Java Institute for Advanced Technology

UNIT NAME – Database Management 1
UNIT ID – H7DX 04
ASSIGNMENT ID – H7DX 04/AS/02

NAME – Piyumi Premachandra
SCN NO
NIC - 200056400356
BRANCH - Colombo

1. What is Normalization?

Database Normalization is deriving the data in inter-dependent relations as non-redundant and consistent data. This helps to retrieve data from a database by reducing data redundancy and enhancing data integrity by forming and arranging a database's features and relationships according to a number of so-called normal forms.

2. Define 1st Normal Form, 2nd Normal Form and 3rd Normal Form

1st Normal Form

If all of the attributes in a relation are solitary valued attributes, the relationship is said to be in first normal form.

In 1st Normal Form, we break the multi valued or non-atomic values in an attribute to atomic values by partitioning to more attributes and placing them. If there are multi valued attributes, we can take the database to 1st NF by partition multi values and place in different rows.

2nd Normal Form

To take to the 2nd Normal Form, the relation should be in the 1st NF. And when a relation in 2nd normal form, it should not have partial dependencies. (if a column depends on a candidate key, it is called as partial dependency)

3rd Normal Form

To take to the 3rd Normal Form, the relation should be in the 2nd NF. And all the attributes should depend on the primary key; this means the attributes cannot be broke more into different tables. Any transitive dependencies should be removed and transitive columns altered to different tables.

3. To which normal form this table belongs to

NICN	Contract No	Hours	Employee Name	Company ID	Company Location
892660522V	IT 2014	27	A.Janith	IT 001	Kandy
880944993V	IT 2014	84	T.Tharaka	IT 001	Kandy
842564347V	IT 1995	42	C.Asela	IT 120	Colombo
892660522V	IT 1995	42	A.Janith	IT 120	Colombo

When analyzing this relation, the provided attributes cannot separate more to make new attributes. Because there are no multi valued or composite valued attributes in this relation.

Therefore, this table is in its' 1st Normal Form.

4. Identify the Primary key of this relation and justify your selection

Primary key is a not null attribute in a relation which uniquely identify the relation's rows(tuples).

In this table, we can't see any single attribute to choose as a primary key. Because we can see all the attribute's values are duplicating. We can't identify a single column uniquely by using a single attribute here. Also there are no candidate keys .

Therefore, we have to create a composite key for uniquely identify the columns. The composite primary key is a combination of multiple columns within the table that are used as the primary key. The columns individually don't stand out on their own, but when combined, they become unique.

By using the following composite key, we can uniquely identify rows.

- **NICN,**
- **Contract No.**

5. Identify Fully Functional Dependencies and Partial Dependencies on the Primary Key

NICN	Contract No	Hours	Employee Name	Company ID	Company Location
892660522V	IT 2014	27	A.Janith	IT 001	Kandy
880944993V	IT 2014	84	T.Tharaka	IT 001	Kandy
842564347V	IT 1995	42	C.Asela	IT 120	Colombo
892660522V	IT 1995	42	A.Janith	IT 120	Colombo

(Employee)

Fully Functional Dependencies –

If an attribute is functionally dependent on another attribute but not on any of its proper subsets, it is regarded as fully functionally dependent on that other attribute.

From the table, we can clearly see that by using NICN or Contract No. individually, we cannot identify the Hours, but can uniquely determine the Hours using composite key consist of NICN and Contract No. So we can say that Hours is fully functionally dependent on {NICN, Contract No. } .

- **{NICN, Contract No } -> Hours**

Also we can see that by using NICN or Contract No. individually, we cannot identify the Company ID, but using composite key consist of NICN and Contract No. we can do so.

Then, Company ID also fully functionally dependent on {NICN, Contract No. } .

- **{NICN, Contract No } -> Company ID**

Partial Functional Dependencies –

Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.

We can identify two partial functional dependencies in this 1st NF table,

1. **Company Location,**
2. **Employee Name**

To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

6. Normalize the above mentioned table to 2nd Normal Form

By removing the partial functional attributes (Company Location, Employee Name) and adding to separate tables, we can get the Employee table to 2nd normal form.

Employee (2nd NF)

NICN	Contract No	Company ID	Hours
892660522V	IT 2014	IT 001	27
880944993V	IT 2014	IT 001	84
842564347V	IT 1995	IT 120	42
892660522V	IT 1995	IT 120	42

Company_Location (3rd NF)

Company ID	Company Location
IT 001	Kandy
IT 120	Colombo

Emp_Name (3rd NF)

NICN	Employee Name
892660522V	A.Janith
880944993V	T.Tharaka
842564347V	C.Asela

After moving the columns of Company Location and Employee Name to different tables; the creating tables, Company_Location and Emp_Name, naturally in 3rd NF now.

7. Identify the Transitive Dependencies on above 2nd Normal Form tables

Transitive Dependency is the term used to describe when an indirect interaction results in functional dependency.

When the value of one non-key attribute is functionally reliant on the value of another non-key attribute, this is known as a transitive dependency. A table must have at least two non-key characteristics in order for a transitive dependency to exist.

A transitive dependence by definition requires three or more attributes (or database columns) that are functionally dependent on one another.

NICN	Contract No	Company ID	Hours
892660522V	IT 2014	IT 001	27
880944993V	IT 2014	IT 001	84
842564347V	IT 1995	IT 120	42

892660522V	IT 1995	IT 120	42

When considering this above table,

NICN -> Contract No.

Contract No. -> Company ID

Thus, the following has a **Transitive Dependency.**

NICN -> Company ID

8. Normalize above mentioned tables to 3rd Normal Form

According to 3NF, each relation's non-key attribute must not be transitively depend on any candidate key.

There is a transitive dependency in the Employee table. **NICN -> Company ID**

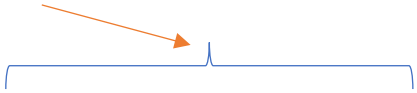
Therefore, we have to Remove above mentioned Transitive Dependency by splitting the table like below. This table has two foreign keys; Emp_Company_id, Emp_Name_id

These are one-to-many relationships.

This relation has a 3rd NF structure now.

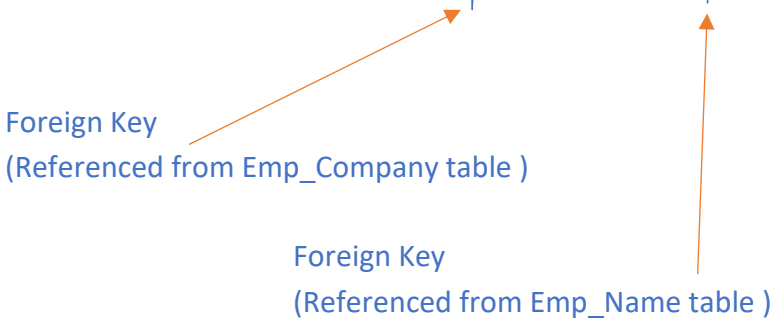
1.Emp_Contract_Hours

Composite Key



NICN	Contract No	Hours	Emp_Company_id	Emp_Name_id
892660522V	IT 2014	27	1	1

880944993V	IT 2014	84	2	2
842564347V	IT 1995	42	3	3
892660522V	IT 1995	42	4	4



2.Company_Location

Primary Key

Company ID	Company Location
IT 001	Kandy
IT 120	Colombo

For this 2nd NF table, The company ID can be select as the primary key because it is not duplicating and has a minimal number of characters.

3.Emp_Company

NICN	Company ID
892660522V	IT 001
880944993V	IT 001
842564347V	IT 120

892660522V	IT 120

In the Emp_Company table, we do not have a natural primary key, then we need to artificially create one in order to uniquely identify a row in the table (id). This is called as **surrogate key.**

This key is auto incremented according to the number of rows.

Also, this table has a **foreign key referenced from the Company_Location table. This is a one-to-one relationship.**

And the primary key referenced from the Company_Location table come to this table as a foreign key.

Surrogate Key

id	NICN	Company_Lcation_Company ID
1	892660522V	IT 001
2	880944993V	IT 001
3	842564347V	IT 120
4	892660522V	IT 120

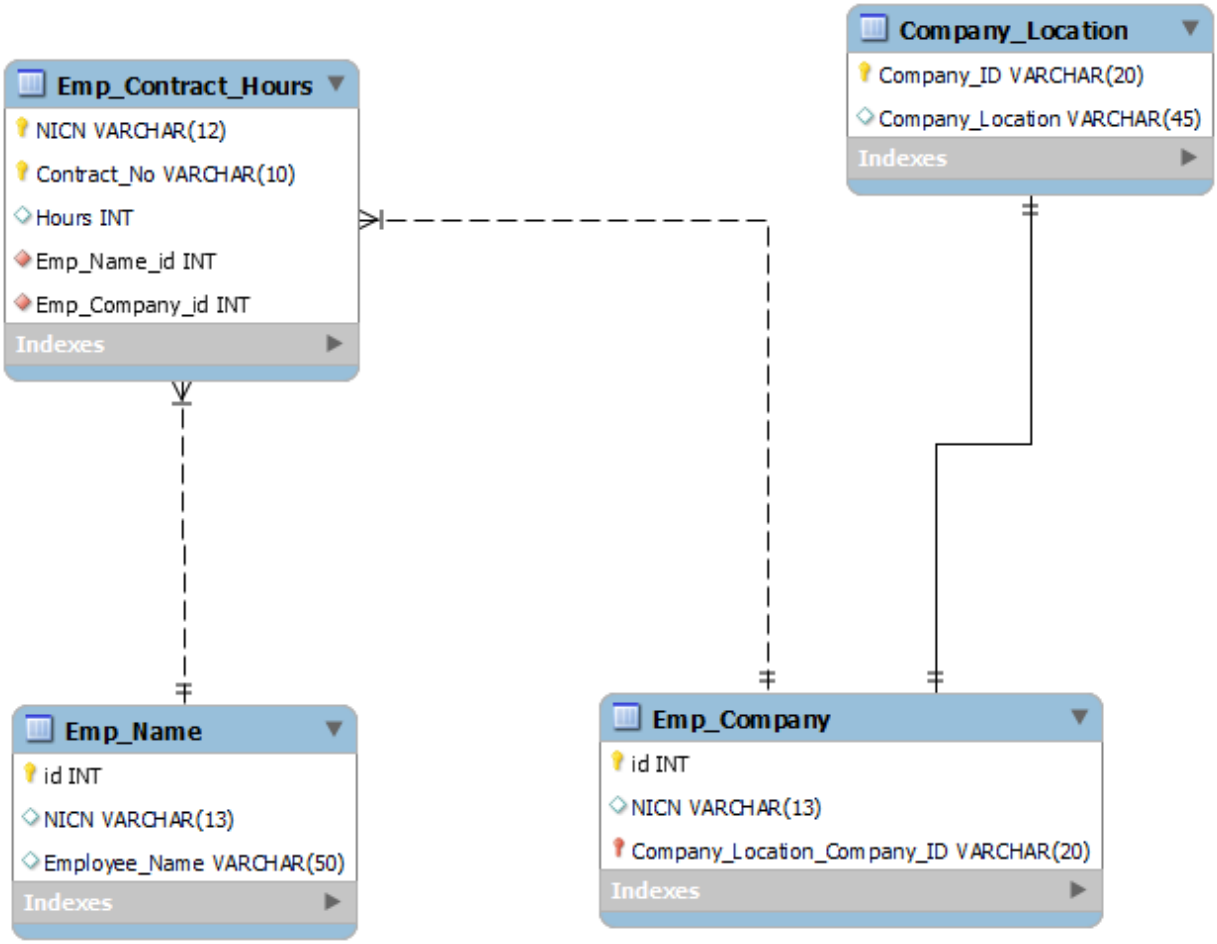
4.Emp_Name

Surrogate Key

id	NICN	Employee Name
1	892660522V	A.Janith
2	880944993V	T.Tharaka
3	842564347V	C.Asela
4	892660522V	A.Janith

We use a surrogate key (id) to normalize this table.

9. Design an ER diagram for 3rd Normal Form.



These all one-to-many relationships are non-identifying relationships. Company_Location and Emp_company table has a one-to-one relationship.

DDL script for the above model:-

```
CREATE SCHEMA IF NOT EXISTS `Exon_db_2` DEFAULT CHARACTER SET utf8 ;

USE `Exon_db_2` ;
```

-- Table `Exon_db_2`.`Emp_Name`

```
CREATE TABLE IF NOT EXISTS `Exon_db_2`.`Emp_Name` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `NICN` VARCHAR(13) NULL,
  `Employee_Name` VARCHAR(50) NULL,
  PRIMARY KEY (`id`)
);
```

-- Table `Exon_db_2`.`Company_Location`

```
CREATE TABLE IF NOT EXISTS `Exon_db_2`.`Company_Location` (
  `Company_ID` VARCHAR(20) NOT NULL,
  `Company_Location` VARCHAR(45) NULL,
```

```
  PRIMARY KEY (`Company_ID`)
);
```

-- Table `Exon_db_2`.`Emp_Company`

```
CREATE TABLE IF NOT EXISTS `Exon_db_2`.`Emp_Company` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `NICN` VARCHAR(13) NULL,
  `Company_Location_Company_ID` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`id`, `Company_Location_Company_ID`),
    FOREIGN KEY (`Company_Location_Company_ID`)
    REFERENCES `Exon_db_2`.`Company_Location` (`Company_ID`)
);
```

-- Table `Exon_db_2`.`Emp_Contract_Hours`

```
CREATE TABLE IF NOT EXISTS `Exon_db_2`.`Emp_Contract_Hours` (
  `NICN` VARCHAR(12) NOT NULL,
  `Contract_No` VARCHAR(10) NOT NULL,
  `Hours` INT NULL,
  `Emp_Name_id` INT NOT NULL,
  `Emp_Company_id` INT NOT NULL,
```

```
  PRIMARY KEY (`NICN`, `Contract_No`),
    FOREIGN KEY (`Emp_Name_id`)
    REFERENCES `Exon_db_2`.`Emp_Name` (`id`)
    FOREIGN KEY (`Emp_Company_id`)
    REFERENCES `Exon_db_2`.`Emp_Company` (`id`)
);
```