Driving Experiences Project

Emiliya Yavarova CS-22

The object of this report is to explain the creation of a database for the Driving Experiences website.

We have a webpage for entering various Driving Experiences for adolescents younger than 18 years old. We have the date, start time and end time for the experience and the distance in kilometers, which are attributes of the Driving Experience that we enter in the web form.

We have various weather conditions, road type, heaviness of the traffic, maneuvers and transmission type, which are their own entities and belong to the Driving Experience.

The values can be the following:

Weather: rainy, sunny, snowy, cloudy, misty-foggy, sandstorm, stormy, tempestuous, partlyCloudy, clear

Road types: concrete, rocks, ice, sand, mud-dirt

Heaviness of traffic: none, light, heavy

Transmission type: automatic, manual

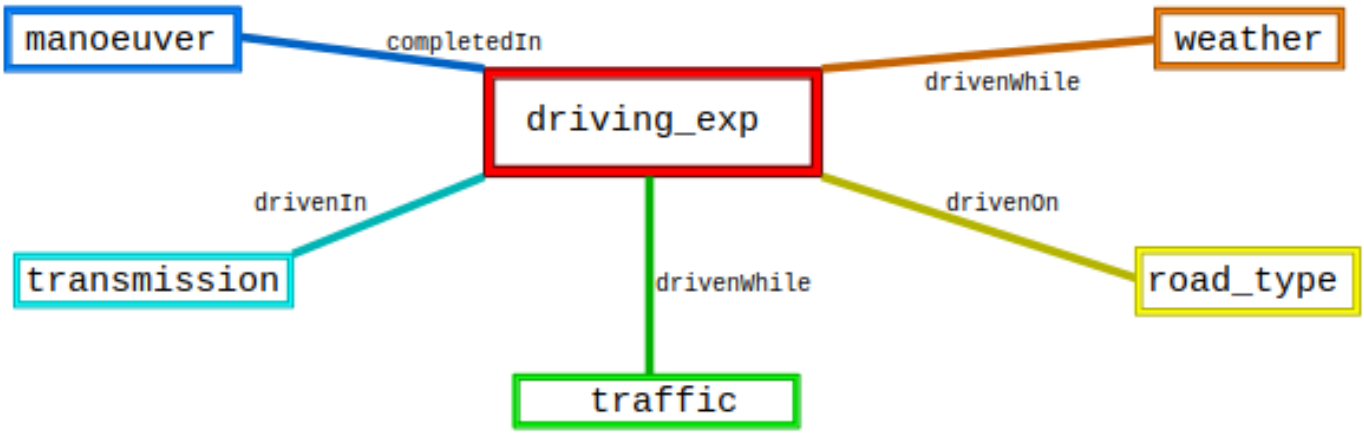
Manoeuvers: reversing, u-turn, hill-start, parallel-parking, gear-changing, three-point turn

The relationship between them is:

Weather – Driving_Experience	One – Many
Road_type – Driving_Experience	Many – Many
Heaviness_of_traffic – Driving_Experience	One – Many
Transmission_type – Driving_Experience	One – Many
Maneuver – Driving_Experience	Many – Many

Now, we begin with the CDM model of our entities and relationships.

STEP 1: CDM



And here is our CDM in text format:

[weather](drivenWhile)[driving_exp]

[road_type](drivenOn)[driving_exp]

[traffic](drivenWhile)[driving_exp]

[transmission](drivenIn)[driving_exp]

[manoeuver](completedIn)[driving_exp]

Now, we continue with creating an LDM and a Data Dictionary.

STEP 2: LDM AND DD

Attribute/Label	Description	Type	Size
driving_exp_id	Id number of the experience	INT	11
date	Date of the experience	DATE	
start_time	Start time of the exp	TINYINT	4
end_time	End time of the exp	TINYINT	4
distance_km	Distance covered (in km)	INT	11
weather_id	Road type id number	TINYINT	4
weather_type	Weather type	VARCHAR	50
road_id	Road type id number	TINYINT	4
road_type	Road type	VARCHAR	50
traffic_id	Traffic heaviness id number	TINYINT	4
traffic_type	Traffic heaviness type	VARCHAR	50
transmission_id	Transmission id number	INT	4
transmission_type	Transmission type	VARCHAR	50
manoeuver_id	Manoeuver id number	TINYINT	4
manoeuver_type	Maoeuver type	VARCHAR	50

Here it is in text format:

[driving_exp](driving_exp_id, date, start_time, end_time, distance_km)<INT, DATE, TINYINT, TINYINT, INT>

[weather](weather_id, weather_type)<TINYINT, VARCHAR>

[road_type](road_id, road_type)<TINYINT, VARCHAR>

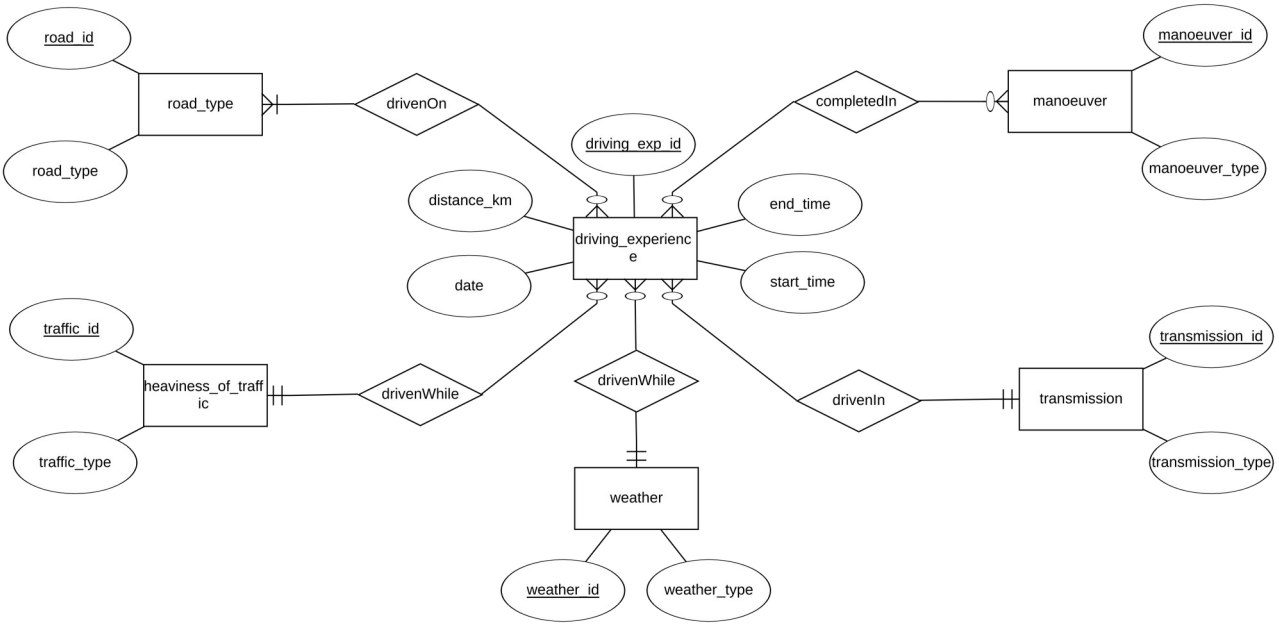
[traffic](traffic_id, traffic_type)<TINYINT, VARCHAR>

[transmission](transmission_id, transmission_type)<TINYINT, VARCHAR>

[manoeuver](manoeuver_id, manoeuver_type)<TINYINT, VARCHAR>

We also have our graph for the LDM:

ERD



Our text format of it is:

[weather](>011)[driving_experience](1-to-many)

[heaviness_of_traffic](>011)[driving_experience](1-to-many)

[transmission](>011)[driving_experience](1-to-many)

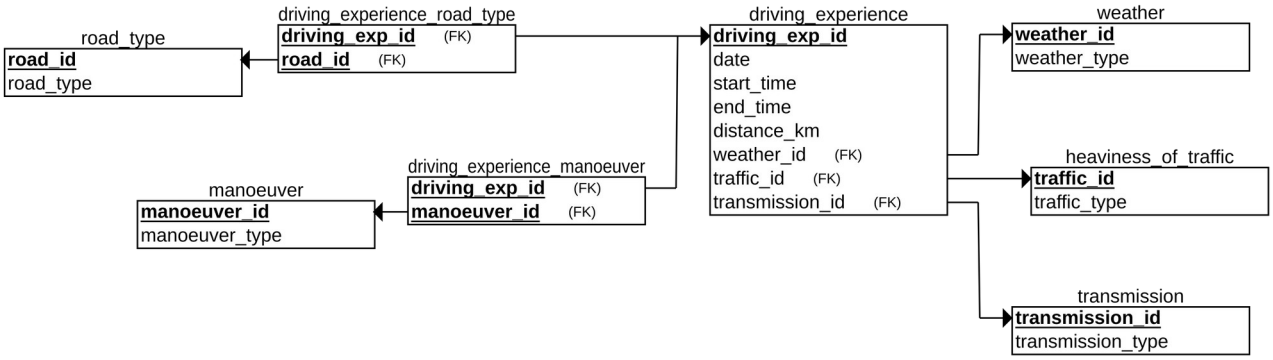
[road_type](>01<)[driving_experience](many-to-many)

[manoeuver](>00<)[driving_experience](many-to-many)

Now, we can move on to the Relational Schema of our database.

STEP 3: RS

In the Relational Schema, we have two additional entities driving_experience_road_type and driving_experience_manoeuver. But why? The reason is because we have a many-to-many relationship. To store the data correctly, we need another table to connect experiences and manoeuvers/experiences and road types, hence the corresponding tables.



[driving_experience, PK:driving_exp_id, FK:weather_id](>011)[weather, PK:weather_id]

[driving_experience, PK:driving_exp_id, FK:traffic_id](>011)[heaviness_of_traffic, PK:traffic_id]

[driving_experience, PK:driving_exp_id, FK:transmission_id](>011)[transmission, PK:transmission_id]

[driving_experience, PK: driving_exp_id](>01<)[driving_experience_road_type, FK: driving_exp_id, FK: road_id]

[driving_experience_road_type, FK: driving_exp_id, FK: road_id](>10<)[road_type, PK: road_id]

[driving_experience, PK: driving_exp_id](>00<)[driving_experience_manoeuver, FK: driving_exp_id, FK: manoeuver_id]

[driving_experience_road_type, FK: driving_exp_id, FK: manoeuver_id](>00<)[manoeuver, PK: manoeuver_id]

driving_experience (PK:driving_exp_id,date,start_time,end_time,distance_km,FK:weather_id)

weather (PK:weather_id,weather_type)

driving_experience (PK:driving_exp_id,date,start_time,end_time,distance_km,FK:traffic_id)

traffic (PK:traffic_id,traffic_type)

driving_experience (PK:driving_exp_id,date,start_time,end_time,distance_km,FK:transmission_id)

transmission (PK:transmission_id,transmission_type)

driving_experience (PK:driving_exp_id,date,start_time,end_time,distance_km)

driving_experience_manoeuver(FK:driving_exp_id,FK:manoeuver_id)

manoeuver(PK:manoeuver_id,manoeuver_type)

driving_experience (PK:driving_exp_id,date,start_time,end_time,distance_km)

driving_experience_road_type(FK:driving_exp_id,FK:road_id)

road_type(PK:road_id,road_type)

From our Relational Schema, we get a database by importing it to PHPMyAdmin. By some tweaks we get a

DATABASE

	Table	Action	Rows	Type	Collation	Size
☐	driving_experience	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	80.0 KiB
☐	driving_experience_manoeuver	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	32.0 KiB
☐	driving_experience_road_type	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	32.0 KiB
☐	heaviness_of_traffic	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB
☐	manoeuver	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	16.0 KiB
☐	road_type	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB
☐	transmission	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB
☐	weather	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB
	8 tables	Sum	75	InnoDB	utf8mb4_general_ci	224.0 KiB

traffic_id	traffic_type
1	light
2	none
3	heavy

weather_id	weather_type
1	rainy
2	sunny
3	snowy
4	cloudy
5	mistyFoggy
6	sandstorm
7	stormy
8	tempestuous
9	partlyCloudy
10	clear

manoeuver_id	manoeuver_type
1	reversing
2	u-turn
3	hill-start
4	parallel-parking
5	gear-changing
6	three-point turn

transmission_id	transmission_type
1	automatic
2	manual

road_id	road_type
1	concrete
2	rocks
3	ice
4	sand
5	mudOrDirt

driving_exp_id	manoeuver_id
1	1
2	2
3	6
4	4
5	5
6	1
7	1
8	2
9	4

driving_exp_id	road_id
1	1
2	2
3	3
4	4
5	5
6	1
7	2
8	3
9	4
10	5
11	1
12	2
13	3
14	4
15	5
16	1
17	2
18	3
19	4
20	5

date	start_time	end_time	distance_km	driving_exp_id	weather_id	traffic_id	transmission_id
2022-04-01	08:00:00	09:30:00	20	1	2	1	1
2022-04-02	10:15:00	12:00:00	15	2	3	2	2
2022-04-03	14:30:00	16:00:00	25	3	1	3	1
2022-04-04	17:00:00	18:45:00	30	4	2	1	2
2022-04-05	09:45:00	11:20:00	18	5	3	2	1
2022-04-06	13:20:00	15:00:00	22	6	1	3	2
2022-04-07	16:30:00	18:10:00	28	7	2	1	1
2022-04-08	11:00:00	12:40:00	16	8	3	2	2
2022-04-09	14:45:00	16:30:00	24	9	1	3	1
2022-04-10	18:15:00	20:00:00	35	10	2	1	2
2022-04-11	08:30:00	10:15:00	23	11	4	2	1
2022-04-12	12:00:00	13:45:00	19	12	5	3	2
2022-04-13	15:45:00	17:30:00	27	13	6	1	1
2022-04-14	19:00:00	20:45:00	32	14	7	2	2
2022-04-15	09:15:00	11:00:00	21	15	8	3	1
2022-04-16	13:00:00	14:45:00	17	16	9	1	2
2022-04-17	16:45:00	18:30:00	26	17	10	2	1
2022-04-18	10:30:00	12:15:00	29	18	1	3	2
2022-04-19	14:15:00	16:00:00	34	19	2	1	1
2022-04-20	18:00:00	19:45:00	36	20	3	2	2

SQL CODE FOR THE DATABASE GENERATED BY ERD PLUS:

```
CREATE TABLE weather
(
  weather_id INT NOT NULL,
  weather_type INT NOT NULL,
  PRIMARY KEY (weather_id)
);

CREATE TABLE road_type
(
  road_id INT NOT NULL,
  road_type INT NOT NULL,
  PRIMARY KEY (road_id)
);

CREATE TABLE heaviness_of_traffic
(
  traffic_id INT NOT NULL,
  traffic_type INT NOT NULL,
  PRIMARY KEY (traffic_id)
);

CREATE TABLE transmission
(
  transmission_id INT NOT NULL,
  transmission_type INT NOT NULL,
  PRIMARY KEY (transmission_id)
);

CREATE TABLE manoeuver
(
  manoeuver_id INT NOT NULL,
  manoeuver_type INT NOT NULL,
  PRIMARY KEY (manoeuver_id)
);

CREATE TABLE driving_experience
(
  date INT NOT NULL,
  start_time INT NOT NULL,
  end_time INT NOT NULL,
  distance_km INT NOT NULL,
  driving_exp_id INT NOT NULL,
  weather_id INT NOT NULL,
  traffic_id INT NOT NULL,
  transmission_id INT NOT NULL,
  PRIMARY KEY (driving_exp_id),
  FOREIGN KEY (weather_id) REFERENCES weather(weather_id),
  FOREIGN KEY (traffic_id) REFERENCES heaviness_of_traffic(traffic_id),
  FOREIGN KEY (transmission_id) REFERENCES transmission(transmission_id)
);

CREATE TABLE driving_experience_road_type
(
  driving_exp_id INT NOT NULL,
  road_id INT NOT NULL,
  PRIMARY KEY (driving_exp_id, road_id),
  FOREIGN KEY (driving_exp_id) REFERENCES driving_experience(driving_exp_id),
  FOREIGN KEY (road_id) REFERENCES road_type(road_id)
);

CREATE TABLE driving_experience_manoeuver
(
  driving_exp_id INT NOT NULL,
  manoeuver_id INT NOT NULL,
  PRIMARY KEY (driving_exp_id, manoeuver_id),
  FOREIGN KEY (driving_exp_id) REFERENCES driving_experience(driving_exp_id),
  FOREIGN KEY (manoeuver_id) REFERENCES manoeuver(manoeuver_id)
);
```

SQL QUERIES

Query to see all the experiences where the road type was “rocks”

```
SELECT de.date, de.start_time, de.end_time, de.distance_km, w.weather_type, rt.road_type, ht.traffic_type, t.transmission_type FROM driving_experience de JOIN driving_experience_road_type dert ON de.driving_exp_id = dert.driving_exp_id JOIN road_type rt ON dert.road_id = rt.road_id JOIN weather w ON de.weather_id = w.weather_id JOIN heaviness_of_traffic ht ON de.traffic_id = ht.traffic_id JOIN transmission t ON de.transmission_id = t.transmission_id WHERE rt.road_type = 'rocks';
```

date	start_time	end_time	distance_km	weather_type	road_type	traffic_type	transmission_type
2022-04-07	16:30:00	18:10:00	28	sunny	rocks	light	automatic
2022-04-17	16:45:00	18:30:00	26	clear	rocks	none	automatic
2022-04-02	10:15:00	12:00:00	15	snowy	rocks	none	manual
2022-04-12	12:00:00	13:45:00	19	mistyFoggy	rocks	heavy	manual

Select Columns: Choose specific details like date, start time, end time, distance, weather type, road type, traffic type, and transmission type.

From Table: Start with the main table containing driving experiences.

Join Tables: Bring in additional information from related tables like road types, weather, traffic, and transmission.

Filter Data: Include only records where the road type is "rocky".

Query to see the most frequent weather type in the database

```
SELECT w.weather_type, COUNT(*) AS experience_count FROM driving_experience de JOIN weather w ON de.weather_id = w.weather_id GROUP BY w.weather_type ORDER BY experience_count DESC LIMIT 1;
```

weather_type	experience_count
sunny	5

Select Columns: Select the weather type and count how many times each weather type appears.

From Table: Start with the table containing driving experiences.

Join Tables: Join the weather table to get the descriptive names of weather types.

Group Data: Group the results by weather type so we can count how many experiences there are for each type.

Order Results: Arrange the weather types in descending order based on the count of experiences.

Limit: Show only the top result, which will be the weather type with the most experiences.

PART 2

Initial Normalized Database Structure

Normalization ensures that updating a piece of data only requires modifying one document, as the data is defined in a single place. For example, collections such as Weather, Road_Condition, Traffic_Condition, Parking_Maneuver, and Driving_Experience are stored separately, which prevents redundant data storage. Each field represents a unique aspect of the driving experience.

To justify the atomic nature of fields in the Driving_Experience collection, the following aggregation pipeline was written:

```
[
  {
    "$unwind": "$data"
  },
  {
    "$lookup": {
      "from": "Weather",
      "let": { "weather_id": "$data.weather_id" },
      "pipeline": [
        { "$unwind": "$data" },
        { "$match": { "$expr": { "$eq": ["$data.weather_id", "$$weather_id"] } } },
        { "$project": { "weather_type": "$data.weather_type" } }
      ],
      "as": "weather_info"
    }
  },
  {
    "$unwind": "$weather_info"
  },
  {
    "$project": {
      "_id": 0,
      "Dr_exp_ID": "$data.Dr_exp_ID",
      "Date": "$data.Date",
      "start_time": "$data.start_time",
      "end_time": "$data.end_time",
      "tot_time": "$data.tot_time",
      "distance_km": "$data.distance_km",
      "weather_id": "$data.weather_id",
      "weather_type": "$weather_info.weather_type"
    }
  }
]
```

Here, weather_id in Driving_Experience refers to an entry in the Weather collection. This setup ensures that any update to the weather type needs to be made only once in the Weather collection, not in every driving experience record.

Denormalized Data Structure

In a denormalized structure, related data that would typically be stored in separate collections is combined into a single document. This eliminates the need for references (foreign keys), as related data is embedded directly within the main document.

Example of a denormalized structure:

```
{
  "Dr_exp_ID": 1,
  "Date": "2023-05-01",
  "start_time": "00:08:00",
  "end_time": "00:08:30",
  "tot_time": "00:00:30",
  "distance_km": 10,
  "weather": {
    "weather_id": 1,
    "weather_type": "Sunny"
  },
  "road_condition": {
    "road_cond_id": 1,
    "road_cond_type": "Dry"
  },
  "traffic_condition": {
    "traffic_cond_id": 1,
    "traffic_cond_type": "Light"
  },
  "maneuvers": [
    {
      "maneuver_id": 1,
      "maneuver_type": "Angle Parking"
    },
    {
      "maneuver_id": 2,
      "maneuver_type": "Parallel Parking"
    }
  ]
}
```

Benefits of Denormalization

Accessing related data does not require joins, as all relevant information is stored together.It reduces latency, which is crucial for real-time applications. Queries become simpler, requiring fewer operations to fetch data. Updates to related data: either the entire document is updated, or none of it is, simplifying error handling.

Aggregation Pipeline

Denormalized_Driving_Experience Collection:

```
{
  "Dr_exp_ID": 1,
  "Date": "2023-05-01",
  "start_time": "00:08:00",
  "end_time": "00:08:30",
  "tot_time": "00:00:30",
  "distance_km": 10,
  "weather_type": "Sunny",
  "road_condition_type": "Dry",
  "traffic_condition_type": "Light",
  "maneuvers": [
    { "maneuver_id": 1, "maneuver_type": "Parallel Parking" }
  ]
}
```

This structure simplifies data retrieval by embedding all related information within a single document, optimizing query performance and making the data easier to manage in a MongoDB environment.