



SECD2523 DATABASE

TOPIC 2

THE RELATIONAL MODEL AND DATABASE DESIGN

Content adapted from Connolly, T., Begg, C., 2015. Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition. Pearson Education.

Innovating Solutions

LECTURE LEARNING OUTCOME

By the end of this lecture, students should be able to:

- 01** Terminology of relational model.
- 02** How tables are used to represent data.
- 03** Properties of database relations.
- 04** How to identify candidate, primary, foreign keys.
- 05** Meaning of entity integrity and referential integrity.
- 06** Purpose and advantages of views.

LECTURE LEARNING OUTCOME

By the end of this lecture, students should be able to:

- 07** Define and describe the stages of the database system development life cycle.
- 08** Define the activities and deliverables in the main phases of database design: conceptual, logical and physical design
- 09** Build a conceptual data model based on information given in a view of the enterprise using ER Modeling.
- 10** Validate result of conceptual model to ensure it is a true and accurate representation of a view of the enterprise.
- 11** Document the process of conceptual database design.

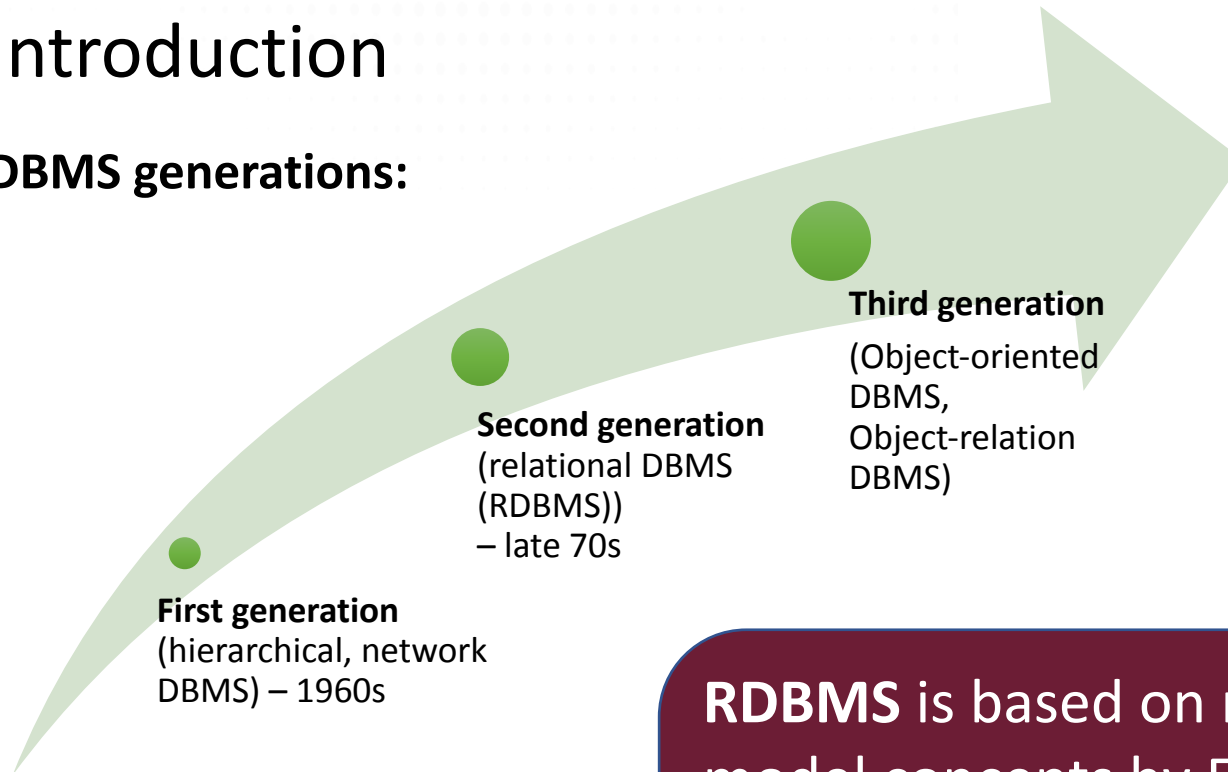
01 THE RELATIONAL MODEL

02 DATABASE DESIGN

- CONCEPTUAL DESIGN

Introduction

DBMS generations:



RDBMS is based on relational data model concepts by E.F. Codd (1970)

- Simple logical structure
- Sound theoretical foundation
- Data is logically structured within relations

Relational Model Terminology

- A **relation** is a table with columns and rows.
 - Only applies to logical structure of the database, not the physical structure.
- **Attribute** is a named column of a relation.
- **Domain** is the set of allowable values for one or more attributes.

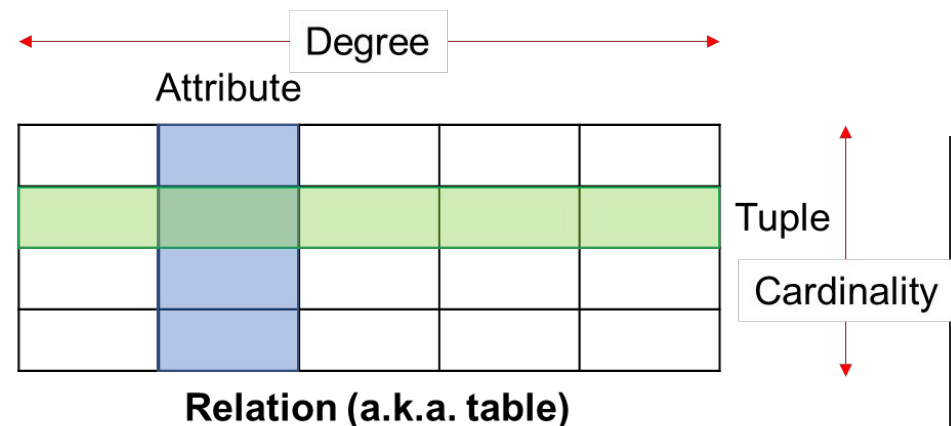
Examples of Attributes & Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

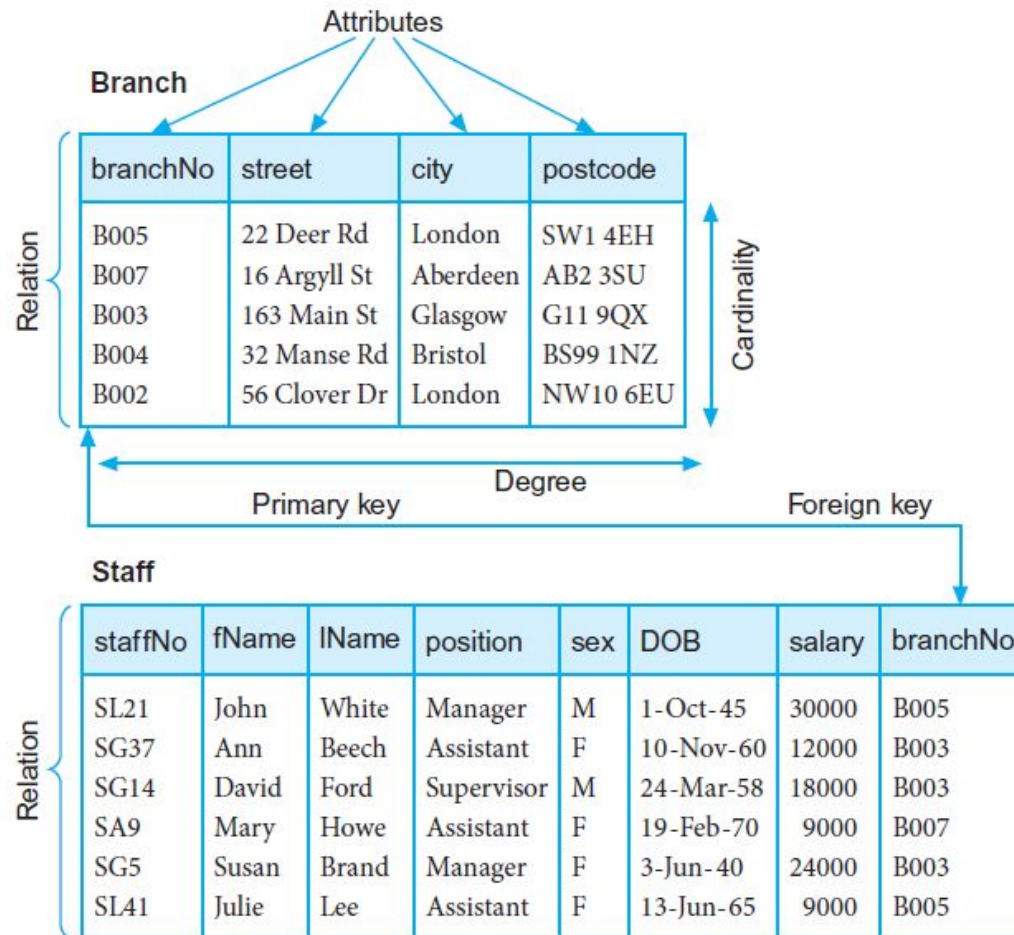
Relational Model Terminology

- **Tuple** is a row of a relation.
- **Degree** is the number of attributes in a relation.
- **Cardinality** is the number of tuples in a relation.

Relational Database is a collection of relations that is appropriately structured (normalized) with distinct relation names.




Examples



Alternative Terminology

Table: Alternative terminology for relational model terms

FORMAL TERMS	ALTERNATIVE 1	ALTERNATIVE 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field



Database Relations

Database Relations

- **Relation schema**

- Named relation defined by a set of attribute and domain name pairs.
- Common convention:
 - *RelationName (attribute_1, attribute_2,, attribute_n)*
 - Example: Branch (branchNo, street, city, postcode)

- **Relational database schema**

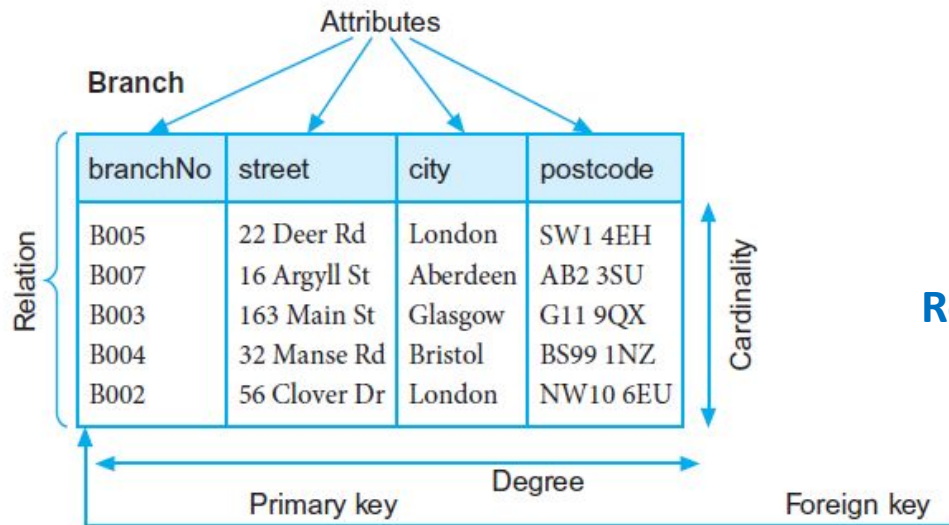
- Set of relation schemas, each with a distinct name.

- **Relation instance**

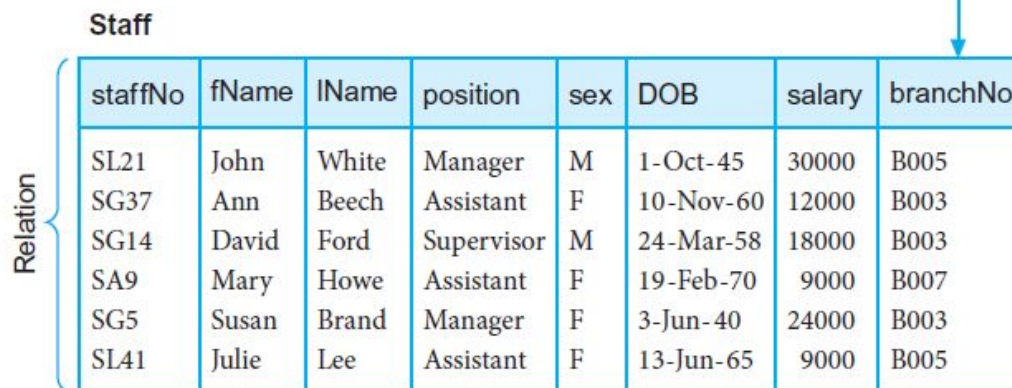
- A tuple at a specific moment of time
- Eg: (B005, 55 Jln Dobi, Johor Bahru, 80100)

Examples

Relation schema



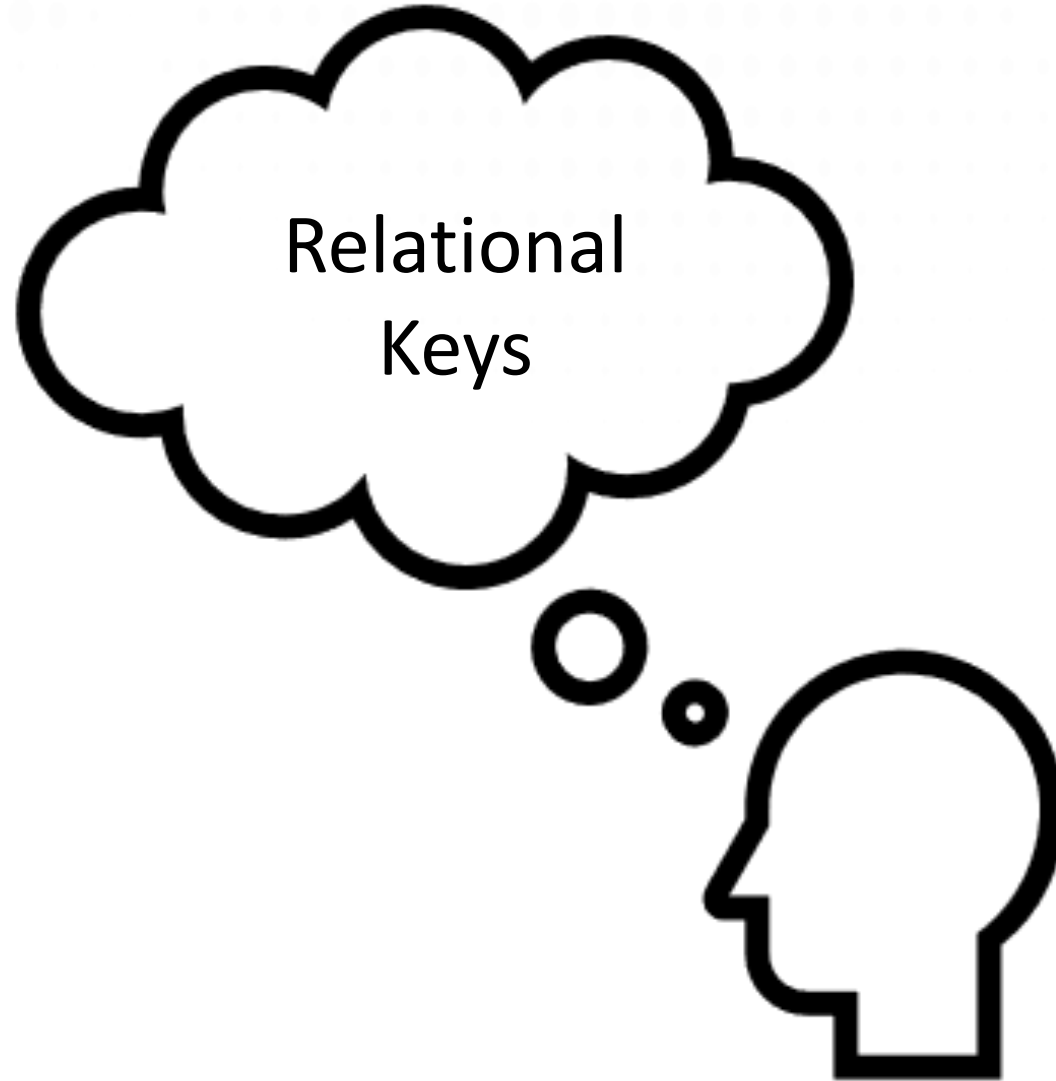
Relational database schema



Relation instance

Properties of Relations

- **Relation name is distinct** from all other relation names in relational schema.
- Each cell of relation contains **exactly one atomic (single) value**.
- Each attribute has a **distinct name**.
- Values of an attribute are all from the same domain.
- Each tuple is distinct; there are no duplicate tuples.
- Order of attributes has no significance.
- Order of tuples has no significance, theoretically.



Relational Keys

- **Relational keys**

- One or more attributes in a relation that uniquely identifies each tuple in a relation
- Types:
 - Superkey
 - Candidate key
 - Composite key
 - Primary key
 - Alternate key
 - Foreign key

Relational Keys

Superkey

- An attribute, or a set of attributes, that uniquely identifies a tuple within a relation.
- Identify superkeys that contain only the minimum number of attributes necessary for unique identification

Relational Keys

- Superkey example
 - Branch (branchNo, street, city, postCode)
 - Each branch has a unique branch number
 - What is/are superkey(s) for relation Branch?
 - Analyze each attribute singularly, given a value for the attribute, determine whether a unique tuple can be derived from the relation, if so, then attribute is a superkey
 - If none of the attribute (singularly) can be a candidate key, then try to combine two (or more) attributes and determine whether the combined attributes can derive a unique tuple, if so, then the combined attributes is a superkey.

- Example of Superkey:

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Relational Keys

Candidate Key

- A superkey such that no proper subset is a superkey within the relation.
- Two properties:
 - In each tuple of relation R , values of K uniquely identify that tuple (**uniqueness**).
 - No proper subset of K has the uniqueness property (**irreducibility**).
- If subset of K is a superkey itself, then K is not a candidate key
- There can be more than one candidate keys in a relation

Relational Keys

- Example (candidate key - CK):
 - Assume we have this list of superkeys for a relation.
Determine whether these superkeys are candidate keys.
 - `branchNo`
 - `branchNo, city`
 - `postCode, branchNo`
 - 1st superkey `branchNo`
 - This superkey has no subset that is a superkey itself, hence `branchNo` is a CK
 - 2nd superkey `branchNo, city`
 - This superkey has a subset that is a superkey itself (i.e. `branchNo`), hence `branchNo, city` is NOT a CK
 - 3rd superkey `postCode, branchNo`
 - This superkey has a subset that is a superkey itself (i.e. `branchNo`), hence `postCode, branchNo` is NOT a CK

Relational Keys

Composite Key

- When a key consists of more than one attribute.

Primary Key (PK)

- The candidate key that is selected to identify tuples uniquely within the relation.

Alternate Key

- The candidate keys that are not selected to be the primary key.

Relational Keys

Foreign Key

- An attribute, or set of attributes, within one relation that matches the candidate key of some (possibly the same) relation.
- Indicate a relationship between relations. E.g:
 1. **Branch (branchNo, street, city, postCode)**
 - PK branchNo
 2. **Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)**
 - PK staffNo
 - FK branchNo references Branch(branchNo)

Relational Keys

Superkey

Candidate
Key

**Primary
Key**

Foreign
Key

Integrity Constraints

- **Integrity constraints** are to ensure correctness / accuracy of data.
 - Domain constraints; integrity rules.
- **Integrity rules**
 - Constraints or restrictions that apply to all instances of the database.
 - Two principal rules:
 - Entity integrity
 - Referential integrity

Integrity Constraints

Null

- Represents value for an attribute that is currently “unknown” or not applicable for tuple.
- Deals with incomplete or exceptional data.
- Represents the absence of a value and is not the same as zero or spaces, which are values.

Integrity Constraints

- **Entity Integrity**

- In a base relation, no attribute of a primary key can be null.

- **Referential Integrity**

- If foreign key exists in a relation, the foreign key value:
 1. must match a candidate key value of some tuple in its home relation;
OR
 2. foreign key value must be wholly null.

- **General Constraints**

- Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.

Views

Base Relation

- A named relation corresponding to an entity in conceptual schema, whose tuples are physically stored in database.

View

- The dynamic result of one or more relational operations operating on base relations to produce another relation.

Views

- A **virtual relation** that does not necessarily actually exist in the database but is produced **upon request**, at time of request.
- Contents of a view are defined as a query on one or more base relations.
- Views are dynamic, meaning that changes made to base relations that affect view attributes are immediately reflected in the view.

Purpose of Views

- Provides powerful and flexible security mechanism by hiding parts of database from certain users.
- Permits users to access data in a customized way, so that same data can be seen by different users in different ways, at same time.
- Can simplify complex operations on base relations.

Updating Views

- All updates to a base relation should be immediately reflected in all views that reference that base relation. If view is updated, underlying base relation should reflect change.
- There are restrictions on types of modifications that can be made through views:
 - Updates are allowed if query involves a single base relation and contains a candidate key of base relation.
 - Updates are not allowed when query involves multiple base relations.
 - Updates are not allowed when query involves aggregation or grouping operations.

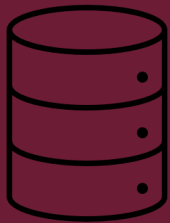
Summary

- Terminologies – relation, relational schema
- Keys – super key, candidate key, composite key, primary key, foreign key, alternate key
- Integrity constraint
- View

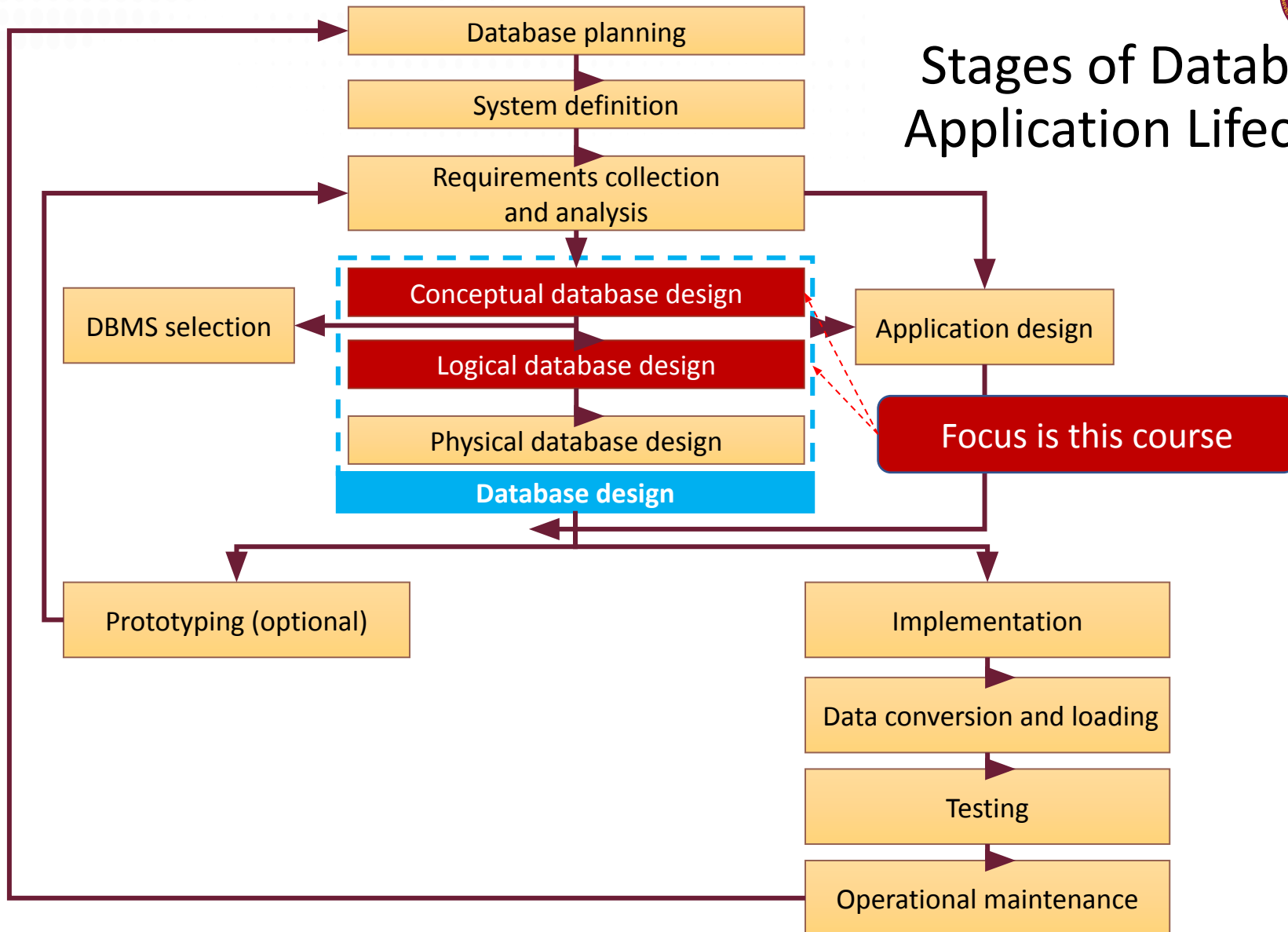
01 THE RELATIONAL MODEL

02 DATABASE DESIGN

- CONCEPTUAL DESIGN



Stages of Database Application Lifecycle



1) Database Planning

- Management activities that allow stages of database application lifecycle to be realized as efficiently and effectively as possible.
- Must be integrated with overall information system (IS) strategy of the organization.

1) Database Planning

Define a clear mission statement

- **major aims** of database application
- clarify **purpose** of the database project
- provides clearer path towards the efficient and effective creation of required database application.
- Defined by ones who are driving database project – director or owner of the project

Identify mission objectives

- What **tasks** that must be supported by the system
- Each objective is one specific task

1) Database Planning

Other information accompanied in mission statement and objectives:

- The work to be done
- The resources with which to do it
- The money to pay for it

Database planning should also include development of standards that govern:

- how data will be collected,
- how the format should be specified,
- what necessary documentation will be needed,
- how design and implementation should proceed.

2) System Definition

- Describes scope and boundaries of database application and the major user views.
 - System boundaries should involve current and future application areas and users
- **User view** defines what is required of a database application from the perspective of:
 - a particular job role (such as Manager or Supervisor) or
 - enterprise application area (such as marketing, personnel, or stock control)

2) System Definition

- Database application may have [one or more user views](#).
- Identifying user views helps ensure that no major users of the database are forgotten when developing requirements for new application.
- A user view provides requirements:
 - What data to be included in the system
 - What transactions to be performed on the data
 - Requirements may be distinct or overlap with other views

3) Requirements Collection and Analysis

- The process of **collecting and analyzing information** about the part of organization to be supported by the database application and using this information to identify users' requirements of new system.
- Information is gathered for each major user view including:
 - a description of data used or generated;
 - details of how data is to be used/generated;
 - any additional requirements for new database application.

3) Requirements Collection and Analysis

- Information is analyzed to identify requirements to be included in new database application
 - Documented in a [Requirements specification](#) for new database system
 - Presented in a structured/organized using requirement specification techniques, such as SAD techniques, DFD and HIPO charts
- Three main approaches in requirements collections and analysis from multiple user views:
 - **centralized approach;**
 - **view integration approach;**
 - **combination of both approaches.**

3) Requirements Collection and Analysis

Centralized approach

- Requirements for each user view are merged into a single set of requirements for the new database system.
- A global data model is created based on the merged requirements (which represents all user views).
- Generally, this approach is preferred when there is a significant overlap in requirements for each user view and the database system is not overly complex.

3) Requirements Collection and Analysis

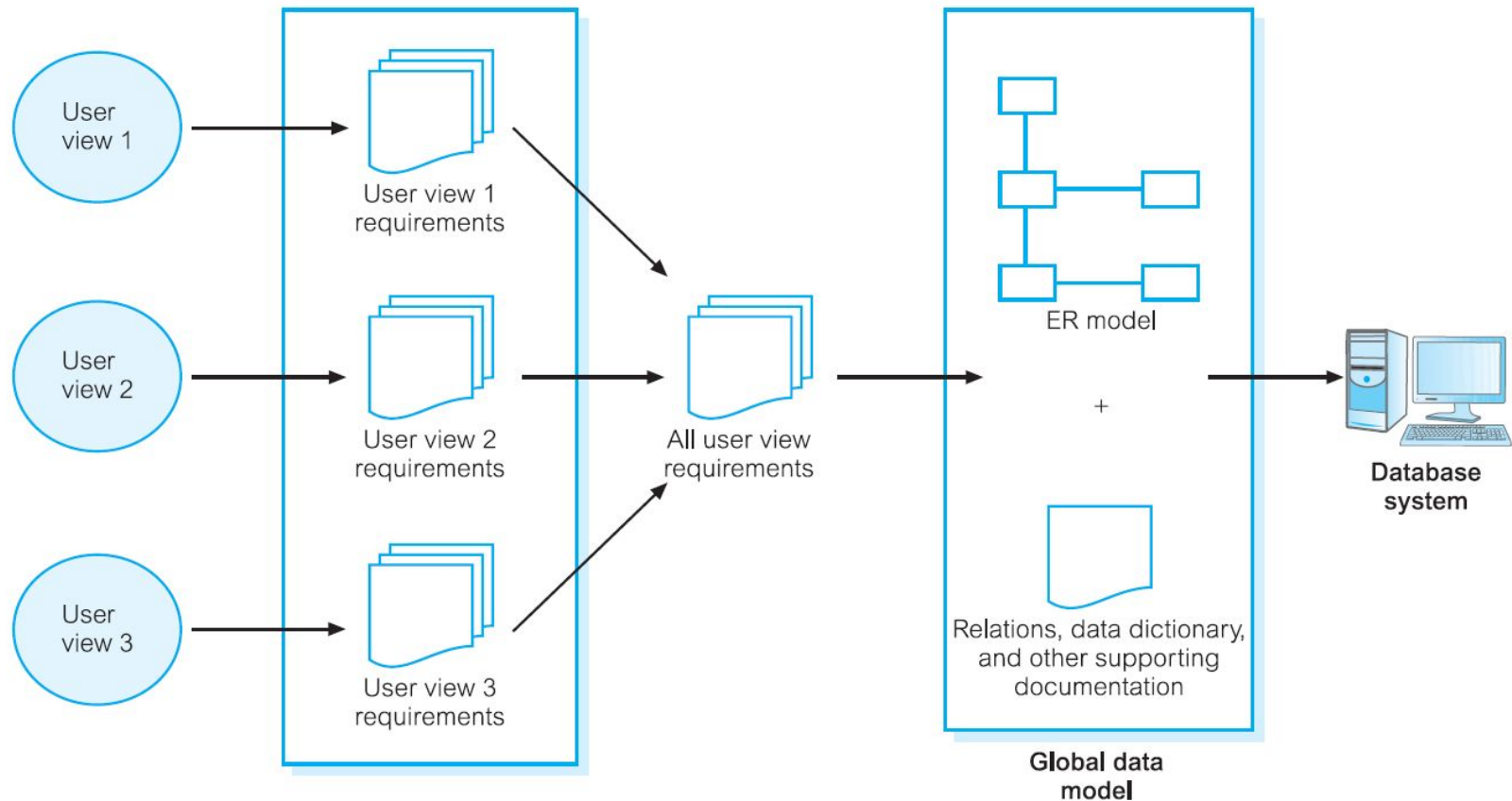


Figure 10.3 The centralized approach to managing multiple user views 1 to 3.

Centralized Approach

3) Requirements Collection and Analysis

View integration approach

- Requirements for each user view are used to build a separate data model.
- Data model representing single user view is called a **local data model**, composed of diagrams and documentation describing requirements of a particular user view of database.
- Local data models are then merged to produce **a global data model**, which represents all user views for the database.

3) Requirements Collection and Analysis

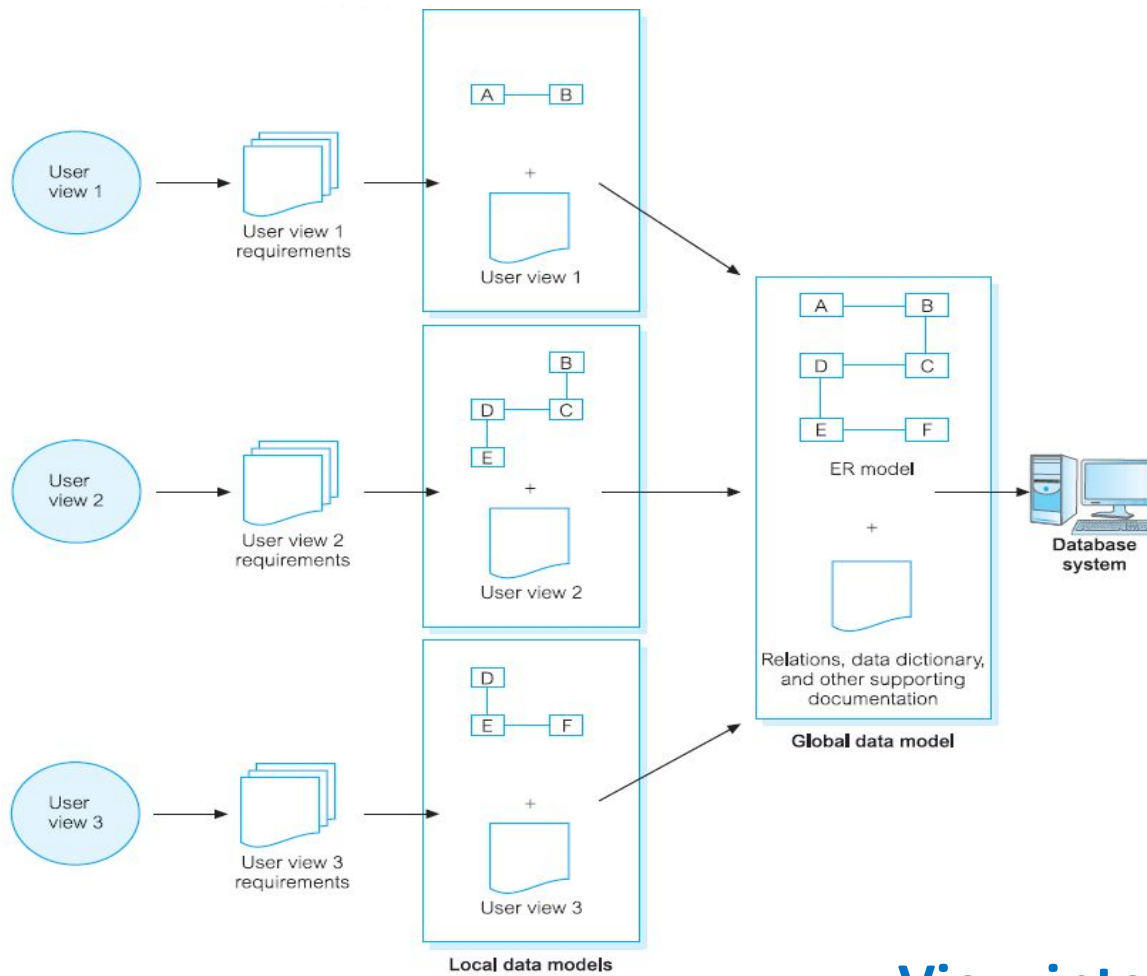


Figure 10.4 The view integration approach to managing multiple user views 1 to 3.

View integration approach

4) Database Design

- The process of creating a design for a database that will support the enterprise's operations and objectives.
- Major aims:
 - Represent data and relationships between data required by all major application areas and user groups.
 - Provide data model that supports any transactions required on the data.
 - Specify a minimal design that is appropriately structured to achieve stated performance requirements for the system (such as response times).

4) Database Design

Approaches include:

Top-down

- Starts with developing data models which contain high-level entities and relationships.
- Then refine this high-level data models into a lower-level entities, relationships and attributes.
- Use **Entity Relationship Model (ERM)**

Bottom-up

- Begins at the fundamental analysis of the associations between attributes
- Grouped into relations that represent types of entities and relationship between entities
- Use **Normalization process**

Inside-out

- Begins with identifying major entities
- Then spreading out to consider other entities, attributes associated to those first identified.

Mixed

- The mixed strategy approach uses both the bottom-up and top-down approach for various parts of the model before finally combining all parts together.

4) Database Design

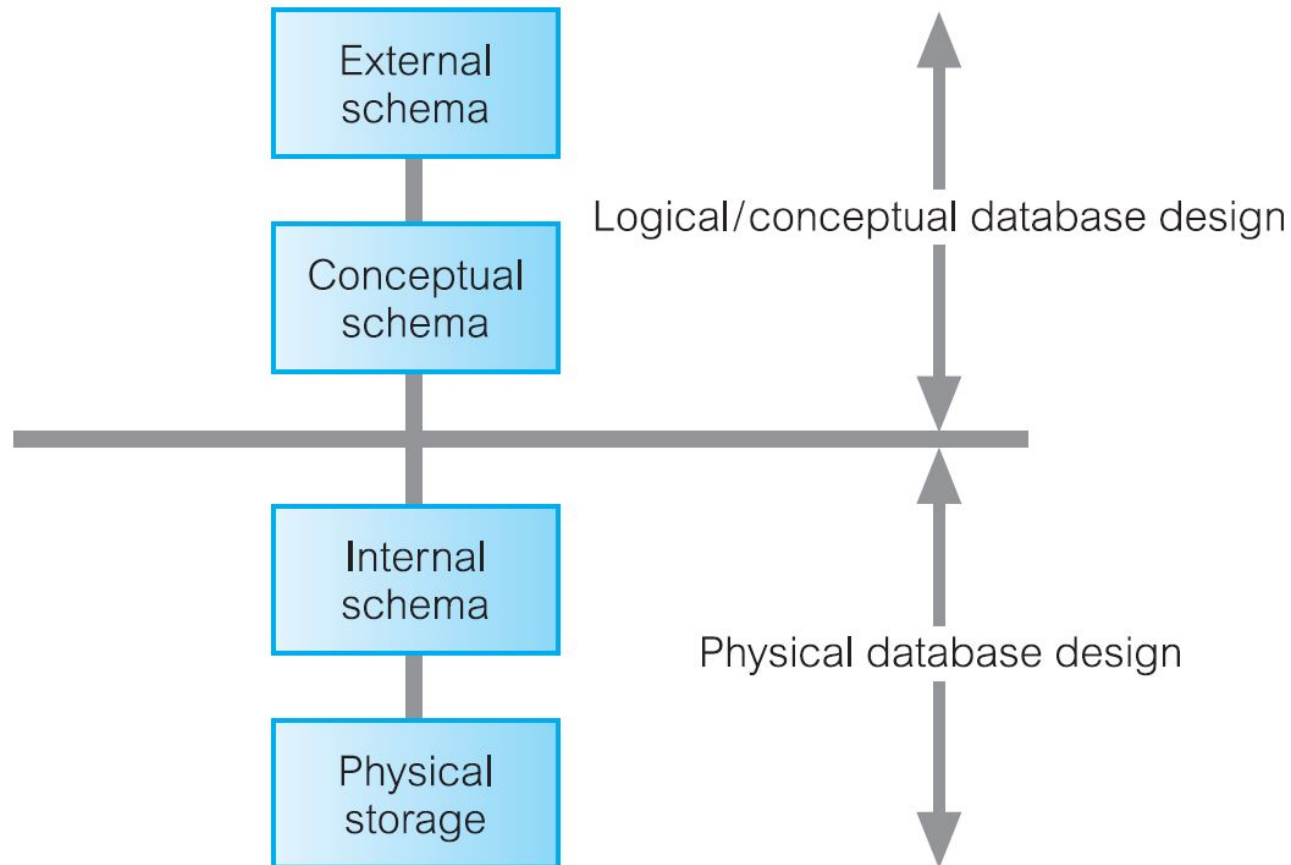
- Main purposes of data modeling include:
 - to assist in understanding the meaning (semantics) of the data;
 - to facilitate communication about the information requirements.
- Building **data model** requires answering questions about entities, relationships, and attributes.
 - A data model ensures we understand:
 - each user's perspective of the data;
 - nature of the data itself, independent of its physical representations;
 - use of data across user views.

4) Database Design

- Three phases of database design:
 - Conceptual, Logical and Physical

Conceptual	<ul style="list-style-type: none"> • Process of constructing a model of information used in an enterprise, independent of all physical considerations. • Entity Relationship Model; Data dictionary
Logical	<ul style="list-style-type: none"> • Process of constructing a model of information used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations. • Normalized database (relational) schemas
Physical	<ul style="list-style-type: none"> • Process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.

Three-level ANSI-SPARC Architecture and Phases of Database Design



Other Stages

- DBMS selection
- Application design – User interface (UI) design and application program design
- Prototyping
- Implementation
- Data Conversion and loading
- Testing
- Operational Maintenance

01 THE RELATIONAL MODEL

02 DATABASE DESIGN

- CONCEPTUAL DESIGN



Design Methodology

- Structured approach that uses procedures, techniques, tools, and documentation aids to support and facilitate the process of design.
- Database design methodology has 3 main phases:
 - Conceptual database design;
 - Logical database design;
 - Physical database design.

Database Design Phases

Conceptual

- Process of constructing a **model of information** used in an enterprise, independent of all physical considerations.
- **Entity Relationship Model; Data dictionary**

Logical

- Process of constructing a model of information used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations.
- **Normalized database (relational) schemas**

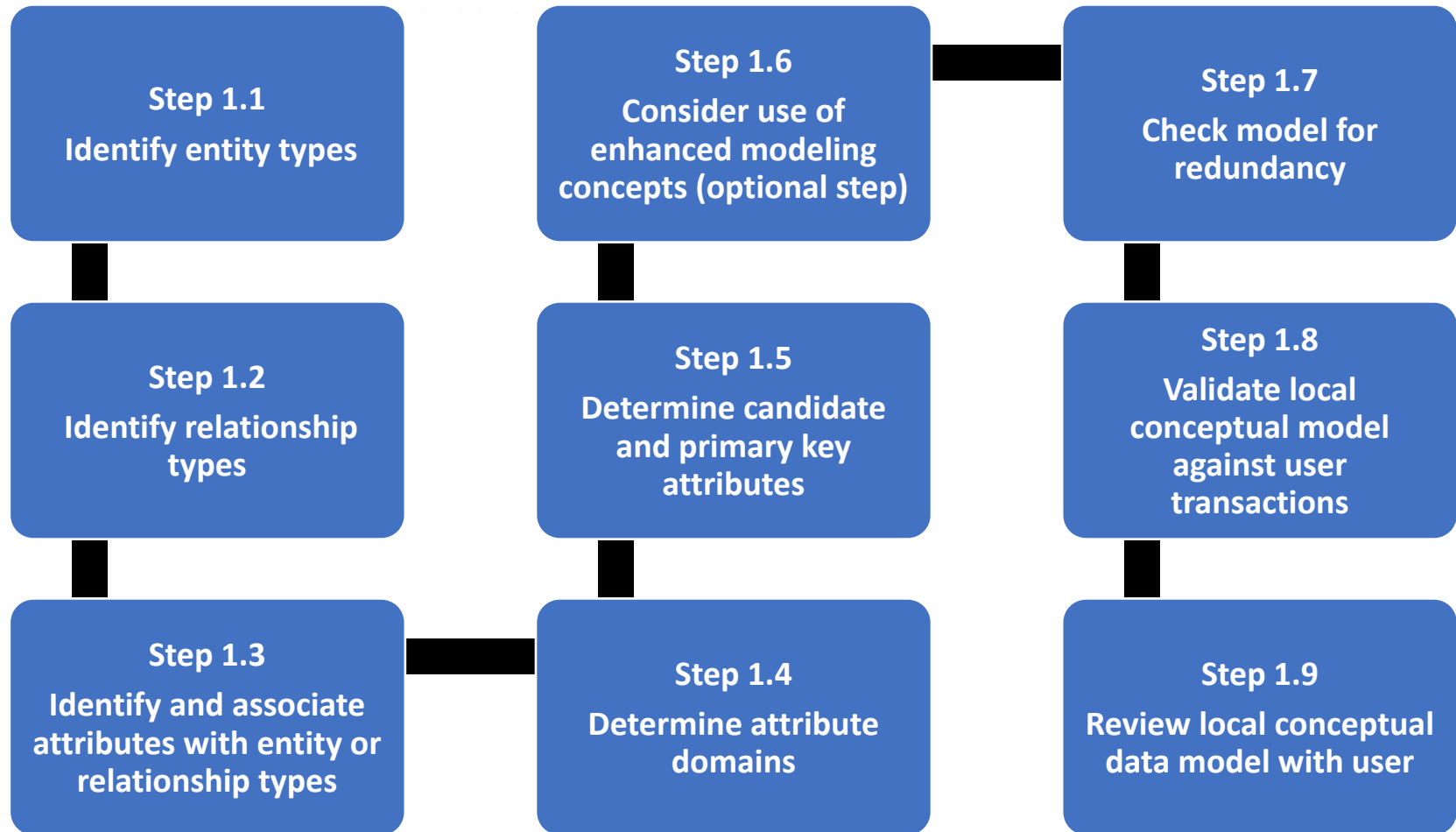
Physical

- Process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.

Critical Success Factors in DB Design

- Work interactively with users as much as possible.
- Follow a structured methodology throughout the data modelling process.
- Employ a data-driven approach.
- Incorporate structural and integrity considerations into the data models.
- Combine conceptualization, normalization, and transaction validation techniques into the data modelling methodology.

CD: Construct local conceptual data model for each user view - ERM



Conceptual Design: Steps and Outputs

Steps	Output
1.1	List of entities □ initial data dictionary (DD - Fig 1)
1.2	Relationships between entities □ ERD (entity, relationship and multiplicity constraints are shown (Fig 2); a more complete DD (Fig 3)
1.3	Attributes belong to entities & relationships □ DD (Fig 4)
1.4	Domain for each attribute □ a more complete DD
1.5	Candidate keys and primary keys for every entity □ ERD with PK (Fig 5)
1.6	ERD or Enhance ERD (EERD – Fig 6)
1.7	Revised ERD/EERD with no redundancies
1.8	Validated (against user transactions) ERD/EERD with pathways(Fig 7)
1.9	Reviewed (with users) ERD/EERD

Let's go to Entity Relationship Model (ERM)

We'll come back to this slide once we're done with ERM

Conceptual Data Model Steps & Output (1/14)

• Step 1.1: Identify entities and document them in Data Dictionary

- Example of data dictionary that documents the entities for Staff of *DreamHome* (Example as figure below)

<i>Entity name</i>	<i>Description</i>	<i>Aliases</i>	<i>Occurrence</i>
Staff	General term describing all staff employed by <i>DreamHome</i> .	Employee	Each member of staff works at one particular branch.
PropertyForRent	General term describing all property for rent.	Property	Each property has a single owner and is available at one specific branch, where the property is managed by one member of staff. A property is viewed by many clients and rented by a single client, at any one time.

Figure 1

Conceptual Data Model Steps & Output (2/14)

• Step 1.2: Identify relationship type between entities – show in ERM and Data Dictionary

- Example of ER diagram for Staff user views of *DreamHome* (Figure below)

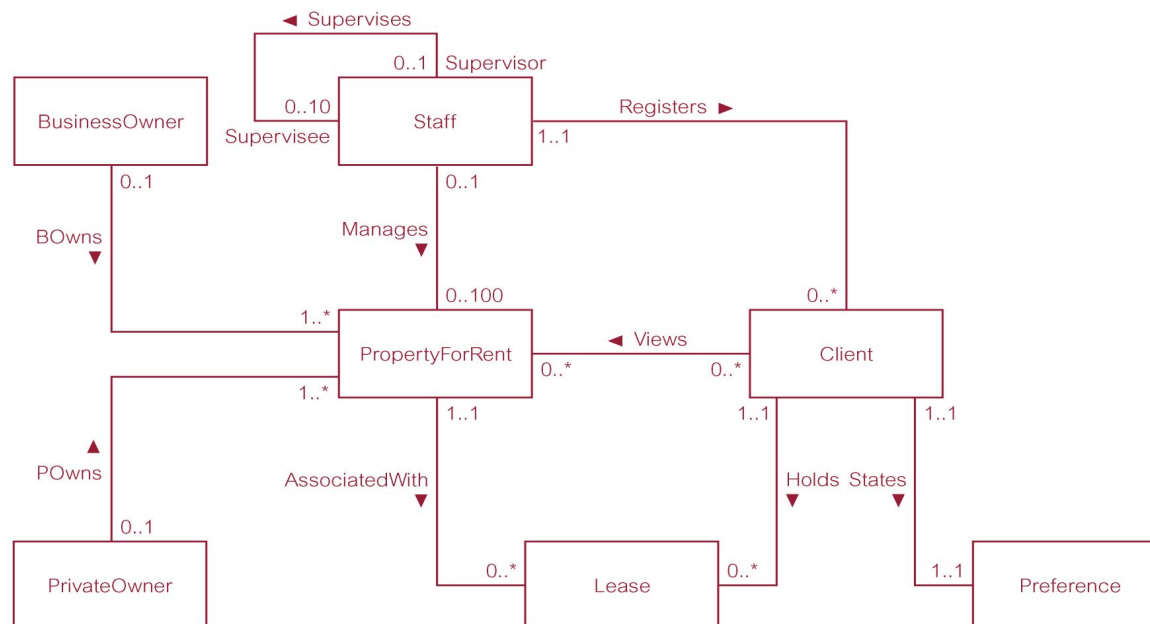


Figure 2

Conceptual Data Model Steps & Output (3/14)

- Example of the data dictionary for the Staff of *DreamHome* showing the description of relationship

<i>Entity name</i>	<i>Multiplicity</i>	<i>Relationship</i>	<i>Entity name</i>	<i>Multiplicity</i>
Staff	0..1 0..1	<i>Manages</i> <i>Supervises</i>	PropertyForRent	0..100 0..10
PropertyForRent	1..1	<i>AssociatedWith</i>	Lease	0..*

Figure 3

Conceptual Data Model Steps & Output (4/14)

- **Step 1.3: Identify and associate attributes with entity or relationship types**

- To identify and associate attributes with the appropriate entity or relationship types and document the details of each attribute.
- How?
 - Ask question: What information are we required to hold on each entity or attribute?
 - Answer can be found in system's specification and/or from users
- Determine attributes classifications: composite, single, multi-valued, derived
- Document the identified attributes (refer next slide)

Conceptual Data Model Steps & Output (5/14)

- Example of data dictionary for Staff of *DreamHome* showing the description of attributes

Entity name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	...
Staff	staffNo	Uniquely identifies a member of staff	5 variable characters	No	No	
	fName	First name of staff	15 variable characters	No	No	
	lName	Last name of staff	15 variable characters	No	No	
	position	Job title of member of staff	10 variable characters	No	No	
	sex	Gender of member of staff	1 character (M or F)	Yes	No	
	DOB	Date of birth of member of staff	Date	Yes	No	
PropertyForRent	propertyNo	Uniquely identifies a property for rent	5 variable characters	No	No	

Figure 4

Conceptual Data Model Steps & Output (6/14)

• Step 1.4: Determine attribute domains

- To determine domains for the attributes and document the details of each domain
- Example:
 - “sex” attribute of the “Staff” entity as being either “M” or “F”. The domain of this attribute is a single character string consisting of values “M” or “F”.
- The domains for each attributes includes:
 - Allowable set of values for attributes
 - Sizes and format of the attribute

Conceptual Data Model Steps & Output (7/14)

• Step 1.5: Determine candidate and primary key

- To identify the candidate key(s) for each entity and if there is more than one candidate key, choose one to be the primary key.
- Use following guidelines to help make the selection:
 - The candidate key with minimal set of attributes
 - The candidate key that is least likely to have its values changed
 - The candidate key with fewest characters (textual attributes)
 - The candidate key with smallest maximum value (numerical attributes)
 - The candidate key that is easiest to use from the users' point of view

Conceptual Data Model Steps & Output (8/14)

- Example of ER diagram for Staff user view of *DreamHome* with primary key added

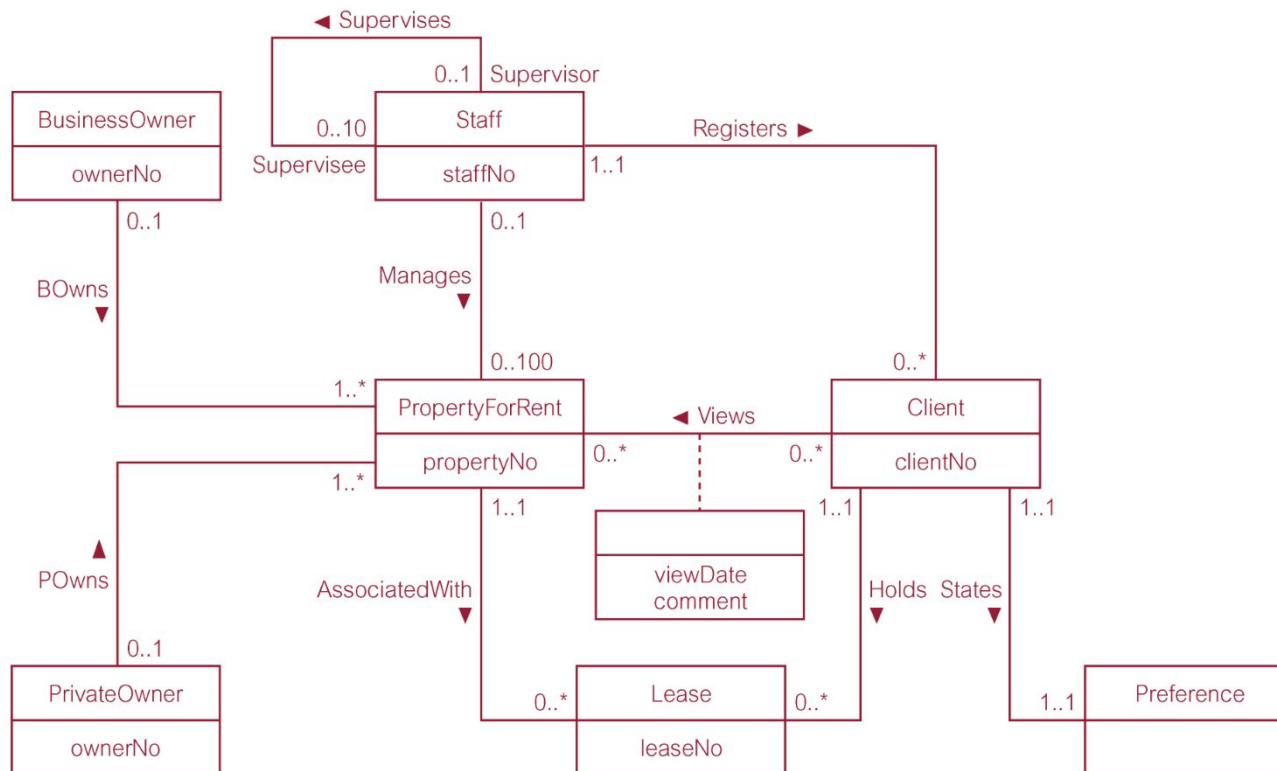


Figure 5

Conceptual Data Model Steps & Output (9/14)

- **Step 1.6: Consider use of enhanced modelling concepts (optional step)**

Example of ER diagram for Staff user view of *DreamHome* with specialization / generalization

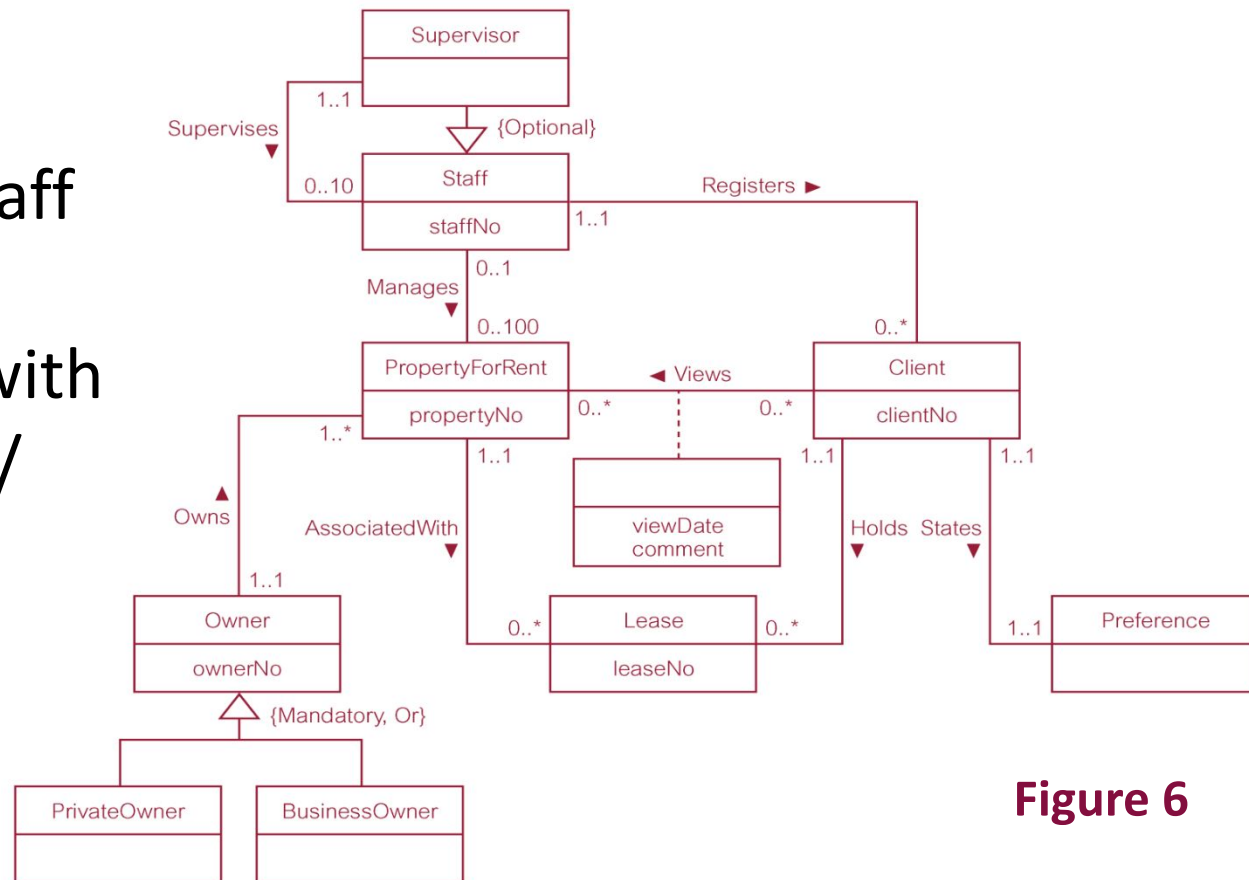


Figure 6

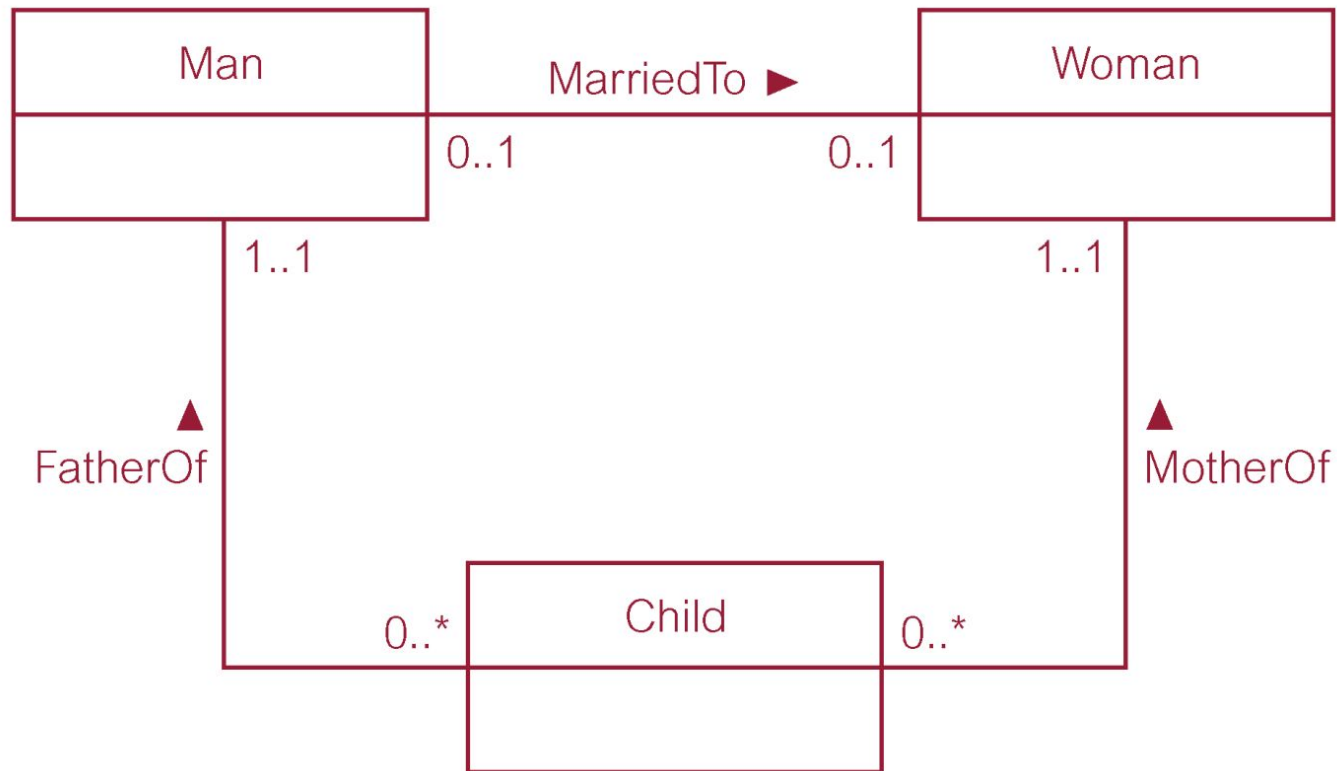
Conceptual Data Model Steps & Output (11/14)

• Step 1.7: Check model for redundancy

- To check for the presence of any redundancy in the model.
- Activities in this step are:
 - Re-examine one to one (1:1) relationships.
 - Example: Staff and employee that are actually the same. In this case two entities should be merged.
 - Remove redundant relationships.
 - Consider time dimension.
 - Example: Slide 19 shows the relationships between Man, Woman and Child. Consider the situation
- It is important to examine the meaning of each relationship between entities when assessing redundancy.

Conceptual Data Model Steps & Output (12/14)

- Example of non-redundant relationship



Conceptual Data Model Steps & Output (13/14)

- **Step 1.8: Validate local conceptual model against user transactions**

- To ensure that the model supports the transactions required by the view.
- Two approaches:
 - Describing transactions.
Example: Transaction (d) – List the details of properties managed by a named member of staff at the branch.
 - Using transaction pathways.
Example: Refer next slide.

Conceptual Data Model Steps & Output (14/14)

- Example of using pathway to check the model supports the user transaction for Staff user view

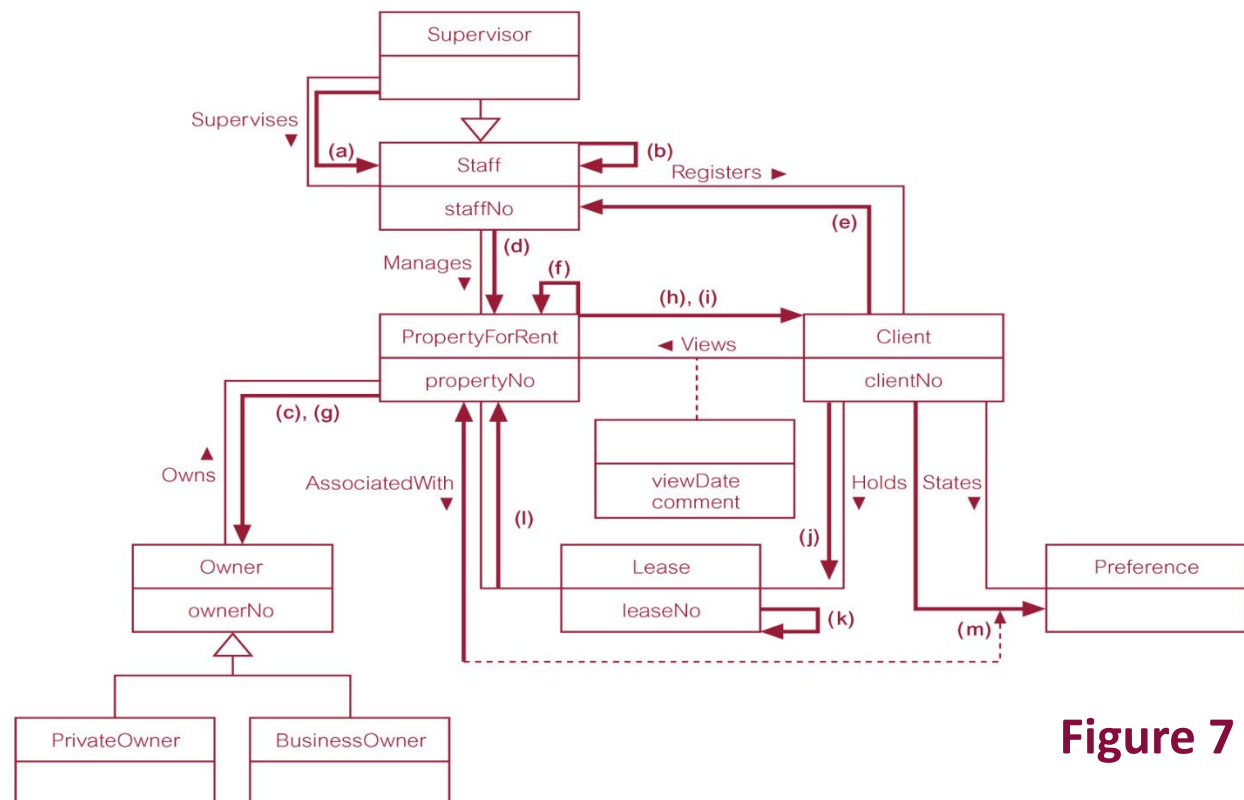
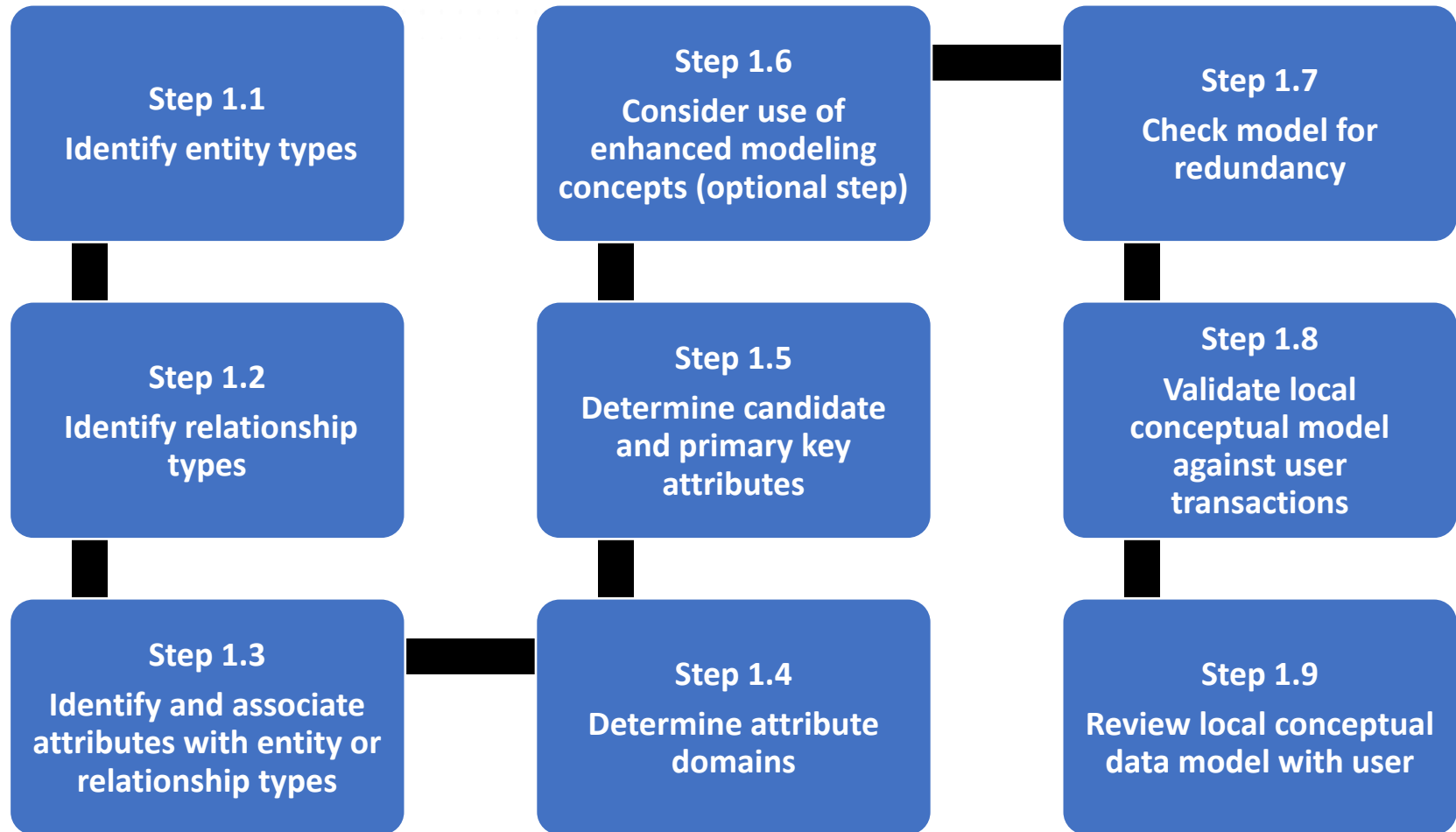


Figure 7

Summary (1/2)



Summary (2/2)

Steps	Output
1.1	List of entities □ initial data dictionary (DD - Fig 1)
1.2	Relationships between entities □ ERD (entity, relationship and multiplicity constraints are shown (Fig 2); a more complete DD (Fig 3)
1.3	Attributes belong to entities & relationships □ DD (Fig 4)
1.4	Domain for each attribute □ a more complete DD
1.5	Candidate keys and primary keys for every entity □ ERD with PK (Fig 5)
1.6	ERD or Enhance ERD (EERD – Fig 6)
1.7	Revised ERD/EERD with no redundancies
1.8	Validated (against user transactions) ERD/EERD with pathways(Fig 7)
1.9	Reviewed (with users) ERD/EERD



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

***Innovating Solutions
Menginovasi Penyelesaian***