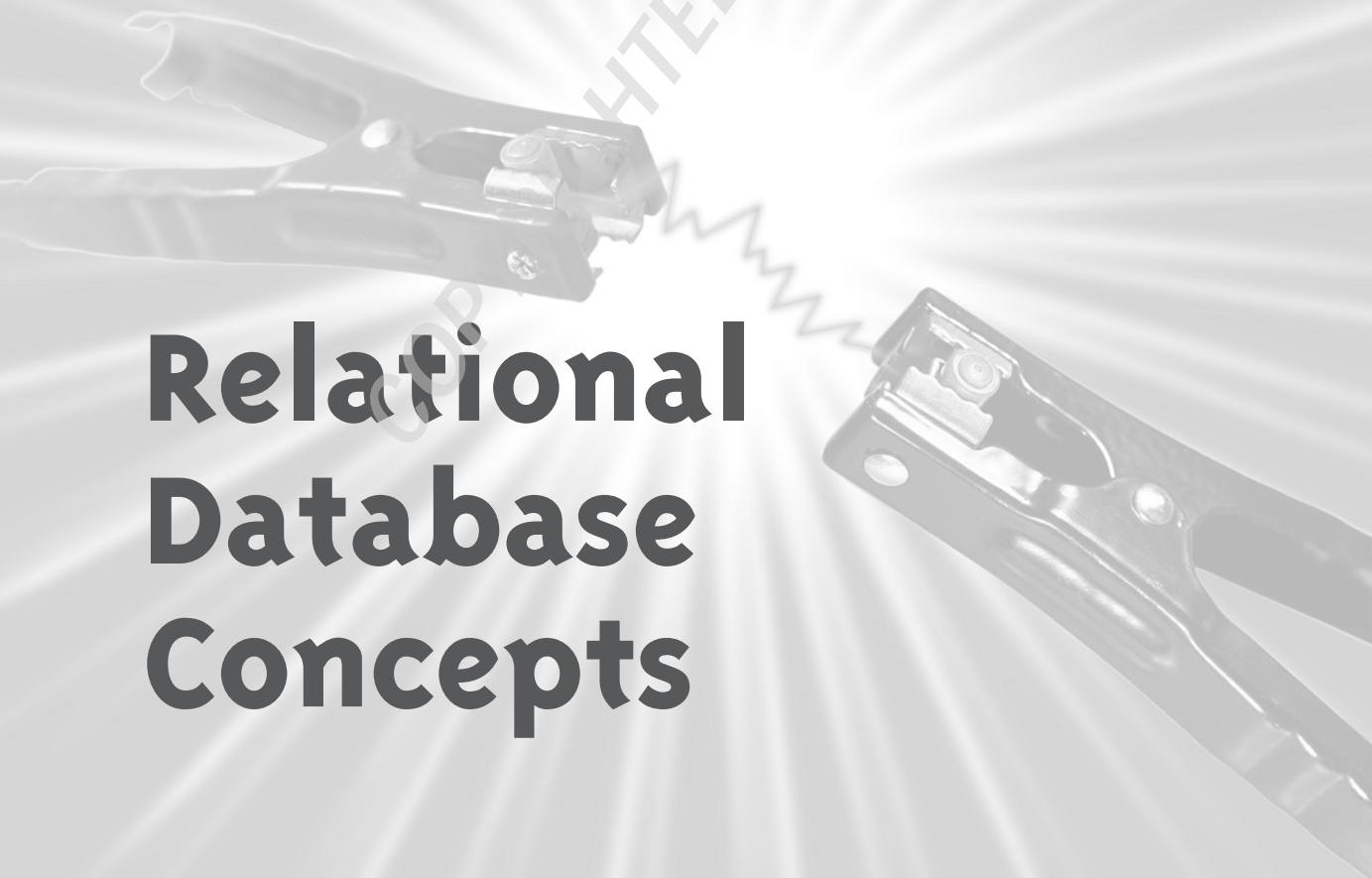



# Chapter

# I

## Relational Database Concepts





**E**very organization has data that needs to be collected, managed, and analyzed. A relational database fulfills these needs. Along with the powerful features of a relational database come requirements for developing and maintaining the database. Data analysts, database designers, and database administrators (DBAs) need to be able to translate the data in a database into useful information for both day-to-day operations and long-term planning.

Relational databases can be a bit intimidating at first, even if you're a specialist in some other informational technology area, such as networking, web development, or programming. This chapter will give you a good overview of current relational and object-relational database concepts. It begins by comparing a database with another tool that most everyone has used—a spreadsheet (also known as the “poor man’s” database). Then you’ll learn about the basic components of a relational database, the data modeling process, and object-relational database features.

In this chapter, you will learn about:



How spreadsheets compare with databases



Relational database concepts



Data modeling concepts



Object-relational database concepts

# Are Spreadsheets Like Databases?

Most people are familiar with some kind of spreadsheet, such as Microsoft Excel. Spreadsheets are easy and convenient to use, and they may be employed by an individual much like a database is used in the enterprise. Let's look at the features of spreadsheets to see how good of a database tool they actually are.

Similar to databases, spreadsheets are commonly used to store information in a tabular format. A spreadsheet can store data in rows and columns, it can link cells on one sheet to those on another sheet, and it can force data to be entered in a specific cell in a specific format. It's easy to calculate formulas from groups of cells on the spreadsheet, create charts, and work with data in other ways. But there are many ways in which a spreadsheet is not like a traditional database table:

Spreadsheet	Database
More than one datatype can be stored in a spreadsheet column.	Usually, only one datatype can be stored in a database table column.
Cells in a spreadsheet can be defined as a formula, making the contents variable depending on other cells.	Columns in a database table have a fixed value.
A spreadsheet has only the physical row number to make it unique, and no built-in way to enforce uniqueness of a given spreadsheet row.	Single rows of a database table are uniquely identified by a unique value (typically a primary key, as described later in this chapter).
Usually, only one user can have write access to the spreadsheet at any given time; anyone else is locked out, even if the second user is on a different part of the spreadsheet.	Multiple users can access a database table at the same time, with various combinations of read and write capabilities in different parts of the database.
A spreadsheet does not have any built-in transaction-control capabilities, such as ensuring that a group of changes to the sheet is completely applied or not applied at all. The Save button is about the best a spreadsheet can do to simulate transaction control.	A database usually has transaction-control capabilities, making it possible to "roll back" a change if something happened to prevent it from completing successfully (such as a power failure).

## Relational Database Concepts

### Spreadsheet

A corrupt spreadsheet cannot usually be repaired; the entire spreadsheet must be restored from a backup, which may have occurred yesterday, last week, or never!

### Database

There are many tools for repairing and recovering databases.

This is not to say that a spreadsheet isn't a valuable tool in the enterprise for ad-hoc and "what-if" analyses. Furthermore, most spreadsheet products have some way to connect to an external database as the data source for analysis.

## Relational Databases

The relational model is the basis for any relational database management system (RDBMS). A relational model has three core components: a collection of objects or relations, operators that act on the objects or relations, and data integrity methods. In other words, it has a place to store the data, a way to create and retrieve the data, and a way to make sure that the data is logically consistent.

### Hierarchical and Network Databases

The relational model was first proposed by Dr. E. F. Codd in 1970. At that time, databases were primarily either of the hierarchical or network type.

A hierarchical database is similar in nature to a filesystem, with a root or parent node and one or more children referencing the parent. This makes for a very fast data-access path, but it has the disadvantages of low flexibility, lack of an ad-hoc query capability, and high application maintenance.

A network database has some advantages over the hierarchical model, including a data definition language, a data manipulation language, and data integrity. However, like hierarchical databases, network databases suffer from rigidity in database structure and high application maintenance costs.

Hierarchical and network-based databases are still used for extremely high-volume transaction-processing systems. IBM claims that 95% of the Fortune 1000 companies in the world still use IMS, a hierarchical database management system that is also web-enabled.

**relational database**

A collection of tables that stores data without any assumptions as to how the data is related within the tables or between the tables.

**table**

The basic construct of a relational database that contains rows and columns of related data.

**relation**

A two-dimensional structure used to hold related information, also known as a table.

**row**

A group of one or more data elements in a database table that describes a person, place, or thing.

**column**

The component of a database table that contains all of the data of the same name and type across all rows.

**field**

The smallest piece of information that can be retrieved by the database query language. A field is found at the intersection of a row and a column in a database table.

A **relational database** uses relations, or two-dimensional tables, to store the information needed to support a business. Let's go over the basic components of a traditional relational database system and look at how a relational database is designed. Once you have a solid understanding of what rows, columns, tables, and relationships are, you'll be well on your way to leveraging the power of a relational database.

**NOTE**

While this book focuses on the Oracle RDBMS for all of its examples and techniques, it's good to know how Oracle fits in with other database vendors and platforms. Appendix C, "Common Database Platforms," has an overview of the major RDBMS vendors and their products.

## Tables, Rows, and Columns

A **table** in a relational database, alternatively known as a **relation**, is a two-dimensional structure used to hold related information. A database consists of one or more related tables.

**NOTE**

Don't confuse a relation with relationships. A relation is essentially a table, and a relationship is a way to correlate, join, or associate the two tables.

A **row** in a table is a collection or instance of one thing, such as one employee or one line item on an invoice. A **column** contains all the information of a single type, and the piece of data at the intersection of a row and a column, a **field**, is the smallest piece of information that can be retrieved with the database's query language. (Oracle's query language, SQL, is the topic of the next chapter.) For example, a table with information about employees might have a column called `LAST_NAME` that contains all of the employees' last names. Data is retrieved from a table by filtering on both the row and the column.

**NOTE**

SQL, which stands for Structured Query Language, supports the database components in virtually every modern relational database system. SQL has been refined and improved by the American National Standards Institute (ANSI) for more than 20 years. As of Oracle9i, Oracle's SQL engine conforms to the ANSI SQL:1999 (also known as SQL3) standard, as well as its own proprietary SQL syntax that existed in previous versions of Oracle. Until Oracle9i, only SQL:1992 (SQL2) syntax was fully supported.

# Primary Keys, Datatypes, and Foreign Keys

The examples throughout this book will focus on the hypothetical work of Scott Smith, database developer and entrepreneur. He just started a new widget company and wants to implement a few of the basic business functions using the Oracle relational database to manage his Human Resources (HR) department.



### NOTE

Most of Scott's employees were hired away from one of his previous employers, some of whom have over 20 years of experience in the field. As a hiring incentive, Scott has agreed to keep the new employees' original hire date in the new database.

You'll learn about database design in the following sections, but let's assume for the moment that the majority of the database design is completed and some tables need to be implemented. Scott creates the EMP table to hold the basic employee information, and it looks something like this:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Notice that some fields in the Commission (COMM) and Manager (MGR) columns do not contain a value; they are blank. A relational database can enforce the rule that fields in a column may or may not be empty. (Chapter 3, "Oracle Database Functions," covers the concept of empty, or NULL, values.) In this case, it makes sense for an employee who is not in the Sales department to have a blank Commission field. It also makes sense for the president of the company to have a blank Manager field, since that employee doesn't report to anyone.

**primary key**

A column (or columns) in a table that makes the row in the table distinguishable from every other row in the same table.

**foreign key**

A column (or columns) in a table that draws its values from a primary or unique key column in another table. A foreign key assists in ensuring the data integrity of a table.

**referential integrity**

A method employed by a relational database system that enforces one-to-many relationships between tables.

**data modeling**

A process of defining the entities, attributes, and relationships between the entities in preparation for creating the physical database.

On the other hand, none of the fields in the Employee Number (EMPNO) column are blank. The company always wants to assign an employee number to an employee, and that number must be different for each employee. One of the features of a relational database is that it can ensure that a value is entered into this column and that it is unique. The EMPNO column, in this case, is the **primary key** of the table.

Notice the different datatypes that are stored in the EMP table: numeric values, character or alphabetic values, and date values. The Oracle database also supports other variants of these types, plus new types created from these base types. Datatypes are discussed in more detail throughout the book.

As you might suspect, the DEPTNO column contains the department number for the employee. But how do you know what department name is associated with what number? Scott created the DEPT table to hold the descriptions for the department codes in the EMP table.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

The DEPTNO column in the EMP table contains the same values as the DEPTNO column in the DEPT table. In this case, the DEPTNO column in the EMP table is considered a **foreign key** to the same column in the DEPT table. With this association, Oracle can enforce the restriction that a DEPTNO value cannot be entered in the EMP table unless it already exists in the DEPT table. A foreign key enforces the concept of **referential integrity** in a relational database. The concept of referential integrity not only prevents an invalid department number from being inserted into the EMP table, but it also prevents a row in the DEPT table from being deleted if there are employees still assigned to that department.

## Data Modeling

Before Scott created the actual tables in the database, he went through a design process known as **data modeling**. In this process, the developer conceptualizes and documents all the tables for the database. One of the common methods for modeling a database is called ERA, which stands for entities, relationships, and attributes. The database designer uses an application that can maintain entities, their attributes, and their relationships. In general, an entity corresponds to a table in the database, and the attributes of the entity correspond to columns of the table.





### NOTE

Various data modeling tools are available for database design. Examples include Microsoft Visio ([www.microsoft.com/office/visio](http://www.microsoft.com/office/visio)) and more robust tools such as Computer Associate's ERwin ([www3.ca.com/Solutions/Product.asp?ID=260](http://www3.ca.com/Solutions/Product.asp?ID=260)) and Embarcadero's ER/Studio ([www.embarcadero.com/products/erstudio/index.asp](http://www.embarcadero.com/products/erstudio/index.asp)).

The data-modeling process involves defining the entities, defining the relationships between those entities, and then defining the attributes for each of the entities. Once a cycle is complete, it is repeated as many times as necessary to ensure that the designer is capturing what is important enough to go into the database. Let's take a closer look at each step in the data-modeling process.

### Defining the Entities

First, the designer identifies all of the entities within the scope of the database application. The entities are the persons, places, or things that are important to the organization and need to be tracked in the database. Entities will most likely translate neatly to database tables. For example, for the first version of Scott's widget company database, he identifies four entities: employees, departments, salary grades, and bonuses. These will become the EMP, DEPT, SALGRADE, and BONUS tables.

### Defining the Relationships between Entities

Once the entities are defined, the designer can proceed with defining how each of the entities is related. Often, the designer will pair each entity with every other entity and ask, "Is there a relationship between these two entities?" Some relationships are obvious; some are not.

In the widget company database, there is most likely a relationship between EMP and DEPT, but depending on the business rules, it is unlikely that the DEPT and SALGRADE entities are related. If the business rules were to restrict certain salary grades to certain departments, there would most likely be a new entity that defines the relationship between salary grades and departments. This entity would be known as an **associative** or **intersection table**, and would contain the valid combinations of salary grades and departments.

In general, there are three types of relationships in a relational database:

**One-to-many** The most common type of relationship is **one-to-many**. This means that for each occurrence in a given entity, the parent entity, there may be one or more occurrences in a second entity, the child entity, to which it is related.

#### **associative table**

A database table that stores the valid combinations of rows from two other tables and usually enforces a business rule. An associative table resolves a many-to-many relationship.

#### **intersection table**

See *associative table*.

#### **one-to-many relationship**

A relationship type between tables where one row in a given table is related to many other rows in a child table. The reverse condition, however, is not true. A given row in a child table is related to only one row in the parent table.



### one-to-one relationship

A relationship type between tables where one row in a given table is related to only one or zero rows in a second table. This relationship type is often used for subtyping. For example, an EMPLOYEE table may hold the information common to all employees, while the FULL-TIME, PARTTIME, and CONTRACTOR tables hold information unique to full time employees, part time employees and contractors respectively. These entities would be considered subtypes of an EMPLOYEE and maintain a one-to-one relationship with the EMPLOYEE table.

### many-to-many relationship

A relationship type between tables in a relational database where one row of a given table may be related to many rows of another table, and vice versa. Many-to-many relationships are often resolved with an intermediate *associative table*.

For example, in the widget company database, the DEPT entity is a parent entity, and for each department, there could be one or more employees associated with that department. The relationship between DEPT and EMP is one-to-many.

**One-to-one** In a **one-to-one** relationship, a row in a table is related to only one or none of the rows in a second table. These relationships are not as common as one-to-many relationships, because if one entity has an occurrence for a corresponding row in another entity, in most cases, the attributes from both entities should be in a single entity.

**Many-to-many** In a **many-to-many** relationship, one row of a table may be related to many rows of another table, and vice versa. Usually, when this relationship is implemented in the database, a third entity is defined as an intersection table to contain the associations between the two entities in the relationship. For example, in a database used for school class enrollment, the STUDENT table has a many-to-many relationship with the CLASS table—one student may take one or more classes, and a given class may have one or more students. The intersection table STUDENT\_CLASS would contain the combinations of STUDENT and CLASS to track which students are in which class.

## Assigning Attributes to Entities

Once the designer has defined the entity relationships, the next step is to assign the attributes to each entity. This is physically implemented using columns, as shown here for the SALGRADE table as derived from the salary grade entity.

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999
6	10000	12500

## Iterate the Process: Are We There Yet?

After the entities, relationships, and attributes have been defined, the designer may iterate the data modeling many more times. When reviewing relationships, new entities may be discovered. For example, when discussing the widget inventory table and its relationship to a customer order, the need for a shipping restrictions table may arise.

Once the design process is complete, the physical database tables may be created. This is where the DBA usually steps in, although the DBA probably has attended some of the design meetings already! It's important for the DBA to be

involved at some level in the design process to make sure that any concerns about processor speed, disk space, network traffic, and administration effort can be addressed promptly when it comes time to create the database.

Logical database design sessions should not involve physical implementation issues, but once the design has gone through an iteration or two, it's the DBA's job to bring the designers "down to earth." As a result, the design may need to be revisited to balance the ideal database implementation versus realities of budgets and schedules.

# Object-Relational Databases

An **object-relational database** system supports everything a relational database system supports, as well as constructs for object-oriented development and design techniques. Object-oriented constructs are found in modern programming languages such as Java and C++. The Oracle9i database fully supports all of the traditional object-oriented constructs and methods.

While the full range of object-oriented techniques are beyond the scope of this book, you will get a good idea of some of the object-oriented capabilities of Oracle, including abstraction, methods, encapsulation, and inheritance. Let's define those terms now.

## Abstraction

One of the ways in which Oracle supports the object-relational model is by using abstraction. As noted earlier, Oracle has many built-in datatypes, such as numeric, string, date, and others. Additionally, you can define user-defined objects as an aggregate of several other datatypes. These new user-defined types are called **abstract datatypes**.

For example, when Scott's widget company grows, there may be other systems where he needs to represent an employee or a customer, or in more general terms, a person. Scott can define a datatype called PERSON that stores a first name, last name, middle initial, and a gender. When the new customer tables are being built, Scott just needs to use the new PERSON type in the table definition. This brings to the table two immediate benefits: reusability and standards. Creating the new table is faster, since the datatype has already been defined, and it's less error prone than creating four individual fields. In addition, any developer who moves from an employee-oriented project to a customer-oriented project at Scott's company will find familiarity in common objects and naming conventions.

### object-relational database

A relational database that includes additional operations and components to support object-oriented data structures and methods.

### abstract datatypes

New datatypes, usually user-created, that are based on one or more built-in datatypes and can be treated as a unit.

## Methods and Encapsulation

Another way in which object-oriented techniques are reflected in the Oracle object-relational database is through the use of methods and encapsulation.

**Methods** define which operations can be performed on an object. **Encapsulation** restricts access to the object other than via the defined methods.

Take a simple example of an employee object: it contains characteristics such as the employee name, address, and salary. A method against an employee object might be to get the name, or change the name. Another method might be to increase the salary, but never to decrease the salary. The encapsulation of the employee object prevents the direct manipulation of the characteristics of an employee object other than what the methods, driven by business rules, dictate.

## Inheritance

**Inheritance** allows objects that are derived from other objects to use the methods available in the parent object. If a new object is created with an existing object as a base, all of the methods available with the existing object will also be available with the new object.

For example, if Scott were to implement a new `EMPLOYEE` type and a new `CUSTOMER` type using the `PERSON` type as the base, then any methods that already exist for `PERSON` would be available when using one of the two new types. The method `ChangeLastName`, defined with the `PERSON` type only once, can be used with objects defined with the `CUSTOMER` or `EMPLOYEE` type.

## Object-Relational Support

Oracle9i provides additional features to ease the transition to an object-oriented database application. **Object views** allow the developer to define an object-oriented structure over an existing relational database table. In this way, existing applications do not need to change immediately, and any new development can use the object-oriented definitions of the table. This makes the transition from a relational to an object-relational database relatively painless, because object definitions can reference existing relational components.

### methods

Operations on an object that are exposed for use by other objects or applications.

### encapsulation

An object-oriented technique that may hide, or abstract, the inner workings of an object and expose only the relevant characteristics and operations on the object to other objects.

### inheritance

Acquiring the properties of the parent, or base object, in a new object.

### object view

A database construct that overlays an object-oriented structure over an existing relational database table. As a result, the table can be accessed as a relational table or as an object table and make the transition to a fully object-oriented environment easier.

# Review Questions

1. Name the most important element of a relational database and its components.  
\_\_\_\_\_
2. Which type of table relationship associates more than one record in a given table with more than one record in another table?  
\_\_\_\_\_
3. What type of key can be used to enforce referential integrity between two tables in a database?  
\_\_\_\_\_
4. What are some reasons why using a spreadsheet is not a good alternative to using a large-scale database?  
\_\_\_\_\_
5. What are some of the benefits of abstraction in an object-relational database management system?  
\_\_\_\_\_
6. What object-relational feature of Oracle eases the transition between relational and object-relational applications?  
\_\_\_\_\_
7. What are the three steps in the ERA process for database design?  
\_\_\_\_\_

## Terms to Know

- abstract datatypes
- associative table
- column
- data modeling
- encapsulation
- field
- foreign key
- inheritance
- intersection table
- many-to-many relationship
- methods
- object view
- object-relational database
- one-to-many relationship
- one-to-one relationship
- primary key
- referential integrity
- relation
- relational database
- row
- table