 4:** Compute the safety index of each seating position as the fraction of total incidents where the victim suffered no injury. The position with the highest safety index is the safest, while the one with the lowest is the most unsafe. List the most safe and unsafe victim seating position along with its safety index.

***Description of logic:***

In this query first we need to extract “no injury” instances which we then combine with the seating positon.

***SQL statement***

```
SELECT T.VICTIM_SEATING_POSITION, T.Safety_index
FROM
  (
   Select no_inj.VICTIM_SEATING_POSITION,
   no_inj.count_no_inj/inj.count_inj Safety_index,
   row_number() over (order by no_inj.count_no_inj/inj.count_inj desc) as rn,
   count(*) over () as total_count
    From (
     SELECT V.VICTIM_SEATING_POSITION, Count(*) count_no_inj
     FROM "Victims" V, "Collisions" Col, "Victim_degree_of_injury" Vdi
     WHERE Col.CASE_ID = V.CASE_ID and
          V.VICTIM_DEGREE_OF_INJURY = Vdi.INT_DEG_INJURY and
          Vdi.VICTIM_DEGREE_OF_INJURY LIKE '%no injury%'
     GROUP BY  V.VICTIM_SEATING_POSITION
     ) no_inj, (
     SELECT V.VICTIM_SEATING_POSITION, Count(*) count_inj
     FROM "Victims" V, "Collisions" Col, "Victim_degree_of_injury" Vdi
     WHERE Col.CASE_ID = V.CASE_ID and
          V.VICTIM_DEGREE_OF_INJURY = Vdi.INT_DEG_INJURY
     GROUP BY  V.VICTIM_SEATING_POSITION
     ) inj
    Where no_inj.VICTIM_SEATING_POSITION = inj.VICTIM_SEATING_POSITION
         ) T
Where rn = 1 or rn = total_count
Order by 2 desc;
```

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

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

| | VICTIM_SEATING_POSITION | SAFETY_INDEX |
|---|---|---|
| 1 | 5 | 0.836203126366692074123130365242599557660 3 |
| 2 | 1 | 0.008904977375565610859728506787330316742081 |

**Query 5:** How many vehicle types have participated in at least 10 collisions in at least half of the cities?

***Description of logic:***
First we started with outputting vehicle types which have participated in at least 10 collision.
To find half of the cities we used Count(DISTINCT L.COUNTY_CITY_LOCATION) in an nested query.
And finally we just counted the instances.

***SQL statement***

```
Select Count(*)
From (
        SELECT T.STATEWIDE_VEHICLE_TYPE, Count(*)
        FROM (
        SELECT L.COUNTY_CITY_LOCATION, Svt.STATEWIDE_VEHICLE_TYPE, Count(*)
        FROM "Collisions" Col,
                "Vehicle" V,
                "Parties" P,
                "Statewide_vehicle_type" Svt,
                "Location" L
        WHERE Col.CASE_ID = P.CASE_ID
                and P.VEHICLE = V.INT_VEHICLE
                and V.STATEWIDE_VEHICLE_TYPE = Svt.INT_VEH_TYPE
                and Col.Location = L.int_location
        GROUP BY Svt.STATEWIDE_VEHICLE_TYPE, L.COUNTY_CITY_LOCATION
        Having Count(V.STATEWIDE_VEHICLE_TYPE) > 10) T
        GROUP BY T.STATEWIDE_VEHICLE_TYPE
        HAVING COUNT(*) >= (SELECT Count(DISTINCT L.COUNTY_CITY_LOCATION) From "Location" L) / 2
        Order by 2 Desc
        );
```

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

| | COUNT(*) |
|---|---|
| 1 | 14 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 6:** For each of the top-3 most populated cities, show the city location, population, and the bottom-10 collisions in terms of average victim age (show collision id and average victim age).

***Description of logic:***
We write a first nested query to select the top-3 most populated cities (the condition L.Population <=7 ensures that the selection does not include rural areas), a second query to compute the average victim age for each county_city_location together with the rank of each case in terms of average victim age and a final selection to display the city location, population, case_id, avg_age for the cases where the rank is less than 10.

***SQL statement***

```
SELECT COUNTY_CITY_LOCATION, POPULATION, CASE_ID, AVERAGE_AGE
FROM
        (
  SELECT top.COUNTY_CITY_LOCATION, top.POPULATION, Col.CASE_ID, AVG(V.VICTIM_AGE) AVERAGE_AGE,
        ROW_NUMBER() over (partition by top.COUNTY_CITY_LOCATION order by AVG(V.VICTIM_AGE)) rank
  FROM (
        SELECT L.INT_LOCATION, L.COUNTY_CITY_LOCATION, L.POPULATION
        FROM "Location" L
        WHERE L.POPULATION = (SELECT MAX(L.POPULATION) From "Location" L Where L.POPULATION <= 7)
        FETCH FIRST 3 rows only
        ) top, "Victims" V, "Collisions" Col
  WHERE V.CASE_ID = Col.CASE_ID and
        Col.LOCATION = top.INT_LOCATION

  GROUP BY top.COUNTY_CITY_LOCATION, top.POPULATION, Col.CASE_ID
        )
where rank <= 10;
```

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

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

| | COUNTY_CITY_LOCATION | POPULATION | CASE_ID | AVERAGE_AGE |
|---|---|---|---|---|
| 1 | 1005 | 7 | 0270716 | 0 |
| 2 | 1005 | 7 | 3307915 | 0 |
| 3 | 1005 | 7 | 2291876 | 0 |
| 4 | 1005 | 7 | 1743393 | 0 |
| 5 | 1005 | 7 | 1127018 | 0 |
| 6 | 1005 | 7 | 1005010920090100384 | 0 |
| 7 | 1005 | 7 | 1005010430213000798 | 0 |
| 8 | 1005 | 7 | 0981396 | 0 |
| 9 | 1005 | 7 | 0970471 | 0 |
| 10 | 1005 | 7 | 0219887 | 0 |
| 11 | 1942 | 7 | 0027566 | 0 |
| 12 | 1942 | 7 | 9590010109200016257 | 0 |
| 13 | 1942 | 7 | 9580010326141514565 | 0 |
| 14 | 1942 | 7 | 9580010117071513408 | 0 |
| 15 | 1942 | 7 | 3531642 | 0 |
| 16 | 1942 | 7 | 3474774 | 0 |
| 17 | 1942 | 7 | 3457495 | 0 |
| 18 | 1942 | 7 | 3426303 | 0 |
| 19 | 1942 | 7 | 3424265 | 0 |
| 20 | 1942 | 7 | 3403451 | 0 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 7:** Find all collisions that satisfy the following: the collision was of type pedestrian and all victims were above 100 years old. For each of the qualifying collisions, show the collision id and the age of the eldest collision victim.

***Description of logic:***

For each case_id, we select the maximum victim age where the collision was of type pedestrian, under the condition that the minimum victim age is larger than 100. This last condition ensures that all victims are above 100 years old.

***SQL statement***

```
SELECT C.CASE_ID,  MAX(V.VICTIM_AGE)
FROM "Collisions" C, "Victims" V, "Type_of_collision" coltype
WHERE C.CASE_ID = V.CASE_ID  AND C.TYPE_OF_COLLISION=coltype.INT_COLL_TYPE AND
coltype.TYPE_OF_COLLISION= 'pedestrian'
GROUP BY C.CASE_ID
HAVING MIN(V.VICTIM_AGE)>100;
```

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

| | CASE_ID | MAX(V.VICTIM_AGE) |
|---|---|---|
| 1 | 1123740 | 102 |
| 2 | 2186400 | 102 |
| 3 | 0999973 | 105 |
| 4 | 2444474 | 102 |
| 5 | 1990860 | 104 |
| 6 | 2715861 | 105 |
| 7 | 3129177 | 101 |
| 8 | 1576798 | 102 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 8:** Find the vehicles that have participated in at least 10 collisions. Show the vehicle id and number of collisions the vehicle has participated in, sorted according to number of collisions (descending order). What do you observe?

***Description of logic:***

For this query, we consider that the tuple (vehicle_make, vehicle_year, vehicle_type) is the vehicle ID. We perform multiple joins going from Collisions to the tables Vehicle, Statewide_vehicle_type and Vehicle_make. Then, we group by the relevant attributes (vehicle_make, vehicle_year, vehicle_type). Next, we count the number of collision occurrences N for each vehicle_id and display the output in descending order under the condition N>10.

***SQL statement***

```
SELECT Svt.STATEWIDE_VEHICLE_TYPE, Vm.VEHICLE_MAKE, V.VEHICLE_YEAR, Count(*) Nb_Collisions
FROM "Vehicle" V, "Collisions" Col, "Parties" P, "Statewide_vehicle_type" Svt, "Vehicle_make" Vm
Where  P.CASE_ID = Col.CASE_ID and V.INT_VEHICLE = P.VEHICLE
        and
        Svt.INT_VEH_TYPE = V.STATEWIDE_VEHICLE_TYPE and V.VEHICLE_MAKE = Vm.INT_MAKE
GROUP BY Svt.STATEWIDE_VEHICLE_TYPE, Vm.VEHICLE_MAKE, V.VEHICLE_YEAR
Having Count(*) >= 10
Order by Nb_Collisions desc;
```

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

We observe that the most common vehicle are those where the investigator didn't input the details of the vehicle. When we consider the human error this makes sense. Especially for pedestrians and bicycles.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

| | STATEWIDE_VEHICLE_TYPE | VEHICLE_MAKE | VEHICLE_YEAR | NB_COLLISIONS |
|---|---|---|---|---|
| 1 | (null) | (null) | (null) | 142112 |
| 2 | pedestrian | (null) | (null) | 84092 |
| 3 | passenger car | (null) | (null) | 75721 |
| 4 | bicycle | (null) | (null) | 67592 |
| 5 | passenger car | TOYOTA | 2000 | 51103 |
| 6 | passenger car | FORD | 2000 | 50835 |
| 7 | passenger car | HONDA | 2000 | 48915 |
| 8 | passenger car | FORD | 1998 | 48153 |
| 9 | passenger car | TOYOTA | 2001 | 46007 |
| 10 | passenger car | FORD | 2001 | 44149 |
| 11 | passenger car | HONDA | 2001 | 43999 |
| 12 | passenger car | TOYOTA | 1999 | 41808 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 9:** Find the top-10 (with respect to number of collisions) cities. For each of these cities, show the city location and number of collisions.

***Description of logic:***

We join the tables “Collisions” and the table “Location” the location attribute. Next we group by COUNTY_CITY_LOCATION, count their occurences, order in descending order and display the 10 first rows.

***SQL statement***

```
SELECT L.COUNTY_CITY_LOCATION, COUNT(*) NUMBER_OF_COLLISIONS
FROM "Location" L, "Collisions" C
WHERE C.LOCATION = L.INT_LOCATION
GROUP BY L.COUNTY_CITY_LOCATION
ORDER BY 2 DESC
FETCH NEXT 10 ROWS ONLY;
```

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

| | COUNTY_CITY_LOCATION | NUMBER_OF_COLLISIONS |
|---|---|---|
| 1 | 1942 | 382220 |
| 2 | 1900 | 118446 |
| 3 | 3400 | 78622 |
| 4 | 3711 | 69410 |
| 5 | 109 | 67883 |
| 6 | 3300 | 61453 |
| 7 | 3404 | 52762 |
| 8 | 4313 | 50876 |
| 9 | 1941 | 49166 |
| 10 | 3801 | 48450 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 10:** Are there more accidents around dawn, dusk, during the day, or during the night? In case lighting information is not available, assume the following: the dawn is between 06:00 and 07:59, and dusk between 18:00 and 19:59 in the period September 1 - March 31; and dawn between 04:00 and 06:00, and dusk between 20:00 and 21:59 in the period April 1 - August 31. The remaining corresponding times are night and day. Display the number of accidents, and to which group it belongs, and make your conclusion based on absolute number of accidents in the given 4 periods.

***Description of logic:***
First we try to extract dawn,dusk from lighting information. Then we extract month and compare times for spring and autumn to decide on the lighting it is.
Then finally we group by time and output the table.

***SQL statement***

```
SELECT T.time, Count(*) Nb_Accidents
From (
 SELECT Col.CASE_ID,
      (CASE
         WHEN L.LIGHTING LIKE '%daylight%' THEN 'during the day'
         WHEN L.INT_LIGHT LIKE '%dark%' THEN 'during the night'
         WHEN (EXTRACT(MONTH FROM Col."Date_Time") >= 9 or
            EXTRACT(MONTH FROM Col."Date_Time") <= 3) and
            to_char("Date_Time", 'HH24:MI:SS') >= '06:00:00' and
            to_char("Date_Time", 'HH24:MI:SS') <= '07:59:00'
           then 'dawn'
         WHEN (EXTRACT(MONTH FROM Col."Date_Time") >= 9 or
            EXTRACT(MONTH FROM Col."Date_Time") <= 3) and
            to_char("Date_Time", 'HH24:MI:SS') >= '18:00:00' and
            to_char("Date_Time", 'HH24:MI:SS') <= '19:59:00'
           then 'dusk'
         WHEN EXTRACT(MONTH FROM Col."Date_Time") >= 4 and
            EXTRACT(MONTH FROM Col."Date_Time") <= 8 and
            to_char("Date_Time", 'HH24:MI:SS') >= '04:00:00' and
            to_char("Date_Time", 'HH24:MI:SS') <= '06:00:00'
           then 'dawn'
         WHEN (EXTRACT(MONTH FROM Col."Date_Time") >= 4 and
            EXTRACT(MONTH FROM Col."Date_Time") <= 8) and
            to_char("Date_Time", 'HH24:MI:SS') >= '20:00:00' and
            to_char("Date_Time", 'HH24:MI:SS') <= '21:59:00'
```

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

```
        then 'dusk'
    WHEN  ((EXTRACT(MONTH FROM Col."Date_Time") >= 9 or
        EXTRACT(MONTH FROM Col."Date_Time") <= 3) and
        to_char("Date_Time", 'HH24:MI:SS') > '07:59:00' and
        to_char("Date_Time", 'HH24:MI:SS') < '18:00:00') OR
       (EXTRACT(MONTH FROM Col."Date_Time") >= 4 and
        EXTRACT(MONTH FROM Col."Date_Time") <= 8 and
        to_char("Date_Time", 'HH24:MI:SS') > '06:00:00' and
        to_char("Date_Time", 'HH24:MI:SS') < '20:00:00')
        then 'during the day'
    WHEN ((EXTRACT(MONTH FROM Col."Date_Time") >= 9 or
        EXTRACT(MONTH FROM Col."Date_Time") <= 3) and
        (to_char("Date_Time", 'HH24:MI:SS') > '19:59:00' or
        to_char("Date_Time", 'HH24:MI:SS') < '06:00:00')) OR
       (EXTRACT(MONTH FROM Col."Date_Time") >= 4 and
        EXTRACT(MONTH FROM Col."Date_Time") <= 8 and
        (to_char("Date_Time", 'HH24:MI:SS') > '21:59:00' or
        to_char("Date_Time", 'HH24:MI:SS') < '04:00:00'))
        then 'during the night'
    ELSE 'Not possible'
    END) time

 FROM "Lighting" L,
     "Collisions" Col
 WHERE L.INT_LIGHT = Col.LIGHTING
) T
GROUP BY T.time
Order by 2 desc;
```

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

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

| | TIME | NB_ACCIDENTS |
|---|---|---|
| 1 | during the day | 2610878 |
| 2 | during the night | 638018 |
| 3 | dusk | 305720 |
| 4 | dawn | 65654 |

The absolute majority of the accidents takes part during the day. Which makes sense as it's the time which has the most drivers. An interesting query would be seing the percentage of fatal or grave accidents, in this query the results could be different.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

# *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:** For the drivers of age groups: underage (less and equal to 18 years), young I [19, 21], young II [22,24], adult [24,60], elder I [61,64], elder II [65 and over), find the ratio of cases where the driver was the party at fault. Show this ratio as a percentage and display it for every age group – if you were an insurance company, based on the results would you change your policies?

**<Initial Running time/IO:** 8.861s

| SQL | Connexion | Horodat... | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT u.Group_ages, 100*(u.count_1/v.count_2)from (SE... | login_G71 | 31.05.21 ... | SQL | 29 | 8.861 |

**Optimized Running time/IO:** 6.612s

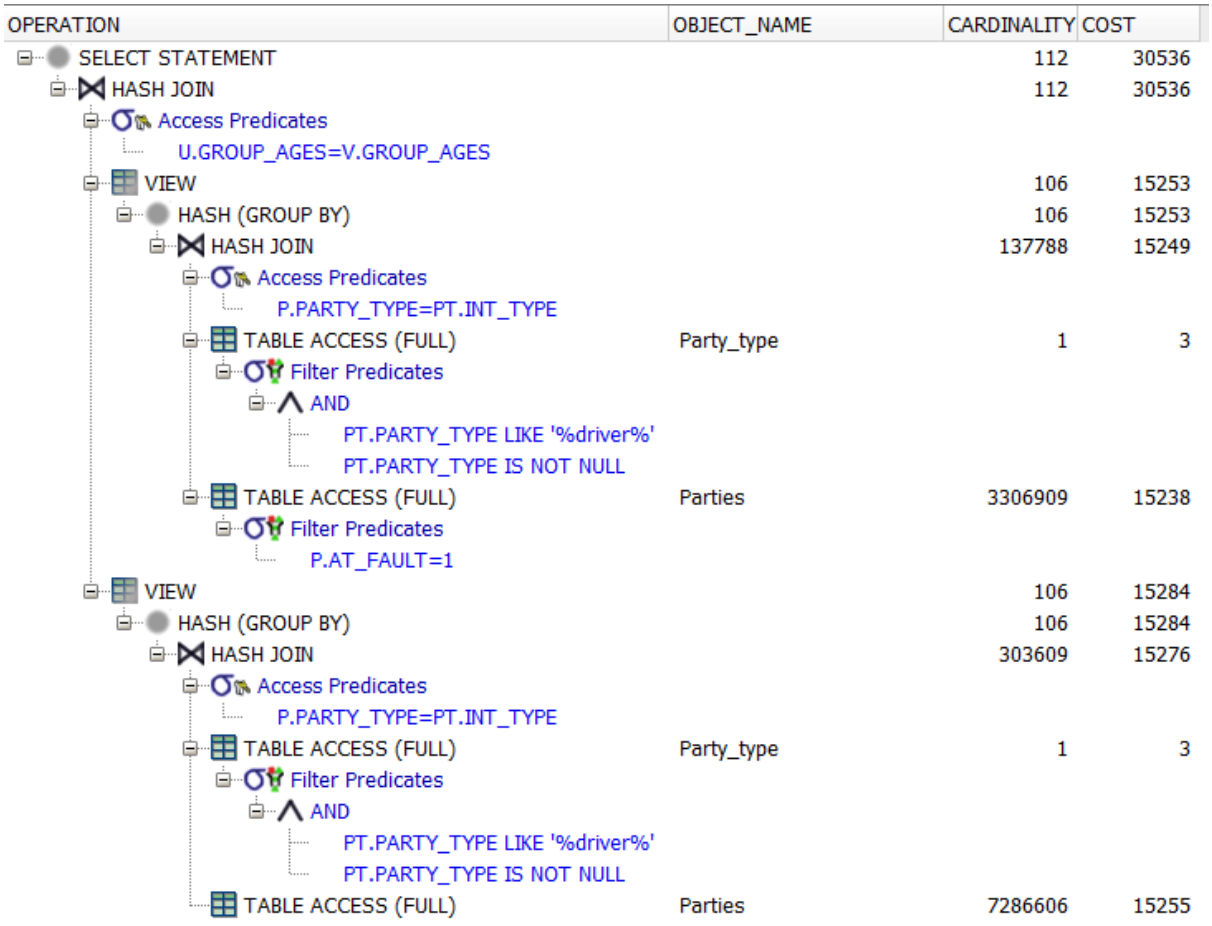
| SQL | Connexion | Horodat... | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT u.Group_ages, 100*(u.count_1/v.count_2)from (SE... | login_G71 | 31.05.21 ... | SQL | 30 | 6.612 |

**Explain the improvement:**
Creating an index that is built on (party_type, party_age, party_at_fault) changed the scan method from a full scan, on the parties table, to a range scan. The optimizer took advantage of this index in the joint operation on the PARTY_TYPE as well.
Creating an index solely on (party_type) would drop the cost at the join operation only.
Considering (party_type,party_at_fault) would also be useful but won't eliminate the first full table access on Parties. Thus, choosing the tuple (party_type, party_age, party_at_fault) is the best option since it helps in the join and selection operations and eliminates a full table access on the Parties table.

Use the command *ALTER INDEX PARTY_AGE_TYPE_FAULT VISIBLE* to make the index visible.

**Initial plan**

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/




**Improved plan>**

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 112 | 6220 |
| HASH JOIN | | 112 | 6220 |
| Access Predicates | | | |
| U.GROUP_AGES=V.GROUP_AGES | | | |
| VIEW | | 106 | 3109 |
| HASH (GROUP BY) | | 106 | 3109 |
| NESTED LOOPS | | 137788 | 3105 |
| TABLE ACCESS (FULL) | Party_type | 1 | 3 |
| Filter Predicates | | | |
| AND | | | |
| PT.PARTY_TYPE LIKE '%driver%' | | | |
| PT.PARTY_TYPE IS NOT NULL | | | |
| INDEX (RANGE SCAN) | PARTY_AGE_TYPE_FAULT | 551151 | 3102 |
| Access Predicates | | | |
| AND | | | |
| P.PARTY_TYPE=PT.INT_TYPE | | | |
| P.AT_FAULT=1 | | | |
| Filter Predicates | | | |
| P.AT_FAULT=1 | | | |
| VIEW | | 106 | 3111 |
| HASH (GROUP BY) | | 106 | 3111 |
| NESTED LOOPS | | 303609 | 3104 |
| TABLE ACCESS (FULL) | Party_type | 1 | 3 |
| Filter Predicates | | | |
| AND | | | |
| PT.PARTY_TYPE LIKE '%driver%' | | | |
| PT.PARTY_TYPE IS NOT NULL | | | |
| INDEX (RANGE SCAN) | PARTY_AGE_TYPE_FAULT | 1214434 | 3101 |
| Access Predicates | | | |
| P.PARTY_TYPE=PT.INT_TYPE | | | |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 2:** Find the top-5 vehicle types based on the number of collisions on roads with holes. List both the vehicle type and their corresponding number of collisions.

**<Initial Running time/IO:** 6.732s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT Svt.STATEWIDE_VEHICLE_TYPE, Count(*) Nb_Colli... | login_G71 | 31.05.21 10:59 | SQL | 38 | 6.732 |

**Optimized Running time/IO:** 4.152s

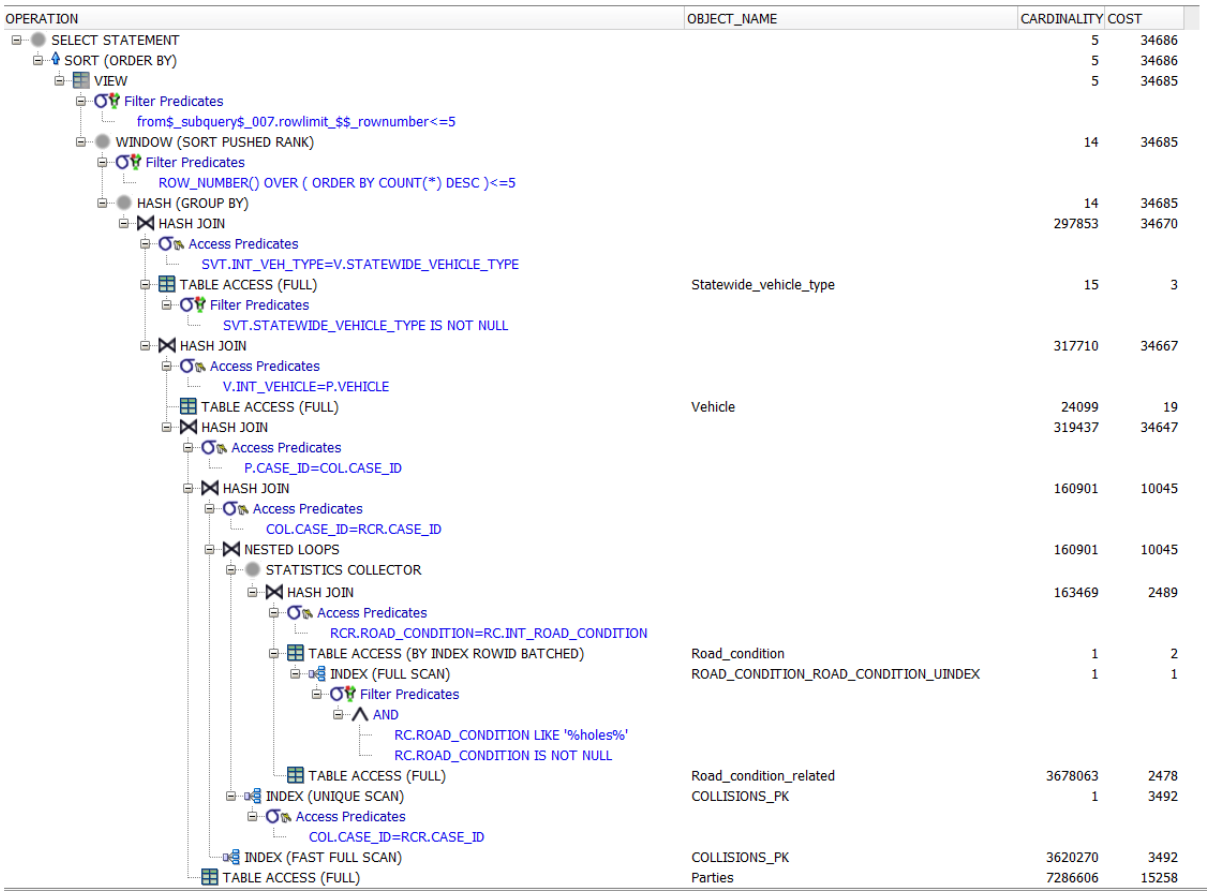
| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT Svt.STATEWIDE_VEHICLE_TYPE, Count(*) Nb_Colli... | login_G71 | 31.05.21 11:00 | SQL | 39 | 4.152 |

**Explain the improvement:**
Creating index on (CASE_ID, VEHICLE) in the parties table drops the overall cost as it will be useful in eliminating the full scan on the Parties table. However, using the same index is more computationally expensive while performing the join operation between the collision case_id and the relation table that maps road_condition to collision.
Note that both the COLLISION Primary key and Road_cdt_related table Primary key indexes are used in this query plan. If we had kept the road_condition attribute inside the collision table we would have considered this field as an optimization path.

Use the command *ALTER INDEX PARTY_CASE_ID_VEHICLE VISIBLE* to make the index visible.

**Initial plan**

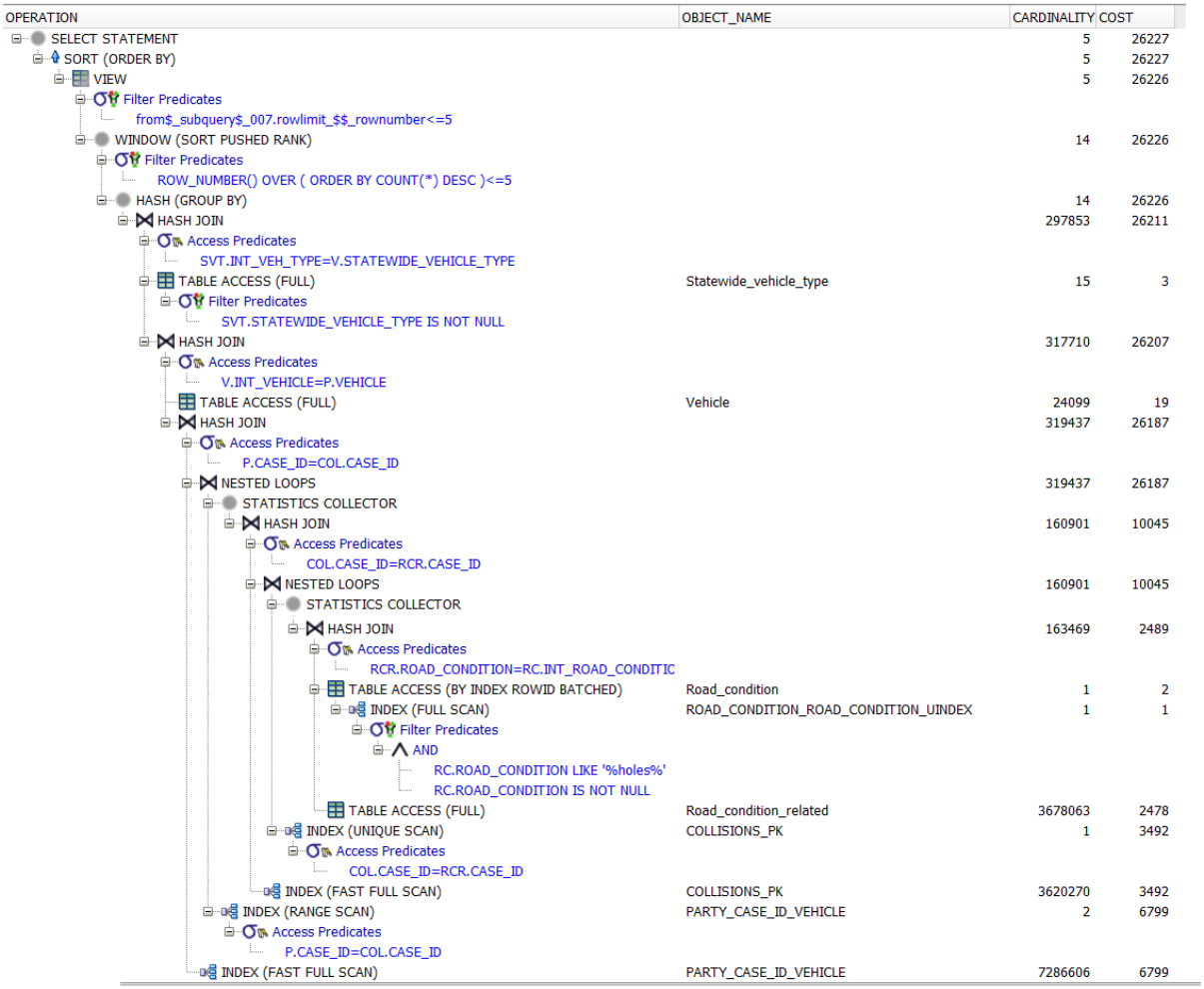
**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/



| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 5 | 34686 |
| SORT (ORDER BY) | | 5 | 34686 |
| VIEW | | 5 | 34685 |
| Filter Predicates | | | |
| from$_subquery$_007.rowlimit_$$_rownumber<=5 | | | |
| WINDOW (SORT PUSHED RANK) | | 14 | 34685 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=5 | | | |
| HASH (GROUP BY) | | 14 | 34685 |
| HASH JOIN | | 297853 | 34670 |
| Access Predicates | | | |
| SVT.INT_VEH_TYPE=V.STATEWIDE_VEHICLE_TYPE | | | |
| TABLE ACCESS (FULL) | Statewide_vehicle_type | 15 | 3 |
| Filter Predicates | | | |
| SVT.STATEWIDE_VEHICLE_TYPE IS NOT NULL | | | |
| HASH JOIN | | 317710 | 34667 |
| Access Predicates | | | |
| V.INT_VEHICLE=P.VEHICLE | | | |
| TABLE ACCESS (FULL) | Vehicle | 24099 | 19 |
| HASH JOIN | | 319437 | 34647 |
| Access Predicates | | | |
| P.CASE_ID=COL.CASE_ID | | | |
| HASH JOIN | | 160901 | 10045 |
| Access Predicates | | | |
| COL.CASE_ID=RCR.CASE_ID | | | |
| NESTED LOOPS | | 160901 | 10045 |
| STATISTICS COLLECTOR | | | |
| HASH JOIN | | 163469 | 2489 |
| Access Predicates | | | |
| RCR.ROAD_CONDITION=RC.INT_ROAD_CONDITION | | | |
| TABLE ACCESS (BY INDEX ROWID BATCHED) | Road_condition | 1 | 2 |
| INDEX (FULL SCAN) | ROAD_CONDITION_ROAD_CONDITION_UINDEX | 1 | 1 |
| Filter Predicates | | | |
| AND | | | |
| RC.ROAD_CONDITION LIKE '%holes%' | | | |
| RC.ROAD_CONDITION IS NOT NULL | | | |
| TABLE ACCESS (FULL) | Road_condition_related | 3678063 | 2478 |
| INDEX (UNIQUE SCAN) | COLLISIONS_PK | 1 | 3492 |
| Access Predicates | | | |
| COL.CASE_ID=RCR.CASE_ID | | | |
| INDEX (FAST FULL SCAN) | COLLISIONS_PK | 3620270 | 3492 |
| TABLE ACCESS (FULL) | Parties | 7286606 | 15258 |

**Improved plan>**

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/



| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 5 | 26227 |
| SORT (ORDER BY) | | 5 | 26227 |
| VIEW | | 5 | 26226 |
| Filter Predicates | | | |
| from$_subquery$_007.rowlimit_$$_rownumber<=5 | | | |
| WINDOW (SORT PUSHED RANK) | | 14 | 26226 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=5 | | | |
| HASH (GROUP BY) | | 14 | 26226 |
| HASH JOIN | | 297853 | 26211 |
| Access Predicates | | | |
| SVT.INT_VEH_TYPE=V.STATEWIDE_VEHICLE_TYPE | | | |
| TABLE ACCESS (FULL) | Statewide_vehicle_type | 15 | 3 |
| Filter Predicates | | | |
| SVT.STATEWIDE_VEHICLE_TYPE IS NOT NULL | | | |
| HASH JOIN | | 317710 | 26207 |
| Access Predicates | | | |
| V.INT_VEHICLE=P.VEHICLE | | | |
| TABLE ACCESS (FULL) | Vehicle | 24099 | 19 |
| HASH JOIN | | 319437 | 26187 |
| Access Predicates | | | |
| P.CASE_ID=COL.CASE_ID | | | |
| NESTED LOOPS | | 319437 | 26187 |
| STATISTICS COLLECTOR | | | |
| HASH JOIN | | 160901 | 10045 |
| Access Predicates | | | |
| COL.CASE_ID=RCR.CASE_ID | | | |
| NESTED LOOPS | | 160901 | 10045 |
| STATISTICS COLLECTOR | | | |
| HASH JOIN | | 163469 | 2489 |
| Access Predicates | | | |
| RCR.ROAD_CONDITION=RC.INT_ROAD_CONDITIC | | | |
| TABLE ACCESS (BY INDEX ROWID BATCHED) | Road_condition | 1 | 2 |
| INDEX (FULL SCAN) | ROAD_CONDITION_ROAD_CONDITION_UINDEX | 1 | 1 |
| Filter Predicates | | | |
| AND | | | |
| RC.ROAD_CONDITION LIKE '%holes%' | | | |
| RC.ROAD_CONDITION IS NOT NULL | | | |
| TABLE ACCESS (FULL) | Road_condition_related | 3678063 | 2478 |
| INDEX (UNIQUE SCAN) | COLLISIONS_PK | 1 | 3492 |
| Access Predicates | | | |
| COL.CASE_ID=RCR.CASE_ID | | | |
| INDEX (FAST FULL SCAN) | COLLISIONS_PK | 3620270 | 3492 |
| INDEX (RANGE SCAN) | PARTY_CASE_ID_VEHICLE | 2 | 6799 |
| Access Predicates | | | |
| P.CASE_ID=COL.CASE_ID | | | |
| INDEX (FAST FULL SCAN) | PARTY_CASE_ID_VEHICLE | 7286606 | 6799 |

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Query 3:** Find the top-10 vehicle makes based on the number of victims who suffered either a severe injury or were killed. List both the vehicle make and their corresponding number of victims.

**<Initial Running time/IO:** 5.491s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT VehM.VEHICLE_MAKE, COUNT(*)FROM "Parties" P, ... | login_G71 | 31.05.21 12:13 | SQL | 19 | 5.491 |

**Optimized Running time/IO:** 4.1s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT VehM.VEHICLE_MAKE, COUNT(*)FROM "Parties" P, ... | login_G71 | 31.05.21 12:08 | SQL | 18 | 4.1 |

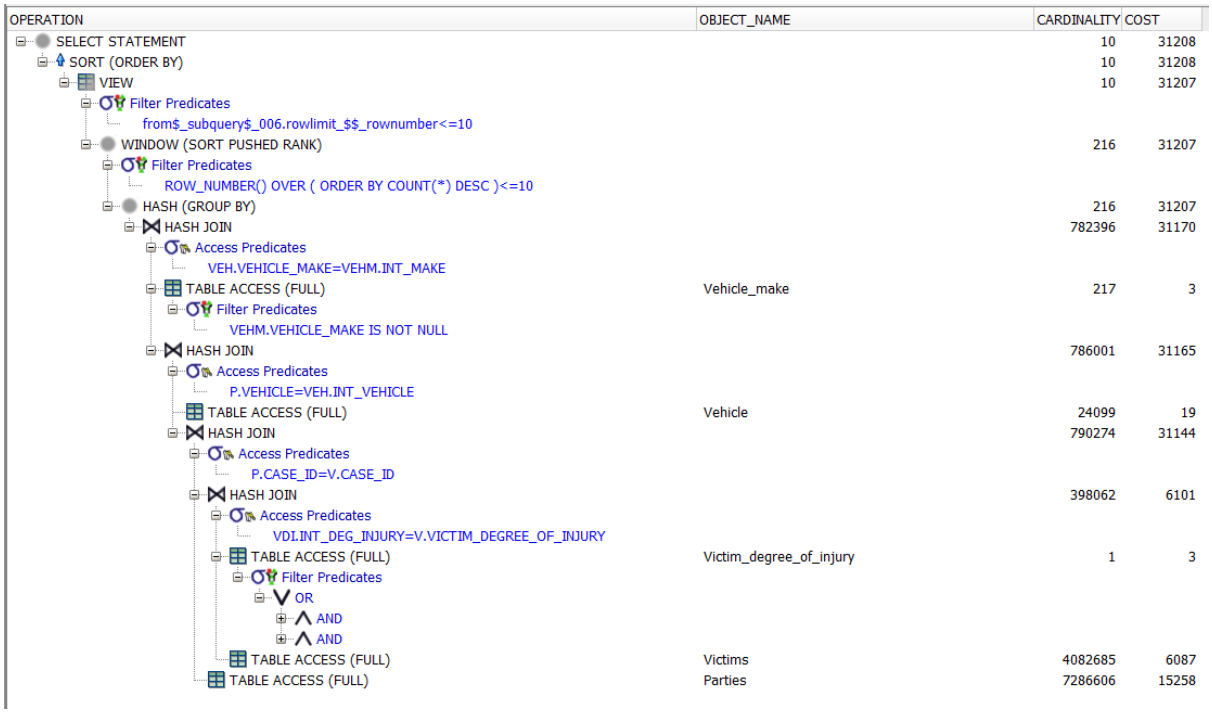
**Explain the improvement:**

An index created on the tuple (CASE_ID, DEG_OF_INJ) in the Victims table. We also took advantage of the previously created index on (CASE_ID, VEHICLE) in the parties table. Those will help turn the FULL SCAN on the tables Parties and Victims into fast full scan and a range scan while joining both tables on CASE_ID.

Use the command *ALTER INDEX* VICTIM_CASE_ID_DEG_INJ *VISIBLE* and *ALTER INDEX PARTY_CASE_ID_VEHICLE VISIBLE* to make the indexes visible.

**Initial plan**

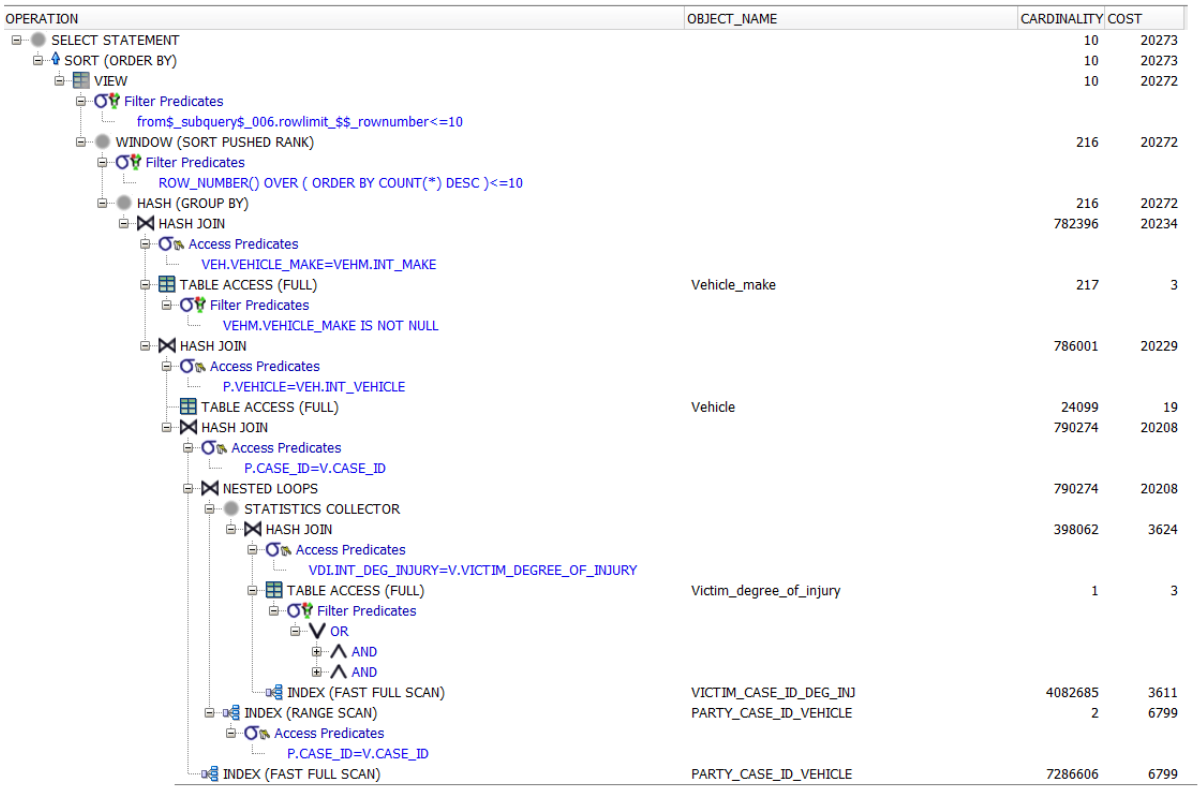
**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/



| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 10 | 31208 |
| SORT (ORDER BY) | | 10 | 31208 |
| VIEW | | 10 | 31207 |
| Filter Predicates | | | |
| from$_subquery$_006.rowlimit_$$_rownumber<=10 | | | |
| WINDOW (SORT PUSHED RANK) | | 216 | 31207 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=10 | | | |
| HASH (GROUP BY) | | 216 | 31207 |
| HASH JOIN | | 782396 | 31170 |
| Access Predicates | | | |
| VEH.VEHICLE_MAKE=VEHM.INT_MAKE | | | |
| TABLE ACCESS (FULL) | Vehicle_make | 217 | 3 |
| Filter Predicates | | | |
| VEHM.VEHICLE_MAKE IS NOT NULL | | | |
| HASH JOIN | | 786001 | 31165 |
| Access Predicates | | | |
| P.VEHICLE=VEH.INT_VEHICLE | | | |
| TABLE ACCESS (FULL) | Vehicle | 24099 | 19 |
| HASH JOIN | | 790274 | 31144 |
| Access Predicates | | | |
| P.CASE_ID=V.CASE_ID | | | |
| HASH JOIN | | 398062 | 6101 |
| Access Predicates | | | |
| VDI.INT_DEG_INJURY=V.VICTIM_DEGREE_OF_INJURY | | | |
| TABLE ACCESS (FULL) | Victim_degree_of_injury | 1 | 3 |
| Filter Predicates | | | |
| OR | | | |
| AND | | | |
| AND | | | |
| TABLE ACCESS (FULL) | Victims | 4082685 | 6087 |
| TABLE ACCESS (FULL) | Parties | 7286606 | 15258 |

**Improved plan>**

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/



| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 10 | 20273 |
| SORT (ORDER BY) | | 10 | 20273 |
| VIEW | | 10 | 20272 |
| Filter Predicates | | | |
| from$_subquery$_006.rowlimit_$$_rownumber<=10 | | | |
| WINDOW (SORT PUSHED RANK) | | 216 | 20272 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=10 | | | |
| HASH (GROUP BY) | | 216 | 20272 |
| HASH JOIN | | 782396 | 20234 |
| Access Predicates | | | |
| VEH.VEHICLE_MAKE=VEHM.INT_MAKE | | | |
| TABLE ACCESS (FULL) | Vehicle_make | 217 | 3 |
| Filter Predicates | | | |
| VEHM.VEHICLE_MAKE IS NOT NULL | | | |
| HASH JOIN | | 786001 | 20229 |
| Access Predicates | | | |
| P.VEHICLE=VEH.INT_VEHICLE | | | |
| TABLE ACCESS (FULL) | Vehicle | 24099 | 19 |
| HASH JOIN | | 790274 | 20208 |
| Access Predicates | | | |
| P.CASE_ID=V.CASE_ID | | | |
| NESTED LOOPS | | 790274 | 20208 |
| STATISTICS COLLECTOR | | | |
| HASH JOIN | | 398062 | 3624 |
| Access Predicates | | | |
| VDI.INT_DEG_INJURY=V.VICTIM_DEGREE_OF_INJURY | | | |
| TABLE ACCESS (FULL) | Victim_degree_of_injury | 1 | 3 |
| Filter Predicates | | | |
| OR | | | |
| AND | | | |
| AND | | | |
| INDEX (FAST FULL SCAN) | VICTIM_CASE_ID_DEG_INJ | 4082685 | 3611 |
| INDEX (RANGE SCAN) | PARTY_CASE_ID_VEHICLE | 2 | 6799 |
| Access Predicates | | | |
| P.CASE_ID=V.CASE_ID | | | |
| INDEX (FAST FULL SCAN) | PARTY_CASE_ID_VEHICLE | 7286606 | 6799 |

**Query 6:** For each of the top-3 most populated cities, show the city location, population, and the bottom-10 collisions in terms of average victim age (show collision id and average victim age).

**<Initial Running time/IO:** 6.24s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT COUNTY_CITY_LOCATION, POPULATION, CASE_ID, ... | login_G71 | 31.05.21 12:22 | SQL | 35 | 6.24 |

**Optimized Running time/IO:** 3.571s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT COUNTY_CITY_LOCATION, POPULATION, CASE_ID, ... | login_G71 | 31.05.21 12:23 | SQL | 36 | 3.571 |

**Explain the improvement:**

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

We created an index on the Location attribute of the Collisions table as well as (Case_id, Victim_age) in the Victims table. Here again, the index was used in the join operation on Collisions Location and helped avoid a full scan of the Collisions and Victims tables. Note that using an index on the Population attribute of the Location entity would also drop the cost in the selection operations on Population. But the change will not be significant as the Location entity has a small number of rows compared to the Collision table.

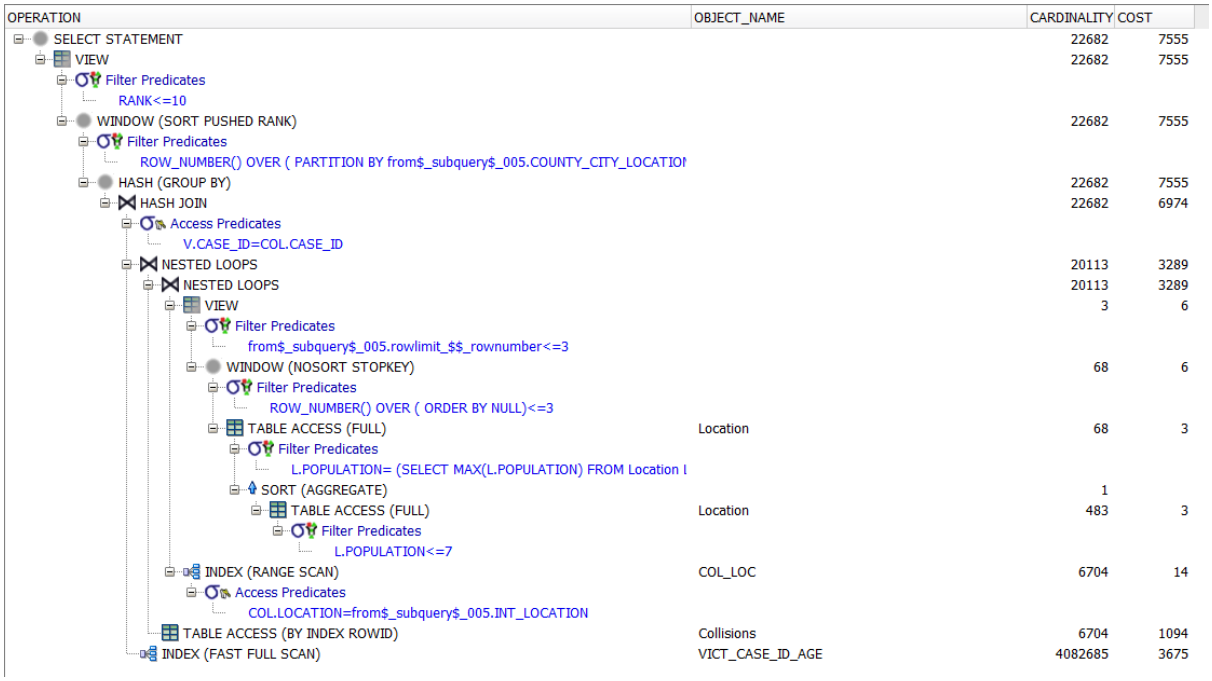
```
ALTER INDEX COL_LOC VISIBLE;
ALTER INDEX VICT_CASE_ID_AGE VISIBLE;
```

**Initial plan**

| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 22682 | 15533 |
| VIEW | | 22682 | 15533 |
| Filter Predicates | | | |
| RANK<=10 | | | |
| WINDOW (SORT PUSHED RANK) | | 22682 | 15533 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( PARTITION BY from$_subquery$_005.COUNTY_CITY_LOCATIOI | | | |
| HASH (GROUP BY) | | 22682 | 15533 |
| HASH JOIN | | 22682 | 14952 |
| Access Predicates | | | |
| V.CASE_ID=COL.CASE_ID | | | |
| HASH JOIN | | 20113 | 8864 |
| Access Predicates | | | |
| COL.LOCATION=from$_subquery$_005.INT_LOCATION | | | |
| VIEW | | 3 | 6 |
| Filter Predicates | | | |
| from$_subquery$_005.rowlimit_$$_rownumber<=3 | | | |
| WINDOW (NOSORT STOPKEY) | | 68 | 6 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY NULL)<=3 | | | |
| TABLE ACCESS (FULL) | Location | 68 | 3 |
| Filter Predicates | | | |
| L.POPULATION= (SELECT MAX(L.POPULATION) FROM Location L W | | | |
| SORT (AGGREGATE) | | 1 | |
| TABLE ACCESS (FULL) | Location | 483 | 3 |
| Filter Predicates | | | |
| L.POPULATION<=7 | | | |
| TABLE ACCESS (FULL) | Collisions | 3620270 | 8849 |
| TABLE ACCESS (FULL) | Victims | 4082685 | 6077 |

**Improved plan>**

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/



| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 22682 | 7555 |
| VIEW | | 22682 | 7555 |
| Filter Predicates | | | |
| RANK<=10 | | | |
| WINDOW (SORT PUSHED RANK) | | 22682 | 7555 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( PARTITION BY from$_subquery$_005.COUNTY_CITY_LOCATIOI | | | |
| HASH (GROUP BY) | | 22682 | 7555 |
| HASH JOIN | | 22682 | 6974 |
| Access Predicates | | | |
| V.CASE_ID=COL.CASE_ID | | | |
| NESTED LOOPS | | 20113 | 3289 |
| NESTED LOOPS | | 20113 | 3289 |
| VIEW | | 3 | 6 |
| Filter Predicates | | | |
| from$_subquery$_005.rowlimit_$$_rownumber<=3 | | | |
| WINDOW (NOSORT STOPKEY) | | 68 | 6 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY NULL)<=3 | | | |
| TABLE ACCESS (FULL) | Location | 68 | 3 |
| Filter Predicates | | | |
| L.POPULATION= (SELECT MAX(L.POPULATION) FROM Location l | | | |
| SORT (AGGREGATE) | | 1 | |
| TABLE ACCESS (FULL) | Location | 483 | 3 |
| Filter Predicates | | | |
| L.POPULATION<=7 | | | |
| INDEX (RANGE SCAN) | COL_LOC | 6704 | 14 |
| Access Predicates | | | |
| COL.LOCATION=from$_subquery$_005.INT_LOCATION | | | |
| TABLE ACCESS (BY INDEX ROWID) | Collisions | 6704 | 1094 |
| INDEX (FAST FULL SCAN) | VICT_CASE_ID_AGE | 4082685 | 3675 |

**Query 7:** Find all collisions that satisfy the following: the collision was of type pedestrian and all victims were above 100 years old. For each of the qualifying collisions, show the collision id and the age of the eldest collision victim.

**<Initial Running time/IO:** 4.812s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT C.CASE_ID, MAX(V.VICTIM_AGE)FROM "Collisions"... | login_G71 | 31.05.21 12:26 | SQL | 32 | 4.812 |

**Optimized Running time/IO:** 2.832s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT C.CASE_ID, MAX(V.VICTIM_AGE)FROM "Collisions"... | login_G71 | 31.05.21 12:26 | SQL | 33 | 2.832 |

**Explain the improvement:**
Optimized query:
We use the same logic as before. By creating an index on (case_id, collision_type) in the Collisions table and using the previously created index VICT_CASE_ID_AGE (case_id,victim_age), we managed to eliminate the full table access and therefore drop the running time of the query

ALTER INDEX VICT_CASE_ID_AGE VISIBLE;
ALTER INDEX COL_CASE_ID_TYPE VISIBLE;

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

**Initial plan**

| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 22682 | 22214 |
| FILTER | | | |
| Filter Predicates | | | |
| MIN(V.VICTIM_AGE)>100 | | | |
| HASH (GROUP BY) | | 22682 | 22214 |
| HASH JOIN | | 453632 | 20492 |
| Access Predicates | | | |
| C.CASE_ID=V.CASE_ID | | | |
| NESTED LOOPS | | 402252 | 8864 |
| TABLE ACCESS (BY INDEX ROWID) | Type_of_collision | 1 | 1 |
| INDEX (UNIQUE SCAN) | TYPE_OF_COLLISION_TYPE_OF_COLLISION_UINDEX | 1 | 0 |
| Access Predicates | | | |
| COLTYPE.TYPE_OF_COLLISION='pedestrian' | | | |
| TABLE ACCESS (FULL) | Collisions | 402252 | 8863 |
| Filter Predicates | | | |
| C.TYPE_OF_COLLISION=COLTYPE.INT_COLL_TYPE | | | |
| TABLE ACCESS (FULL) | Victims | 4082685 | 6077 |

**Improved plan>**

| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 22682 | 14077 |
| FILTER | | | |
| Filter Predicates | | | |
| MIN(V.VICTIM_AGE)>100 | | | |
| HASH (GROUP BY) | | 22682 | 14077 |
| HASH JOIN | | 453632 | 12354 |
| Access Predicates | | | |
| C.CASE_ID=V.CASE_ID | | | |
| NESTED LOOPS | | 402252 | 3128 |
| TABLE ACCESS (BY INDEX ROWID) | Type_of_collision | 1 | 1 |
| INDEX (UNIQUE SCAN) | TYPE_OF_COLLISION_TYPE_OF_COLLISION_UINDEX | 1 | 0 |
| Access Predicates | | | |
| COLTYPE.TYPE_OF_COLLISION='pedestrian' | | | |
| INDEX (FAST FULL SCAN) | COL_CASE_ID_TYPE | 402252 | 3127 |
| Filter Predicates | | | |
| C.TYPE_OF_COLLISION=COLTYPE.INT_COLL_TYPE | | | |
| INDEX (FAST FULL SCAN) | VICT_CASE_ID_AGE | 4082685 | 3675 |

**Query 9:** Find the top-10 (with respect to number of collisions) cities. For each of these cities, show the city location and number of collisions.

**<Initial Running time/IO:** 2.591s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT L.COUNTY_CITY_LOCATION, COUNT(*) NUMBER_O... | login_G71 | 31.05.21 13:49 | SQL | 17 | 2.591 |

**Optimized Running time/IO:** 0.91s

| SQL | Connexion | Horodatage | Type | Exécuté | Durée (secondes) |
|---|---|---|---|---|---|
| SELECT L.COUNTY_CITY_LOCATION, COUNT(*) NUMBER_O... | login_G71 | 31.05.21 13:49 | SQL | 18 | 0.91 |

**Explain the improvement:**

Here we use the index COL_LOC that was created on the Location attribute of Collisions to optimize the join operation (C.LOCATION=L.INT_LOCATION) between the Collisions table and the dictionary Location.

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

ALTER INDEX COL_LOC VISIBLE;

### Initial plan

| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 10 | 9046 |
| SORT (ORDER BY) | | 10 | 9046 |
| VIEW | | 10 | 9045 |
| Filter Predicates | | | |
| from$_subquery$_003.rowlimit_$$_rownumber<=10 | | | |
| WINDOW (SORT PUSHED RANK) | | 540 | 9045 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=10 | | | |
| HASH (GROUP BY) | | 540 | 9045 |
| HASH JOIN | | 3620270 | 8861 |
| Access Predicates | | | |
| C.LOCATION=L.INT_LOCATION | | | |
| TABLE ACCESS (FULL) | Location | 540 | 3 |
| TABLE ACCESS (FULL) | Collisions | 3620270 | 8849 |

### Improved plan>

| OPERATION | OBJECT_NAME | CARDINALITY | COST |
|---|---|---|---|
| SELECT STATEMENT | | 10 | 2189 |
| SORT (ORDER BY) | | 10 | 2189 |
| VIEW | | 10 | 2188 |
| Filter Predicates | | | |
| from$_subquery$_003.rowlimit_$$_rownumber<=10 | | | |
| WINDOW (SORT PUSHED RANK) | | 540 | 2188 |
| Filter Predicates | | | |
| ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC )<=10 | | | |
| HASH (GROUP BY) | | 540 | 2188 |
| HASH JOIN | | 3620270 | 2004 |
| Access Predicates | | | |
| C.LOCATION=L.INT_LOCATION | | | |
| NESTED LOOPS | | 3620270 | 2004 |
| STATISTICS COLLECTOR | | | |
| TABLE ACCESS (FULL) | Location | 540 | 3 |
| INDEX (RANGE SCAN) | COL_LOC | 6704 | 1991 |
| Access Predicates | | | |
| C.LOCATION=L.INT_LOCATION | | | |
| INDEX (FAST FULL SCAN) | COL_LOC | 3620270 | 1991 |

# General Comments

All team members have contributed to this work. Each team member has focused on a specific part of the project (ER diagram, Data cleaning and preparation in Python , DDL, Queries and Optimization) while helping the others with their parts.