Normalization

Normalization

- Normalization is a process for converting complex data structures into simple, stable data structures.

Why Normalization is necessary ?

- The database design must be efficient.
- The amount of data should be reduced if possible.
- The design should be free of update, insertion and deletion anomalies.
- The design must comply with rules regarding relational databases.
- The design has to show pertinent relationship between entities.
- The design should permit simple retrieval, simplify data maintenance and reduce the need to restructure data.

Functional Dependency

- Normalization is based on the analysis of functional dependence.
- A functional dependency is a particular relationship between two attributes.
- For any relation R, attribute B is functionally dependent on attribute A if, for every valid instance of A, that value of A uniquely determines the value of B.
- This is usually represented by an arrow, as follows:

A → B

Functional Dependency

- An attribute may be functionally dependent on two (or more) attributes, rather than on a single attribute.
- Example
  - ORDER (ORDER-NO, PART-NO, NO-ORDERED, PART-DESC, QUOTED-PRICE)
  - ORDER-NO, PART-NO → NO-ORDERED, PART-DESC, QUOTE-PRICE
- The attribute on the left-hand side of the arrow is called a **Determinant**.

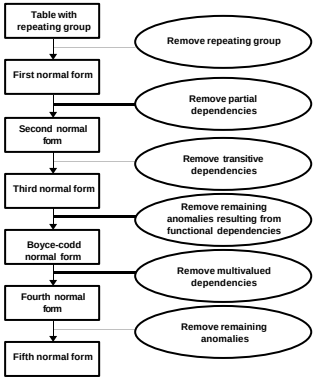
Functional Dependencies

- Examples
  - CUST-NO → CUST-NAME, ADDRESS, COMPANY
  - INVOICE-NO → INVOICE-DATE, CUST-NO, ORDER-NO
  - CUST-NO and INVOICE-NO are examples of determinants.

Primary Key

- An attribute (or field), K, if
  - All columns (all the fields in the table) are functionally dependent on K.
  - Each value is unique.
  - If K is a composite/concatenate key then it must comply with the following conditions:
    - No portion of the key should be a primary key.
    - All attributes that make up the key are not null.

Steps in Normalization



Steps in Normalization

- Step 1: First normal form (1NF).
  - Any repeating groups have been removed, so that there is a single value at the intersection of each row and column of the table.
- Step 2: Second normal form (2NF).
  - Any partial functional dependencies have been removed.
- Step 3: Third normal form (3NF).
  - Any transitive dependencies have been removed.
- If a relation meets the criteria for 3NF, it also meets criteria for 2NF and 1NF. Most design problems can be avoided if the relations are in 3NF.

Example

- Given a Table in UNF
  - (Order-no, Date, {Part-no, Qty-ordered, Part-description, Quote-price}, Cust-no, Cust-name, Cust-address)
- Since (Part-no, Part-description, Quote-price, etc…) is the repeated group, we need to remove it.

Result of 1NF

UNF	1NF	
Order-no	**Order no**	**Order-no**
Date	Date	Part-no
Part-no	Cust-no	Qty-ordered
Qty-ordered	Cust-name	Part-description
Part-description	Cust-address	Quote-price
Quote-price		
Cust-no		
Cust-name		
Cust-address		

Second Normal Form

- A relation is in 2NF if
  - It is in 1NF, and
  - All non-key attributes are fully functionally dependent on the primary key and not on only a portion of the primary key.

Steps to Transform into 2NF

- Identify all functional dependencies in 1NF.
- Make each determinant the primary key of a new relation.
- Place all attributes that depend on a given determinant in the relation with that determinant as non-key attributes.

Example

- All the functional dependencies in this case are:
  - ORDER-NO → DATE, CUST-NO, CUSTNAME, CUST-ADDRESS
  - PART-NO → PART-DESC
- In this case, we say that PART-NO is only partially functional dependent on the key.
  - (ORDER-NO, PART-NO) → QTY-ORDERED, QUOTE-PRICE

Example

- The partial functional dependency in ITEM (ORDER-NO, PART-NO, QTY-ORDERES, QUOTE-PRICE) creates redundancy in that relation, which results in anomalies when the table is updated.

Anomaly

- Insertion Anomaly
  - To insert a row for the ITEM table, we must provide the part description information too.
- Deletion Anomaly
  - If we delete a row for the ITEM table, we may lose some PART information.
- Modification Anomaly
  - If a PART's description changes, we must record the change in multiple rows in the ITEM table.

Result for 2NF

1NF

Order-no	**Order-no**
Date	Part-no
Cust-no	Qty-ordered
Cust-name	Part-description
Cust-address	Quoted-price

2NF

Order-no	**Order-no**
Date	**Part-no**
Cust-no	Quoted-price
Cust-name	Quoted-price
Cust-address	
	Part-no
	Part-description

Note of 2NF

- A relation that is in first normal form will be in second normal form if any one of the following conditions apply:
- The primary key consists of only one attribute (such as the attribute ORDER-NO in ORDER).
- No non-key attributes exist in the relation.
- Every non-key attribute is functionally dependent on the full set of primary key attributes.

Third Normal Form

- A relation is in 3NF if:
  - It is in 2NF, and
  - No transitive dependencies.
- Transitive dependencies are when A → B → C. Thus it can be split into A → B and B → C.

Steps to Transform into 3NF

- Create one relation for each determinant in the transitive dependency.
- Make the determinants the primary keys in their respective relations.
- Include as non-key attributes those attributes that depend on the determinant.

Example

- In the functional dependency:
  - ORDER(ORDER-NO, DATE, CUST-NO, CUST-NAME, CUST-ADDRESS)
- There is a transitive dependency.
- That is, one of the non-key attribute can be used to determine other attributes.
  - CUST-NO → CUST-NAME, CUST ADDRESS

Update Anomaly

- Insertion Anomaly
  - A new customer is found and cannot be entered until it has made an order.
- Deletion Anomaly
  - If an order-no is deleted from the ORDER table, we may lose some CUSTOMER information.
- Modification Anomaly
  - If the address of a customer changes, we have to update all the associated past order records.
- To remove such anomalies, we can decompose the ORDER relation into two relations.

Result of 3NF

- Notice that CUST-NO is the primary key of a new relation and is a foreign key in the ORDER relation.

2NF		**3NF**	
Order-no	**Order-no**	**Order-no**	**Order-no**
Date	**Part-no**	Date	Part-no
Cust-no	Qty-priced	Cust-no	Qty-ordered
Cust-name	Quoted-price		Quoted-price
Cust-address		**Cust-no**	
	Part-no	Cust-name	**Part-no**
	Part-description	Cust-address	Part-description

Foreign Key

- A foreign key is an attribute that appears as a non-key attribute in one relation and as a primary key attribute in another relation.

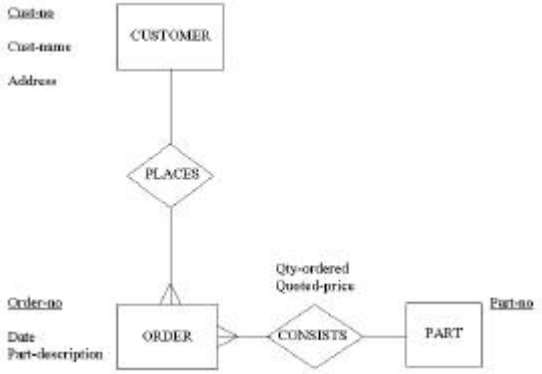
Final Result after 3NF

- ORDER (ORDER-NO, DATE, CUST-NO)
- ITEMS (ORDER-NO, PART-NO, NO-ORDERED, QUOTED-PRICE)
- CUSTOMER (CUST-NO, CUST-NAME, CUST-ADDRESS)
- PART (PART-NO, PART-DESC)

Transforming E-R diagrams into Relations

- Each entity type in an E-R diagram is transformed into a relation.
- The primary key of the entity type becomes the primary key of the corresponding relation.

Represent Entities



Binary 1:N Relationships

- A binary one-to-many (1:N) relationship in an E-R diagram is represented by adding the primary key attribute of the entity on the one-side of the relationship, as a foreign key.
- The CUSTOMER and ORDER relations in the E-R diagram are then transformed into
  - ORDER(ORDER-NO, DATE, CUST-NO)
  - CUSTOMER (CUST-NO, CUST-NAME, CUST-ADDRESS)
- CUST-NO is a foreign key in the ORDER relation but a primary key in the CUSTOMER relation.

Binary M:N Relationships

- For a binary many-to-many relationship between two entity types A and B, create a separate relation C. The primary key of this C relation is the composite key consisting of the primary keys for entities A and B.
- In the entities types PART and ORDER, a relation called ORDER_LINE is created which consists of the two primary keys in the PART and ORDER as well as the attributes QTY-ORDERED, QUOTED-PRICE.
  - ORDER-LINE (ORDER-NO, PART-NO, QTY-ORDERED, QUOTED-PRICES)

Unary Relationships

- In a unary relationship (recursive relationship), the primary key of that relation is the same as for the entity type.
- A foreign key is added to the relation that references the primary key values.
- This is known as the **Recursive Foreign Key**.
  - EMPLOYEE (EMP-ID, NAME, BIRTHDATE, MANAGER-ID)

Merging Relations

- As part of the logical design process, normalized relations may have been created from a number of separate E-R diagrams and other user views.
- Some of these relations may be redundant and can be integrated with other relations (view integration).

Example

- Suppose that modeling a user view results in 3NF relation:
  - STUDENT1 (STUDENTID, NAME, ADDRESS, PHONE, GUARDIAN).
- Modeling a second user view might result in the following relation:
  - STUDENT2 (STUENTID, NAME, ADDRESS, DEPT)
- Since these two relations have the same primary key (STUDENTID), they describe the same entity and may be merged into one relation.
  - STUDENT (STUDENTID, NAME, ADDRESS, PHONE, GUARDIAN, DEPT)
- This reduces duplication of NAME and ADDRESS.