Database Design for Mere Mortals
A Hands-On Guide to Relational Database Design
By Michael J. Hernandez

Introduction

It is important to have a properly designed database so that accurate information can be provided to an organization. It is much easier to create an effective design initially so that necessary modifications to the database are kept at a minimum. Discovering problems after a database has been put into operation can be detrimental to a business, institution or organization.

There is a three-phase process in developing a database:

- *Logical design*: defining tables, fields, Primary and Foreign keys, establishing table relationships and levels of data integrity.
- *Implementation* of the logical design: using a DBMS to create tables and their relationships, using the tools to implement levels of data integrity.
- *Development* of end-user application.

This book focuses only on the logical design of a database.

Chapter 1: What is a Relational Database

There are two types of databases: operational and analytical.

Operational is used in everyday businesses, institutions and organizations. They are primarily used to store data that is collected, maintained and modified (dynamic data).

Analytical is used to track historical data (static data). Track trends, view statistical data.

Insertion anomalies and redundant data are problems associated with an early database model known as a hierarchical table (parent-child table). Network database (owner-member table) models were problematic as well. These two models led to the development of the relational database model.

Dr. E. F. Codd applied mathematical theories known as first order predicate logic and set theory to design relational databases. These theories are the foundation for the Relational Database Model (RDM). They are important because they makes the RDM predictable and reliable. It is not necessary to fully understand these theories to develop a sound database design.

In a RDM, data are stored in a *relation* or *table* (those terms may be used interchangeably.) Each table contains rows or records, (also called *tuples*), and columns which represent attributes or *fields*. Each record or row is represented by a unique field known as the Primary key.

The categories of relationships in a RDM are *one-to-one*, *one-to-many*, and *many-to-many*. A many-to-many relationship must be broken down into numerous one-to-many relationships. If a pair of tables share a relationship, data can be retrieved based on matching values of a shared field between the tables.

Data is retrieved by specifying fields and tables using a standard query language known as Structured Query Language (SQL). Most DBMSs (Database Managements Systems) use SQL to build, modify, maintain and manipulate databases. Thorough knowledge of SQL isn't always necessary since most DMBSs use a graphical interface to generate SQL statements and retrieve data. It is good, however, to have basic knowledge of SQL.

Large volumes of centrally located shared data have come about in recent history, creating the need for client/server software. Data security and integrity can be implemented through the database server.

Chapter 2: Design Objectives

A logical design of the database is highly important. Wizards that help you create tables are of no real use if the database itself is designed improperly. Accuracy, integrity and consistency of data will be dependent upon a good design. Many problems can arise if there are design flaws, such as inaccurate retrieval of information.

Objectives of Good Design:

- Supports required and ad hoc information retrieval
- Contains efficiently constructed tables
- Imposes data integrity at the field, table and relationship level
- Data must provide accurate and valid information that is meaningful to the organization
- The database structure should be easily modified for possible future growth

Advantages of Good Design:

- Easy to modify and maintain the structure
- Easy to modify data
- Easy retrieval of information
- Easy to develop and build user applications

There are three traditional design methods:

- *A requirements analysis phase* involves examining the business being modeled, interviewing users and management to assess the current system and to analyze future needs and determining information requirements for the business as a whole

- *A data modeling phase* involves modeling the database structure itself by using a method such as entity relationship diagramming (ER diagramming). This provides a means of visually representing various aspects of the database structure, such as the tables, table relationships and relationship characteristics.

- *The Normalization phase* is the process of decomposing large tables into smaller tables in order to eliminate redundant data, duplicate data and avoid problems with inserting, modifying or deleting data. Table structures are tested against normal forms, which are a specific set of rules that can be used to test a table structure to be sure it is sound and free of problems. These normal forms are: First through Fifth Normal Forms, Boyce-Codd Normal Form and Domain/Key Normal Form.

Chapter 3: Terminology

Four categories of terms are described in this chapter: value-related, structure-related, relationship-related, and integrity-related.

Value Related Terms

- *Data* are the values that are stored in the database. They are static in the sense that they remain in the same state until they are modified.

- *Information* is data that has been processed in a way that makes it meaningful. It can be shown as the result of a query, either displayed on-screen, or printed on a report.

- *Null* is a value that is either missing or unknown. A null value represents neither zero nor blank, as they are actual values and can be meaningful in certain circumstances. A drawback to null values is that they cannot be evaluated by mathetatical expressions.

Structure Related Terms

- A *table* is the main structure in a relational database. It is composed of fields and records, the order of which is completely unimportant. It always represents a single, specific subject, which can be an object or an event.

- A *field* (also known as an *attribute*) is the smallest structure in a relational database. It represents a characteristic of the subject of the table. A field may be multipart, multi-valued or the result of a calculation (concatenated).

- A *record* (also known as a *tuple*) is a structure within a table that represents a unique instance of the subject of the table.

- A *View* is a virtual table that is composed of the fields of one or more tables. It draws its data from the tables on which it is based. They are commonly implemented as saved queries.

STUDENTS TABLE

Student ID	Student First Name	Student Last Name	Student Phone
60003	Zachary	Erlich	553-0223
60928	Susan	McLain	790-3992
60765	Joe	Rosales	551-4993

INSTRUMENTS TABLE

Instrument ID	Student ID	Instrument Type	Instrument Desc
11128	60003	Guitar	Stratocaster
11185	60928	Drums	Ludwig Pro
11147	60765	Guitar	Les Paul

STUDENT INSTRUMENTS (VIEW)

Student ID	Student Last Name	Instrument Desc
60003	Erlich	Stratocaster
60928	McLain	Ludwig Pro
60765	Rosales	Les Paul

In this example, the STUDENT INSTRUMENTS View is composed of fields taken from both the STUDENTS table and the INSTRUMENTS table. Data in the View is drawn from both tables simultaneously, based on matching values between the Student ID field in the STUDENTS table and the Student ID field in the INSTRUMENTS table.

Keys are special fields that serve a specific purpose within a table. A Primary key is a field that uniquely identifies a record within a table. A Foreign key is the field that is used to establish a relationship between a pair of tables.

In the following example, Agent ID is the Primary key of AGENTS because it uniquely identifies each record in that table. Similarly, Client ID is the Primary key of CLIENTS because it also uniquely identifies each of the table's records. Agent ID in the CLIENTS table is a Foreign key because it is used to establish a relationship between the CLIENTS and the AGENTS table.

AGENTS

Agent ID	Agent First Name	Agent Last Name	Date of Hire	Agent Home Phone
100	Mike	Hernandez	05/16/95	553-3992
101	Greg	Pierce	10/15/95	790-3992
102	Katherine	Ehrlich	03/01/96	551-4993

CLIENTS

Client ID	Agent ID	Client First Name	Client Last Name	Client Home Phone
9001	100	Stewart	Jameson	553-3992
9002	101	Shannon	McLain	790-3992
9003	102	Estelle	Parker	551-4993

Relationship-related Terms

Relationships establish a connection between a pair of tables. This relationship exists when a pair of tables is connected by a Primary and Foreign key.

Types of Relationships

One-to-One: Exists between a pair of tables if a single record in the first table is related to one and only one record in the second table.

EMPLOYEES

Employee ID	Emp First Name	Emp Last Name	Home Phone
100	Zachary	Erlich	553-3992
101	Susan	McLain	790-3992
102	Joe	Rosales	551-4993

COMPENSATION

Employee ID	Hourly Rate	Commission Rate
100	25.00	50%
101	19.75	3.5%
102	22.50	5.0%

One-to-Many: Exists between a pair of tables if a single record in the first table is related to one or more records in the second table, but a single record in the second table can be related to only one record in the first table. This is the most common type of relationship.

STUDENTS

Student ID	Student First Name	Student Last Name	Student Phone
60003	Zachary	Erlich	553-3992
60928	Susan	McLain	790-3992
60765	Joe	Rosales	551-4993

INSTRUMENTS

Instrument ID	Student ID	Instrument Type	Instrument Desc
11128	60003	Guitar	Stratocaster
11185	60765	Drums	Ludwig Pro
11147	60765	Guitar	Les Paul

Many-to-Many: Exists between a pair of tables if a single record in the first table can be related to one or more records in the second table, and a single record in the second table can be related to one or more records in the first table. Establishing a direct connection between these two tables is difficult because it will produce a large amount of redundant data in one of the tables.

STUDENTS

Student ID	Student First Na	Student Last Na	Student Phone
60003	Zachary	Erlich	553-3992
60928	Susan	McLain	790-3992
60765	Joe	Rosales	551-4993

CLASSES

Class ID	Class Name	Instructor ID
900001	Intro to Political Science	220087
900002	Adv. Music Theory	220039
900003	American History	220148

Types of Participation

There are two types of participation that a table can have within a relationship: *mandatory* and *optional*. If records in Table A must exist before any records can be entered into Table B, then Table A's participation within the relationship is mandatory. If not, it is considered optional.

Each table in a relationship has a *degree of participation*, which is the minimum and maximum number of records in one table that can be related to a single record in the other table. Consider Agents and Clients tables. If we say that an agent should have at least one client but no more than eight, then the degree of participation for the Clients table is 1,8.

Integrity-related Terms

A *field specification* (also known as a domain) represents all the elements of a field. Each field specification has three types of elements: general, physical and logical.

A field's *general elements* include such items as field name, description and source table. *Physical elements* include items such as data type, length, and display format. *Logical elements* describe the values stored in a field, such as required value, range of values and default values.

Data integrity refers to the validity, consistency and accuracy of the data in a database. The four types of data integrity are:

- *Table-level integrity* ensures that the field that identifies each record within the table is unique and is never missing its value.

- *Field-level integrity* ensures that the structure of every field is sound, that the values in each field are valid, consistent and accurate.

- *Relationship-level integrity* ensures that the relationship between a pair of tables is sound and that there is synchronization between the two tables whenever data is entered, updated or deleted.

- *Business rules* impose restrictions or limitations on certain aspects of a database based on the ways an organization perceives and uses its data.

Chapter 4: Conceptual Overview

The first phase in the database design process is to define a *mission statement* and *mission objective*. This establishes the purpose of the database and provides a focus for the developer.

The second phase involves analyzing the current database, if one exists. It will typically be a legacy (one that has been in use for several years) or paper-based (forms, index cards, folders, etc.) database.

It is very important to conduct interviews with users and management to identify how they interact with the database on a daily basis. With this information, you then compile a list of fields. This list will be refined as the design is developed.

The third phase is creating the data structures: tables, fields, establishing keys and defining field specifications.

Tables are the first structures you define in the database. Once subjects are identified, they're established as tables, then fields are associated with the appropriate tables. Tables should be review to be sure they represent only one subject and that no fields are duplicated.

Next, fields are reviewed to be sure there are no multipart or multivalued fields. If so, you modify those fields so each fields stores a single value. Then a Primary key is established, making sure it uniquely identifies each record within the table.

Finally, field specifications are established. Interviews should be conducted with users to help identify any specific field characteristics that may be important to them.

In the fourth phase, table relationships are established. Interviews are conducted with users and management to identify relationships, relationship characteristics and establish relationship-level integrity.

Once relationships have been identified, it is necessary to establish the logical connection for each relationship. Depending upon the type of relationship, you would use either a Primary key or a "linking" or "composite" table to make the connection between a pair of tables based upon the type of relationship you want to establish.

The fifth phase of the database design process is to define the business rules. How an organization views and uses its data will determine limitations and requirements that must be built into the database. Again, this information is gained through interviews with users and management.

Next, *validation tables* are defined. For example, if certain fields are found to have a finite range of values due to the way they are used by an organization, validation tables are used to ensure the consistency and validity of the values stored in those fields.

Determining and establishing *views* is the sixth phase of the database design process. Interviewing users and management will help identify the different ways data is viewed. One group may view data from a different perspective than another group. Another group may only need to view one specific field from a certain table.

The last phase is reviewing the final database structure for data integrity. First, each table is reviewed to ensure that it meets proper design criteria. Then field specifications are reviewed and checked. Then, you test the validity of each relationship. Finally, business rules are reviewed and confirmed.

Chapter 5: Starting the Process

To begin the design process, you must identify the purpose of the database as well as a list of tasks that can be performed against the data.

Conducting interviews provides valuable information that affects the design of the database structure. Having a list of prepared questions is highly recommended. It's important to ask open-ended questions. This gives the participant an opportunity to provide complete and objective answers to questions.

The following are suggestions for interview guidelines:

- Set a limit of six people or less for each interview.
- Conduct separate interviews for users and managers.
- If several groups are interviewed, designate a group leader for each group.
- Prior to the interview, inform the participants of what will be discussed and how the interview will be conducted.

- Make sure everyone understands you appreciate their participation and that their responses are valuable to the overall design.
- Conduct interview in well-lit room, separated from distracting noise, large table and comfortable chairs and have coffee and munchies on hand.
- If you're not good at taking notes, assign the task to a dependable transcriber or get the group's permission to use a tape recorder.
- Give everyone your equal and undivided attention.
- Make sure everyone understands that you're the official arbitrator.
- Keep the pace of the interview moving.
- Always maintain control of the interview.

Defining the Mission Statement

A good mission statement is succinct and to the point. It should be very general and should not describe specific tasks. Interviewing management and staff will bring an overall understanding of the organization and general comprehension of why the database is necessary.

Defining the Mission Objectives

Mission objectives are statements that represent the general tasks performed against the data in the database. Each statement represents a single task and should not contain unnecessary detail. Mission objectives are used to help define table structures, field specifications, relationship characteristics and Views. Information used to define the mission objectives is gathered through interviews with users and management. General tasks are determined by asking open-ended questions. The interviews should be very general in nature to get an overall idea of the general tasks the database should support.

Chapter 6: Analyzing the Current Database

The database currently in use can provide a great resource for developing the new database. Many features can remain useful, whereas others can and should be discarded. An analysis is conducted reviewing the various ways data is collected and presented, as well as through interviews with users and management. A preliminary field list is defined, as well as a list of tables that should be included in the initial database structure.

Data that is literally collected, stored and maintained on paper is known as a *paper-based database.* Some common formats include index cards, hand-written reports and various types of preprinted forms. Typically these types of databases contain inconsistent data, erroneous data, duplicate data, redundant data, incomplete entries and data that should have been purged from the database long ago. The only reason to analyze this type of database is to identify various items that will be incorporated into the new database.

A database that has been in use for five years or more is considered to be a *legacy database*. The term "legacy" may also mean that the individual who originally created the database is working elsewhere and the database has become his or her legacy to the organization. Many of these legacy databases are improperly structured or inefficiently designed. Many times they are based on hierarchical or network database models and store duplicate fields and redundant data.

To conduct the analysis of the current database, one must first review the ways in which data is collected. Begin by reviewing all paper-based items and gather a single sample of each type. Next find sample screen shots in the database programs that best represent how the various programs are used.

The second step in the analysis process is to review any methods used to present information, such as a report. A report is a way to present data that is meaningful to those viewing it. Gather samples of all reports.

The final step is to review any on-screen presentations that use the data in the database. Make screen shots of the slides that are used in the presentations.

Next, conducting interviews with users and management is useful in determining how the organization uses its data. For example,

- They provide details about the samples you assembled in reviewing how data is collected and how information is presented.
- They provide information on the way the organization uses its data.
- They are instrumental in defining preliminary field and table structures.
- They help to define future information requirements.

It's better to speak to the users first because they have the clearest picture of the details connected with the day-to-day operations of the organization. Use open-ended question to focus on specific subjects, use closed questions to obtain specific details on a certain subject. You can identify subjects by looking for nouns within the sentences that make up the responses. Subjects are always represented by nouns and identify an object or an event. You can then use these subjects to come up with further questions during the interview process. The purpose is to gain as much detailed information as possible about the subjects you've identified.

You'll also want to identify nouns that represent characteristics of the subjects. These will ultimately become fields in the database. This technique is known as the *characteristic identification technique.* It's important to use a separate sheet of paper for listing the characteristics.

The first part of the interview process involves conducting user interviews, which will focus on:

- The types of data users are currently using;
- How users are currently using their data;

- The data-collection samples, report samples, and on-screen presentation samples; and
- The types of information users need in conjunction with their daily work.

The next part of the discussion should focus on any additional information that is not being supplied to them currently. Once this is identified, you'll define new data structures to support this extra information.

The last part of the interview process concerns future information that may be required by a growing organization. Once such information is identified, you can be sure that the data structures needed to support that information are defined in the database.

The second part of the interview process involves conducting interviews with management, which will focus on:

- The types of information managers currently receive;
- The types of additional information they need to receive;
- The types of information they foresee themselves needing; and
- Their perception of the business's overall information requirements.

To begin, review current information requirements. Identify information that management routinely receives and determine whether they currently receive any reports that are not represented in your group of report samples. If so, obtain a sample of each "new" report.

The next subject concerns management's needs for additional information. If there is any information that is missing from the reports they currently receive, that information must be identified.

Next, review future information requirements, and finally, review overall information requirements. If there is any data that the organization needs to maintain, it will need to be accounted for in the database structure.

Compiling a Complete List of Fields

Now that an analysis of the current database is complete and the interviews with users and management have been conducted, a *preliminary field list* can be created. This list represents the fundamental data requirements of the organization and constitutes the core set of fields that will be defined in the database.

First, review and refine the list of characteristics you have complied. Then determine whether there are any characteristics in the data collection samples, report samples, and on-screen presentation samples that need to be added to the primary field list. Be sure that each item on your list represents a characteristic (field) and not a subject (table). And, finally, remove any fields that are *calculated* and place them on a separate list. A *calculated field* is one that stores the result of a mathematical calculation as its value. Be sure to review both lists with users and management.

Chapter 7: Establishing Table Structures

Defining the Preliminary Table List

During this phase, a *preliminary table list* will be defined. Three procedures are used to develop this list. The first involves using the *preliminary field list*, the second involves using the *list of subjects* gathered during the interview process, and the third involves using the *mission objectives* that were defined at the beginning of the database design process.

The process of defining the tables begins with a review of the preliminary field list to determine what subjects are implied by these items. If you can infer a subject from the listed fields, enter that subject on the new *preliminary table list*. Next, create a second version of the preliminary table list by merging the list you created during the interviews with users and management with the first version of the preliminary table list. Remove any duplicate items, resolving items that represent the same subject and combine the remaining items together into one master list.

Finally, use the mission objectives to determine whether any subjects were overlooked during the previous procedures. This is a final opportunity to add tables to the preliminary table list.

Defining the Final Table List

It's time to transform the preliminary table list into the final table list. To do this, you'll need to add two elements: *table type* and *table description*.

There are four table types:

- *Data*: This type of table stores data used to supply information.
- *Linking*: This table is used to establish a link between two tables in a many-to-many relationship.
- *Subset*: This table contains supplemental fields that are related to a particular data table and further describe the subject of that table in a very specific manner.
- *Validation*: This type of table is used to implement data integrity.

Table descriptions are used to provide a clear definition of the subject represented by the table. One final task in developing the final table list is to refine the table names.

Refining the Table Names

There are certain guidelines that should be used when creating a table name. These guidelines ensure that a name is clear and unambiguous, that it is descriptive and meaningful, and that each table is named in a consistent manner.

Guidelines for creating a table name:

- Create a unique, descriptive name that is meaningful to the entire organization.
- Create a table name that accurately, clearly and unambiguously identifies the subject of the table.
- Use the minimum number of words necessary to convey the subject of the table.
- Do not use words that convey physical characteristics.
- Do not use acronyms and abbreviations.
- Do not use proper names and other words that will unduly restrict the data that can be entered into the table.
- Do not use names that implicitly or explicitly identify more than one subject.
- Use the plural form of the name (in contrast, field names are always singular).

Indicating the Tables Types

You'll indicate each table's type on the final table list (i.e., data, linking, subset and validation). Each item on the list will be a *data* table because it represents a single subject that is important to the organization. *Linking* and *validation* tables will be missing because you have not yet defined any relationships or data integrity. These issues will be addressed later in the design process. *Subset* tables are missing as well because they are defined after fields have been assigned to data tables.

Composing the Table Descriptions

Table descriptions are important for understanding why each table exists and why the organization is concerned with collecting the data for each table. It must explicitly define the table and state its importance. If you are unable to explain the importance of a table, you'll need to further investigate when and how the table was identified and whether it really is necessary.

Guidelines for Composing a Table Description:

- Include a definition statement that accurately identifies the table.
- Include a statement that explains why this table is important to the organization.
- Compose a description that is clear and succinct.
- Do not include implementation-specific information in your table description.
- Do not make the table description for one table dependent on the table description of another table.
- Do not use examples in a table description.

Associating Fields with Each Table

The next stage of the database design process is to assign fields to each table on the final table list. These fields are taken from the preliminary field list.

To assign fields to a table, determine which fields best represent characteristics of the table's subject and assign them to that table. Repeat this procedure for every table on the final table list.

To begin this process, write the names of each table across the top of the paper. Repeat this procedure, using as many sheets as you need to account for every table on the list. Assign fields from the preliminary field list to each table. Start with the first table and determine which fields best describe its subject. List these fields under the table name. After assigning all fields you believe to be appropriate for the table, move on to the next table and repeat the process. Continue in this manner until you've assigned fields to all of the tables.

Refining the Fields

This step is preparation for the process of refining the table structures, which entails establishing that the appropriate fields have been assigned to the table and that the table is structurally sound.

Guidelines for Creating Field Names

- Create a unique, descriptive name that is meaningful to the entire organization.
- Create a name that accurately, clearly and unambiguously identifies the characteristic represented by the field.
- Use the minimum number of words necessary to convey the meaning of the characteristic the field represents.
- Do not use acronyms, and be discriminating in the use of abbreviations.
- Do not use words that could confuse the meaning of the field name.
- Do not use names that implicitly or explicitly identify more than one characteristic.
- Use the singular form of the name.

Using the Ideal Field to Resolve Anomalies

Unless you know what to look for, it's hard to determine whether any of the fields in a table is going to cause problems. The best way to identify potentially problematic fields is to determine whether they are in accordance with the Elements of the Ideal Field:

- It represents a characteristic of the subject of the table.
- It contains only a single value.
- It cannot be deconstructed into smaller components.
- It does not contain a calculated or concatenated value.
- It is unique within the entire database structure.
- It retains all of its characteristics if it appears in more than one table.

Resolving Multipart Fields

Multipart Fields are difficult to work with because they contain more than one item of data. It's hard to retrieve information from a multipart field and it's hard to sort or group the records in the table by the field's value. To resolve this, you need only identify the separate items making up the field and treat each one as an individual field.

Resolving Multivalued Fields

Multivalued fields bring about the same set of problems as do multipart fields. Unlike a multipart field, whose value represents two or more separate items, a multivalued field represents two or more occurrences of the same value. This poses a problem if you try to enter more occurrences of the value than the field will allow.

To resolve a multivalued field, first remove the field from the table and use it as the basis for a new table. Next use a field (or set of fields) from the original table to link the original table with the new table. Assign an appropriate name, type and description to the new table and add the table to the final table list.

Refining the Table Structures

The objective in this phase of the design process is to make sure that the appropriate fields have been assigned to each table and that each table's structure is properly defined.

Redundant Data and Duplicate Fields

Redundant data is a value that is repeated in a field as a result of the fields' use as a link between two tables, or is the result of some field or table anomaly. In the first instance, the redundant data is appropriate; by definition, fields used to link tables together contain redundant data. In the second instance, the redundant data is entirely unacceptable because it poses problems with data consistency and data integrity.

Duplicate fields are fields that appear in two or more tables for any of the following reasons: they are used to link a set of tables together, they indicate multiple occurrences of a particular type of value, or they are there as a result of a perceived need for supplemental information. The only instance in which they are necessary is in the case of linking two tables together.

Elements of the Ideal Table:

- It represents a single subject, which can be an object or event.
- It has a Primary key.
- It does not contain multipart fields.
- It does not contain multivalued fields.

- It does not contain calculated fields.
- It does not contain unnecessary duplicate fields.
- It contains only an absolute minimum amount of redundant data.

Chapter 8

Why Keys are Important

- They ensure that each record in a table can be properly identified.
- They help establish and enforce various types of integrity.
- They are used to establish table relationships.

When the keys are appropriate, you guarantee that the table structures are sound, redundant data within each table is minimal, and the relationships between tables are solid.

Establishing Keys for Each Table

The function of a key within a table is determined by the key type. There are four types of keys: Candidate, Primary, Foreign and Non-keys.

Candidate Keys. It is from the pool of available Candidate keys that a Primary key is chosen. It is a field or set of fields that uniquely identifies a single instance of the subject represented by the table.

Elements of a Candidate Key:

- It must uniquely identify each record in the table.
- It must contain unique values.
- It cannot be null.
- It cannot be a multipart field.
- It comprises a minimum number of fields necessary to define uniqueness.
- Its value is not optional in whole or in part.
- It must directly identify the value of each field in the table.
- Its value can only be modified in rare or extreme cases.

Artificial Candidate Keys. If none of the fields in a table qualifies as a Candidate key, you can use an Artificial Candidate key. You establish an Artificial Candidate key by creating a new field that conforms to the elements of a Candidate key and then adding that field to the table.

Primary Keys. The next step is to establish a Primary key. A Primary key must conform to the exact same elements as the Candidate key. It is the key that officially identifies the table throughout the database structure. It is used to enforce table-level integrity, helps establish relationships with other tables and accurately identifies and refers to a particular record within the table. Here are a couple of guidelines to use when selecting an appropriate Primary key:

- If you have a "simple" (single field) Candidate key and a Composite Candidate key, choose the "simple" Candidate key.
- Choose the Candidate key that uses a field that incorporates part of the table name within its name.

Choose the key that you believe is the most meaningful to everyone in the organization.

Elements of a Primary Key

- It must uniquely identify each record in the table.
- It must contain unique values.
- It cannot be null.
- It cannot be a multipart field.
- It should contain the minimum number of fields necessary to define uniqueness.
- It is not optional in whole or in part.
- It must directly identify the value of each field in the table.
- Its value can only be modified in rare or extreme cases.

Rules for Establishing a Primary Key

- Each table must have one and only one Primary key.
- Each Primary key within the database should be unique – no two tables should have the same Primary key unless one of them is a subset table.

Table-Level Integrity

- There are no duplicate records in a table.
- Every record in a table is identified by a Primary key value.
- Every Primary key value is unique.
- Primary key values are not null.

Reviewing the Initial Table Structures

Once the fundamental table definitions are complete, you'll need to conduct interviews with users and management to review the work you've done so far. During these interviews, you will:

- Ensure that the appropriate subjects are represented in the database.
- Make certain that the table names and table descriptions are suitable and meaningful to everyone.

- Make certain that the field names are suitable and meaningful to everyone.
- Verify that all the appropriate fields are assigned to each table.

Chapter 9: Field Specifications

Why Field Specifications are Important

- Field-level integrity is established and enforced as a result of defining Field Specifications.
- Defining Field Specifications for each field enhances overall data integrity.
- Defining Field Specifications compels you to acquire a complete understanding of the nature and purpose of the data in the database.
- Field Specifications are valuable when you implement the database in an RDBMS program.

Field-Level Integrity

A field has field-level integrity after a full set of Field Specifications has been defined for the field. Field Specifications help to ensure that:

- The identity and purpose of each field is clear, and that all of the tables in which it appears are properly identified.
- Field definitions are consistent throughout the database.
- The values of the field are consistent and valid.
- The types of modifications, comparisons and operations that can be applied to the values in the field are clearly identified.

Elements of the Ideal Field

- It represents a characteristic of the subject of the table.
- It contains only a single value.
- It cannot be broken down into smaller components.
- It does not contain a calculated or concatenated value.
- It is unique within the entire database structure.
- It retains all of its characteristics if it appears in more than one table.

Anatomy of a Field Specification

- *General Elements*: Field Name, Parent Table, Label, Shared By, Alias(es), Description;

- *Physical Elements*: Data Type, Character Support, Length, Decimal Places, Input Mask, Display Format;

- *Logical Elements*: Type of Key, Uniqueness, Required Value, Null Support, Edit Rule, Comparisons Allowed, Operations Allowed, Values Entered By, Default Value, Range of Values; and

- *Specification Information*: Specification Type, Based on Existing Specification,m Source Specification.

General Elements

Field Name. The field name is the unique identifier for the field itself. It is the set of absolute minimal words used to identify a particular field throughout the database.

Label. A label is an alternate name for the field that may be used to identify the field in a RDBMS program. For example, a label for a field named *Quantity on Hand* might be *Qty on Hand*. A label is typically a shorter form of the Field Name.

Parent Table. A field represents a characteristic of a particular table's subject. The table that represents this subject is referred to as the parent table of the field, and it is the only table in which the field will appear.

Shared By. This element is used to list the names of other tables that share this field. The only table names that should appear here are those that use this field as a connection to its parent table.

Alias(es). An alias is a name that a field assumes under very rare circumstances. One instance when a field would use an alias is when there must be two occurrences of the field in the same table.

Description. In the description you provide a complete interpretation of the field.

Guidelines for Composing a Field Description

- Use a statement that accurately identifies the field and clearly states it purpose
- Write a statement that is clear and succinct
- Refrain from restating or rephrasing the field name
- Avoid using technical jargon, acronyms, or abbreviations
- Do not include implementation-specific information
- Do not make this description statement dependent on the description of another field
- Do not use examples

Physical Elements

The Physical Elements category pertains to the structure of a field.

Data Type. In the Data Type setting you indicate the nature of the data that is stored in the field. The most common types of data are:

- Alphanumeric
- Numeric

- Date
- Time

Character Support. This element is used to indicate the characters that are permitted to be entered into the field.

- Letters;
- Numbers (0-9);
- Extended characters: Any character (other than letters and numbers) that can be entered from the keyboard; and
- Special characters: Any character that must be entered by some means other than the keyboard, such as through a special software program.

Length. The total number of characters that can be entered into a particular field is indicated by this element. The maximum number of characters you allow for a field will depend on the RDBMS program you use to implement the database.

Decimal Places. The number of digits to the *right* of the decimal point is indicated by this element.

Input Mask. This element is used to indicate the manner in which the data should be entered into the field. Controlling the way data is entered results in consistent entries within the field.

Display Format. This element allows you to indicate how the value of the field should be presented. For example, you may want to enter a date as "mm/dd/yy," but it is much more meaningful when it is displayed as "January 1, 1996."

Logical Elements

This category of elements pertains to the values of the field. These elements indicate whether the values should be unique, when they should be entered, whether they can be edited, the types of comparisons and operations that can be performed on the values, and the range of acceptable values that can be entered into the field.

Type of Key. In the Type of Key element you indicate the role of the values within the field, e.g., Primary keys, Non-keys, Foreign keys, etc.

Uniqueness. This element determines whether the value of a field should be unique. In the case of a Primary key, this element will be set to "Unique."

Required Value. Whether a user must enter a value into a given field is indicated by this element.

Null Support. This element indicates whether null values should be allowed in the field.

Edit Rule. This element indicates at what point in time a value must be entered into the field and whether that value can be modified.

Comparisons Allowed. The types of comparisons that can be made to a particular value of this field are indicated by the Comparisons Allowed element. There are six types of comparisons: equal to, not equal to, greater than, less than, greater than or equal to, and less than or equal to.

Operations Allowed. This element indicates the types of operations that can be performed on the values in the field. Four types of operations are allowed: addition, subtraction, multiplication and division.

Values Entered By. This element indicates how values are entered into a field. Either the user will enter each value manually of the values will be entered automatically by the database application.

Default Value. A *default value* is a value that used when an entry is required but not yet available, and when Nulls are not allowed. You should only use a default value if it is meaningful.

Range of Values. The Range of Values element lets you determine every possible value that can be entered into a field. It can be represented with a lower and upper limit (1000-9999) or with a specific list of values ("WA," OR," "ID," "MT").

Specification Information

The elements under this category pertain to the nature of the field specification as a whole.

Specification Type. A Field Specification falls into one of the following categories:

- Unique
- Generic
- Replica

Based on Existing Specifications. This element indicates whether any of the elements in this specification have drawn their settings from another field specification.

Source Specification. This element indicates the name of the generic Field Specification upon which the current specification is based.

Chapter 10: Table Relationships

A relationship is a crucial part of the database because

- It establishes a connection between a pair of tables that are logically related to each other in some form.
- It helps to further refine table structures and minimize redundant data.
- It is the mechanism that allows data from multiple tables to be drawn together simultaneously.
- Relationship-level integrity is established when a relationship is properly defined.

Type of Relationships

One-to-One Relationships. A pair of tables are defined as having a one-to-one relationship if a single record in the first table is related to one and *only* one record in the second table.

One-to-Many Relationships. A one-to-many relationship is defined as one in which a single record in the first table can be related to one or more records in the second table, *but* a single record in the second table can be related to only one record in the first table.

Many-to-Many Relationships. Many-to-many relationships exist between a pair of tables if a single record in the first table can be related to one or more records in the second table, and a single record in the second table can be related to one or more records in the first table.

Problems with Many-to-Many Relationships

Before you can use the data from tables involved in many-to-many relationships, several problems must be resolved:

- One of the tables involved in the relationship will contain a large amount of redundant data.
- Both tables will contain some amount of duplicate data because of the redundancies.
- It will be difficult to insert, update and delete data in the participating tables.

Identifying Existing Relationships

In the first step, you'll identify the relationships that currently exist between the tables. Only look for *direct* relationships. Tables that are indirectly related will be implicitly connected through a third connecting table.

Two types of questions that can be asked:

- *Associative*: Can a single record in (name of first table) be associated with one or more records in (name of second table)?
- *Contextual*: This type of question contrasts a single instance of the subject represented by the first table against multiple instances of the subject represented by the second table.

Establishing Each Relationship

The One-to-One Relationship. In this type of relationship, one of the tables is referred to as the "main" table and assumes a dominant role in the relationship; the other table is referred to as the "subordinate" table.

A one-to-one relationship is established by taking a copy of the Primary key from the main table and inserting it into the subordinate table, where it becomes the Foreign key.

The One-to-Many Relationship. The technique used to establish a one-to-many relationship is similar to the one used to establish a one-to-one relationship. In this case, you take a copy of the Primary key from the table on the "one" side of the relationship and insert it into the table on the "many" side, where it becomes a Foreign key.

The Many-to-Many Relationship. A many-to-many relationship is established using a *linking* table. You create the linking table by taking a copy of the Primary key from each table involved in the relationship and using those Primary keys to create the new linking table. Next, you give the linking table a meaningful name, one that represents the nature of the relationship between the two tables.

Establishing Relationship Characteristics

These characteristics indicate what will occur when a record is deleted, the type of participation each table bears within the relationship, and to what degree each table participates in the relationship.

Establishing a Deletion Rule for Each Relationship

This rule defines what will happen if a user wants to delete a record in the main table of a one-to-one relationship or in the "one" side of a one-to-many relationship. This helps to guard against "orphaned" records, which are records that exist in a subordinate table of a one-to-one relationship but have no related records in a main table, or records that exist in the "many" side of a one-to-many relationship that have no related records in the "one" side.

Two options are available for the deletion rule:

- *Restrict*: The requested record cannot be deleted if there are related records in the subordinate table of a one-to-one relationship or the "many" side of a one-

to-many relationship. Related records must be deleted before the requested record can be deleted.
- *Cascade*: The requested record will be deleted as well as related records in the subordinate table of a one-to-one relationship or the "many" side of a one-to-many relationship.

Identifying the Type of Participation for Each Table

A table's *type of participation* determines whether a record must exist in that table before a record can be entered into the other table. There are two types of participation:

- *Mandatory*: There must be at least one record in this table before you can enter any records into the other table.
- *Optional*: There is no requirement for any records to exist in this table before you can enter any records into the other table.

Identifying the Degree of Participation for Each Table

This is a matter of identifying the total number of records in one table that can be related to a single record in the other table. The degree of participation is symbolized by two numbers, separated by a comma, and enclosed in parentheses. In this instance (1,8) a single record in one table can be related to a minimum of one record in a maximum of eight records in the other table.

Relationship-Level Integrity

Relationship-level integrity is a result of properly establishing a table relationship and defining its characteristics in the proper manner. You'll need to:

- Make certain that the connection between two tables in a relationship is sound.
- Ensure your ability to insert new records into each table in a meaningful manner.
- Ensure your ability to delete an existing record without creating adverse affects.
- Establish a meaningful limit to the number of records that can be interrelated within the relationship.

Chapter 11: Business Rules

A *Business Rule* is a statement that imposes some form of constraint on elements within a field specification for a particular field or on characteristics of a relationship between a specific pair of tables. For example, a "ship date" cannot be prior to an "order date" for any given order.

Types of Business Rules

There are two major types of Business Rules: *database-oriented* and *application-oriented*.

Database-oriented Business Rules are those that impose constraints that can be established within the logical design of the database.

Application-oriented Business Rules are statements that impose constraints that cannot be established by modifying a Field Specification or relationship diagram; they must be established within the physical design of the database or within the design of a database application.

Categories of Business Rules

- *Field-Specific Business Rules*: Business Rules under this category impose constraints on the elements of a Field Specification for a particular field.
- *Relationship-Specific Business Rules*: Constraints imposed by relationship-specific business rules affect the characteristics of a relationship between a particular pair of tables.

Defining and Establishing Business Rules

These rules are based on the manner in which your organization perceives and uses its data, which will depend on the way the organization functions. You'll need to work with users and management to define and establish the appropriate Business Rules.

Field-Specific Business Rules

- Select a table.
- Review each field and determine whether you need to impose any constraints on it.
- Define the necessary Business Rules for the field.
- Establish the rules by modifying the appropriate Field Specifrication elements.
- Determine what actions test the rule.
- Record the rule on a Business Rule Specification sheet.

Establishing Relationship-Specific Business Rules

- Select a pair of tables that share a relationship.
- Review each relationship characteristic and determine whether a constraint is warranted by the way the organization functions.
- Define the necessary Business Rule.
- Establish the rule by modifying the relationship characteristics.
- Determine what actions will test the rule.
- Record the rule on the Business Rule Specification sheet.

Validation Tables

A *validation table* holds data specifically used to implement data integrity. There are instances where a rule affects the range of values for a particular field. It commonly limits the range of values to a specific set of valid entries. In many cases, the set of values is made up of a relatively fixed number of entries with values that will rarely change. These entries can be stored in a *validation table*.

Chapter 12: Views

A View is a *virtual table* that comprises the fields of one or more tables in the database (it may also contain fields from other views). It does not store or contain data – it gets its data from the tables upon which it is based. Only a View's structure is saved in the database. It is recreated each time it is accessed.

Views can be based on a single table, multiple tables, other Views, or a combination of tables and views.

There are three categories of Views:

- Data Views are used to examine and manipulate data from base tables.

- Aggregate Views are used to display information that is the result of aggregating a particular set of data in a specific manner. An aggregate View includes one or more calculated fields that incorporate the functions that aggregate the data, and one or more data fields to group the aggregated data.

- Validation Views are similar to validation tables in that they are used to implement data integrity.

Establishing Views

- Use calculated fields where appropriate
- Impose criteria to filter the data
- Use a View specification sheet to record the View

Chapter 13: Reviewing Data Integrity

Reviewing data integrity is simple if you sequentially review each component of the overall data integrity at the:

Table Level. Make certain the table conforms with all of the following points:

- There are no duplicate fields in the table.
- There are no calculated fields in the table.
- There are no multivalued fields in the table.

- There are no multipart fields in the table.
- There are no duplicate records in the table.
- Every record in the table is identified by a Primary key value.
- Each Primary key conforms to the Elements of a Primary key.

Field Level. Make sure field-level integrity has been properly established by making certain that:

- Each field conforms to the Elements of the Ideal Field.
- A set of Field Specifications has been defined for each field.

Relationship Level. Make certain that:

- The table relationship is properly established.
- The appropriate deletion rule has been established.
- The type of participation has been correctly identified.
- The proper degree of participation has been established.

Business-Rule Level. Make certain that:

- Each rule imposes a meaningful constraint.
- The proper category has been determined for the rule.
- Each rule is properly defined and established.
- The appropriate Field Specification elements or table relationship characteristics have been properly modified.
- The appropriate validation tables have been established.
- A Business Rule Specification sheet has been completed for each rule.

View Level. Make certain that:

- Each View contains the base tables necessary to provide the required information.
- Each view has been assigned the appropriate fields.
- Each calculated field provides pertinent information or enhances the manner in which the data is displayed.
- Each filter returns the appropriate set of records.
- Each View has a View diagram.
- Each View is accompanied by a View Specifications sheet.

Assembling the Database Documentation

Throughout the database design process, many lists, specification sheets, and diagrams were generated. They should now be assembled into a central repository. The design repository should consist of the:

- Final table list;
- Field Specifications sheet;

- Calculated field list;
- Table structure diagrams;
- Relationships diagrams;
- Business Rule Specifications sheets;
- View diagrams; and
- View Specifications sheets.

The design documentation is vital for three reasons:

- It provides a complete record of the structure of the database.
- It provides a complete set of specifications and instructions on how the database should be created during the implementation process.
- It can be used to determine the effects and consequences of any modifications, should it become necessary to modify the database structure during the implementation process.

Now that the integrity review is complete and you've assembled all of the documentation, the logical database design process is complete. You can be sure that you have a properly designed database and that its implementation will proceed smoothly.

Chapter 14: Bad Design – What Not To Do

Flat-File Design. This type of single-table design has many problems:

- Multipart fields
- Calculated fields
- Unnecessary duplicate fields
- No true Primary key
- The table represents more than one subject

Spreadsheet Design. A spreadsheet design is a good tool if used properly. It does not make a good relational database. Potential problems are:

- Duplicate fields
- Multipart fields
- Multivalued fields
- This type of database is difficult to use.

Chapter 15: Bending or Breaking the Rules

There are two specific circumstances under which it is permissible to break the rules of proper database design: when designing an *analytical database*, because this type of database is used to store and track historical data. Another common reason for breaking the rules, is when a multitable query is slow to process and the solution is to modify a table so that every field necessary for the query is included in the table. One

should take caution when doing so, however, because it introduces a number of new problems.

Other ways to improve performance:

- Enhance or upgrade the hardware;
- Fine-tune the operating system;
- Review the database structure;
- Review the implementation of the database; and
- Review the application program.

If you still find it is necessary to break the rules of proper database design, you must document each rule you break and each action you take. The items that should be recorded include:

- The reason you're by-passing the rules;
- The design principle you are violating;
- Identify the aspect of the database that you're modifying;
- The method by which you make the modifications;
- The anticipated effects on the database and the application program.

In conclusion, you should now possess the knowledge and skills necessary to design a sound and reliable relational database structure.