CHAP 6- SQL

1) TABLE

```
CREATE TABLE Staff (staffno,= VARCHAR(5), salary int);
INSERT INTO Staff VALUES ('SG16', 'Brown' ,8000);
SELECT staff, salary FROM Staff;
SELECT * FROM Staff;
SELECT DISTINCT propertyNo FROM Viewing ;  //shows only diff. types of propertyNo
SELECT COUNT(staffno)AS myCount, SUM(salary) AS mySum FROM Staff;
…GROUP BY branchNo  HAVING...          //column yg sama
…ORDER BY salary DESC  //susun by rows,  ACS-ascending
SELECT….FROM… WHERE position IN ('Manager', 'Sec');
SELECT….FROM… WHERE address LIKE '%Gombak%';
```

2) SUBQUERIES

```
SELECT staffNo, fName FROM Staff   //two table (equality)
    WHERE branchNo=
        (SELECT branchNo FROM Branch
        WHERE street = 'jalan labu')

SELECT staffNo, fName,                  //using aggregate
    salary – (SELECT AVG(salary) FROM STAFF) AS SalDif
    FROM Staff
    WHERE salary> (SELECT AVG(salary) FROM STAFF);

SELECT staffNo, fName FROM Staff
    WHERE salary> SOME (SELECT salary FROM Staff
    WHERE branchNo='B003');

SELECT staffNo, fName FROM Staff
    WHERE salary> ALL (SELECT salary FROM Staff
    WHERE branchNo='B003');
```

3) EDIT

Add column

ALTER TABLE *table_name*
ADD *column_name datatype*;

Drop column

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

UPDATE

```
UPDATE Staff SET salary = salary*1.03;
```

DELETE

```
DELETE from Viewing WHERE propertyNo='PG4';
```

4) JOIN

```
SELECT c.clientNo, fName, propertyNo
    FROM Client c, Viewing v
    WHERE c.client = v.clientNo;
```

CHAP 7- DATA DEFINITION (1)

Domain Constraints

(a)CHECK

```
    sex CHAR NOT NULL
        CHECK (sex IN ('M','F'))
```

(b)CREATE DOMAIN

```
    CREATE DOMAIN DomainName [AS] dataType
        [DEFAULTdefaultOption]
            [CHECK(searchCondition)]

Eg. CREATE DOMAIN SexType AS CHAR
        CHECK(VALUE IN('M','F'));
sex         sexType         NOTNULL
```

Referential Integrity

CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.
SETNULL: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL.
SETDEFAULT: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.
NO ACTION: Reject delete from parent. Default.

Main SQL DDL statements are:

CREATE SCHEMA	DROP SCHEMA
CREATE/ALTER DOMAIN	DROP DOMAIN
CREATE/ALTER TABLE	DROP TABLE
CREATE VIEW	DROP VIEW
CREATE INDEX	DROPINDEX

Schema = collection of related database objects.

- Objects in a schema can be tables, views, domains, assertions, collations, translations, and character sets. All have same owner.

CREATE SCHEMA

```
CREATE SCHEMA [Name|
    AUTHORIZATION CreatorId]
DROP SCHEMA Name [RESTRICT|CASCADE]
```

• With RESTRICT (default), schema must be empty or operation fails.
• With CASCADE, operation cascades to drop all object as associated with schema in order defined above. If any of these operations fail, DROP SCHEMA fails.

CHAP 7- DATA DEFINITION (2)

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- Can specify a DEFAULT value for the column.
- Primary keys should always be specified as NOT NULL.
- FOREIGNKEY clause specifies FK along with the referential action

```
Eg.   CREATE DOMAIN StaffNumber AS VARCHAR(5)
           CHECK (VALUE IN (SELECT staffNo FROM Staff));
```

ALTER TABLE

```
ALTER TABLE Staff
     ALTER position DROP DEFAULT;      //remove default
ALTER TABLE Staff
     ALTER sex SET DEFAULT 'F';        //set default

ALTER TABLE PropertyForRent        //remove constraint
     DROP CONSTRAINT StaffNotHandlingTooMuch;
ALTER TABLE Client                 //new column
     ADD prefNoRooms PRooms;

DROP TABLE PropertyForRent;
```

VIEW

```
CREATE VIEW Manager3Staff AS SELECT * FROM Staff
     WHERE branchNo = 'B003';
```

Advantages of Views
Data independence
Currency
Improved security
Reduced complexity
Convenience
Customization
Data integrity

Disadvantages of Views
Update restriction
Structure restriction
Performance

PRIVILEGES
REFERENCES Reference columns of named table in integrity constraints.
USAGE Use domains, collations, character sets, and translations.

ISO specifies that a view is updatable if

- NO-->DISTINCT, nested SELECT, GROUP BY
- Every element in SELECT list of defining query is a column name and no column appears more than once.
- FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.

WITH CHECK OPTION prohibits a row migrating out of the view.
WITH GRANT OPTION allows privileges to be passed on.
REVOKE takes away privileges granted with GRANT.

View materialization stores view as temporary table when view is first queried.

View maintenance aims to apply only those changes necessary to keep view current.

CHAP 14- NORMALIZATION

= technique for producing a set of suitable relations that support the data requirements of an enterprise.

PROPERTIES OF DECOMPOSITION
Lossless-join property enables us to find any instance of the original relation from corresponding instances in the smaller relations.
Dependency preservation property enables us to enforce a constraint on the original relation by enforcing some constraint on each of the smaller relations.

B is **functionally dependent** on A (denoted $A \rightarrow B$)
Full functional dependency **=** Determinants should have the minimal number of attributes necessary to maintain the functional dependency with the attribute(s) on the right hand-side.

eg. staffNo, sName→branchNo

- *Has one-to-one* relationship
- Holds for *all* time
- Determinant has the *minimal* number of attributes

Transitive Dependency

staffNo →sName, position, salary, branchNo, bAddress
branchNo →bAddress

The Process of Normalization
As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

UNF(unnormalized table) **to 1NF**
Remove the repeating group by:

- Entering appropriate data into the empty columns of rows containing the repeating data ('flattening' the table).

-OR-

- Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

1NF to 2NF

- Identify the primary key for the 1NF relation.
- Identify the functional dependencies in the relation.
- If partial dependencies exist on the primary key remove them by placing then in a new relation along with a copy of their determinant.

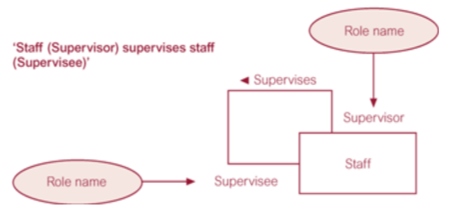
2NF to 3NF

- Identify the primary key in the 2NF relation.
- Identify functional dependencies in the relation.
- If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their dominant.

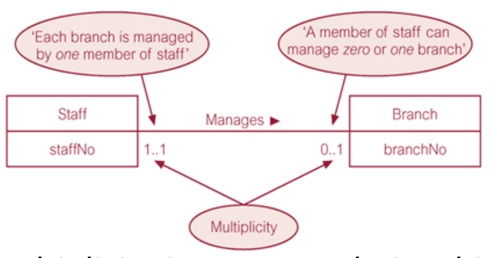
General Definitions of 2NF and 3NF
Second normal form (2NF)
A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on *any candidate key*.
Third normal form (3NF)
A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on *any candidate key*.

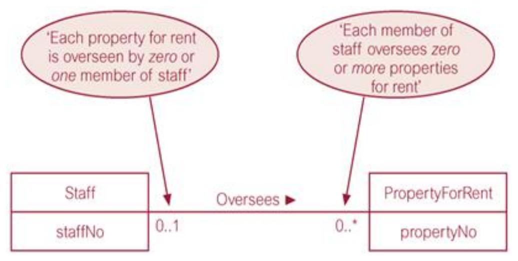
Chapter 12 CONCEPT OF ER MODEL

entity types:
Relationship types:
Degree of a relay - Number of participating entities in relationship.
Relationship of degree -Two is binary, three is ternary, four is quaternary.
Recursive Relationship - Relationship type where same entity type participates more than once in different roles.

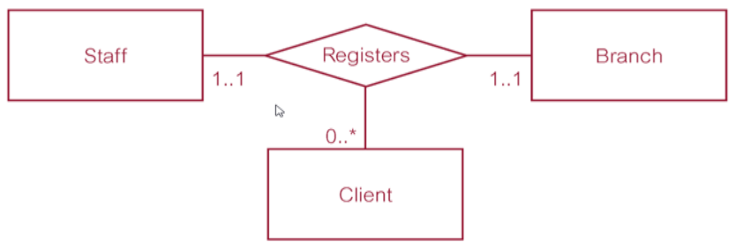


Attribute: Property of an entity or a relationship type.
Attribute Domain - Set of allowable values for one or more attributes.
Simple Attribute - Attribute composed of a single component with an independent existence.
Composite Attribute - Attribute composed of multiple components, each with an independent existence
Single-valued Attribute - Attribute that holds a single value for each occurrence of an entity type.
Multi-valued Attribute - Attribute that holds multiple values for each occurrence of an entity type.
Derived Attribute - Attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity type.
STRUCTURAL CONSTRAINTS
Structural constraints in Binary relationships - one-to-one (1:1), one-to-many (1:*), many-to-many (*:*) and others.

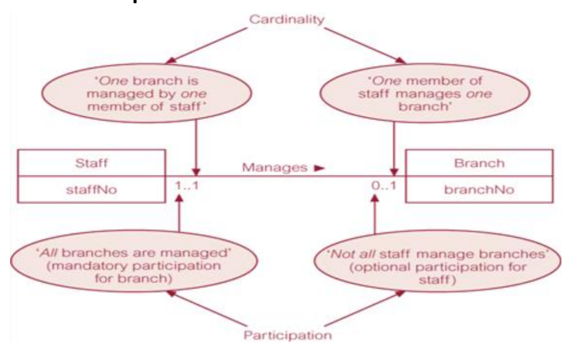

'Each branch is managed by one member of staff'
'A member of staff can manage zero or one branch'
Staff
staffNo
Manages ►
Branch
branchNo
1..1
0..1
Multiplicity


'Each property for rent is overseen by zero or one member of staff'
'Each member of staff oversees zero or more properties for rent'
Staff
staffNo
Oversees ►
PropertyForRent
propertyNo
0..1
0..*

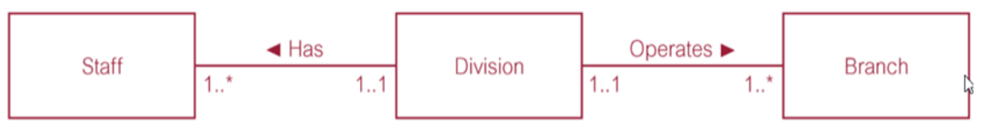
Multiplicity in ternary relationships:



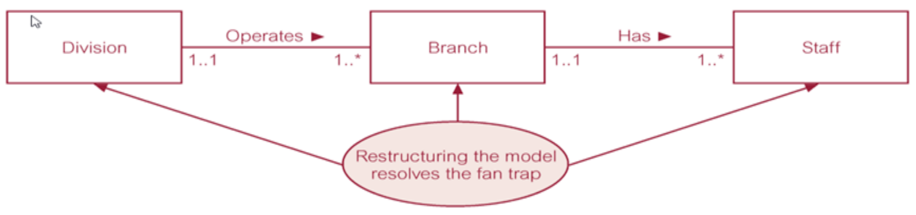
Cardinality - Describes maximum number of possible relationship occurrences for an entity participating in a given relationship type.
Participation - Determines whether all or only some entity occurrences participate in a relationship.


Cardinality
'One branch is managed by one member of staff'
'One member of staff manages one branch'
Staff
staffNo
Manages ►
Branch
branchNo
1..1
0..1
'All branches are managed' (mandatory participation for branch)
'Not all staff manage branches' (optional participation for staff)
Participation

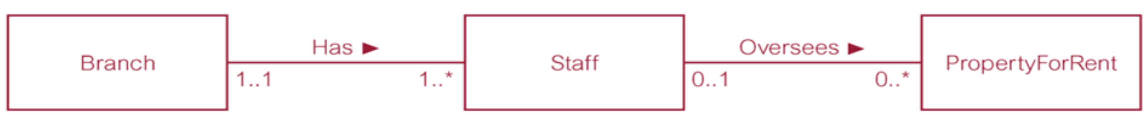
PROBLEMS WITH ER MODEL:
Fan Trap - Where a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous
EXAMPLE OF FAN TRAP:



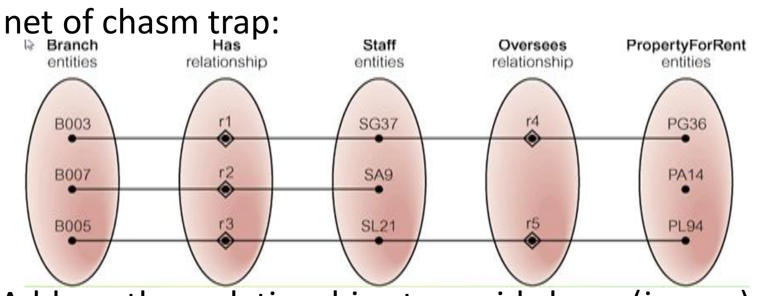
SOLUTION: Restructuring the fan trap



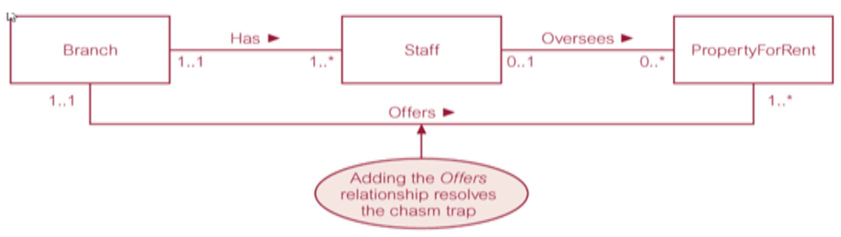
Chasm Trap - Where a model suggests the existence of a relationships between entity types, but pathway does not exist between certain entity occurrences.
Example of chasm trap:



Semantic net of chasm trap:



Solution: Add another relationships to avoid chasm(jurang).

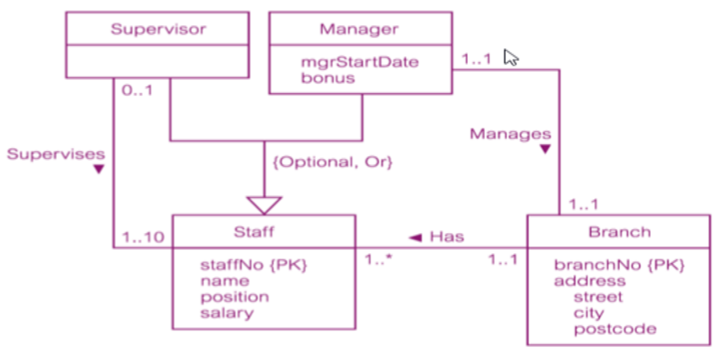


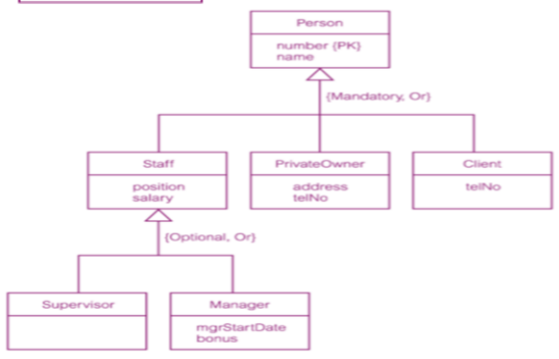
Chapter 13: Specialization / Generalization

Superclass - An entity type that includes one or more distinct subgroupings of its occurrences. (staff)
Subclass - A distinct subgrouping of occurrences of an entity type(manager , secretary).

- Superclass may contain overlapping or distinct subclasses.(staff who is manager and sales personnel)
- Not all members of a superclass need be a member of a subclass.(ordinary staff).

Constraints on Specialization / Generalization -
Participation constraints - Determines whether every member in superclass must participate as a member of a subclass. (mandatory or optional).
Disjoint constraints: Describes relationship between members of the subclasses and indicates whether member of a superclass can be a member of one, or more than one, subclass.(disjoint or no disjoint/and & or).
Examples of constraints:





Aggregation - Represents a "has-a" or "is-part-of" relationship between entity types, where one represents the "whole" and the other the "part".

Example: Branch (whole) has Staff (part),
Branch (whole) offers PropertyForRent (part).

Aggregation - Place an open diamond shape next to the entity that represents the "whole".

Composition: A specific form of aggregation that represents an association between entities, where there is a strong ownership and coincidental lifetime between the "whole" and the "part".

The "whole" manages the creation and destruction of its "parts".
Example: Newspaper (whole) displays Advert (part).

Composition - Place a filled-in diamond shape next to the entity that represents the "whole"

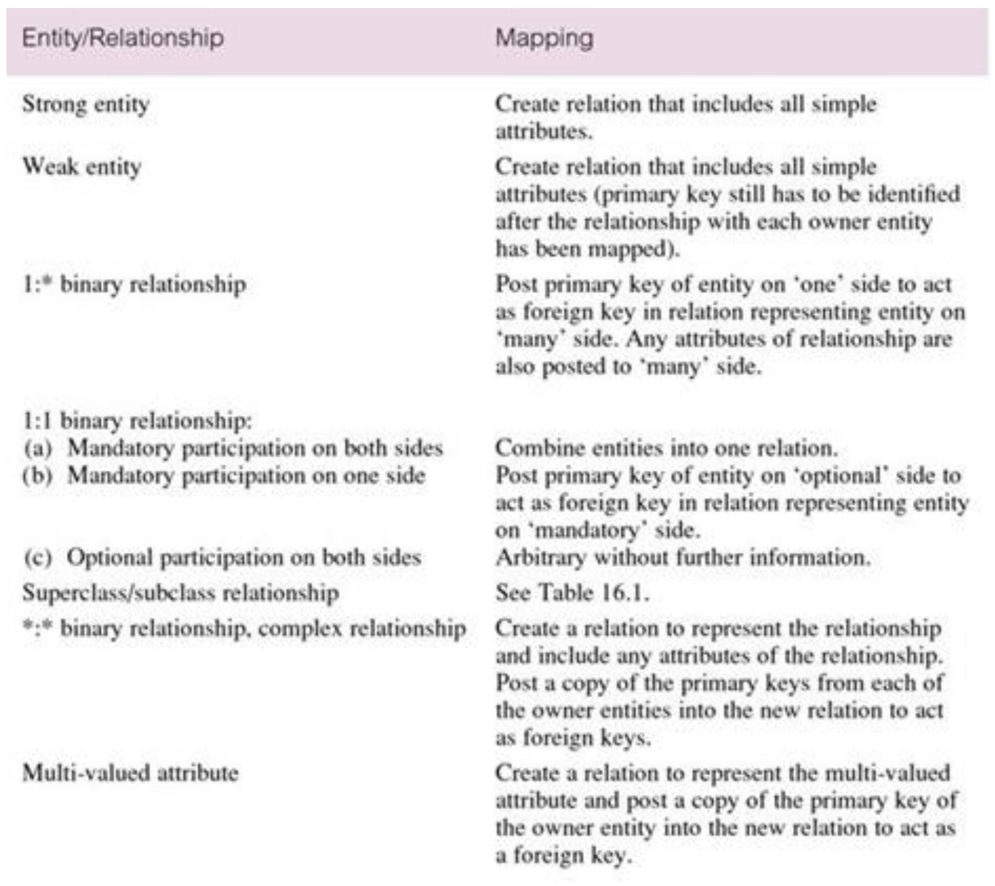
CHAPTER 17 Methodology Logical Database Design for the Relational Model

Build and Validate Logical Data Model

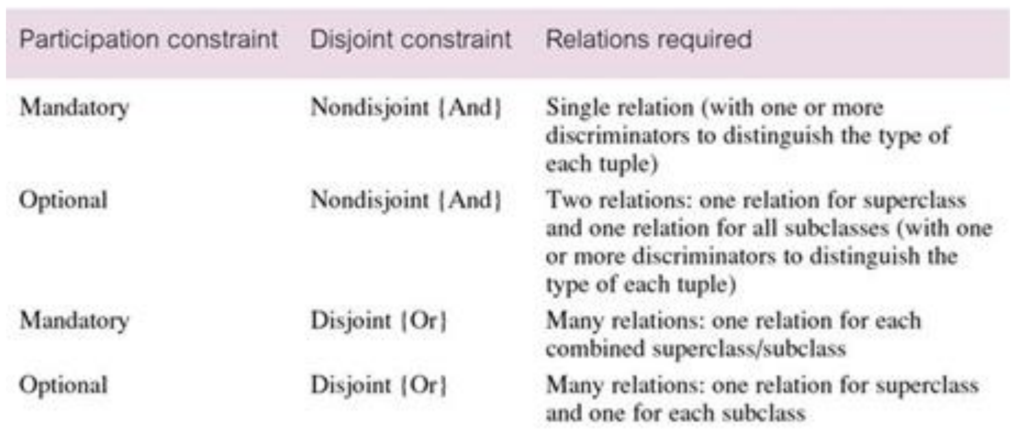
- To translate the conceptual data model into a logical data model and then to validate this model to check that it is structurally correct using normalization and supports the required transactions.

Step 2.1 Derive relations for logical data model

- To create relations for the logical data model to represent the entities, relationships, and attributes that have been identified.



Entity/Relationship	Mapping
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1:* binary relationship	Post primary key of entity on 'one' side to act as foreign key in relation representing entity on 'many' side. Any attributes of relationship are also posted to 'many' side.
1:1 binary relationship:	
(a) Mandatory participation on both sides	Combine entities into one relation.
(b) Mandatory participation on one side	Post primary key of entity on 'optional' side to act as foreign key in relation representing entity on 'mandatory' side.
(c) Optional participation on both sides	Arbitrary without further information.
Superclass/subclass relationship	See Table 16.1.
: binary relationship, complex relationship	Create a relation to represent the relationship and include any attributes of the relationship. Post a copy of the primary keys from each of the owner entities into the new relation to act as foreign keys.
Multi-valued attribute	Create a relation to represent the multi-valued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.



Participation constraint	Disjoint constraint	Relations required
Mandatory	Nondisjoint {And}	Single relation (with one or more discriminators to distinguish the type of each tuple)
Optional	Nondisjoint {And}	Two relations: one relation for superclass and one relation for all subclasses (with one or more discriminators to distinguish the type of each tuple)
Mandatory	Disjoint {Or}	Many relations: one relation for each combined superclass/subclass
Optional	Disjoint {Or}	Many relations: one relation for superclass and one for each subclass

2.2 Validate relations using normalization

- To validate the relations in the logical data model using normalization.
- Purpose normalization
    - Ensure set of relations has minimal and sufficient number of attributes necessary to support data requirement of the enterprise
    - Relations should have minimal data redundancy to avoid the problems of update anomalies

Step 2.3 Validate relations against user transactions

- To ensure that the relations in the logical data model support the required transactions.
    - To validate the logical data model
    - Ensure that conceptual data model supported the required transactions

Step 2.4 Check integrity constraints

- To check integrity constraints are represented in the logical data model.
    - That include all important integrity constraints in a 'true' representation of the data requirement for enterprise.
- This includes identifying:
    - Required data
    - Attribute domain constraints
    - Multiplicity
    - Entity integrity
    - Referential integrity
    - General constraints
    - Multiplicity
    - Entity integrity
    - Referential integrity
    - General constraints

Step 2.5 Review logical data model with user

- To review the logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of the enterprise.

Step 2.6 Merge logical data models into global Model (optional step)

- To merge logical data models into a single global logical data model that represents all user views of a database.
- Local logical data model represent one or more but not all user views of database where global logical data model represent all user views of database.

Step 2.6.1 Merge local logical data models into global model

- To merge local logical data model into a single global logical data model.
- This activities in this step include:

<u>Step 2.6.1 Merge local logical data models into global model</u>

- Review the names and contents of entities/relations and their candidate keys.
- Review the names and contents of relationships/foreign keys.
- Merge entities/relations from the local data models
- Include (without merging) entities/relations unique to each local data model
- Merge relationships/foreign keys from the local data models.
- Include (without merging) relationships/foreign keys unique to each local data model.
- Check for missing entities/relations and relationships/foreign keys.
- Check foreign keys.
- Check Integrity Constraints.
- Draw the global ER/relation diagram
- Update the documentation.

<u>Step 2.6.2 Validate global logical data model</u>

To validate the relations created from the global logical data model using the technique of normalization and to ensure they support the required transactions, if necessary.

<u>Step 2.6.3 Review global logical data model with users.</u>

To review the global logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of an enterprise.