PART V

DESIGN AND IMPLEMENTATION

CHAPTER 11

COMPLETING THE LOGICAL DESIGN

CHAPTER OBJECTIVES

- Understand the place and significance of logical design in the database development life cycle (DDLC)
- Study how data modeling fits in with logical design
- Review the various steps necessary to complete logical design
- Focus on logical design for the relational data model
- Learn how to document the logical design effort

In Chapter 4, while covering the database development life cycle (DDLC), we briefly discussed the design phase. The ultimate goal of a database development project is to have proper data content and structures in the database system. You must make sure that all the necessary real-world information requirements for your organization are correctly included in the database system. Before you determine how to do this and create a design, you need to know exact details about the real-world information—its exact nature and extent and how the information is used to run the business. You find out the details during the requirements definition phase. Once you have the details of all the information requirements, somehow you have to represent all of this in your design of the database system.

How do you keep data in the database? Data are stored in physical storage in the form of records, blocks, and files. Therefore, to properly store data in the database, you need to design the files, blocks, and records. This part of the design phase relates to the physical design of the database. Physical design must consider the stipulations of the DBMS selected for your organization. The selected DBMS lays down conditions, rules, and formats for the file structures.

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4 Copyright © 2003 by John Wiley and Sons, Inc.

Logical design precedes the physical design activity. In the logical design activity, you create a conceptual design, not in terms of physical files, blocks, and records, but in terms of how data are perceived by the users and developers. If the target DBMS is relational, data are perceived as two-dimensional tables. For a hierarchical DBMS, data are seen as being organized in the form of hierarchical segments. In a network database environment, data are perceived as network nodes. The tables, segments, or nodes do not represent how data are physically kept in data storage—they just indicate how we perceive data as being organized. Logical design consists of the tasks necessary to arrive at the design of the tables, segments, or nodes, depending on the type of target DBMS.

SIGNIFICANCE OF LOGICAL DESIGN

Think back to the days of designing file-oriented data systems before the database era. Considering the business processes, the analyst would prepare the layouts for the master files and the transaction files. For an order entry application, the analyst would design the order master file and order transaction file. He or she then defined the necessary index files. This was the extent of the data design. No separation of data design into logical design and physical design existed. In fact, such separation did not make sense in a file-oriented data environment.

Although this was the best available design approach at that time, two problems were encountered. The first problem related to the lack of data independence. Every time changes in business conditions warranted revisions to data structures, the design of the file layouts had to be changed. The design changes affected the physical storage of data. The second problem referred to lack of simplicity for users in understanding the data. Users could not comprehend the technical details of physical file layouts. They need to perceive and understand data in simpler ways. They can easily perceive data as tables, segments, or nodes—not expressed in terms of data blocks, records, fields, storage addresses, and so on.

Logical Structure Versus Physical Structure

Database management systems provide means for separating the logical structure and manipulation of data from their physical representation in computer hardware. The relational revolution further confirmed the need for the separation of the logical structure from the physical structure. Logical data design simply covers the tasks necessary for creating the logical structure as distinct from the physical structure.

Before proceeding further, try to get a clear picture of what we mean by the logical and physical structures. Consider a small database system consisting of data about customers, orders, and details of orders. This is a very simple database just for the sake of illustration of the difference between the logical and physical structures. Figure 11-1 illustrates this example and distinguishes between the logical and physical structures.

This figure illustrates the logical and physical structures implementing the database system based on relational technology. If the database system were to be based on the hierarchical data model, the logical and physical structures would be differ-

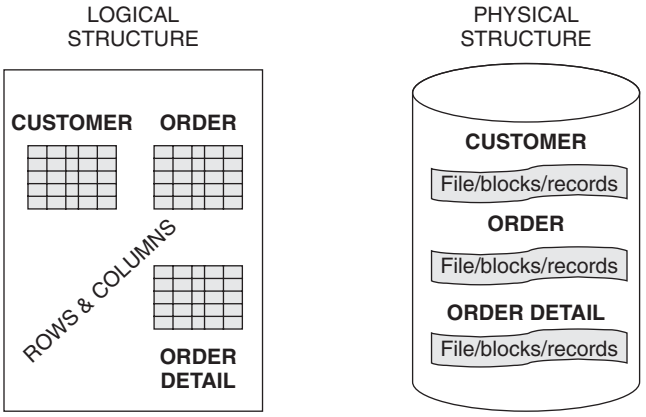

Figure 11-1 Logical and physical structures.

ent. Note that the logical structure is based on the conventional data model adopted for the implementation. The physical structure is determined by the specific DBMS used for the particular conventional data model.

Logical Design Phase in DDLC

Let us retrace the steps discussed so far in the development of a database system. Quickly review the DDLC steps from the beginning and note where logical design phase fits in the DDLC. After initial planning and feasibility study, you launch requirements analysis and definition. You have to determine exactly what information is required to be contained in the final database system. You have gone through the process of creating a generic data model to truly represent the real-world information requirements. You already know the necessity for creating a generic data model first. You know that only a generic data model unrestricted by any constraints can truly represent all the details of real-world information.

After the creation of a generic data model, what comes next? You know that a generic semantic data model created through a data modeling technique such as the E-R modeling technique has no flavor of the relational, hierarchical, or network approaches. If you are developing a relational database, your semantic data model must be transformed into a relational data model. What is the relational data model at this stage of the development? It does not represent how data are to be stored in the final database system. Then what does it represent? It indicates the logical structure, at a conceptual level, of the data perceived as two-dimensional tables. Figure 11-2 shows the place of logical design phase in DDLC.

What is the logical design phase? It is the set of tasks in DDLC intended for the design of the logical structure. Observe the presentation in Figure 11-2. Note the illustration of logical design phase for the small orders database above.

Why This Phase Is Necessary

Imagine a situation in which you plan to bypass the logical design phase and go directly to the physical design phase from requirements definition. You will be

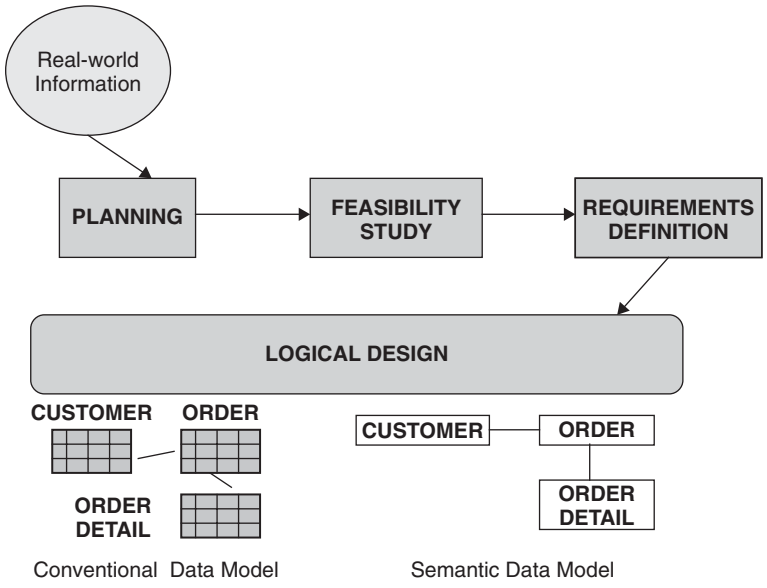

Figure 11-2 Logical design in DDLC.

taking the semantic data model and transforming it into the physical design. You will be designing files and records based on the features of the hardware. Your issues and concerns will be centered more on hardware features than on the information requirements. As hardware changes occur, your design will have to be modified.

Let us say that you are designing a relational database system by adopting this plan of bypassing the logical design. You need to define the files and records for data elements determined in your requirements definition phase. You are not going to think in terms of relational tables, but only about designing files and records. How are you going to ensure that your files will be free from data redundancy and that they will conform to the relational rules? How are you going to ensure that your files will be free from update, deletion, and addition anomalies? Remember, the relational model rests on the understanding that data may be perceived as two-dimensional, relational tables. Relational DBMSs work on this premise.

This logical design phase is essential for the following reasons:

- The output of the logical design phase represents data as perceived as tables, segments, or nodes—the way data are considered in the relational, hierarchical, and network models.
- Users do not understand the output of the physical design phase in the form of hardware files, records, and so on. Tables, segments, or nodes are better understood.
- Logical design enables data independence, a crucial advantage of the database approach.
- The design effort becomes easier and more manageable when you separate it into logical and physical design phases.

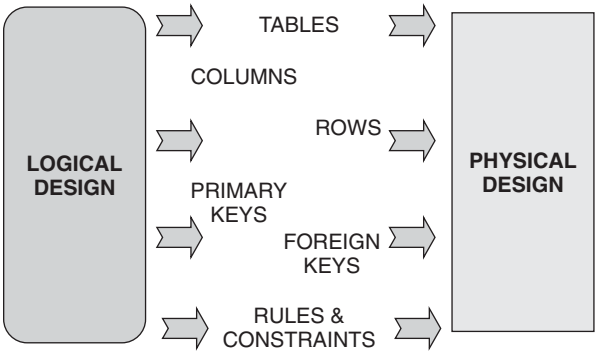

Figure 11-3 Output from logical design as input to physical design.

- In the logical design phase, you can focus on designing the tables, segments, or nodes without being bogged down by hardware considerations.

Input to Physical Design

When you finish the logical design phase, you produce a representation of the information requirements in the form of tables, segments, or nodes. If you are designing for the relational data model, your output from the logical design phase consists of two-dimensional tables with their columns and rows. The logical design phase takes the database development effort to this stage.

However, you know that data do not exist as tables, or segments, or nodes in physical storage. Therefore, to implement the target database system, you have to define the representation of data in formats that are suitable for physical storage. How do you accomplish the design of the physical structure? The output of the logical design phase serves as the input to the physical design phase. In the physical design phase, you take each table from the output of the logical design phase and transform it into file and record layouts.

Figure 11-3 illustrates how the output from the logical design phase is used as input to the physical design phase. Note the components of the logical design output. Also, observe once again how design is simplified by separating it into logical and physical design phases.

Ensuring Design Completeness

When you are near the end of the logical design phase, usually it is a good place in DDLC to ensure that the design is complete. You started from the real-world information that needs to be represented in the target database system. You gathered requirements and defined the content, extent, and scope of the information requirements. On the basis of the requirements definition, you created a semantic data model to represent the real-world information. Then you transformed the semantic data model into a relational data model. If your target database system were hierarchical or network, you would have transformed the semantic data model into a hierarchical or network data model. If you had adopted the traditional method of

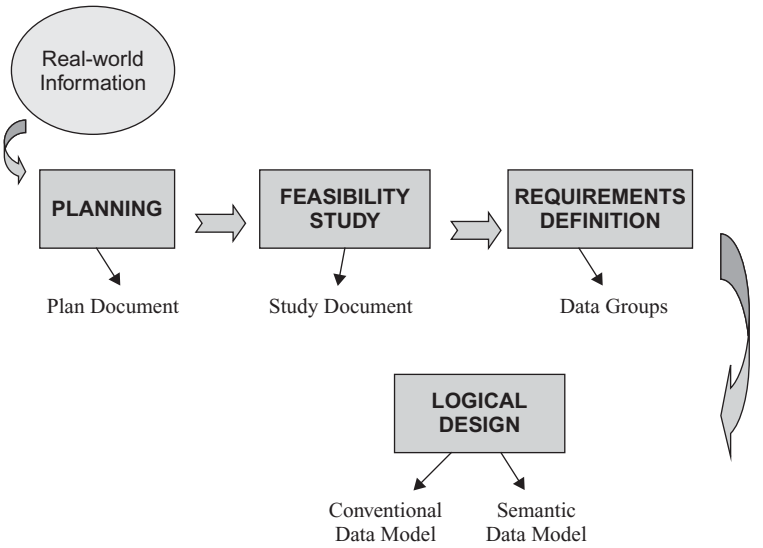

Figure 11-4 DDLC steps up to logical design.

designing the relational data model, you would have created it with the data normalization method. In any case, you now have a relational data model. Figure 11-4 recapitulates these steps.

Having gone through these distinct steps, how are you sure that the output of the logical design phase truly and completely represents the real-world information? Has something been missed along the way? Has something been misrepresented in the transition from step to step? Especially if the database project is large with multiple analysts working in each step of DDLC, there are more chances of things going wrong. So, this is a good stage in DDLC to pause and ensure the completeness and correctness of the design. Adjustments can be made with comparative ease at this point rather than after defining the files and records in the physical design phase.

Go back and review the outputs from all the steps leading up to this point and adjust the logical design accordingly. Pay attention to the following critical factors in your review:

- Real-world information requirements could create variations in the way business objects and the relationships among these are perceived in your organization. Ensure that your semantic data model represents all the nuances of the information requirements in your particular environment.
- Semantic data modeling techniques have all the necessary components to represent every aspect of real-world information. Become familiar with the features and purposes of the data model components and employ these suitably.
- If the scope of the information requirements for your organization is large and complex, your database project team will be dividing up the data modeling task. The team will be creating partial data models. Ensure that nothing gets lost

when consolidating the partial data models into the global semantic data model.

- When the semantic data model is transformed into the appropriate conventional data model—relational, hierarchical, or network—ensure that none of the essential properties of the conventional data model is overlooked.
- If your target database is relational, make sure that the transformed data model conforms to all the relational rules and constraints.

DATA MODELING IN LOGICAL DESIGN

Think of logical design phase as consisting of two distinct steps—creation of the semantic data model and transformation of the semantic data model into the desired conventional model. Sometimes, semantic data modeling is taken to be part of the requirements definition phase, perhaps considering the semantic model as the output of the requirements definition phase. We would rather place semantic data modeling in the logical database design phase. Both the semantic and the conventional data models are conceptual or logical models. They are conceptual representations of real-world information requirements.

Also, the skills needed for semantic data modeling and for gathering and defining information requirements are not the same. Analysts working on the requirements definition phase must possess special proficiency in analyzing business operations, reviewing business processes, and interviewing users to gather the requirements. Data modelers must have expertise in modeling techniques and knowledge of leading CASE tools.

Steps for Completing Logical Design

In the previous section, we listed a number of critical factors to be concerned about in the logical design process. Essentially, steps for completing the logical design relate to the fulfillment of these factors. Semantic data modeling is the beginning of the logical design phase. We have covered semantic data modeling in elaborate detail in earlier chapters. You know the components of a semantic data model; you are very familiar with data modeling techniques. Let us trace the steps to complete the logical design from the semantic data model.

Figure 11-5 shows a semantic data model for a small florist business. Note the components in the data model diagram. Observe the entities, attributes, and relationships. As this exercise is meant just to illustrate the steps for completing the logical design, we are not considering all the possible entities and attributes for a typical florist business. Let us begin with this data model and complete the logical design.

We will consider the relational data model as the conventional data model while studying the steps for completing the logical design. Similar steps are applicable if the conventional data models were to be hierarchical or network. Here are the obvious basic steps to complete the logical design from the semantic data model:

- Represent entities in the conventional data model.
- Represent attributes in the conventional data model.

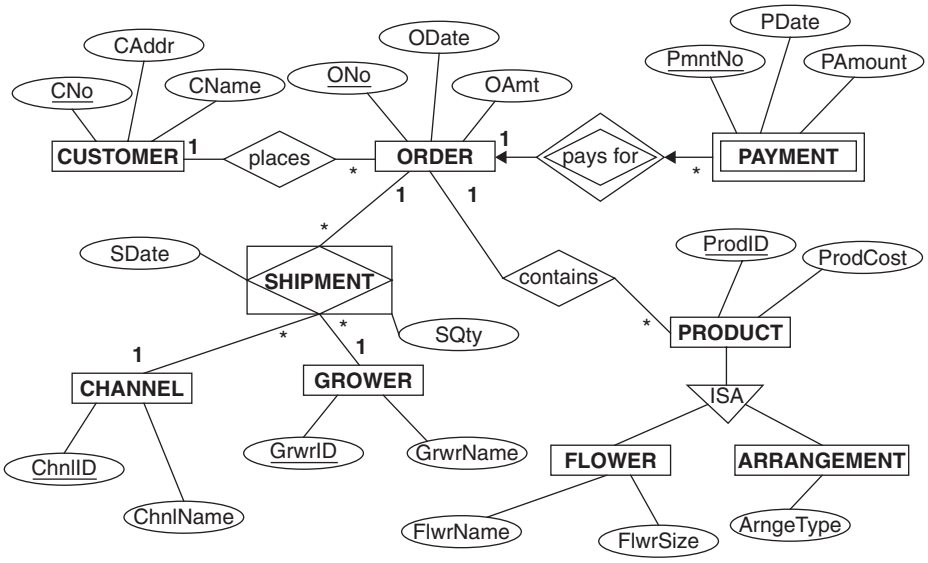

Figure 11-5 ERD for florist business.

- Represent relationships in the conventional data model.
- Include the necessary rules and constraints in the conventional data model.
- Review conventional data model for completeness.

Representing Entities

Figure 11-6 illustrates the representation of entities in the relational data model for the florist business. Note how each entity is represented by a relation or table in the relational data model.

Representing Attributes

Figure 11-7 illustrates the representation of attributes in the relational data model for the florist business. Note how the representation is done as columns for attributes. Also, observe how entity identifiers or primary keys are represented in the tables.

Representing Relationships

Figure 11-8 illustrates the representation of relationships in the relational data model for florist business. As you know, relationships are represented through foreign keys in the relational data model. Note the foreign key columns in the appropriate tables. CustNo is a foreign key column in the ORDER relation. Also, observe the foreign key columns in the PAYMENT, SHIPMENT, FLOWER, and ARRANGEMENT relations.

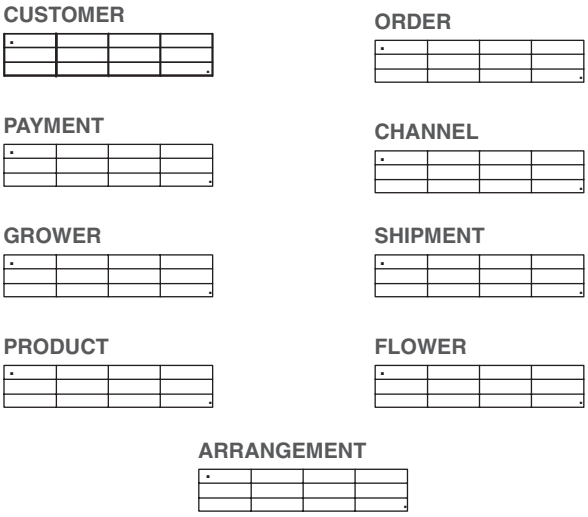

Figure 11-6 Representing entities for florist business.

CUSTOMER

CustNo	CustAddr	CustName
.		
		.

ORDER

OrdrNo	OrdrDate	OrdrAmt
.		
		.

PAYMENT

OrdrNo	PmntNo	PmntDate	PmntAmt
.			
			.

CHANNEL

ChnlID	ChnlName
.	

GROWER

GrwrID	GrwrName
.	

SHIPMENT

OrdrNo	ChnlID	GrwrID	ShipDate	ShipQty
.				
				.

PRODUCT

ProdID	ProdCost
.	

FLOWER

ProdID	ProdCost	FlwrName	FlwrSize
.			
		.	

ARRANGEMENT

ProdID	ProdCost	ArngeType
.		
		.

Figure 11-7 Representing attributes for florist business.

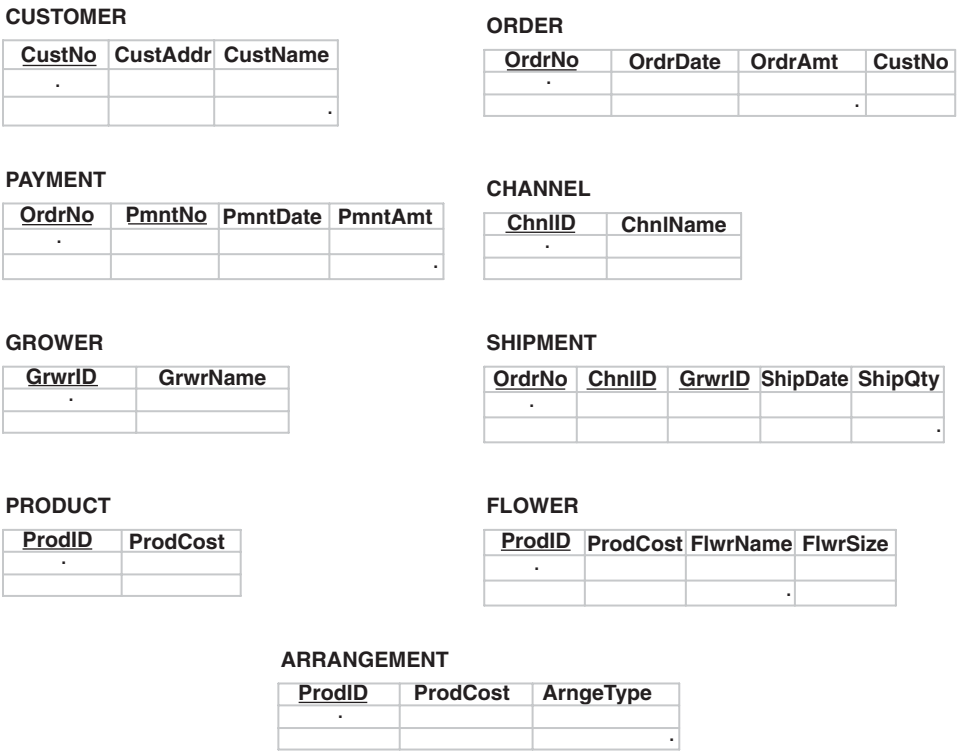

Figure 11-8 Representing relationships for florist business.

Rules and Constraints

This step in the logical design includes representation of the following in the relational data model:

- Reduction of many-to-many relationships into multiple one-to-many relationships
- Representation of foreign keys as additional columns
- Indication of optional and mandatory relationships through minimum cardinality indicators

Once this step is completed, logical design comes to an end, to be followed by physical design.

DESIGN FOR THE RELATIONAL DATA MODEL

Before going through the completion of the logical design for the relational data model, let us recapitulate the basic features of the relational model. This refresher will help you appreciate the special requirements for going through this phase of

completing the logical design. Here is a recapitulation of the significant properties of the relational data model.

- In the relational data model, data are perceived as in two-dimensional tables that conform to relational rules and constraints.
- A table represents a single entity type or business object set.
- Each row in a table representing an object refers to one instance of that object set.
- Each column in a table refers to an attribute of the object set. Data values in all the columns for a row denote the values of the attributes for that single object.
- Each row in a table is unique. The primary key value identifies each row uniquely.
- Each value of every attribute is atomic from a certain domain.
- The order of the columns in a relation is immaterial; the sequence of the rows is insignificant.
- No part of the primary key of any tuple in a relation can have a null value (entity integrity rule).
- The value of a foreign key in a table must either be null or be one of the values of the primary key in the related table (referential integrity rule).

Relation as the Single Design Concept

While completing the logical design for your relational database, keep the following fundamental notion as the main focus:

> The underlying design concept for a relational data model is the two-dimensional table or relation.

All of the logical design rests on the premise that data is formulated and perceived as two-dimensional tables. Within each table you have columns and rows. Columns and rows are used to express the different aspects of each business object represented by a table or relation. One or more columns are designated as the primary key for the relation. Every relationship among the tables is established by the values in a column known as the foreign key. All of the real-world information requirements are expressed as a complete set of two-dimensional tables with columns and rows.

What is the implication of stressing the fundamental design concept for the relational data model? When you complete the logical design and produce the logical schema for your relational database system, the logical schema essentially consists of table descriptions. The relational schema contains a list of all the tables. For each table, you include the column names, and for each column you provide the descriptions and data types. You indicate the primary key columns for each table. You add the foreign key columns to establish each relationship among the tables. Then you

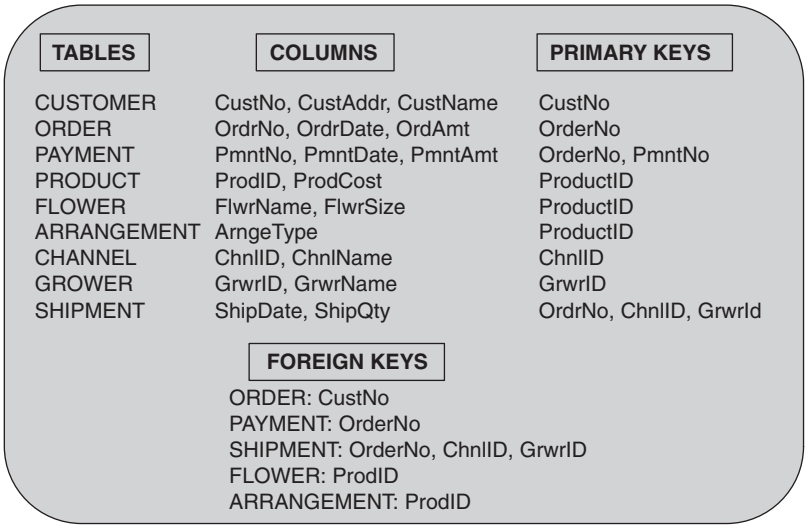

Figure 11-9 List of logical design components for florist business.

note the relational constraints about the applicable columns in each table. In short, the logical schema is nothing but descriptions of the relational tables.

Logical Design Components

In the previous section, we discussed a florist business and covered the representation of individual components. Let us pull the separate representations of individual components together and list the complete set of logical design components. Figure 11-9 displays such a list.

Note each component. Again, note how every component is part of the underlying design concept—the two-dimensional table or relation.

Logical Schema

Creating and defining the logical schema essentially completes the logical design phase. The logical schema for the relational data model contains the representation and descriptions of the relational tables. Let us take the complete relational data model for the florist business and define the logical schema. Each table in the relational data model must be defined. Every column in each table must be expressed in the logical schema.

Figure 11-10 displays the logical schema for the relational data model representing the information requirements for the florist business. We have used standard relational notation to express the schema. Data definition languages are used to define the schema to the data dictionary.

Carefully note the definition for each table in the logical schema. Also, for each table, observe how the columns are defined. Make a note of the definition of relational constraints in the schema.

CUSTOMER (CustNo, CustAddr, CustName)

ORDER (OrdrNo, OrdrDate, OrdrAmt, CustNo)
Foreign Key: **CustNo** REFERENCES **CUSTOMER** ON DELETE CASCADE

PAYMENT (OrdrNo, PmntNo, PmntDate, PmntAmt)
Foreign Key: **OrdrNo** REFERENCES **ORDER**

PRODUCT (ProdID, ProdCost)

FLOWER (ProdID, ProdCost, FlwrName, FlwrSize)
Foreign Key: **ProdID** REFERENCES **PRODUCT**

ARRANGEMENT (ProdID, ProdCost, ArngeType)
Foreign Key: **ProdID** REFERENCES **PRODUCT**

CHANNEL (ChnlID, ChnlName)

GROWER (GrwrID, GrwrName)

SHIPMENT (OrdrNo, ChnlID, GrwrID, ShipDate, ShipQty)
Foreign Key: **OrdrNo** REFERENCES **ORDER**
ChnlID REFERENCES **CHANNEL**
GrwrID REFERENCES **GROWER**

Figure 11-10 Relational schema for florist business.

Special Considerations

Before declaring that the logical design phase is complete, you need to examine each statement in the logical schema carefully. This is the basis for defining the physical files, records, fields, and restrictions. The logical schema, therefore, must truly define and represent the real-world information requirements. Here is a list of special considerations in verifying the logical schema for the relational model:

- Ensure that the logical schema defines each and every table in the relational model.
- Define each table and its columns with meaningful names. Remember, these names will be used for accessing tables and the columns.
- For each column, define appropriate data types and data lengths.
- If there are any edits to be specified for specific columns, do so clearly for these columns.
- If any column may not contain null values, express this clearly.
- Clearly indicate the columns that make up the primary key for each table. Primary keys cannot contain null values.
- Show the foreign key columns distinctly. Indicate the table to which a foreign key establishes a relationship.
- Ensure that all intersection tables are clearly defined with proper primary key and foreign key columns.
- Indicate the columns for which duplicate values are not allowed in the table.

- Express the referential integrity constraints for each relationship in terms of how to handle additions and deletion of related rows.

DOCUMENTATION OF LOGICAL DESIGN

Let us go back and review the role of documentation in the database development life cycle. In the requirements definition phase, analysts skilled in interviewing techniques and analysis of business operations complete the study of information requirements. After the requirements definition phase comes the design phase. The first part of the design phase consists of the logical design activity. At this stage, data modelers and database designers enter the scene in the database development life cycle. These professionals possess different sets of skills and training. However, they have to pick up from where the analysts leave off and build on the outputs from the requirements definition phase. What enables the data modelers and database designers to embark on the design phase? The answer is the documentation produced during the requirements definition phase. Whenever you switch gears and move into the next phase in the database development life cycle, you need adequate documentation to proceed further.

What happens at the end of the logical design activity? Data modelers and designers complete the semantic data model. If your target database system is relational, the semantic data model is transformed into a relational data model. If the designers adopt the traditional approach, they design the relational tables, normalize the structures, and produce the relational data model. At this point, the logical design step is deemed to be complete. What happens next? The database administrators take up the project and proceed to the next step, namely, physical design. How can they perform their physical design tasks? Again the answer is the documentation produced during the logical design step. Proper documentation enables the database administrators to correctly perform the physical design. As you complete the logical design, ensure that adequate documentation exists and that the outputs are clearly identified.

Logical Design Outputs

What are the major outputs from the logical design step? These outputs are generated from the tasks in the logical design activity. The first major set of tasks center on the creation of the semantic data model. Next, you have the mapping of the components of the semantic data model to those of the relational data model. The outputs from the logical design step must, therefore, include the semantic data model and the transformation of the semantic data model into the relational data model.

Here is a summary of the logical design outputs:

- Partial semantic data models represented in standard format
- Methodology for consolidating partial semantic data models
- Consolidated semantic data model diagram with detailed descriptions

- Mapping of each individual component of the consolidated semantic data model to one or more components of the relational data model
- Completion of the relational data model schema, expressed with standard conventions and notations
- List of any special considerations

Usage of Outputs

The outputs from the logical design step serve both users and database developers. This point in the database development life cycle is a good time to review the semantic data model diagram with the key users and get their further confirmation. In the requirements definition document, the users get the initial knowledge of how real-world information requirements are reviewed and captured. But the semantic data model crystallizes the representation of real-world information requirements in a clearer manner. You can walk the key users through the semantic model diagram, explain how the model is a true replica of real-world information, and relate the components of the model to the business processes in which the users are engaged.

When the data modelers and designers review the consolidated semantic model diagram, they can ensure that nothing is missed from the requirements definition. The completeness of the model is the confirmation of one major step in the database development life cycle.

Database administration function is more concerned with the completeness of the relational schema. This forms the basis for the physical design. In addition, database administrators also need information for defining the data views to the database management system. As you recall, each data view is the portion of the database in which a particular user group is interested. The data views form the external schemas. All the data views combined together make up the complete logical schema.

Use of CASE Tools

CASE (Computer-Aided Software Engineering) tools offer major benefits during the logical design step in the database development life cycle. You can use drag-and-drop techniques to create semantic data models. The ease of use of most of the data modeling tools provides enormous advantages and saves considerable time for data modelers in completing their tasks. The software tools generate the outputs in the form of E-R diagrams, supporting the standard methods for data modeling.

In addition to providing facilities for creating semantic data models, most of the CASE tools contain features known as forward engineering and backward engineering.

Forward Engineering. As soon as a generic semantic data model is created with the CASE tool, you can use this feature to transform the logical data model into a physical data model for the target database system. The database administrator can use this powerful feature to take the physical schema generated by the forward engineering feature and define the database in the data dictionary of the target DBMS.

Backward Engineering. This feature is the converse of forward engineering. Assume that your new database system includes data already part of some earlier databases in the organization. The database designers and analysts need to get details of the data structure in the older databases. They can get the physical schema of those databases and transform back into generic logical data model. The backward engineering feature of CASE tools accomplishes this.

Let us summarize the major benefits of CASE tools:

- Provide data modelers with toolbox capabilities to create the semantic data model with utmost ease.
- Provide adequate documentation for use by database analysts and developers to communicate among themselves and with users.
- Give a simple, clear picture of the information requirements in the form of a semantic data model diagram.
- Provide forward engineering feature, greatly reducing the effort of the database administrator in defining the physical schema.

Documentation Outline

When a logical design document is issued, you arrive at the conclusion of the logical design step in the database development life cycle. Therefore, it is important that the results of all the steps leading up to the finish of the logical design process are reviewed and confirmed with key users. The issuance of the logical design document marks the beginning of the physical design and leads into the concluding phases of implementation and ongoing maintenance.

Here is a suggested outline for the logical design document. Review the outline, amend it, and adapt it for your specific database project.

Introduction. State the overall scope and content of the logical design step. Include an executive summary.

Semantic data modeling technique. Describe the data modeling technique adopted—object-based or entity-relationship modeling or any other.

CASE tool features. If a CASE tool is used for the logical design, highlight the important features of the tool and its use in the logical design.

Partial semantic data models. List the partial data models. Describe why and how the real-world information requirements are separated out for partial data modeling.

Consolidated semantic data model. Include the consolidated data model diagram. Include descriptions of the various components.

Relational schema of logical model. Include the schema definition of the logical data model.

Data views. Derive and present the subschemas for all the data views of individual user groups.

Physical data model. If you use a CASE tool and are able to forward engineer, include the physical data model.

Special issues. List any special issues, considerations, and assumptions.

Confirmation of logical design. Describe the procedure used for having the logical data model reviewed with user groups. State the results of the reviews and any special requests for revisions.

CHAPTER SUMMARY

- The logical data structure represents how data are perceived; in a relational data model, logically, data are perceived as two-dimensional tables.
- The physical data structure refers to how data are stored in physical storage.
- The logical data design phase in DDLC defines the logical structure of data. This phase is a prerequisite for physical design.
- Users understand the logical structure of data better than physical structures. The logical design phase is necessary to produce the logical structure that can be used to communicate with users.
- The design effort is easier and more manageable when it is divided into logical design and physical design.
- The outputs of the logical design phase are used as inputs in the physical design phase.
- The first part of the logical design phase consists of creating a semantic data model truly representing real-world information requirements.
- In the second part of the logical design phase, the generic semantic data model is transformed into the desired logical data model. To implement a relational database system, the semantic data model is transformed into a relational data model.
- Entities, attributes, relationships, and rules and constraints are represented in the target relational data model.
- CASE tools offer major benefits during the logical design phase.
- The issuance of the logical design documentation marks the conclusion of the logical design phase and the commencement of the physical design phase.

REVIEW QUESTIONS

1. Distinguish between logical data structures and physical data structures.
2. Explain briefly the difference between logical and physical data structures in a relational database system.
3. Describe briefly how and where logical design fits in the database development life cycle (DDLC).
4. List any three reasons why the logical design phase is necessary.
5. "The output of the logical design phase forms the input to the physical design phase." Explain briefly.
6. Completing the logical design phase involves ensuring design completeness. List any four critical factors for consideration in the review for design completeness.

7. What are the two major activities in the logical design phase?
8. Instead of considering the creation of a semantic data model as part of the logical design, can this considered as part of the requirements definition phase? If you do so, what are some of the advantages and disadvantages?
9. How are CASE tools useful in the logical design phase? List any three major benefits.
10. List the major contents expected to be part of logical design documentation.

EXERCISES

1. Match the columns:

1. output of logical design	A. data as segments
2. logical schema	B. files, blocks, records
3. data views	C. consists of two-dimensional tables
4. consolidated semantic model	D. input to physical design
5. hierarchical data model	E. transformation of semantic data model
6. CASE tools	F. representation at conceptual level
7. logical design of relational DB	G. useful for communicating with users
8. second part of logical design	H. translate into subschemas
9. physical data structure	I. simplify data modeling
10. logical data model	J. represents all relevant information

2. You are the senior analyst in the database project building a relational database for a large bank. Write a memorandum to the CIO explaining the significance of the logical design phase and describe the major activities in that phase.
3. As the project manager for the database project for an insurance company, what special measures will you take to ensure the completeness of the design by end of the logical design phase?
4. Create a semantic data model for a small bakery and transform the semantic data model into a relational data model. Produce the logical schema.
5. Describe the forward and backward engineering features found in most CASE tools. As the lead database designer for a fast-food chain, describe how you propose to use these features of your CASE tool.