Department of CSE

Bangladesh University of Engineering and Technology

Assignment on CSE 6601

Airline Reservation System

Submitted to
Dr. Abu Sayed Md. Latiful Hoque
Professor
Department of CSE, BUET

Prepared By,

Rifat Rahman

0419052028

Md. Ashifur Rahman

1017052030

RDBMS (SQLite):

This is the entity relationship diagram form RBDMS:

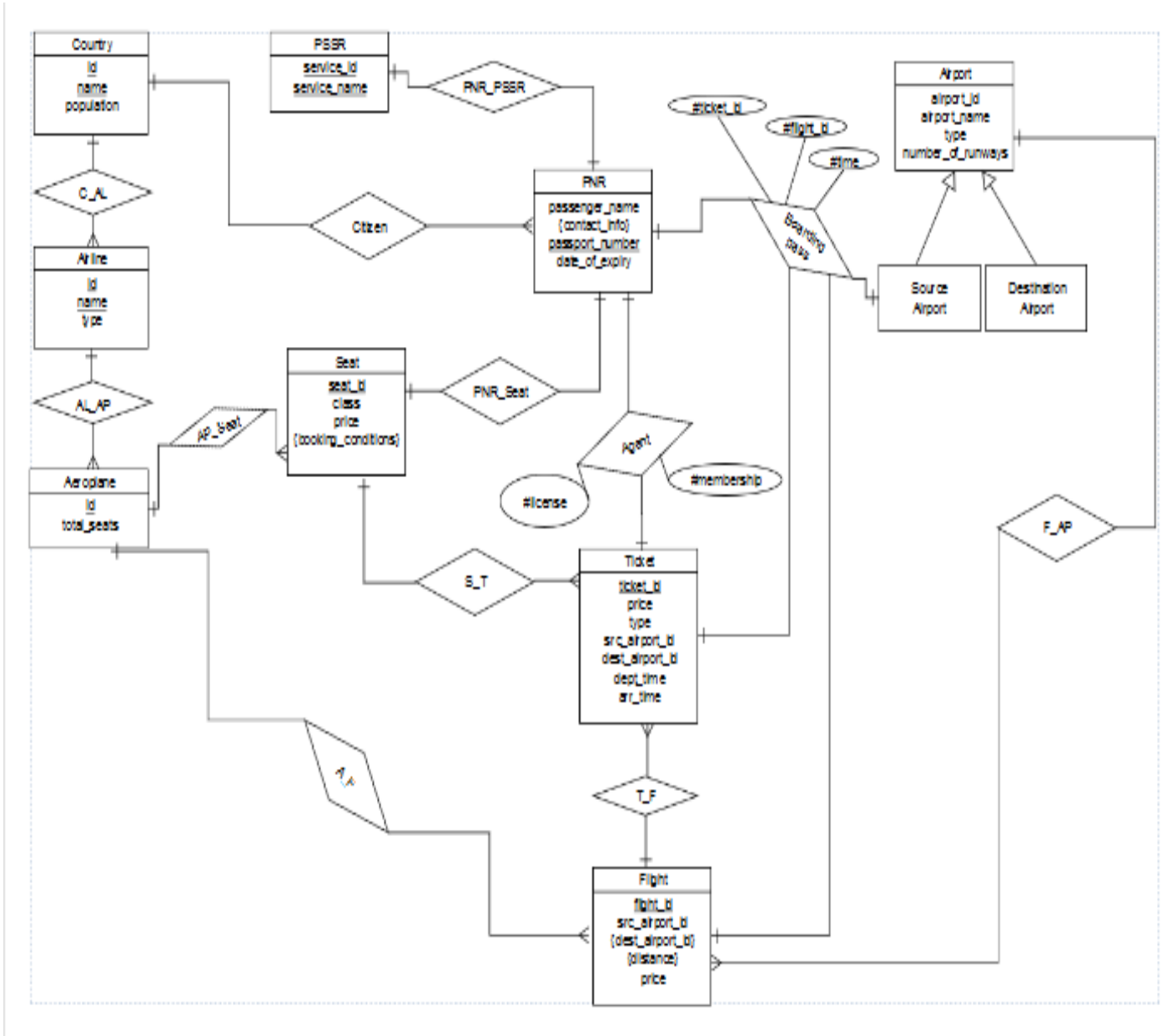


Figure1: ERD for RDBMS

This is a screenshots of the DDL of the schema for relational database.

```
c.execute("""CREATE TABLE IF NOT EXISTS Country (
             country_id text,
             country_name text,
             population integer NOT NULL,
             PRIMARY KEY(country_id,country_name)
            )""")

c.execute("""CREATE TABLE IF NOT EXISTS Airline (
             airline_id text,
             airline_name text,
             airline_type text NOT NULL,
             PRIMARY KEY(airline_id,airline_name)
            )""")

c.execute("""CREATE TABLE IF NOT EXISTS AeroPlane (
             aeroplane_id text PRIMARY KEY,
             seat_number integer NOT NULL
            )""")

c.execute("""CREATE TABLE IF NOT EXISTS PNR (
             name text NOT NULL,
             contact_info text,
             passport_number integer,
             date_of_expiry text,
             PRIMARY KEY(contact_info,passport_number)
            )""")

c.execute("""CREATE TABLE IF NOT EXISTS PSSR (
             service_id text,
             service_name text,
             PRIMARY KEY(service_id,service_name)
            )""")
```

Figure2: DDL for RDBMS

We have used python language for creating table in SQLite database. Figure 2 shows some the DDL according to our ERD

We have manually created 10, 20, 30 date samples and insert them all in our tables. We have measured cost estimation (Man hour), size of the entries (number of rows) for the design.

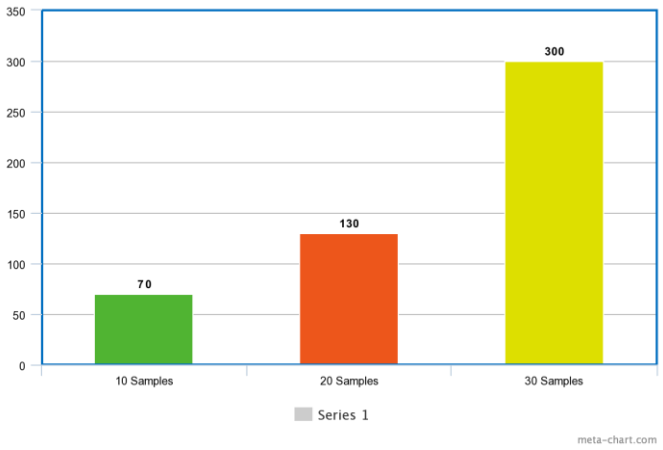


Figure3: Number of entries for different number of samples for RDMS

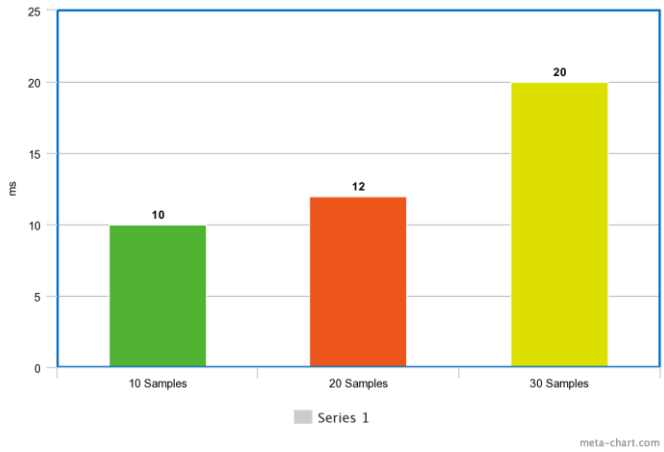


Figure4: Insertion time for different number of samples for RDBM

ORDBMS (SQLite):

The following refers to the ERD for ORDBMS.

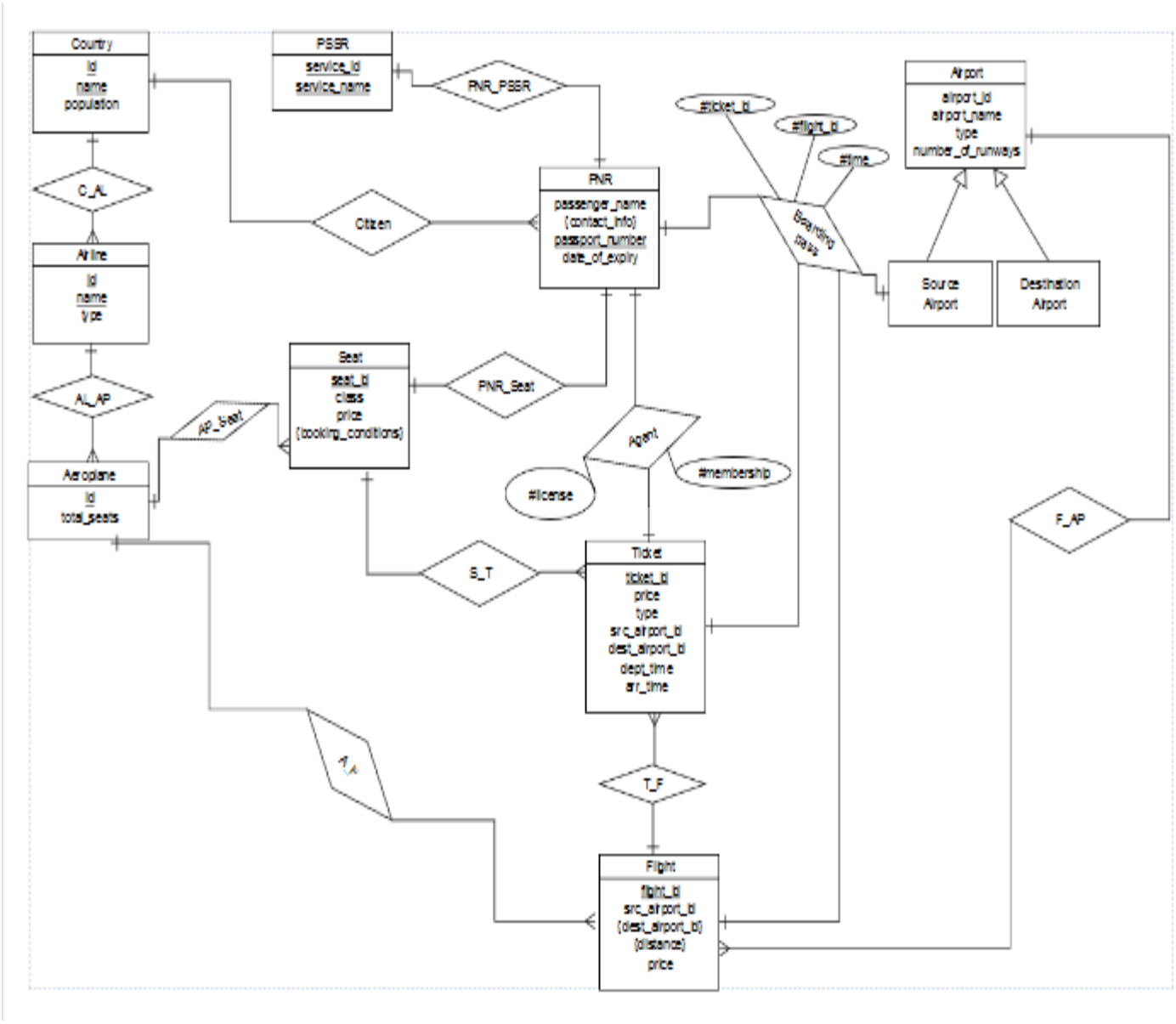


Figure 5: ERD for ORDBMS

This is a screenshot from the DDL of the schema for the object relational database:

```
class Countryy(Base):
    """"""
    __tablename__ = "country"

    country_id = Column(String, primary_key=True)
    country_name = Column(String, primary_key=True)
    population = Column(Integer, nullable=False)

class Airlinee(Base):
    """"""
    __tablename__ = "airline"

    airline_id = Column(String, primary_key=True)
    airline_name = Column(String, primary_key=True)
    airline_type = Column(String, nullable=False)

class AeroPlanee(Base):
    """"""
    __tablename__ = "aeroplane"

    aeroplane_id = Column(String, primary_key=True)
    seat_number = Column(Integer, nullable=False)

class PNRR(Base):
    """"""
    __tablename__ = "pnr"

    passenger_name = Column(String, nullable=False)
    contact_info = Column(PickleType, primary_key=True)
    passport_number=Column(Integer, primary_key=True)
    date_of_expiry=Column(PickleType)
```

Figure 6: DDL for ORDBMS

The above figure shows the DDL for ORDBMS. As this is object relational, we have used an object relational mapping tool (**ORM**) which name is **SQLAlchemy**. We have created class and make tables od that class.

We have manually created 10, 20, 30 date samples and insert them all in our tables. We have measured cost estimation (Man hour), size of the entries (number of rows) for the design.

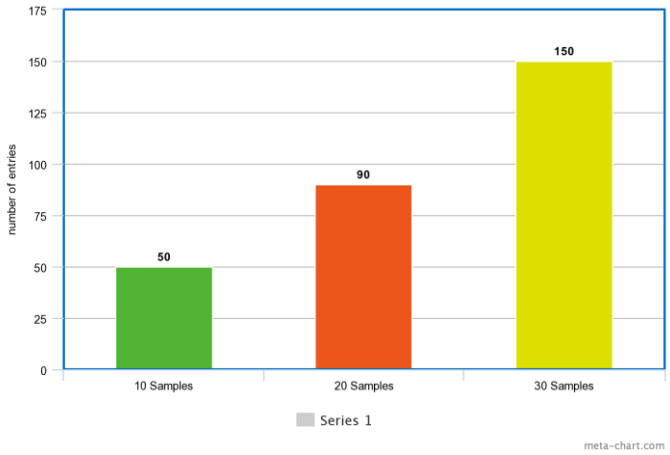


Figure7: Number of rows for different number of samples for ORDMS

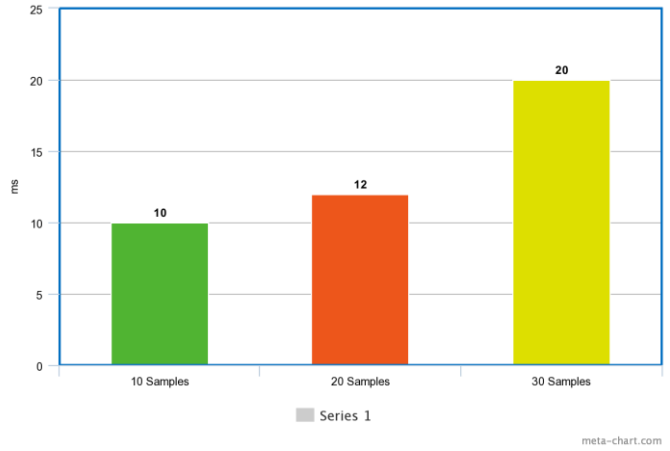


Figure8: Insertion time for different number of samples for ORDBM

From the above analysis between RBMS & ORDBMS, we can say that both relational and object relation databases will follow the schemas. If there is no value for any entries, we have to use NULL at that entry. Both the models have data types. Again in the relational model, multiple values can't be located in a single entry, we have to take multiple rows for multiple entries. On the other hand, in the object relational model we can use set or arrays for multiple values. As a result, the number of rows created for the ORDBMS is less than the RDBMS.

NOSQL (MongoDB)

NoSQL is schema-less database. A NoSQL (Not Only SQL) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. No SQL systems are also referred to as "Notonly SQL“ to emphasize that they do in fact allow SQL-like query languages to be used.

```
flight1={'flight_id':"fl1",
         'avail_seats':["A1","A2","A3","B1","B2","B3"],
         'airline':"AA Air",
         'src_aport':"S",
         'dest_aport':["D1","D2","D3"],
         'dep_date':"23/9/19",
         'dep_time':"12:00",
         'arr_date':["23/9/19","24/9/19","24/9/19"],
         'arr_time':["18:30","2:30","4:30"],
         'flight_price':["200","500","600"],
         'passengers':["pp_221","pp_333"]}

flight2={'flight_id':"fl2",
         'avail_seats':["A4","A5","A6","B1","B2","B3"],
         'airline':"AA Air",
         'src_aport':"S",
         'dest_aport':["D1","D2","D4"],
         'dep_date':"23/9/19",
         'dep_time':"13:00",
         'arr_date':["23/9/19","24/9/19","24/9/19"],
         'arr_time':["18:30","2:30","4:30"],
         'flight_price':["250","500","600"],
         'passengers':["pp_223","pp_343"]}
```

```
passenger1={'pass_num':"pp_221",
            'name':"Mr.A",
            'country':"BD",
            'service':["s1","s2"],
            'flight_id':["fl1"],
            'seat_no':["A4"],
            'dept_date':["23/9/19"],
            'dept_time':["12:00"],
            'src_aport':["S"],
            'dest_aport':["D1"]}

p1=db.passengers.insert_one(passenger1)
```

MongoDB supports document based data model. So, we have created two collections. One is Passenger collection and another is Flight collection. We have added all the entities by use just only two collections. Because document can contain key-value pairs, key- array pairs, even nested documents. We have grouped all the data that we needed.

NoSQL data modeling often starts from the application-specific queries as opposed to relational modeling.

Here is the demonstration of implementation of a-g of task 1.

Available seats for a flight, fl1(flight-1):

```
Available seats of a flight
{'flight_id': 'fl1', 'avail_seats': ['A1', 'A2', 'A3', 'B1', 'B2', 'B3']}
```

```
Fligt list & Departure Time
{'flight_id': 'fl1', 'dep_date': '23/9/19', 'dep_time': '12:00'}
{'flight_id': 'fl2', 'dep_date': '23/9/19', 'dep_time': '13:00'}
{'flight_id': 'fl3', 'dep_date': '24/9/19', 'dep_time': '13:30'}
```

```
Fligt list & Arrival Time
Flight1
{'arr_time': ['18:30', '2:30', '4:30']}
{'arr_date': ['23/9/19', '24/9/19', '24/9/19']}
{'dest_aport': ['D1', 'D2', 'D3']}

Flight2
{'arr_time': ['18:30', '2:30', '4:30']}
{'arr_date': ['23/9/19', '24/9/19', '24/9/19']}
{'dest_aport': ['D1', 'D2', 'D4']}

Flight3
{'arr_time': ['16:30', '4:30', '5:30']}
{'arr_date': ['24/9/19', '25/9/19', '25/9/19']}
{'dest_aport': ['D3', 'D4', 'D7']}
```

Here, we can observe the list of on-board passengers by a single query where in relation model we must need multiple join operations.

On board passengers of flight-fl1, airline-AA Air, date- 23/9/19

```
On board passengers
{'flight_id': 'fl1', 'dep_date': '23/9/19', 'airline': 'AA Air', 'passengers': ['pp
_221', 'pp_333']}
```

Buying ticket online user interface

```
Give passport number: pp_221

Name: Mr.A

Country: BD

Total_Service: 1

Give service id
s3

############ New Booking #################

Departure Date: 24/9/19

Source Airport: S

Destination Airport: D4
```

Available flight for that date, source & destination

```
{'flight_id': 'f13', 'dep_time': '13:30', 'avail_seats': ['A1', 'A6', 'A7', 'B1', '
B4', 'B5']}
```

User input for flight, seat and time.

```
Flight_id: f13

Seat_No: B1

Departure Time: 13:30
```

Selling ticket by agents

```
Available seats of a flight
{'flight_id': 'f13', 'avail_seats': ['A1', 'A6', 'A7', 'B4', 'B5']}
```

Comparison in Mr.A's documents

Before Booking


```
{'flight_id': ['fl1'], 'dept_date': ['23/9/19'], 'name': 'Mr.A', 'pass_num': 'pp_22
1', 'service': ['s1', 's2'], '_id': ObjectId('5d888ed674b09334b0d73fba'), 'country'
: 'BD', 'dept_time': ['12:00'], 'seat_no': ['A4'], 'src_aport': ['S'], 'dest_aport'
: ['D1']}
```

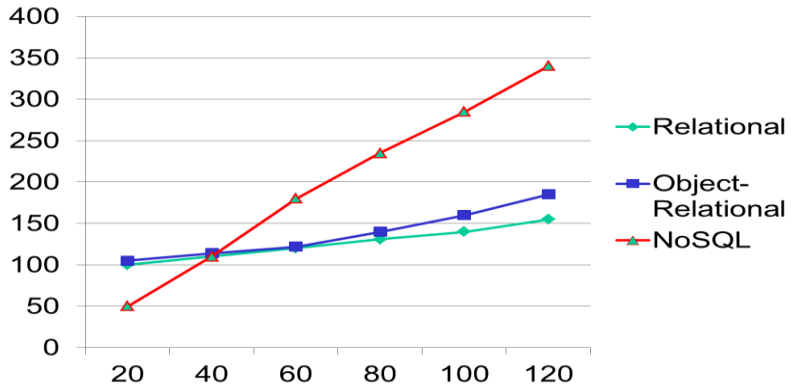

After Booking


```
{'flight_id': ['fl1', 'fl3'], 'dept_date': ['23/9/19', '24/9/19'], 'name': 'Mr.A',
'pass_num': 'pp_221', 'service': ['s1', 's2', 's3'], '_id': ObjectId('5d888ed674b09
334b0d73fba'), 'country': 'BD', 'dept_time': ['12:00', '13:30'], 'seat_no': ['A4',
'B1'], 'src_aport': ['S', 'S'], 'dest_aport': ['D1', 'D4']}
```

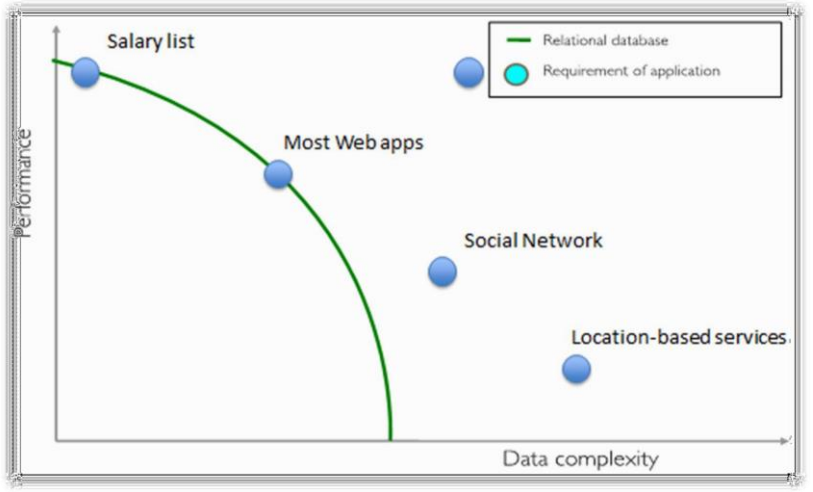

Man Hour Cost estimation:

Task	RDBMS	ODBMS	NoSQL
Database Design	1 day	12 hr	1 hr
Database Development	8 hr	7 hr	1 hr
Task A-G	4 hr	4 hr	1 hr
Total	1 day 12 hr	23 hr	3 hr

Size Estimation:



Data Complexity Vs Performance in RDBMS:



New Application:

Location based service such as “Traffic location tracking in run-time”

Research Questions:

1. The impact of relational or object relational schema on system development using relational or object relational databases. Problem of scalability.

   By analyzing all the databases, we can see that both relational and object relational model are not fully suitable for scalability. When many samples are inserted scale is needed. The solution is using NoSQL database. It has scalability property.

2. The impact of data redundancy of NoSQL model in maintaining the consistency in database application.

   Data redundancy of NoSQL can increase the cost of maintaining consistency. Suppose, any entry name has been changed then we have to change all the documents manually. But in RDBMS, we just need to change only one entry. The solution in NoSQL is using separate collection.

3. Transactional ACID property support in NoSQL database: problems & existing solutions.

   NoSQL follows CAP theorem. As a result, NoSQL doesn’t support transactional ACID property. We can handle ACID property in front end. Again we can do NoSQL in middle layer.