Normalisation Process

Scenario:

Mr. Steve who is a local businessman and real estate owner decided to expand his business into painting, which is why he decided to register an internet business called Masterpieces Limited. The business leases paintings of various kinds of painters and artists to people who are interested and willing to buy those paintings. Being a new business, the records regarding the painting, painters, artists, staff and customers has to be maintained which is why a database to manage it is required. The objective of this project is to create a functional database by taking every aspect of the business in consideration, so as to make it easier for the owner, Mr. Steve, to manage his internet business.

1. UNF

This is the first step for normalisation. In this step an entity is created with all the repeating data and repeating group based on our assumptions. A repeating group is the data which is seen multiple times in the database where as repeating data is the data seen only once in the database. The repeating groups are enclosed with in curly brackets.

The un-normalised form for this project is:

Customer (**customer_id**, customer_name, customer_contact_number, customer_type, customer_category, discount, {transaction_id, transaction_type, rental_amount, sold_price, transaction_date, staff_id, staff_name, staff_contact_number, salary, {painting_id, painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_type, owner_contact_number}})

Here, the entity identified is Customer and the unique identifier is customer_id for the entire entity.

2. 1NF

In this step, the repeating groups are separated into different entities and keys are assigned to create relation between the entities. This step creates a single value at the intersection of each row and column of the table.

Arriving at 1NF

Customer – 1 (**customer_id**, customer_name, customer_contact_number, customer_type, customer_category, discount)

Transaction – 1 (**transaction_id**, **customer_id***, transaction_type, rental_ amount, sold_price, transaction_date, **staff_id**, staff_name, staff_contact_number, salary)

Painting – 1 (**painting_id**, **transaction_id***, **customer_id***, **staff_id***, painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_type, owner_contact_number)

Hence, 3 entities are created after the separation of repeating groups. Each entity has a unique identifier to identify the entity. The repeating group gets the primary key of repeating data as foreign key.

The 1NF step is completed as we can see that there are only single valued attributes.

3. 2NF

The second normal form is based on the concept of full functional dependency. The 2NF form applies when an entity has composite keys. A composite key is formed by the combination of two or more columns which identify a particular row in the table. In this step partial dependency is removed. Partial dependency occurs when any one of the composite key has a relation with non-key attribute. After the removal of such partial dependency, a relation is said to be in 2NF. (ggeksforgeeks.org, 2019)
Applying 2NF to the above 1NF relations.

Checking partial dependency for:
Customer – 1 (**customer_id**, customer_name, customer_contact_number, customer_type, customer_category, discount)

- This relation is already in 2NF as it has only single attribute as Unique identifier.

Checking partial dependency for:
Transaction – 1 (**transaction_id**, **customer_id*****, transaction_type, rental_ amount, sold_price, transaction_date, **staff_id**, staff_name, staff_contact_number, salary)

transaction_id → transaction_type, rental_ amount, sold_price, transaction_date

customer_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

staff_id → staff_name, staff_contact_number, salary

transaction_id, customer_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

transaction_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

customer_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

transaction_id, customer_id, **staff_id** → gives details of transaction made by a staff but entity not created as the same data can be retrieved from another entity.

Transaction – 2 (**transaction_id**, transaction_type, rental_ amount, sold_price, transaction_date)

Staff – 2 (**staff_id**, staff_name, staff_contact_number, salary)

Transaction_by_staff – 2 (**transaction_id*, customer_id***, **staff_id*)**

Checking partial dependency for:

Painting – 1 (**painting_id**, **transaction_id***, **customer_id*, staff_id***, painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_type, owner_contact_number)

painting_id → painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_type, owner_contact_number

transaction_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

customer_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

painting_id, transaction_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

painting_id, customer_id → gives details of painting leased by a customer but entity not created as the data can be retrieved from another entity.

painting_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

transaction_id, customer_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

transaction_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

customer_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

painting_id, transaction_id, customer_id→ Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

painting_id, transaction_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

transaction_id, customer_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

painting_id, customer_id, staff_id → Nothing

- It has no Functional Dependent Attribute therefore it does not have Partial Functional Dependency. Hence entity is not created.

painting_id, transaction_id, customer_id, staff_id → gives details of transaction of the painting done by a staff for a particular customer.

Painting – 2 (**painting_id**, painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_typem owner_contact_number)

Leased_painting – 2 (**painting_id*, transaction_id*, customer_id*, staff_id*)**

Therefore Final 2NF,

Customer – 2 (**customer_id**, customer_name, customer_contact_number, customer_type, customer_category, discount)

Staff – 2 (**staff_id**, staff_name, staff_contact_number, salary)

Painting – 2 (**painting_id**, painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_type, owner_contact_number)

Transaction – 2 (**transaction_id**, transaction_type, rental_ amount, sold_price, transaction_date)

Leased_painting – 2 (**painting_id*, transaction_id*, customer_id*, staff_id*)**

4. 3NF

A relation is said to be in Third Normal Form, if there are no transitive dependency. Transitive dependency occurs when a key attribute has relation with a non-key attribute and that non-key attribute again has relation with another non-key attribute. Hence, in this step transitive dependency is removed. The dependent attributes are kept in a new relation.

The transitive dependency are removed in the following way:

Checking for:

Customer – 2 (**customer_id**, customer_name, customer_contact_number, customer_type, customer_category, discount)

customer_id → customer_category → discount

Customer – 3 (**customer_id**, customer_name, customer_contact_number, customer_type, **customer_category***)

Discount – 3 (**customer_category**, discount)

Checking for:

Staff – 2 (**staff_id**, staff_name, staff_contact_number, salary)

- This relation is already in 3 NF as it has only one non key attribute.

Checking for:

Painting – 2 (**painting_id**, painting_name, theme, artist_id, artist_name, artist_contact_number, submitted_date, last_leased_date, rental_price, available_status, owner_id, owner_name, owner_type, owner_contact_number)

painting_id → artist_id → artist_name, artist_contact_number
painting_id → owner_id → owner_name, owner_type, owner_contact_number

Painting – 3 (**painting_id**, painting_name, theme, submitted_date, last_leased_date, rental_price, available_status, **artist_id***, **owner_id***)

Artist – 3 (**artist_id**, artist_name, artist_contact_number)

Owner – 3 (**owner_id**, owner_name, owner_type, owner_contact_number)

Checking for:

Transaction – 2 (**transaction_id**, transaction_type, rental_amount, sold_price, transaction_date)

- This relation is already in 3 NF as it has only one non key attribute.

Checking for:

Leased_painting – 2 (**painting_id*, transaction_id*, customer_id*, staff_id*)**

- This relation is already in 3 NF as it has no non key attribute.

Final table after 3NF,

Customer – 3 (**customer_id**, customer_name, customer_contact_number, customer_type, **customer_category***)

Discount – 3 (**customer_category**, discount)

Staff – 3 (**staff_id**, staff_name, staff_contact_number, salary)

Painting – 3 (**painting_id**, painting_name, theme, submitted_date, last_leased_date, rental_price, available_status, **artist_id***, **owner_id***)

Artist – 3 (**artist_id**, artist_name, artist_contact_number)

Owner – 3 (**owner_id**, owner_name, owner_type, owner_contact_number)

Transaction – 3 (**transaction_id**, transaction_type, rental_amount, sold_price, transaction_date)

Leased_painting – 3 (**painting_id*, transaction_id*, customer_id*, staff_id*)**

The normalisation process in now completed where relations have been derived which solves the issue of data redundancy and anomalies. The primary keys and foreign keys have also been determined. Carrying out operations based on these entities will be much more accurate and efficient.