people, places, or concepts of interest, excluding those nouns that are merely qualities of other objects. Document entity types.

Step 1.2: Identify relationship types

Identify the important relationships that exist between the entity types that have been identified. Use Entity–Relationship (ER) modeling to visualize the entity and relationships. Determine the multiplicity constraints of relationship types. Check for fan and chasm traps. Document relationship types.

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

Associate attributes with the appropriate entity or relationship types. Identify simple/composite attributes, single-valued/multi-valued attributes, and derived attributes. Document attributes.

Step 1.4: Determine attribute domains

Determine domains for the attributes in the conceptual model. Document attribute domains.

Step 1.5: Determine candidate, primary, and alternative key attributes

Identify the candidate key(s) for each entity and, if there is more than one candidate key, choose one to be the primary key. Document primary and alternative keys for each strong entity.

Step 1.6: Consider use of enhanced modeling concepts (optional step)

Consider the use of enhanced modeling concepts, such as specialization/generalization, aggregation, and composition.

Step 1.7: Check model for redundancy

Check for the presence of any redundancy in the model. Specifically, re-examine one-to-one (1:1) relationships, remove redundant relationships, and consider time dimension.

Step 1.8: Validate conceptual data model against user transactions

Ensure that the conceptual data model supports the required transactions. Two possible approaches are describing the transactions and using transaction pathways.

Step 1.9: Review conceptual data model with user

Review the conceptual data model with the user to ensure that the model is a "true" representation of the data requirements of the enterprise.

Step 2: Build Logical Data Model

Build a logical data model from the conceptual data model and then validate this model to ensure that it is structurally correct (using the technique of normalization) and to ensure that it supports the required transactions.

Step 2.1: Derive relations for logical data model

Create relations from the conceptual data model to represent the entities, relationships, and attributes that have been identified. Table D.1 summarizes how to map entities, relationships and attributes to relations. Document relations and foreign key attributes. Also, document any new primary or alternate keys that have been formed as a result of the process of deriving relations.

Step 2.2: Validate relations using normalization

Validate the relations in the logical data model using the technique of normalization. The objective of this step is to ensure that each relation is in at least Third Normal Form (3NF).

Step 2.3: Validate relations against user transactions

Ensure that the relations in the logical data model support the required transactions.

Step 2.4: Check integrity constraints

Identify the integrity constraints, which includes specifying the required data, attribute domain constraints, multiplicity, entity integrity, referential integrity, and general constraints. Document all integrity constraints.

Step 2.5: Review logical data model with user

Ensure that the users consider the logical data model to be a true representation of the data requirements of the enterprise.

TABLE D.1 Summary of how to map entities and relationships to relations.

ENTITY/RELATIONSHIP/ATTRIBUTE	MAPPING TO RELATION(S)
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 D.2.
: 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.
Multivalued attribute	Create a relation to represent the multivalued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.

TABLE D.2 Guidelines for the representation of a superclass/subclass relationship based on the participation and disjoint constraints.

PARTICIPATION CONSTRAINT	DISJOINT CONSTRAINT	MAPPING TO RELATION(S)
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

Step 2.6: Merge logical data models into global model

The methodology for Step 2 is presented so that it is applicable for the design of simple and complex database systems. For example, to create a database with a single user view or multiple user views being managed using the centralized approach (see Section 10.5), Step 2.6 is omitted. If, however, the database has multiple user views that are being managed using the view integration approach (see Section 10.5), Steps 2.1 to 2.5 are repeated for the required number of data models, each of which represents different user views of the database system. In Step 2.6 these data models are merged. Typical tasks associated with the process of merging are:

(1) Review the names and contents of entities/relations and their candidate keys.
(2) Review the names and contents of relationships/foreign keys.
(3) Merge entities/relations from the local data models.
(4) Include (without merging) entities/relations unique to each local data model.
(5) Merge relationships/foreign keys from the local data models.
(6) Include (without merging) relationships/foreign keys unique to each local data model.
(7) Check for missing entities/relations and relationships/foreign keys.
(8) Check foreign keys.
(9) Check integrity constraints.
(10) Draw the global ER/relation diagram.
(11) Update the documentation. Validate the relations created from the global logical data model using the technique of normalization and ensure that they support the required transactions, if necessary.

Step 2.7: Check for future growth

Determine whether there are any significant changes likely in the foreseeable future, and assess whether the logical data model can accommodate these changes.

Step 3: Translate Logical Data Model for Target DBMS

Produce a relational database schema that can be implemented in the target DBMS from the logical data model.

Step 3.1: Design base relations

Decide how to represent the base relations that have been identified in the logical data model in the target DBMS. Document design of base relations.

Step 3.2: Design representation of derived data

Decide how to represent any derived data present in the logical data model in the target DBMS. Document design of derived data.

Step 3.3: Design general constraints

Design the general constraints for the target DBMS. Document design of general constraints.

Step 4: Design File Organizations and Indexes

Determine the optimal file organizations to store the base relations and the indexes that are required to achieve acceptable performance, that is, the way in which relations and tuples will be held on secondary storage.

Step 4.1: Analyze transactions

Understand the functionality of the transactions that will run on the database and analyze the important transactions.

Step 4.2: Choose file organizations

Determine an efficient file organization for each base relation.

Step 4.3: Choose indexes

Determine whether adding indexes will improve the performance of the system.

Step 4.4: Estimate disk space requirements

Estimate the amount of disk space that will be required by the database.

Step 5: Design User Views

Design the user views that were identified during the requirements collection and analysis stage of the relational database system development lifecycle. Document design of user views.

Step 6: Design Security Mechanisms

Design the security measures for the database system as specified by the users. Document design of security measures.

Step 7: Consider the Introduction of Controlled Redundancy

Determine whether introducing redundancy in a controlled manner by relaxing the normalization rules will improve the performance of the system. For example, consider duplicating attributes or joining relations together. Document introduction of redundancy.

Step 8: Monitor and Tune the Operational System

Monitor the operational system and improve the performance of the system to correct inappropriate design decisions or reflect changing requirements.