**NORMALIZATION**

Normalization is the process of analysing the given relational schema based on primary key and functional dependencies.

Design pattern of Normal Form:

1. To minimize the data redundancy.

2. To minimize Insertion, deletion and modification anomalies.

Types of Normal Forms:

1). 1NF: The domain of an attribute must include atomic values.



|  |  |
| --- | --- |
| ID | LOCATION |
| 1 | L1 |
| 1 | L2 |
| 1 | L3 |
| 2 | L2 |
| 2 | L3 |
| 3 | L1 |

2). 2NF: A relational schema R is in 2nd NF if every non prime attribute A in R is fully functional dependent on primary key of R.

EMPLOYEE\_DEPT TABLE:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SSN** | NAME | AGE | ADDRESS | **DNO** | D\_NAME | MGR\_SSN |

EMPLOYEE\_TABLE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SSN** | NAME | AGE | ADDRESS | D\_NO |

DEPARTMENT\_TABLE:

|  |  |  |
| --- | --- | --- |
| **D\_NO** | D\_NAME | MGR\_SSN |

3). 3NF: A relational schema R is in 3rd NF if the relation is already in 2NF and no non-functional prime attribute is transitive dependent on primary key.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROPERTY\_ID** | COUNTRY\_NAME | LOT# | AREA | PRICE | TAX |

|  |  |
| --- | --- |
| COUNTRY\_NAME | TAX |

- BCNF: A relation R is in BCNF if R is in Third Normal Form and for every Functional Dependency, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key.

4). 4NF: A relation is said to be in 3NF and has no multi-valued dependency.

5). 5NF: A relational schema R is in 5NF if the relation is already in 4NF and have no Join dependency.

**MONGODB**

MongoDB is a NoSQL database.

The records or data are stored as documents, in the JSON(Java script object notation) format.

Collections in MongoDB is equivalent to the tables in RDBMS.  
Documents in MongoDB is equivalent to the rows in RDBMS.  
Fields in MongoDB is equivalent to the columns in RDBMS.

**Commands**:

1. To create a database: use database\_name;

2. To check current db user: db

3. To create collection: db.createCollection(“collection\_name”);

3. To show collections: show collections

4. To insert a values: This can be done in 2 ways that is either through single insertion of fields or through array of objects.

(a). db.customer.insert({first\_name:"priyanka",last\_name:"ramesh"});

(b). db.customer.insert([{first\_name:"sneha",last\_name:"ramesh",gender:"female"}, {first\_name:"daya",last\_name:"jegan",gender:"female"}]);

5. To view the contents: db.customer.find();

5.1. db.customer.find.pretty();

pretty() method is used to configure the cursor to display results in an easy-to-read format.

6. To update a record: db.customer.update();

db.customer.update({first\_name:"myth",last\_name:"subu",gender:"female"});

7. Sort the records in a collection in accending or decending order

7.1.Ascending order

db.customer.find({},{\_id:0,first\_name:1,last\_name:1}).sort({"last\_name":1});

7.2. Decending order:

2. db.customer.find({},{\_id:0,first\_name:1,last\_name:1}).sort({"last\_name":-1});



