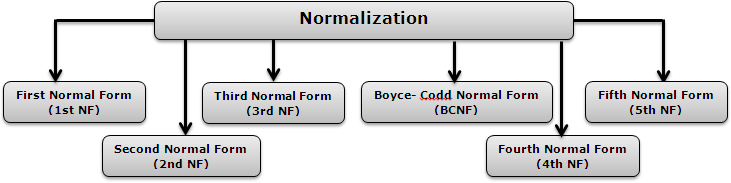
**NORMALIZATION:**  
  
Normalization is the process of organizing data into a related table; it also eliminates redundancy and increases the integrity which improves performance of the query. To normalize a database, we divide the database into tables and establish relationships between the tables.  
  
Database normalization can essentially be defined as the practice of optimizing table structures. Optimization is accomplished as a result of a thorough investigation of the various pieces of data that will be stored within the database, in particular concentrating upon how this data is interrelated.  
  
**Normalization Avoids:**

* **Duplication of Data** - The same data is listed in multiple lines of the database
* **Insert Anomaly** - A record about an entity cannot be inserted into the table without first inserting information about another entity - Cannot enter a customer without a sales order
* **Delete Anomaly** - A record cannot be deleted without deleting a record about a related entity. Cannot delete a sales order without deleting all of the customer's information.
* **Update Anomaly** - Cannot update information without changing information in many places. To update customer information, it must be updated for each sales order the customer has placed

**DE- NORMALIZATION:**   
  
Denormalization is the process of adding redundant data to speed up complex queries involving multiple table JOINS. One might just go to a lower form of Normalization to achieve Denormalization and better performance. Data is included in one table from another in order to eliminate the second table which reduces the number of JOINS in a query and thus achieves performance.  
  
Normalization is a Six stage process - After the first stage, the data is said to be in first normal form, after the second, it is in second normal form, after the third, it is in third normal form and so on.  
  
  
  
**First Normal Form (1st NF)**  
  
In 1st NF:

* The table cells must be of single value.
* Eliminate repeating groups in individual tables.
* Create a separate table for each set of related data.
* Identify each set of related data with a primary key.

**Definition:** An entity is in the first normal form if it contains no repeating groups. In relational terms, a table is in the first normal form if it contains no repeating columns. Repeating columns make your data less flexible, waste disk space, and make it more difficult to search for data.  
  
**IMP:** In 1NF relation the order of tuples (rows) and attributes (columns) does not matter.   
  
**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Customer** | **Contact Person** | **Total** |
| 1 | Rishabh | Manish | 134.23 |
| 2 | Preeti | Rohan | 521.24 |
| 3 | Rishabh | Manish | 1042.42 |
| 4 | Rishabh | Manish | 928.53 |

The above relation satisfies the properties of a relation and is said to be in first normal form (or 1NF). Conceptually it is convenient to have all the information in one relation since it is then likely to be easier to query the database.   
  
**Second Normal Form (2nd NF)**  
  
In 2nd NF:

* Remove Partial Dependencies.
* Functional Dependency: The value of one attribute in a table is determined entirely by the value of another.
* Partial Dependency: A type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key).
* Create separate table with the functionally dependent data and the part of the key on which it depends. Tables created at this step will usually contain descriptions of resources.

**Definition:** A relation is in 2NF if it is in 1NF and every non-key attribute is fully dependent on each candidate key of the relation.   
  
**Example:**  
  
**The following relation is not in Second Normal Form:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Customer** | **Contact Person** | **Total** |
| 1 | Rishabh | Manish | 134.23 |
| 2 | Preeti | Rohan | 521.24 |
| 3 | Rishabh | Manish | 1042.42 |
| 4 | Rishabh | Manish | 928.53 |

In the table above, the order number serves as the primary key. Notice that the customer and total amount are dependent upon the order number -- this data is specific to each order. However, the contact person is dependent upon the customer. An alternative way to accomplish this would be to create two tables:

|  |  |
| --- | --- |
| **Customer** | **Contact Person** |
| Rishabh | Manish |
| Preeti | Rohan |

|  |  |  |
| --- | --- | --- |
| **Order** | **Customer** | **Total** |
| 1 | Rishabh | 134.23 |
| 2 | Preeti | 521.24 |
| 3 | Rishabh | 1042.42 |
| 4 | Rishabh | 928.53 |

The creation of two separate tables eliminates the dependency problem. In the first table, contact person is dependent upon the primary key -- customer name. The second table only includes the information unique to each order. Someone interested in the contact person for each order could obtain this information by performing a Join Operation.  
  
**Third Normal Form (3rd NF)**  
In 3rd NF:

* Remove transitive dependencies.
* Transitive Dependency A type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key. Thus its value is only indirectly determined by the primary key.
* Create a separate table containing the attribute and the fields that are functionally dependent on it. Tables created at this step will usually contain descriptions of either resources or agents. Keep a copy of the key attribute in the original file.

A relation is in third [normal](http://www.watchesn.com) form, if it is in 2NF and every non-key attribute of the relation is non-transitively dependent on each candidate key of the relation.   
  
**Non-transitive dependency:**  
  
Let A, B and C be three attributes of a relation R such that Aïƒ B and Bïƒ C. From these FDs, we may derive Aïƒ C. This dependence Aïƒ C is transitive.   
  
**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Company** | **City** | **State** | **ZIP** |
| ABC Ltd. | Mumbai | MH | 10169 |
| XYZ Ltd. | Noida | UP | 33196 |
| ASD Ltd. | Chennai | TN | 21046 |

The above table is not in the 3NF.  
  
In this example, the city and state are dependent upon the ZIP code. To place this table in 3NF, two separate tables would be created -- one containing the company name and ZIP code and the other containing city, state, ZIP code pairings.

|  |  |
| --- | --- |
| **Company** | **ZIP** |
| ABC Ltd. | 10169 |
| XYZ Ltd. | 33196 |
| ASD Ltd. | 21046 |

|  |  |  |
| --- | --- | --- |
| **City** | **State** | **ZIP** |
| Mumbai | MH | 10169 |
| Noida | UP | 33196 |
| Chennai | TN | 21046 |

This may seem overly complex for daily applications and indeed it may be. Database designers should always keep in mind the tradeoffs between higher level normal forms and the resource issues that complexity creates.   
  
**Boyce-Codd Normal Form (BCNF)**  
  
In BCNF:

* When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.
* 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys
* i.e. composite candidate keys with at least one attribute in common.
* BCNF is based on the concept of a determinant.
* A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
* A relation is in BCNF is, and only if, every determinant is a candidate key.

**Definition:** A relation is in Boyce-Codd Normal Form (BCNF) if every determinant is a candidate key. (See the links in the box at right for definitions of determinant and candidate key.)  
  
The difference between 3NF and BCNF is that for a functional dependency A ïƒ  B, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key,  
   
Whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.  
  
**Example:**  
  
**CLIENT INTERVIEW:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClientNo** | **InterviewDate** | **InterviewTime** | **StaffNo** | **RoomNo** |
| CR76 | 13-may-11 | 10:30 | SG5 | G101 |
| CR76 | 13-may-11 | 12:00 | SG5 | G101 |
| CR74 | 13-may-11 | 12:00 | SG37 | G102 |
| CR56 | 02-july-11 | 10:30 | SG5 | G102 |

â€¢ FD1 ClientNo, InterviewDate -> InterviewTime, StaffNo, RoomNo (Primary Key)  
â€¢ FD2 StaffNo, InterviewDate, InterviewTime -> ClientNo (Candidate key)  
â€¢ FD3 RoomNo, InterviewDate, InterviewTime -> ClientNo, StaffNo (Candidate key)  
â€¢ FD4 StaffNo, InterviewDate -> RoomNo (not a candidate key)  
  
As a consequece the ClientInterview relation may suffer from update anomalies.  
  
To transform the ClientInterview relation to BCNF, we must remove the violating functional dependency by creating two new relations called Interview and StaffRoom as shown below,  
  
**Interview (ClientNo, InterviewDate, InterviewTime, StaffNo)  
StaffRoom (StaffNo, InterviewDate, RoomNo)**  
**INTERVIEW**

|  |  |  |  |
| --- | --- | --- | --- |
| **ClientNo** | **InterviewDate** | **InterviewTime** | **StaffNo** |
| CR76 | 13-may-11 | 10:30 | SG5 |
| CR76 | 13-may-11 | 12:00 | SG5 |
| CR74 | 13-may-11 | 12:00 | SG37 |
| CR56 | 02-july-11 | 10:30 | SG5 |

**STAFFROOM**

|  |  |  |
| --- | --- | --- |
| **StaffNo** | **InterviewDate** | **RoomNo** |
| SG5 | 13-may-11 | G101 |
| SG37 | 13-may-11 | G102 |
| SG5 | 02-july-11 | G102 |

BCNF Interview and StaffRoom relations.  
  
An entity is in Fourth Normal Form (4NF) when it meets the requirement of being in Third Normal Form (3NF) and additionally:

* Has no multiple sets of multi-valued dependencies. In other words, 4NF states that no entity can have more than a single one-to-many relationship within an entity if the one-to-many attributes are independent of each other.
* Many:many relationships are resolved independently.

**Fourth Normal Form (4th NF)**  
  
In 4th NF:  
  
An entity is in Fourth Normal Form (4NF) when it meets the requirement of being in Third Normal Form (3NF) and additionally:

* Has no multiple sets of multi-valued dependencies. In other words, 4NF states that no entity can have more than a single one-to-many relationship within an entity if the one-to-many attributes are independent of each other.
* Fourth Normal Form applies to situations involving many-to-many relationships.

In relational databases, many-to-many relationships are expressed through cross-reference tables.  
  
**Definition:** A table is in fourth normal form (4NF) if and only if it is in BCNF and contains no more than one multi-valued dependency.   
  
**Example:**  
  
Take an example of Employee Table  
info(Employee, Skills, Hobbies)

|  |  |  |
| --- | --- | --- |
| **Employee** | **Skills** | **Hobbies** |
| 1 | Programming | Golf |
| 1 | Programming | Bowling |
| 1 | Analysis | Golf |
| 1 | Analysis | Bowling |
| 2 | Analysis | Golf |
| 2 | Analysis | Gardening |
| 2 | Management | Golf |
| 2 | Management | Gardening |

This table is difficult to maintain since adding a new hobby requires multiple new rows corresponding to each skill. This problem is created by the pair of multi-valued dependencies EMPLOYEE -> SKILLS and EMPLOYEE -> HOBBIES. A much better alternative would be to decompose INFO into two relations:

|  |  |
| --- | --- |
| **Employee** | **Skills** |
| 1 | Programming |
| 1 | Analysis |
| 2 | Analysis |
| 2 | Management |

Hobbies(Employee, Hobby)

|  |  |
| --- | --- |
| **Employee** | **Hobbies** |
| 1 | Golf |
| 1 | Bowling |
| 2 | Golf |
| 2 | Gardening |

**Fifth Normal Form (5th NF)**  
  
In 5th NF:

* A relation that has a join dependency cannot be decomposed by a projection into other relations without spurious results
* A relation is in 5NF when its information content cannot be reconstructed from several smaller relations i.e. from relations having fewer attributes than the original relation

**Definition:** A table is in fifth normal form (5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables.  
  
Fifth normal form, also known as join-projection normal form (JPNF), states that no non-trivial join dependencies exist. 5NF states that any fact should be able to be reconstructed without any anomalous results in any case, regardless of the number of tables being joined. A 5NF table should have only candidate keys and it's primary key should consist of only a single column.  
  
**Example:**  
  
Take an example of a buying table. This is used to track buyers, what they buy, and from whom they buy. Take the following sample data:

|  |  |  |
| --- | --- | --- |
| **Buyer** | **Vendor** | **Item** |
| Shalley | Kashmir House | Jeans |
| Mary | Kashmir House | Jeans |
| Shalley | Radhika Sarees | Saree |
| Mary | Radhika Sarees | Saree |
| Shalley | Radhika Sarees | Suit |

The problem with the above table structure is that if Claiborne starts to sell Jeans then how many records must you create to record this fact? The problem is there are pair wise cyclical dependencies in the primary key. That is, in order to determine the item you must know the buyer and vendor, and to determine the vendor you must know the buyer and the item, and finally to know the buyer you must know the vendor and the item.  
  
And the solution is to break this one table into three tables; Buyer-Vendor, Buyer-Item, and Vendor-Item. So following tables are in the 5NF.  
  
**BUYER-VENDOR**

|  |  |
| --- | --- |
| **Buyer** | **Vendor** |
| Shalley | Kashmir House |
| Mary | Kashmir House |
| Shalley | Radhika Sarees |
| Mary | Radhika Sarees |

**BUYER-ITEM**

|  |  |
| --- | --- |
| **Buyer** | **Item** |
| Shalley | Jeans |
| Mary | Jeans |
| Shalley | Saree |
| Mary | Saree |
| Shalley | Suit |

**VENDOR- ITEM**

|  |  |
| --- | --- |
| **Vendor** | **Item** |
| Kashmir House | Jeans |
| Radhika Sarees | Saree |
| Radhika Sarees | Suit |