Appendix B

Summary of the database design methodology

In this chapter you will learn:

- Database design is composed of two main phases: logical and physical database design.
- The steps involved in the main phases of the database design methodology.

In this book, we present a database design methodology for relational databases. This methodology is made up of the two main phases logical database design and physical database design, which were described in detail in Chapters 9, 10, and 12 to 16. In this appendix, we summarize the steps involved in these phases for those readers who are already familiar with database design.

Step 1 Create and check ER model

During analysis, you will have identified a number of user views. Depending on the amount of overlap, for manageability you may decide to merge some of these views. The purpose of this step is to build a logical data model of the organization (or part of the organization) for each of these (possibly merged) views.

Step 1.1 Identify entities

Identify and document the main entities in the organization.

Step 1.2 Identify relationships

Identify the important relationships that exist between the entities that you have identified. Determine the multiplicity constraints of the relationships. Document relationships. Use Entity–Relationship (ER) modeling when necessary.

Step 1.3 Identify and associate attributes with entities or relationships

Associate attributes with the appropriate entities or relationships. 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 ER model. Document attribute domains.

Step 1.5 Determine candidate, primary key, and alternate 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, the others becoming alternate keys. Document candidate, primary, and alternate keys for each strong entity.

Step 1.6 Specialize/Generalize entities (optional step)

Identify superclass and subclass entities, where appropriate.

Step 1.7 Check model for redundancy

Examine the ER model to ensure there is no redundancy. Specifically, re-examine 1:1 relationships and remove redundant relationships.

Step 1.8 Check model supports user transactions

Ensure that the ER model supports the transactions required by the users.

Step 1.9 Review model with users

Step 2 Map ER model to tables

Map the ER model to a set of tables and check the structure of the tables.

Step 2.1 Map tables

In this step, you produce a set of tables to represent the entities, relationships, attributes, and constraints for the ER model created in Step 1. The structures of the tables are derived from the information that describes the ER model. This information includes the data dictionary, and any other documentation that describes the model. Also, document any new primary or candidate keys that have been formed as a result of the process of creating tables for the ER model.

The basic rules for creating tables are as follows:

(a) For each entity, create a table that includes all the entity's simple attributes.

(b) Relationships can be represented by the primary key/foreign key mechanism. In deciding where to post the foreign key, you must identify the 'parent' and 'child' entities in the relationship. The parent entity then posts a copy of its primary key into the child table, to act as the foreign key.

A summary of the rules for creating tables from an ER model is shown in Table B.1.

Table B.1 Summary of how to represent entities, relationships, and multi-valued attributes as tables.

Entity/Relationship/Attribute	Representation as table(s)
Strong or weak entity	Create table that includes all simple attributes.
1:* binary relationship	Post copy of primary key of entity on 'one' side to table representing entity on 'many' side. Any attributes of relationship are also posted to 'many' side.
1:* recursive relationship	As entity on 'one' and 'many' side is the same, the table representing the entity receives a second copy of the primary key, which is renamed, and also any attributes of the relationship.
1:1 binary relationship:	
Mandatory participation on *both* sides	Combine entities into one table.
Mandatory participation on *one* side	Post copy of primary key of entity with optional participation to table representing entity with mandatory participation. Any attributes of relationship are also posted to table representing entity with mandatory participation.

Table B.1 *Continued*

Entity/Relationship/Attribute	Representation as table(s)
Optional participation on *both* sides	Without further information, post copy of primary key of one entity to the other. However, if information is available, treat entity that is closer to having mandatory participation as being the child entity.
: binary relationship/ complex relationship	Create a table to represent the relationship and include any attributes associated with the relationship. Post a copy of the primary key from each parent entity into the new table to act as foreign keys.
Multi-valued attribute	Create a table to represent the multi-valued attribute and post a copy of the primary key of the parent entity into the new table to act as a foreign key.

You may use Step 1.6 to introduce specialization/generalization into your ER model. For each superclass/subclass relationship, you identify the superclass as the parent entity and the subclass as the child entity. There are various options on how you may best represent such a relationship as one or more tables. The selection of the most appropriate option is dependent on the participation and disjoint constraints on the superclass/subclass relationship. A summary of how to map tables from your EER model is shown in Table B.2.

Table B.2 Options available for the representation of a superclass/subclass relationship based on the participation and disjoint constraints.

Participation constraint	Disjoint constraint	Tables required
Mandatory	Nondisjoint {And}	Single table
Optional	Nondisjoint {And}	Two tables: one table for superclass and one table for all subclasses
Mandatory	Disjoint {Or}	Many tables: one table for each combined superclass/subclass
Optional	Disjoint {Or}	Many tables: one table for superclass and one for each subclass

Step 2.2 Check table structures using normalization

The purpose of this step is to examine the groupings of columns in each table created in Step 2.1. You check the composition of each table using the rules of normalization. Each table should be in at least third normal form (3NF).

Step 2.3 Check tables support user transactions

In this step, you ensure that the tables support the required transactions, which are described in the users' requirements specifications.

Step 2.4 Check business rules

Check that all business rules are represented in the logical database design. These include specifying the required data, attribute domain constraints, entity integrity, multiplicity, referential integrity, and any other business rules. Document all business rules.

Step 2.5 Review logical database design with users

Ensure that the logical database design is a true representation of the data requirements of the organization (or part of the organization) to be supported by the database.

Step 2.6 Build and check global logical data model*

Combine the individual local logical data models into a single global logical data model that represents the data requirements of the organization (or part of the organization) to be supported by the database.

Step 2.6.1 Merge local logical data models into global model

Merge the individual local logical data models into a single global logical data model. Some typical tasks of this step are as follows:

(1) Review the names and contents of entities/tables and their primary keys.
(2) Review the names and contents of relationships/foreign keys.
(3) Merge entities/tables from the local data models.
(4) Include (without merging) entities/tables unique to each local data model.
(5) Merge relationships/foreign keys from the local data models.

* Step 2.6 is only required when creating a multi-user view database, using the view integration approach (described in Appendix C).

(6) Include (without merging) relationships/foreign keys unique to each local data model.
(7) Check for missing entities/tables and relationships/foreign keys.
(8) Check foreign keys.
(9) Check business rules.
(10) Draw the global ER/table diagram.
(11) Update the documentation.

Step 2.6.2 Check global logical data model

This step is equivalent to Steps 2.3 and 2.4, where you check the structure of the tables created for the global data model using normalization and then check that these tables are capable of supporting all user transactions.

Step 2.6.3 Check for future growth

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

Step 2.6.4 Review global logical data model with users

Ensure that the global logical data model is a true representation of the data requirements of the organization (or the part of the organization) to be supported by the database.

Step 3 Translate logical database design for target DBMS

Produce a basic working set of tables from the logical data model.

Step 3.1 Design base tables

Decide how to represent the base tables you have identified in the logical data model in the target DBMS. Document design of tables.

Step 3.2 Design representation of derived data

Consider how derived data will be represented. The choice is to calculate derived data each time it's needed or to introduce redundancy and store the derived data as a column in a table. Document design of derived data.

Step 3.3 Design remaining business rules

Design the remaining business rules for the target DBMS. Document design of the remaining business rules.

Step 4 Choose file organizations and indexes

Determine the file organizations that will be used to store the base tables; that is, the way in which tables and records will be held on secondary storage. Consider the addition of indexes to improve performance.

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 table.

Step 4.3 Choose indexes

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

Step 5 Design user views

Design the user views that you identified during the requirements collection and analysis stage.

Step 6 Design security mechanisms

Design the security measures for the database implementation as specified by the users during the requirements collection and analysis stage. Document the design of the 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. Consider duplicating columns or joining tables together to achieve improved performance. In particular, consider combining one-to-one (1:1) relationships, duplicating nonkey columns in one-to-many (1:*) relationships to reduce joins, duplicating foreign key columns in one-to-many (1:*) relationships to reduce joins, duplicating columns in many-to-many (*:*) relationships to reduce joins, introducing repeating groups, creating extract tables, and partitioning tables.

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.