

Lyft - Ride Sharing Company (San - Francisco)



Guided by
Dr. Vidhyacharan Bhaskar
CS457

Presented by
Shoumya Singh
19566

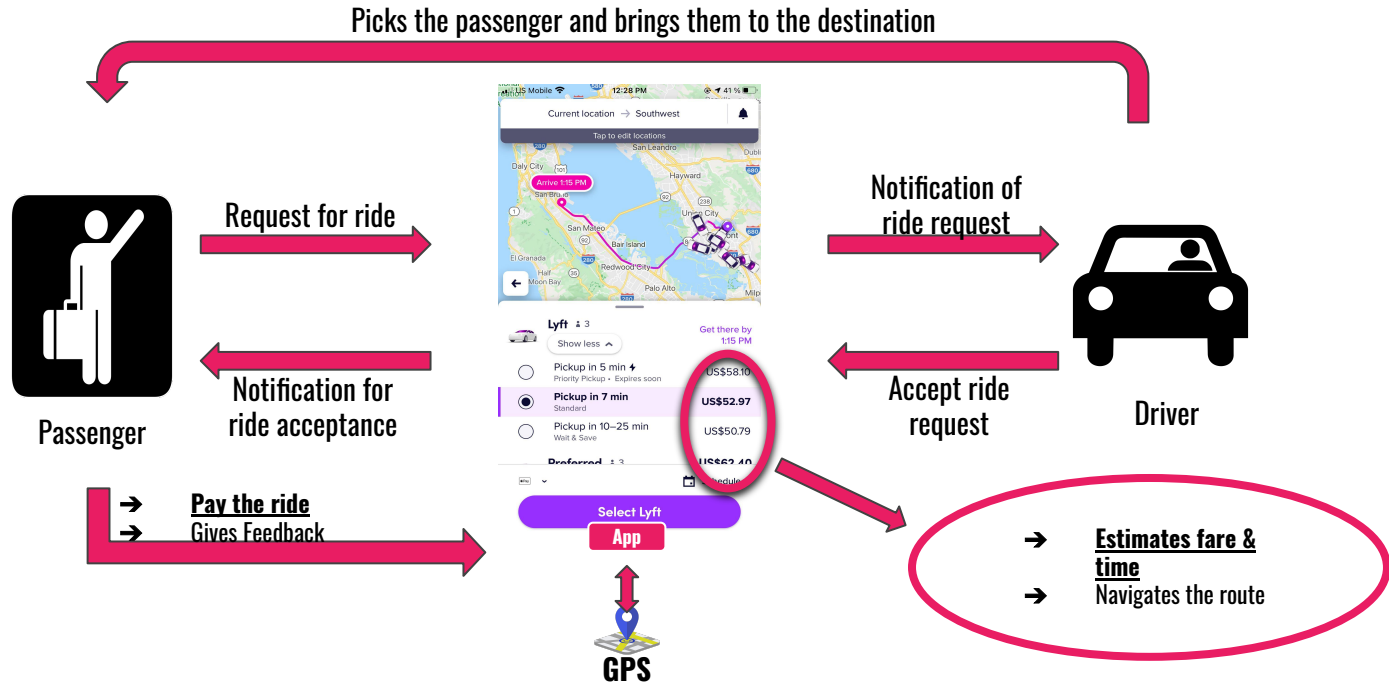
Contents

- ★ What is Lyft and it's Business Model?
- ★ Lyft Database Development Cycle
- ★ What is ER/EER Model?
 - Components of EER Model
 - Relationship Cardinality Ratio
 - Participation Constraints
- ★ Superclass & Subclass
- ★ Specialization & Generalization
- ★ Constraints on Specialization
- ★ EER diagram of Lyft -Ride sharing Company
- ★ Entity Tables
- ★ Queries

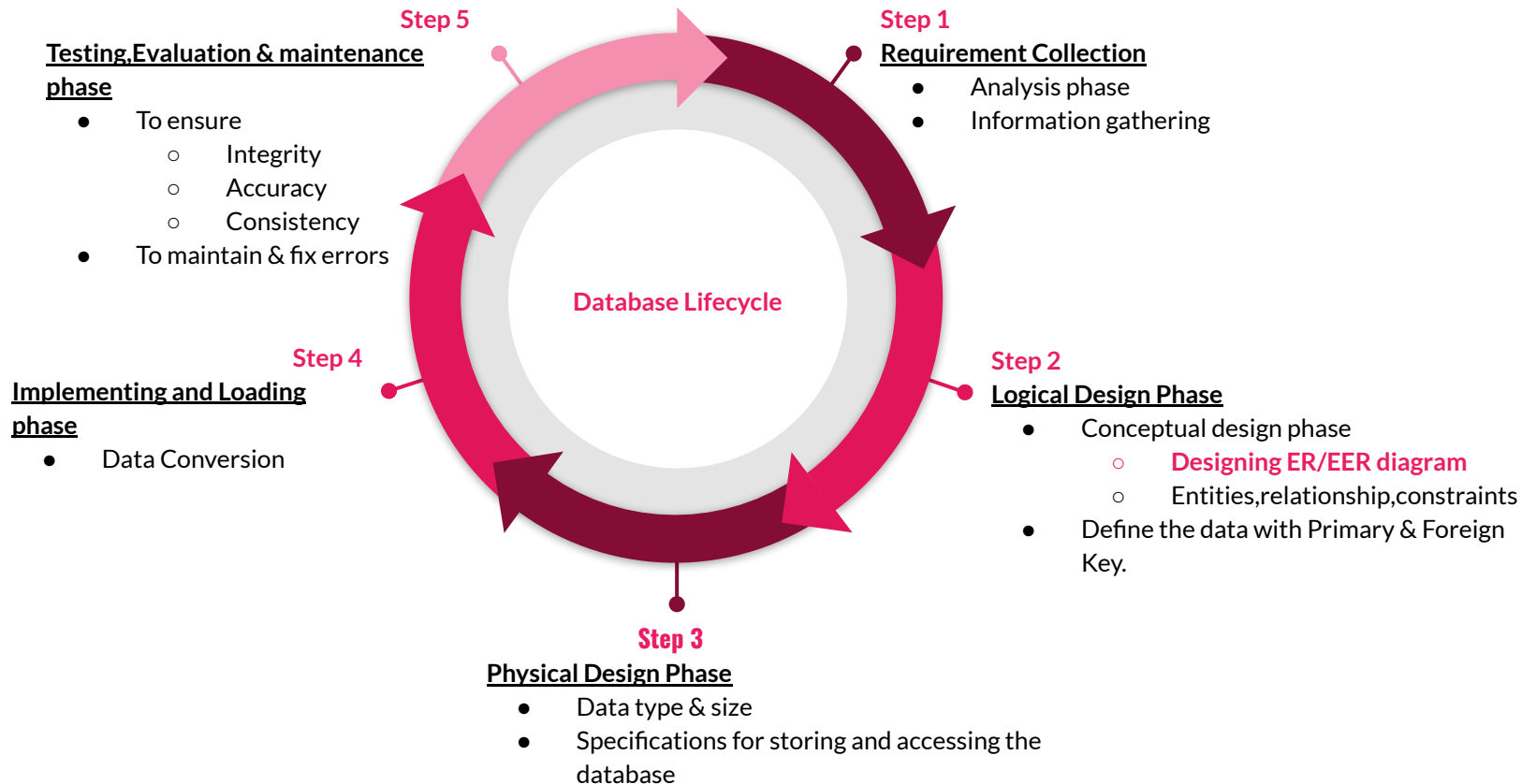


What is Lyft and it's Business Model?

- **Lyft, Inc.** develops, markets, and operates a mobile app, offering vehicles for local rides, motorized scooters, and a bicycle-sharing system.
- The company is based in San Francisco, California .
- The passenger can have a local ride car for pick up and drop off at the desired location.
- Lyft business model rotates around passenger ,driver and admin.



Lyft Database Development Cycle



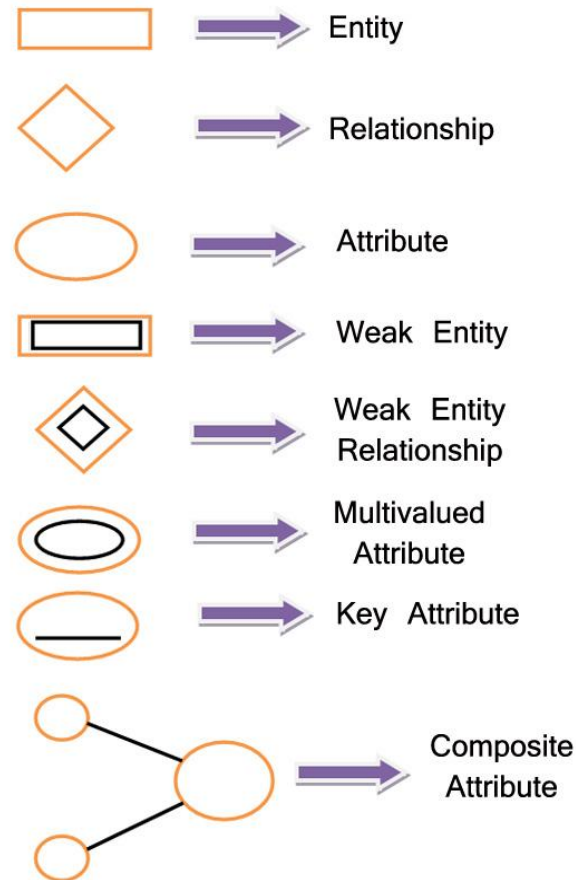
What is ER/EER Model?

- ER model stands for Entity Relationship model and EER model stands for Enhanced Entity Relationship model.
- In this project, EER model will be used for further .
- EER model creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model and its the detailed version of ER model.
- It includes the concept of specialization and generalization.
- Each entity, attribute, and relationship, should have appropriate names that can be easily understood by the non-technical people as well.

Components of EER Model

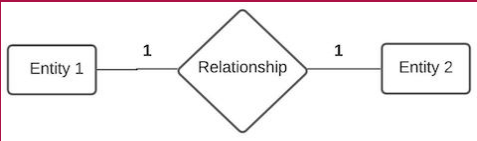
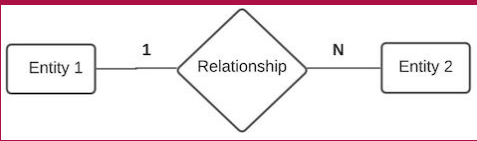
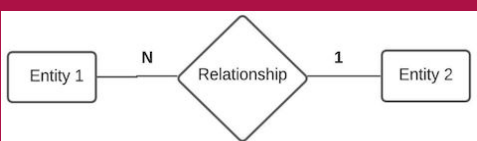
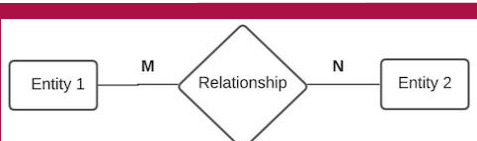
→ There are three main components

- ◆ **Entity** - An entity is an object or component of data.
 - Strong Entity: An entity which doesn't depend on other entity.
 - Weak Entity: An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity.
- ◆ **Attributes** - An attribute describes the property of an entity.
 - Key Attribute: It represents key attribute of an entity which have a unique value in a table also known as Primary Key.
 - Derived Attribute: It represents the derived attribute which can be derived from the value of related attribute.
 - Multivalued Attribute: It represents multivalued attribute which can have many values for a particular entity.
 - Composite Attribute: A composite attribute can be subdivided into smaller components which further form attributes.
- ◆ **Relationship**- Each entity is related to another entity with a relationship (Relationship cardinality).
 - Strong Relationship
 - Weak relationship



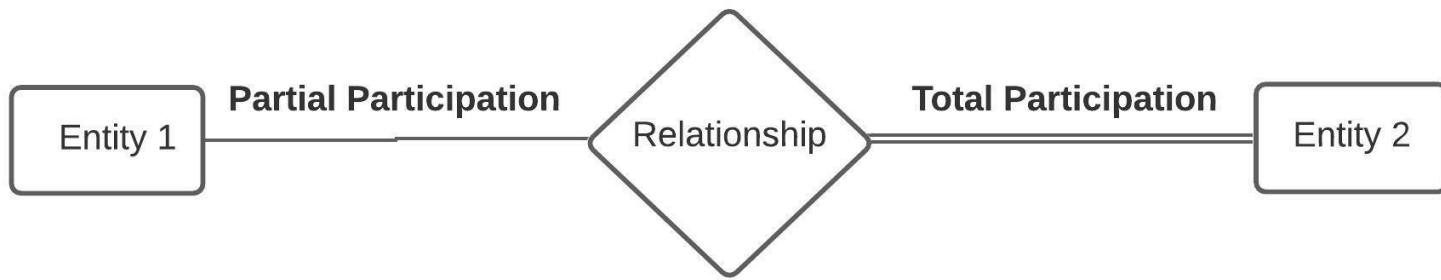
Relationship Cardinality Ratio

It shows the relationship among the entities. There are four types of entity relationship ratios:

One to One (1 : 1)		→ When a single instance of an entity is associated with a single instance of another entity then it is called <u>one to one relationship</u> .
One to Many (1 : N)		→ When a single instance of an entity is associated with more than single instance of another entity then it is called <u>one to many relationship</u> .
Many to One (N : 1)		→ When more than single instance of an entity is associated with a single instance of another entity then it is called <u>many to one relationship</u> .
Many to Many (M : N)		→ When more than single instance of an entity is associated with more than single instance of another entity then it is called <u>many to many relationship</u> .

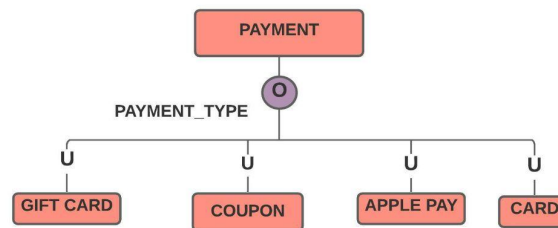
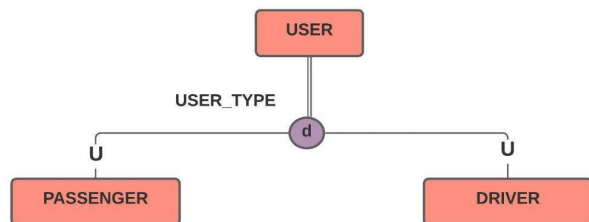
Participation Constraints

- **Total Participation** – Each entity is involved in the relationship. Total participation is represented by double lines. **Ex: User in LYFT has total participation. A user must be a PASSENGER or DRIVER.**
- **Partial participation** – Not all entities are involved in the relationship. Partial participation is represented by single lines. **Ex: PAYMENT to CARD type it is a partial participation.**



Superclass & Subclass

- Subclass and Superclass relationship leads the concept of Inheritance.
- Superclass is an entity type that has a relationship with one or more subtypes.
- Subclass inherits properties and attributes from its superclass.
- In this project we are using it in two entity USER and PAYMENT.
- USER entity type is a superclass to 2 different subclasses based on the USER_TYPE as PASSENGER and DRIVER.
- PAYMENT entity type is a superclass to 4 different subclasses based on the PAYMENT_TYPE as GIFT_CARD, COUPON, APPLE_PAY and CARD.



Specialization & Generalization

- **Generalization** is the process of generalizing the entities which contains the properties of all the generalized entities which means two lower level entities combine to form a higher level entity.

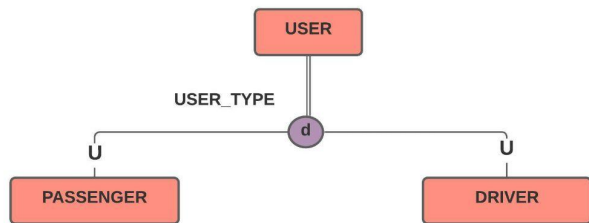
Example: PASSENGER and DRIVER can be generalized as USERS.

- **Specialization** is a process of defining a set of subclasses of an entity type based on their characteristics, this entity type is called the superclass of the specialization. IT means one higher entity can be broken down into many lower level entities.

Example: USER can be specialized as PASSENGER or DRIVER based on the USER_TYPE.

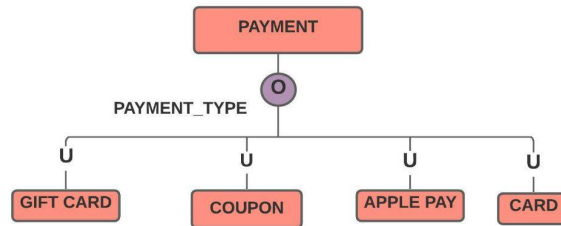
- Specialization has some constraints as Disjoint and Overlapping.

Constraints on Specialization



Disjoint Specialization

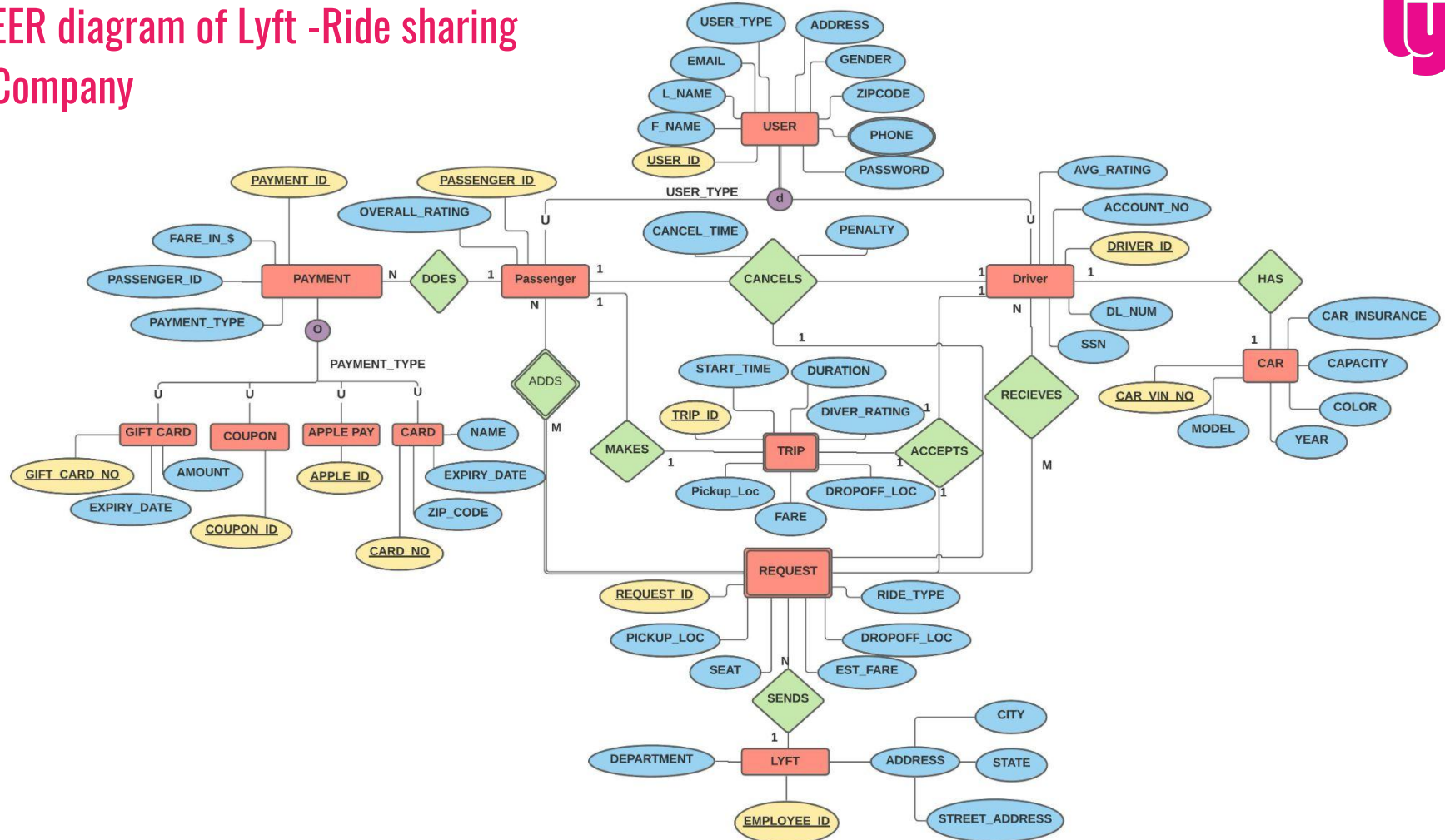
- Disjoint specialization means that an entity can be a member of at most one of the subclasses of the specialization.
- In this project USER entity can be only one of the following category at a time that is PASSENGER or DRIVER. It cannot be both at the same time.



Overlapping Specialization

- Overlapping specialization means that an entity may be a member of more than one subclass of the specialization.
- In this project PAYMENT can be done in more than one of the following category at a time that is through GIFT_CARD, COUPON, APPLE_PAY and CARD.

EER diagram of Lyft -Ride sharing Company



Relationship Types & Cardinality Ratio

- **ADDS** - This relationship relates PASSENGER and REQUEST. The cardinality ratio is N : M as multiple passenger can add multiple requests.
- **DOES** - This relationship relates PASSENGER and PAYMENT. The cardinality ratio is 1 : N as one passenger can do multiple methods of payment.
- **ACCEPTS** - This relationship relates DRIVER and REQUEST. The cardinality ratio is 1 : 1 as one driver can accepts one requests.

Entity	Relationship	Cardinality Ratio	Entity
PASSENGER	ADDS	N : M	REQUEST
PASSENGER	DOES	1 : N	PAYMENT
PASSENGER	MAKES	1 : 1	TRIP
PASSENGER	CANCELS	1 : 1	REQUEST
DRIVER	ACCEPTS	1 : 1	REQUEST
DRIVER	RECIEVE	N : M	REQUEST
DRIVER	HAS	1 : 1	CAR
DRIVER	CANCELS	1 : 1	REQUEST
DRIVER	ACCEPTS	1 : 1	TRIP
REQUEST	SENDS	N : 1	LYFT

Entity Tables

USER Entity Table

```
MariaDB [19566ss]> select * from USER;
```

USER_ID	F_NAME	L_NAME	E_MAIL	ADDRESS	ZIPCODE	GENDER	PHONE_NO	PASSWORD	USER_TYPE
1	David	Warner	d_warner@gmail.com	1401 Taylor Street, San Jose, CA	95119	M	5012346791	thanks@1401	Driver
2	Bettina	Pauley	bettina_pauley@gmail.com	789 Logan Street, San Francisco, CA	95111	F	5082346793	hell@123	Driver
3	Justin	Dsouza	justin_dsouza@gmail.com	567 Stevenson Street, Richmond, CA	95131	M	1282346793	djdj@123	Driver
4	James	Smith	j_smith@gmail.com	123 Belmont Street, Fresno, CA	94567	M	1282899793	mith_jam	Passenger
5	Emily	Cortes	emily_cortes@gmail.com	123 Morwy Ave, Fremont, CA	94536	F	1289999793	cortes_34	Passenger
6	Sherlock	Holmes	holmes123@gmail.com	458 Walnut Creek, San Francisco, CA	95671	M	1289999788	4hol123	Passenger

6 rows in set (0.000 sec)

- USER_ID: Primary Key
- F_NAME and L_NAME VARCHAR character range 15
- E_MAIL VARCHAR character range 40
- ADDRESS VARCHAR character range 50
- ZIPCODE VARCHAR character range 5
- GENDER VARCHAR(1)
- PHONE_NO VARCHAR(10)
- PASSWORD VARCHAR(15)
- USER_TYPE VARCHAR(10)
- USERS entity type is superclass to 2 different subclasses based on USER_TYPE , DRIVER and PASSENGER.
- This specialization is disjoint (d) meaning, users cannot be of more than one user type at a time .

Entity Tables

DRIVER Entity Table

```
[MariaDB [19566ss]> select * from DRIVER;
```

DRIVER_ID	DL_NUM	SSN	ACCOUNT_NO	AVG_RATING	USER_ID
1	43567283	123456789	1001001234	5	1
2	43567457	133488789	1111001234	4.8	3
3	89567457	135588789	1111003334	4.5	2

```
3 rows in set (0.000 sec)
```

- DRIVER_ID: Primary Key
- DL_NUM VARCHAR character range 8
- SSN Integer not NULL
- ACCOUNT_NO not NULL
- AVG_RATING FLOAT
- USER_ID: Foreign Key reference from USER Table.

PASSENGER Entity Table

```
[MariaDB [19566ss]> select * from PASSENGER;
```

PASSENGER_ID	OVERALL_RATING	USER_ID
1	5	4
2	4.1	5
3	4.7	6

```
3 rows in set (0.000 sec)
```

- PASSENGER_ID: Primary Key
- OVERALL_RATING FLOAT
- USER_ID: Foreign Key reference from USER Table.

Entity Tables

CAR Entity Table

MariaDB [19566ss]> select * from CAR;

CAR_VIN_NO	COLOR	YEAR	CAPACITY	CAR_INSURANCE	DRIVER_ID	MODEL	MAKE
W14550ck78k09we97	White	2020	5	129-6700-8991	1	SUV	MERCEDES
W17890jm78k09we91	Black	2019	4	123-4567-8901	2	SEDAN	TOYOTA
W18950kk78k09fg66	Red	2018	4	199-2300-0990	3	SEDAN	NISSAN

3 rows in set (0.000 sec)

- CAR_VIN_NO: Primary Key
- MODEL VARCHAR character 15
- MAKE VARCHAR character 15
- COLOR VARCHAR character 10
- YEAR Integer not NULL
- CAPACITY Integer not NULL
- CAR_INSURANCE VARCHAR character 10
- DRIVER_ID: Foreign Key reference from DRIVER Table.

LYFT Entity Table

MariaDB [19566ss]> select* from LYFT;

EMPLOYEE_ID	DEPARTMENT	STREET_ADDRESS	CITY	STATE
10	Infrastructure	1Folsom Street	San Francisco	CA
11	Infrastructure	1Folsom Street	San Francisco	CA
12	Security	34Fresno Street	San Francisco	CA
13	Research	4Folsom Street	San Francisco	CA
14	Research	4Folsom Street	San Francisco	CA

5 rows in set (0.000 sec)

- EMPLOYEE_ID: Primary Key
- DEPARTMENT VARCHAR character 30
- STREET_ADDRESS VARCHAR character 15
- CITY VARCHAR character 15
- STATE VARCHAR character 5

Entity Tables

REQUEST Entity Table

```
[MariaDB [19566ss]> select * from REQUEST;
```

REQUEST_ID	PICKUP_LOC	SEAT	DROPOFF_LOC	RIDE_TYPE	EST_FARE_\$	USER_ID
1	123 Belmont Street	4	International SF Airport	Economy	34.4	4
2	International SF Airport	5	1458 Walnut Creek	Luxury	80.65	2
3	International SF Airport	4	123 Mowry Ave	Economy	54.3	5

```
3 rows in set (0.000 sec)
```

- It is weak entity as REQUEST is USER dependent.
- REQUEST_ID: Primary Key
- PICKUP_LOC VARCHAR character range 50
- SEAT Integer not NULL
- DROPOFF_LOC VARCHAR character range 50
- RIDE_TYPE VARCHAR 15
- EST_FARE_\$ in FLOAT
- USER_ID: Foreign Key reference from USER Table.

Entity Tables

TRIP Entity Table

MariaDB [19566ss]> select * from TRIP;

TRIP_ID	PICKUP_LOC	DROPOFF_LOC	START_TIME	DURATION	FARE_\$	DRIVER_RATING	REQUEST_ID	DRIVER_ID
1	International SF Airport	123 Mowry Ave	07:00 am	55 Min	60.34	4.8	3	1
2	International SF Airport	1458 Walnut Creek	10:00 pm	30 Min	81.25	4.9	2	2
3	123 Belmont Street	International SF Airport	10:00 am	40 Min	33.55	4.2	1	3
4	123 Belmont Street	567 Logan drive	11:00 pm	20 Min	20.39	4.5	4	1

4 rows in set (0.000 sec)

- It is weak entity as TRIP is REQUEST dependent which is indirectly USER dependent .
- TRIP_ID: Primary Key
- PICKUP_LOC VARCHAR character range 50
- DROPOFF_LOC VARCHAR character range 50
- START_TIME VARCHAR 10
- DURATION VARCHAR 10
- FARE_\$ in FLOAT
- DRIVER_RATING in FLOAT
- REQUEST_ID: Foreign Key reference from REQUEST Table.
- DRIVER_ID: Foreign Key reference from DRIVER Table.

Entity Tables

PAYMENT Entity Table

```
[MariaDB [19566ss]> select* from PAYMENT;
```

PAYMENT_ID	PAYMENT_TYPE	FARE_\$	PASSENGER_ID	TRIP_ID
1	APPLE PAY	81.25	2	2
2	GIFT_CARD	33.55	1	3
3	CARD	60.34	3	1
4	COUPON	20.39	4	4

4 rows in set (0.000 sec)

- PAYMENT_ID: Primary Key
- PAYMENT_TYPE VARCHAR character range 15
- FARE_\$ in FLOAT
- PASSENGER_ID: Foreign Key reference from PASSENGER Table.
- TRIP_ID: Foreign Key reference from TRIP Table.

- PAYMENT entity type is superclass to 4 different subclasses based on PAYMENT_TYPE , GIFT_CARD,APPLE_PAY,COUPON and CARD.
- This specialization is overlapping (o) meaning, payment can be made from one or more than one payment types at a time .

Queries

Query 1: Display detail information about completed trips.

```
[MariaDB [19566ss]> select R.REQUEST_ID,R.PICKUP_LOC,R.DROPOFF_LOC,U.USER_ID,U.L_NAME,U.F_NAME,U.USER_TYPE,T.TRIP_ID,T.FARE_$,P.PAYMENT_TYPE from ]
REQUEST R INNER JOIN USER U USING (USER_ID) INNER JOIN TRIP T USING (REQUEST_ID) INNER JOIN PAYMENT P USING (TRIP_ID) order by TRIP_ID;
```

REQUEST_ID	PICKUP_LOC	DROPOFF_LOC	USER_ID	L_NAME	F_NAME	USER_TYPE	TRIP_ID	FARE_\$	PAYMENT_TYPE
3	International SF Airport	123 Mowry Ave	5	Cortes	Emily	Passenger	1	60.34	APPLE PAY
2	International SF Airport	1458 Walnut Creek	2	Pauley	Bettina	Driver	2	81.25	APPLE PAY
1	123 Belmont Street	International SF Airport	4	Smith	James	Passenger	3	33.55	GIFT_CARD

3 rows in set (0.001 sec)

This query is giving the detailed information of the completed trips. Here we are joining 4 different tables REQUEST, USER, TRIP and PAYMENT on the common records using INNER JOINS.

Query 2: List of all the user who never book any ride.

```
MariaDB [19566ss]> select u.USER_ID, u.F_NAME,u.L_NAME,u.USER_TYPE,p.PASSENGER_ID FROM PASSENGER p INNER JOIN USER u ON
p.USER_ID = u.USER_ID where p.USER_ID NOT in (SELECT USER_ID FROM REQUEST);
```

USER_ID	F_NAME	L_NAME	USER_TYPE	PASSENGER_ID
6	Sherlock	Holmes	Passenger	3

1 row in set (0.001 sec)

This query is giving the list of passenger who has never taken a trip. Here we are using subquery method and joining 2 different tables USER and PASSENGER on the common records using INNER JOIN.

Queries



Query 3: Right join between REQUEST and USER.

```
MariaDB [19566ss]> select R.REQUEST_ID,R.PICKUP_LOC,R.DROPOFF_LOC,U.USER_ID,U.L_NAME,U.F_NAME,U.USER_TYPE from REQUEST R  
RIGHT JOIN USER U USING (USER_ID);
```

REQUEST_ID	PICKUP_LOC	DROPOFF_LOC	USER_ID	L_NAME	F_NAME	USER_TYPE
1	123 Belmont Street	International SF Airport	4	Smith	James	Passenger
2	International SF Airport	1458 Walnut Creek	2	Pauley	Bettina	Driver
3	International SF Airport	123 Mowry Ave	5	Cortes	Emily	Passenger
NULL	NULL	NULL	1	Warner	David	Driver
NULL	NULL	NULL	3	Dsouza	Justin	Driver
NULL	NULL	NULL	6	Holmes	Sherlock	Passenger

6 rows in set (0.001 sec)

It will return all the data from the right table which is USER and corresponding record from left table REQUEST. In case of no common column match, NULL value will be printed for LEFT table.

Query 4: Natural join between PASSENGER and USER to get the detail of the passenger.

```
MariaDB [19566ss]> select * from USER u JOIN PASSENGER p using (USER_ID);
```

USER_ID	F_NAME	L_NAME	E_MAIL	ADDRESS	ZIPCODE	GENDER	PHONE_NO	PASSWORD	USER_TYPE	PASSENGER_ID	OVERALL_RATING
4	James	Smith	j_smith@gmail.com	123 Belmont Street,Fresno,CA	94567	M	1282899793	mith_jam	Passenger	1	5
5	Emily	Cortes	emily_cortes@gmail.com	123 Morwy Ave,Fremont,CA	94536	F	1289999793	cortes_34	Passenger	2	4.1
6	Sherlock	Holmes	holmes123@gmail.com	458 Walnut Creek,San Francisco,CA	95671	M	1289999788	4hol123	Passenger	3	4.7

3 rows in set (0.001 sec)

This query is giving the list of passenger details. Here we are joining 2 different tables USER and PASSENGER on the common records using NATURAL JOIN OR OUTER JOIN.

Queries



Query 5:Joining 4 tables and LEFT JOIN .

```
MariaDB [19566ss]> SELECT d.DRIVER_ID, d.AVG_RATING, t.PICKUP_LOC, t.DROPOFF_LOC ,c.COLOR, c.MODEL ,u.USER_ID, u.F_NAME, u.L_NAME
FROM DRIVER d INNER JOIN TRIP t on d.DRIVER_ID = t.DRIVER_ID INNER JOIN CAR c ON c.DRIVER_ID = d.DRIVER_ID LEFT JOIN USER u on
d.USER_ID= u.USER_ID;
```

DRIVER_ID	AVG_RATING	PICKUP_LOC	DROPOFF_LOC	COLOR	MODEL	USER_ID	F_NAME	L_NAME
1	5	International SF Airport	123 Mowry Ave	White	MERCEDES	1	David	Warner
2	4.8	International SF Airport	1458 Walnut Creek	Black	TOYOTA	3	Justin	Dsouza
3	4.5	123 Belmont Street	International SF Airport	Red	NISSAN	2	Bettina	Pauley
1	5	123 Belmont Street	567 Logan drive	White	MERCEDES	1	David	Warner

4 rows in set (0.001 sec)

This query will return all the data from USER table along with corresponding records from DRIVER, TRIP and CAR table. We are seeing all the user_id which are present in the driver table.

Query 6:List total number of trip by each driver.

```
[MariaDB [19566ss]> SELECT u.USER_ID,u.USER_TYPE,d.AVG_RATING, COUNT(1) as ride_count from USER u INNER JOIN DRIVER d on u.USER_ID ]
= d.USER_ID group by u.USER_ID;
```

USER_ID	USER_TYPE	AVG_RATING	ride_count
1	Driver	5	1
2	Driver	4.5	1
3	Driver	4.8	1

3 rows in set (0.001 sec)

This query is giving the list of total number of trip by each driver. Here we are joining 2 different tables USER and DRIVER on the common records using INNER JOIN group by user_id.

Queries



Query 7: CROSS JOIN between CAR and TRIP .

```
MariaDB [19566ss]> SELECT t.TRIP_ID,t.DURATION,c.MAKE, c.MODEL,c.COLOR FROM TRIP t CROSS JOIN CAR c ;
```

TRIP_ID	DURATION	MAKE	MODEL	COLOR
1	55 Min	MERCEDES	SUV	White
1	55 Min	TOYOTA	SEDAN	Black
1	55 Min	NISSAN	SEDAN	Red
2	30 Min	MERCEDES	SUV	White
2	30 Min	TOYOTA	SEDAN	Black
2	30 Min	NISSAN	SEDAN	Red
3	40 Min	MERCEDES	SUV	White
3	40 Min	TOYOTA	SEDAN	Black
3	40 Min	NISSAN	SEDAN	Red
4	20 Min	MERCEDES	SUV	White
4	20 Min	TOYOTA	SEDAN	Black
4	20 Min	NISSAN	SEDAN	Red

12 rows in set (0.000 sec)

Q7. CROSS JOIN between car and trip. It will return multiple of number of records from both the car.

Q8. This query will return all the data from USER table along with corresponding records from driver, trip and car table. We are seeing null on the left because user_id from user table are passenger who never booked any ride.

Query 8:Joining 4 tables and RIGHT JOIN .

```
MariaDB [19566ss]> SELECT d.DRIVER_ID, d.AVG_RATING, t.PICKUP_LOC, t.DROPOFF_LOC ,c.COLOR, c.MODEL ,u.USER_ID, u.F_NAME, u.L_NAME FROM DRIVER d INNER JOIN TRIP t on d.DRIVER_ID = t.DRIVER_ID INNER JOIN CAR c ON c.DRIVER_ID = d.DRIVER_ID RIGHT JOIN USER u on d.USER_ID= u.USER_ID;
```

DRIVER_ID	AVG_RATING	PICKUP_LOC	DROPOFF_LOC	COLOR	MODEL	USER_ID	F_NAME	L_NAME
1	5	International SF Airport	123 Mowry Ave	White	MERCEDES	1	David	Warner
1	5	123 Belmont Street	567 Logan drive	White	MERCEDES	1	David	Warner
3	4.5	123 Belmont Street	International SF Airport	Red	NISSAN	2	Bettina	Pauley
2	4.8	International SF Airport	1458 Walnut Creek	Black	TOYOTA	3	Justin	Dsouza
NULL	NULL	NULL	NULL	NULL	NULL	4	James	Smith
NULL	NULL	NULL	NULL	NULL	NULL	5	Emily	Cortes
NULL	NULL	NULL	NULL	NULL	NULL	6	Sherlock	Holmes

7 rows in set (0.001 sec)

References

- FUNDAMENTALS OF **Database Systems** SEVENTH EDITION, Ramez Elmasri and Shamkant B. Navathe.
- <https://www.lyft.com/>
- https://en.wikipedia.org/wiki/Enhanced_entity%E2%80%93relationship_model
- <https://www.tutorialspoint.com/Extended-Entity-Relationship-EE-R-Model>
- <https://www.geeksforgeeks.org/enhanced-er-model/>



Thank You !!!

Questions?