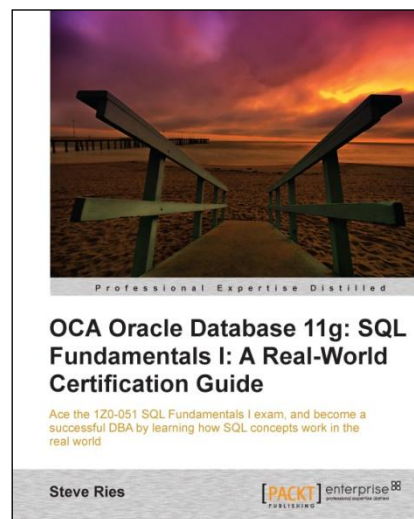


OCA Oracle Database 11g: SQL Fundamentals I: A Real-World Certification Guide

Steve Ries



Chapter No. 1 "SQL and Relational Databases"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO. 1 "SQL and Relational Databases"

A synopsis of the book's content

Information on where to buy this book

About the Author

Steve Ries has been an Oracle DBA for 15 years, specializing in all aspects of database administration, including security, performance tuning, and backup and recovery. He is a specialist in Oracle Real Application Clusters (RAC) and has administered Oracle clustered environments in every version of Oracle since the creation of Oracle Parallel Server. He holds five Oracle certifications as well as the Security+ certification. He currently consults for the Dept of Defense, U.S. Marine Corps, and holds a high-level security clearance. Additionally, Steve has been an adjunct instructor of Oracle technologies at Johnson County Community College for eight years where he teaches classes that prepare students for the Oracle certification exams. He was also a speaker at the 2011 Oracle Open World conference. Steve is an award-winning technical paper writer and the creator of the alt.oracle blog.

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

I would like to thank Gary Hayes, Carol Ross, Matt Sams, Pete Scalzi, Angela Morten, Joe Duvall, Sandee Vandenboom, Karen Buck, Gary Deardorff, and Chad Fletcher for their support and technical advice during the writing of this book. I would also like to thank Debbie Rulo, Keith Krieger, and the staff at the Center for Business at Johnson County Community College for their support. Finally, I would like to thank my wife Dee and daughter Faith for their love, personal support, and patience.

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

OCA Oracle Database 11g: SQL Fundamentals I: A Real-World Certification Guide

There's never been a time in the Information Technology industry where professional certifications have been more important. Because of the specialized nature of technological careers today, certifications are considered by some to be just as important as technological degrees. This focus on certifications has led to the rise of an entire industry around books that assist readers in preparing for various certification tests. In the author's opinion, many, if not most, of these books make a lot of assumptions as to the prior knowledge of the reader and serve more as reference material than a cohesive learning experience.

In my role as an instructor of Oracle technologies, I have noticed a shift in the types of people seeking to learn Oracle. In the past several years, more and more students are seeking to break into an Oracle career path with little or no experience. Whether they come from backgrounds in business analysis, project management, or other non-database technical areas, these students need to be able to learn Oracle from the ground up. When instructing these types of students, I cannot make assumptions as to the knowledge they bring with them. We must start at the beginning and work our way to certification level knowledge. To accomplish this goal in class, the accompanying textbook must be designed in the same way.

Similarly, many certification books today serve only as exam cram books that neglect an application to real world scenarios. Readers of these types of books may find themselves with a certification, yet possess no way to apply the knowledge in their first job.

For More Information:

www.packtpub.com/oqa-oracle-database-11g-sql-fundamentals/book

My goal in writing this book is to address both of these problems. This book attempts to begin at the foundation and continue to the knowledge of the subject required for the certification exam, using real world examples and tips along the way. This book is heavily example-oriented and is intended to serve as step-by-step instruction instead of reference material. In essence, I attempt to bring the classroom experience to the reader, using examples, real world tips, and end-of-the-chapter review. This book is written to be read cover to cover, with the reader completing the examples and questions as they go. Using this process, it is my hope that readers can truly begin at the beginning, regardless of previous experience, and learn SQL in a relevant way that will serve them in their pursuit of an Oracle certification as well as an Oracle career path.

The Oracle Database 11g: SQL Fundamentals I exam is the first stepping stone in earning the Oracle Certified Associate Certification for Oracle Database 11g. The SQL programming language is used in every major relational database today, and understanding the real-world application of it is the key to becoming a successful DBA.

This book gives you the essential real-world skills to master relational data manipulation with Oracle SQL and prepares you to become an Oracle Certified Associate. Beginners are introduced to concepts in a logical manner while practitioners can use it as a reference to jump to relevant concepts directly.

We begin with the essentials of why databases are important in today's information technology world and how they work. We continue by explaining the concepts of querying and modifying data in Oracle using a range of techniques, including data projection, selection, creation, joins, sub-queries, and functions. Finally, we learn to create and manipulate database objects and to use them in the same way as today's expert SQL programmers.

This book prepares you to master the fundamentals of the SQL programming language using an example-driven approach that is easy to understand.

This definitive certification guide provides a disciplined approach to be adopted for successfully clearing the 1Z0-051 SQL Fundamentals I exam, which is the first stepping stone towards attaining the OCA on Oracle Database 11g certification.

Each chapter contains ample practice questions at the end. A full-blown mock test is included for practice so you can test your knowledge and get a feel for the actual exam.

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

What This Book Covers

Chapter 1, SQL and Relational Database, examines the purpose and use of relational database management systems, including the use of entity relationship diagrams and the structure of tables. We then introduce Structured Query Language and the SQL Developer tool.

Chapter 2, SQL SELECT Statements, explores the most foundational SQL clause; the SELECT statement. We use SELECT statements for single and multi-column data retrieval and take a look at using SQL to do basic mathematical operations.

Chapter 3, Using Conditional Statements, examines the concept of data selection using the WHERE clause paired with conditions. In it, we construct selective statements using conditions of both equality and non-equality. We also use range and set conditions with the WHERE clause for further data selectivity. Finally, we examine the concept of sorting data using the ORDER BY clause.

Chapter 4, Data Manipulation with DML, explores the use of Data Manipulation Language to add, modify, and remove table data using INSERT, UPDATE, and DELETE statements. Lastly, we look at transaction control in SQL.

Chapter 5, Combining Data from Multiple Tables, examines the concept of combining data from multiple tables using various join statements. We accomplish this using both ANSI standard and Oracle syntax.

Chapter 6, Row Level Data Transformation, explores the concept of row level data transformation using single-row functions. We use these functions to transform date, character and numeric data.

Chapter 7, Aggregate Data Transformation, explores data transformation using functions, this time with multi-row functions. We combine these functions with the GROUP BY and HAVING statements to perform aggregate data transformation.

Chapter 8, Combining Queries, focuses on using several types of subqueries to combine data from multiple tables. We close the chapter by exploring set theory in Oracle and implement it using SQL set operators.

Chapter 9, Creating Tables, introduces the concept of Data Definition Language and how to use it to create database tables. We also use SQL to write database constraints that enforce business data rules.

Chapter 10, Creating Other Database Objects, examines the use of DDL statements to create some of the other common objects available to us in Oracle. We use these statements to create indexes, views, sequences, and synonyms.

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

Chapter 11, Using SQL in Application Development, examines how SQL is used in real-world programming languages such as PL/SQL, Perl, Python, and Java. We close by offering hints and strategies for taking the SQL certification exam.

Appendix A, Companylink Table Reference, a reference section describing the various tables used as examples in this book.

Appendix B, Getting Started with APEX, shows an alternative method for completing the examples in this book that does not require installing the Oracle database software.

Appendix C, Test Your Knowledge; you can download this appendix that contains answers to the *Test your knowledge* section in all the chapters at http://www.packtpub.com/sites/default/files/downloads/testyourknowledge_answers.pdf.

Mock practice test paper can be downloaded from http://www.packtpub.com/sites/default/files/downloads/mock_test_paper.pdf

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

1

SQL and Relational Databases

We live in a data-driven world. Think for a moment about all the data that exists about you, in computers around the world.

- Your name
- Birth date and information
- Your hobbies
- Purchases you've made
- The identity of your friends
- Your place of employment

The examples are endless. Next, multiply that amount of data by the number of people in the world. The result is a truly staggering amount of information. How is it possible that all this data can be organized and retrieved? In today's data-centric world, it is databases that make this possible. These **Relational Database Management Systems (RDBMS)** are primarily controlled by a programming language called **Structured Query Language (SQL)**.

In this chapter, we will cover the following topics:

- Discussing the purpose of relational database management systems
- Understanding the use of the relational paradigm
- Examining the use of Entity Relationship Diagrams (ERDs)
- Looking at the structure of tables
- Introducing Structured Query Language (SQL)
- Reviewing commonly-used query tools
- Introducing the SQL Developer tool

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

Relational Database Management Systems

Imagine, for a moment, that you have the telephone books for the 20 largest cities in the U.S. I give you the following request: *Please find all the phone numbers for individuals named Rick Clark in the Greater Chicago area.* In order to satisfy the request, you simply do the following:

- Open the Chicago phone book
- Scan to the "C" section of names
- Find all individuals that match "Clark, Rick"
- Report back their phone numbers

Now imagine that I take each phone book, tear out all of the pages, and throw them into the air. I then proceed to shuffle the thousands of pages on the ground into a completely disorganized mess. Now I repeat the same request: *Please find all the phone numbers for individuals named Rick Clark in the Greater Chicago area.* How do you think you would do that? It would be nearly impossible. The data is all there, but it's completely disorganized. Finding the "Rick Clarks" of Chicago would involve individually examining each page to see if it satisfied the request—a very frustrating undertaking, to say the least.

This example underscores the importance of a database, or more accurately, a **Relational Database Management System (RDBMS)**. Today's RDBMSs are what enable the storage, modification, and retrieval of massive amounts of data.

Flat file databases

When the devices that we know as computers first came into existence, they were primarily used for one thing—computation. Computers became useful entities because they were able to do numeric computation on an unprecedented scale. For example, one of the first computers, ENIAC, was designed (although not used) for the US Army to calculate artillery trajectories, a task made simpler through the use of complex sequences of mathematical calculations. As such, originally, computers were primarily a tool for mathematical and scientific research. Eventually, the use of computers began to penetrate the business market, where the company's data itself became just as important as computational speed. As the importance of this data grew, so the need for data storage and management grew as well, and the concept of a database was born.

The earliest databases were simple to envision. Most were simply large files that were similar in concept to a spreadsheet or **comma-separated values (CSV)** file. Data was stored as fields. A portion of these databases might look something like the following:

```
Susan, Bates, 123 State St, Somewhere, VA
Fred, Hartman, 234 Banner Rd, Anywhere, CA
Bill, Frankin, 345 Downtown Rd, Somewhere, MO
Emily, Thompson, 456 Uptown Rd, Somewhere, NY
```

In this example, the first field is determined by reading left to right until a delimiter, in this case a comma, is reached. This first field refers to the first name of the individual. Similarly, the next field is determined by reading from the first delimiter to the next. That second field refers to the last name of the individual. It continues in this manner until we have five fields—first name, last name, street address, city, and state. Each individual line or record in the file refers to the information for a distinct individual. Because this data is stored in a file, it is often referred to as a flat file database. To retrieve a certain piece of information, programs could be written that would scan through the records for the requested information. In this way, large amounts of data could be stored and retrieved in an orderly, programmatic way.

Limitations of the flat file paradigm

The flat file database system served well for many years. However, as time passed and the demands of businesses to retain more data increased, the flat file paradigm began to show some flaws.

In our previous example, our flat file is quite limited. It contains only five fields, representing five distinct pieces of information. If this flat file database contained the data for a real company, five distinct pieces of information would not even begin to suffice. A complete set of customer data might include addresses, phone numbers, information about what was ordered, when the order was placed, when the order was delivered, and so on. In short, as the need to retain more data increases, the number of fields grows. As the number of fields grows, our flat file database gets wider and wider. We should also consider the amount of data being stored. Our first example had four distinct records; not a very realistic amount for storing customer data. The number of records could actually number in thousands or even millions. Eventually, it is completely plausible that we could have a single flat file that is hundreds of fields wide and millions of records long. We could easily find that the speed with which our original data retrieval programs can retrieve the required data is decreasing at a rapid rate and is insufficient for our needs.

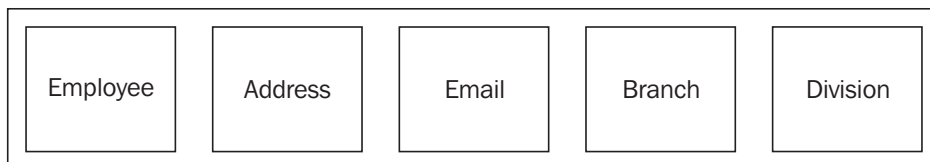
As our data demands increase, we're presented with another problem. If we are storing order information, for example, strictly under the flat file paradigm, we are forced to store a new record each time an order is placed. Consider this example, in which our customer purchases six different items. We store a six-digit invoice number, customer name, and address for the customer's purchase, as follows:

```
487345, Susan, Bates, 123 State St, Somewhere, VA
584793, Susan, Bates, 123 State St, Somewhere, VA
998347, Susan, Bates, 123 State St, Somewhere, VA
126543, Susan, Bates, 123 State St, Somewhere, VA
487392, Susan, Bates, 123 State St, Somewhere, VA
```

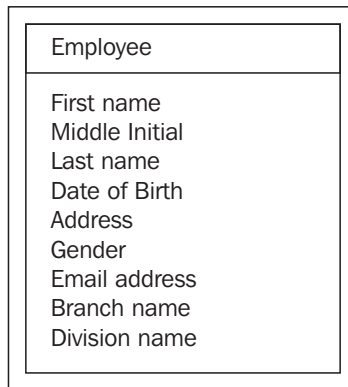
Using this example, notice how much duplicate data we have stored. The fields are invoice number, first name, last name, street address, city, and state, respectively. The only different piece of information in each record is the invoice number, and yet we have repeatedly stored the last five fields—information that is stored in previous records. We refer to these anomalies as repeating values. Repeating values present two problems from a processing standpoint. First, the duplicate data must be re-read each time by our retrieval programs, creating a performance problem for our retrieval operations. Second, those duplicate characters constitute bytes that must be stored on disk, uselessly increasing our storage requirements. It is clear that the flat file paradigm needs to be revised in order to meet the growing demands of our database.

Normalization

The world of databases changed in the early 1970s due in large part to the work of Dr. Edgar "Ted" Codd. In his paper, *A Relational Model of Data for Large Shared Data Banks*, Dr. Codd presented a new paradigm—the *relational paradigm*. The relational paradigm seeks to resolve the issues of repeating values and unconstrained size by implementing a process called normalization. During normalization, we organize our data in such a way that the data and its inter-relationships can be clearly identified. When we design a database, we begin by asking two questions—what data do I have? And, how do the pieces of data relate to each other? In the first step, the data is identified and organized into entities. An entity is any person, place, or thing. An entity also has attributes, or characteristics, that pertain to it. Some example entities are listed in the following diagram:



These entities represent distinct pieces of information: the `Employee` entity represents information about employees, the `Email` entity represents information about e-mail addresses, and so on. These entities, and any others we choose to add, make up our data model. We can also look a little closer at the attributes of a particular entity, as shown in the following diagram:



In our example, data (such as `First name`, `Last name`, `Address`, and `Branch name`) are the attributes of the `Employee` entity—they describe information about the employee. This is by no means exhaustive. There would most likely be many more attributes for an employee entity. In fact, this is part of the problem that we discussed earlier with the flat file database—data tends to accumulate, making our file wider and wider, if you will. Additionally, if we were to actually collect this data in a flat file, it might look something like the following screen:

First Name	Mid Initial	Last Name	Address	Email Address
James	R	Johnson	123 State St, Bell, WA	jj@hotmail.com, jj@yahoo.com
Mary	S	Williams	234 First St, Bigtown, VA	mw@gmail.com
Linda	L	Anderson	345 Fifth Ave, Smalltown, MA	la@yahoo.com, la@gmail.com

At first glance, this structure may appear to be adequate, but, if we examine further, we can identify problems with it. To begin with, we note that there are multiple values stored in the `Address` and `Email Address` fields, which can make structuring queries difficult. This is where the process of normalization can assist us. Normalization involves breaking data into different normal forms. These forms are the steps we take to transform non-relational data into relational data. The **first normal form (1NF)** involves determining a **primary key**—a value in each occurrence of the data that uniquely identifies it. In the previous example of data, what attribute could be used to uniquely identify each occurrence of data? Perhaps we could use `First Name`.

However, it seems fairly clear that there could be more than one employee with the name James or Mary, so that will not suffice. If we were to use `First Name` and `Last Name` together as our primary key values, we would get closer to uniqueness, but it would still be insufficient for common names such as John Smith. For now, let us say that `First Name`, `Mid Initial`, and `Last Name`, together (as indicated earlier), uniquely identify each occurrence of data and thus comprise our primary key for the employee entity.

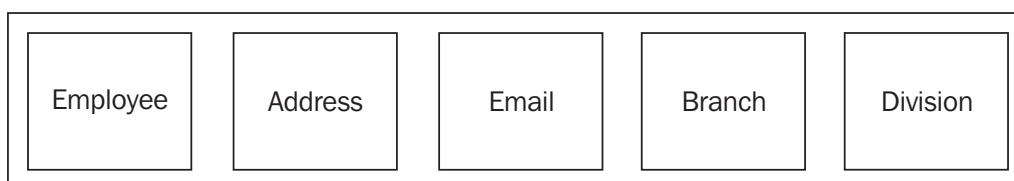
The next issue is the problem of repeating groups. Examine the `Email Address` attribute. It may be required that one or more e-mail addresses be stored for each employee. This presents problems when attempting to query for a particular employee's e-mail address. As each employee can have more than one e-mail address, the `Email` attribute would have to be scanned rather than simply pattern-matched in order to retrieve a particular piece of data. One way to rectify this would be to break each individual occurrence of the e-mail address into two separate records that demonstrates the removal of repeating groups, as shown in the next example. Thus, James R. Johnson, who has two `Email Addresses`, now has two rows in the database—one for the first e-mail address and one for the second:

First Name	Mid Initial	Last Name	Address	Email Address
James	R	Johnson	123 State St, Bell, WA	jj@hotmail.com
James	R	Johnson	123 State St, Bell, WA	jj@yahoo.com
Mary	S	Williams	234 First St, Bigtown, VA	mw@gmail.com
Linda	L	Anderson	345 Fifth Ave, Smalltown, MA	la@yahoo.com
Linda	L	Anderson	345 Fifth Ave, Smalltown, MA	la@gmail.com

We have eliminated the repeating groups, but we have now introduced other problems. First, we have violated our primary key, as first, middle, and last name no longer uniquely identify each row. Second, we have begun to duplicate our data. First name, middle initial, last name, and address are all repeated simply for the sake of removing repeating groups. Lastly, we now realize that it is possible for our employees to have more than one address, which further complicates the problem. Clearly, the first normal form alone is insufficient. It is necessary to transform the data again—this time into the **second normal form (2NF)**.

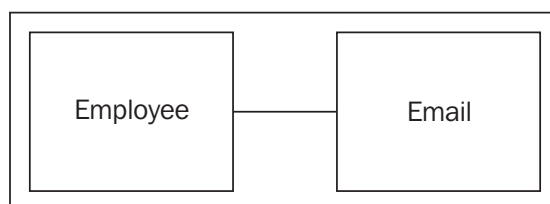
The relational approach

The second normal form involves breaking our employee entity into a number of separate entities, each of which can have a unique primary key and no repeating groups. This is displayed again in the following diagram:

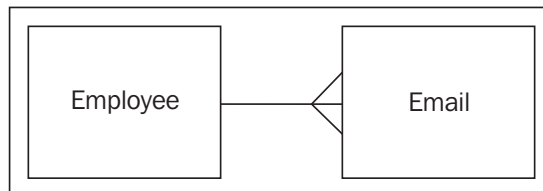


Here, we have separated our employee information into separate entities. We've also added entities that represent the branch and division of which each employee is a part. Now our employee entity contains information such as first name, middle initial, and last name, while our `Email` entity contains the e-mail address information. The other entities operate similarly – each contains information unique to itself.

This may have solved our repeating data problem, but now we simply have five files that have no relation to each other. How do we connect a particular employee to a particular e-mail address? We do this by establishing relationships between the entities; another requirement of the second normal form. A relationship between two entities is formed when they share some common piece of information. How this relationship functions is determined by the business rules that govern our data. Let's say in our model that one, and only one, e-mail address is kept for each employee. We would then say that there is a one-to-one relationship between our `employee` entity and our `Email` entity. Generally, such a relationship is denoted visually with a single bar between the two. We could diagram it as follows:

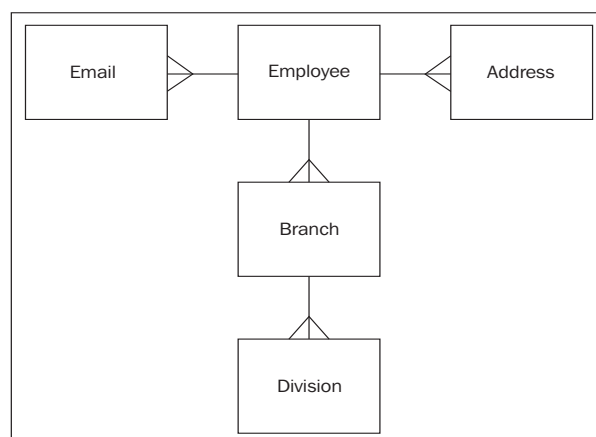


A more realistic relationship, however, would be one where each employee could have more than one e-mail address. This type of relationship is termed a one-to-many relationship. These relationships form the majority of the relationships used in the relational model and are shown in the following diagram. Note the crow's foot connecting to the `Email` entity, indicating many:



As you might expect, there is another type of relationship, one which, under relational rules, we try to avoid. A many-to-many relationship occurs when multiple occurrences of the data in one entity relate to multiple occurrences in the other. For instance, if we allowed employees to have multiple e-mail addresses, but also allowed multiple employees to share a single e-mail address. In the relational paradigm, we seek to avoid these types of relationships, usually by relating an entity between the two that transforms a many-to-many relationship into two distinct one-to-many relationships. The last step in normalization is generally the transformation into the **third normal form (3NF)**. In the 3NF, we remove any columns that are not dependent on the primary key. These are known as **transitive dependencies**. To resolve these dependencies, we move the non-dependent columns into another table. There are higher normal forms, such as **fourth normal form (4NF)** and **fifth normal form (5NF)**, but these forms are less commonly used. Generally, once we have taken our data structure up to the 3NF, our data is considered relational.

When we design the number and types of relationships between our entities, we construct a data model. This data model is the guide for the DBA on how to construct our database objects. The visual representation of a data model is commonly referred to as an entity relationship diagram (ERD). Using the five example entities we listed previously, we can construct a simple entity relationship diagram, as demonstrated in the following example:



From this example model, we can determine that the `employee` entity is more or less the center of this model. An employee can have one or more e-mail addresses. An employee can also have one or more street addresses. An employee can belong to one or more branches, and a branch can belong to one or more divisions. Even though it is highly simplified, this diagram shows the basic concepts of visually modeling data.

Through the use of relational principles and entity relationship diagrams, a database administrator or data architect can take a list of customer data and organize it into distinct sets of information that relate to one another. As a result, data duplication is greatly reduced and retrieval performance is increased. It is because of this efficient use of storage and processing power that the RDBMS is the predominant method used in storing data today.



SQL in the real world

Strictly speaking, the Oracle RDBMS is actually an **Object Relational Database Management System (ORDBMS)** and has been since Oracle version 8. An ORDBMS refers to the ability of Oracle databases to be manipulated using object-oriented concepts.

Bringing it into the Oracle world

To discuss the relational paradigm, we have used relational terminology, which is designed to be generic and not associated with any particular database product. The subject of this book, however, is using SQL with Oracle databases. It is time to relate the terminology used in the relational paradigm to terms that are likely more familiar:

Relational	Flat file	Oracle-specific
Entity	File	Table
Attribute	Field	Column
Tuple	Record	Row

The preceding diagram shows a comparison table of the different terms used to describe basic database components. Up to this point, we have used the relational term, entity, to describe our person, place, or thing. From this point, we will refer to it by its more commonly known name—the table.

Tables and their structure

If you've ever used a spreadsheet before, then you are familiar with the concept of a table. A table is the primary logical data structure in an Oracle database. We use the term logical because a table has no physical structure in itself—you cannot simply login to a database server, open up a file manager, and find the table within the directories on the server. A table exists as a layer of abstraction from the physical data that allows a user to interface with it in a more natural way. Examine the following diagram; like a spreadsheet, a table consists of columns and rows:

The diagram illustrates the structure of a table. At the top, the word "Columns" has three arrows pointing down to the column headers of a table: "FIRST_NAME", "MIDDLE_INITIAL", and "LAST_NAME". To the left of the table, the word "Rows" has five arrows pointing to the first five rows of data. The table contains 15 rows of data, each with five columns: FIRST_NAME, MIDDLE_INITIAL, LAST_NAME, GENDER, and DOB.

FIRST_NAME	MIDDLE_INITIAL	LAST_NAME	GENDER	DOB
James	R	Johnson	M	01-01-60
Mary	S	Williams	F	03-15-64
Linda	L	Anderson	F	10-24-70
Daniel	J	Robinson	M	11-23-59
Matthew	K	Garcia	M	04-14-71
Helen	H	Harris	F	07-13-75
Ken	W	White	M	02-22-58
Donald	A	Perez	M	03-14-79
Lisa	C	Lee	F	06-15-63
Carol	M	Clark	F	08-11-67
Gary	R	Moore	M	11-01-65
Cynthia	B	Hall	F	10-21-55
Sandra	S	Rodriguez	F	05-10-74
Kevin	L	Lewis	M	07-01-76
George	H	Taylor	M	12-24-72
Laura	I	Thomas	F	10-26-81

A column identifies any single characteristic of a particular table, such as first name. A column differs from a row in more than its vertical orientation. Each value within a column contains a particular type of data, or data type. For instance, in the preceding example, the column `FIRST_NAME` denotes that all data within that column will be of the same type and that data type will be consistent with the label `FIRST_NAME`. Such a column would contain only character string data. For instance, in the `FIRST_NAME` column, we have data such as Mary and Matthew, but not the number 42532.84. In the date of birth column, or `DOB`, only date data would be stored. As we will see in the next chapter, in Oracle, string data or text data is not the same thing as date data.

Along the horizontal, we have rows of data. A row of data is any single instance of a particular piece of information. For example, in the first row of the table in our example, we have the following pieces of information:



Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

```
First name = "James"
Middle initial = "R"
Last name = "Johnson"
Gender = "M"
DOB = "01-01-60"
```

This information comprises the sum total of all the information we have in this table for a single individual, namely, James R. Johnson. The following row, for Mary S. Williams, contains the same types of information, but different values. This construct allows us to store and display data that is orderly in terms of data types, but still flexible enough to store the data for many different individuals. Together, the columns and rows of data form a relational table: the heart of the Oracle database. However, in order to retrieve and manipulate this table data, we need a programming language; for relational databases, that language is SQL.

Structured Query Language

SQL was developed by Donald Chamberlain and Raymond Boyce in the early 1970s as a language to retrieve data from IBM's early relational database management systems. It was accepted as a standard by the **American National Standards Institute (ANSI)** in 1986. SQL is generally referred to as a **fourth-generation language (4GL)**, in contrast with **third-generation languages (3GLs)** such as C, Java, and Python. As a 4GL, the syntax of SQL is designed to be even closer to human language than 3GLs, making it relatively natural to learn. Some do not refer to SQL as a programming language at all, but rather a data sub-language.

A language for relational databases

Before we look at what SQL (pronounced either 'S-Q-L' or 'sequel') is, it is important to define what it is not. First, SQL is not a product of Oracle or any other software company. While most relational database products use some implementation of SQL, none of them own it. The structure and syntax of SQL are governed by the American National Standards Institute and the **International Organization for Standardization (ISO)**. It is these organizations that decide, albeit with input from other companies such as Oracle, what comprises the accepted standard for SQL. The current revision is SQL:2008.

Second, while the ANSI standard forms the basis for the various implementations of SQL used in different database management systems, this does not mean that the SQL syntax and functionality in all database products is the same; in fact, it is often quite different. For instance, the SQL language permits the concatenation of two column values into one; for example, the values *hello* and *there* concatenated would be *hellothere*. Oracle and Microsoft SQL Server both use symbols to denote concatenation, but they are different symbols. Oracle uses the double-pipe symbol, '||', and SQL Server uses a plus sign, '+'. MySQL, on the other hand, uses a keyword, `CONCAT`. Additionally, RDBMS software manufacturers often add functionalities to their own SQL implementations. In Oracle version 10g, a new type of syntax was included to join data from two or more tables that differs significantly from the ANSI standard. Oracle does, however, still support the ANSI standard as well.



SQL in the real world

Although the SQL implementations of the major RDBMS products differ, they all conform to the basic ANSI standard. That means if you learn how SQL is used in one database product, such as Oracle, much of your acquired knowledge should transfer easily to other database products.

Last, SQL should not be confused with any particular database product, such as Microsoft SQL Server or MySQL. Microsoft SQL Server is sometimes referred to by some as SQL; a confusing distinction.

What SQL does provide for developers and database administrators is a simple but rich set of commands that can be used to do the following:

- Retrieve data
- Insert, modify, and delete data
- Create, modify, and delete database objects
- Give or remove privileges to a user
- Control transactions

One of the interesting things about SQL is its dataset-oriented nature. When programmers use third-generation languages such as C++, working with the kinds of datasets we use in SQL is often a cumbersome task involving the explicit construction of variable arrays for memory management. One of the benefits of SQL is that it is already designed to work with arrays of data, so the memory management portion occurs implicitly. It is worth noting, however, that third-generation languages can do many things that SQL cannot. For instance, by itself, SQL cannot be used to create standalone programs such as video games and cell phone applications. However, SQL is an extremely effective tool when used for the purpose for which it was designed – namely, the retrieval and manipulation of relational data.



SQL in the real world

Standard programming constructs such as flow control, recursion, and conditional statements are absent from SQL. However, Oracle has created PL/SQL, a third-generation overlay language that adds these and other basic programmatic constructs to the SQL language. Because of the strength of the SQL language in manipulating data, PL/SQL is often the choice of developers when programming the portions of their applications that interact with Oracle databases.

The goal of this book is to teach you the syntax and techniques to use the SQL language to make data do whatever you want it to do. *Chapter 2, SQL SELECT Statements* and beyond address these topics. However, before we can learn more about the SQL language, we are going to need to choose a tool that can interact with the database and process our SQL.

Commonly-used SQL tools

Because SQL is the primary interface into relational databases, there are many SQL manipulation tools from which to choose. There are benefits and drawbacks to each, but the choice of tool to use is generally about your comfort level with the tool and its feature set. Some tools are free, some are open source, and some require paid licenses; however, each tool uses the same syntax for SQL when it connects to an Oracle database. Following are some commonly-used SQL tools:



SQL in the real world

While your choice of SQL tool is an important one, in the industry it is one that is sometimes dictated by the toolset standards of your employer. It's important that you don't completely dedicate yourself to one tool. If you become an expert at one and then transfer to a different employer whose standards don't allow for the use of your tool, you may find yourself with an initial learning curve.

SQL*Plus

SQL*Plus is the de facto standard of SQL tools to connect to an Oracle database. Since Oracle's early versions, it has been included as a part of any Oracle RDBMS installation. SQL*Plus is a command-line tool and is launched on all Oracle platforms using the command, `sqlplus`. This command-line tool has been a staple of Oracle database administrators for many years. It has a powerful, interactive command interface that can be used to issue SQL statements, create database objects, launch scripts, and startup databases. However, compared with some of the newer tools, it has a significant learning curve. Its use of line numbering and mandatory semicolons for execution is often confusing to beginners. Oracle has also released a GUI version of SQL*Plus for use on Windows systems. Its rules for use, however, are still generally the same as the command-line interface, and its confinement to the Windows platform limits its use. As of Oracle version 11g, the GUI version of SQL*Plus is no longer included with a standard Oracle on Windows installation. Whatever your choice of SQL tool, it is very difficult for a database administrator to completely avoid using SQL*Plus. The following is a screenshot of the command-line SQL*Plus tool:

```

CA Command Prompt - sqlplus companylink/companylink

SQL> select first_name, last_name, dob from employee;

FIRST_NAME      LAST_NAME      DOB
-----
James           Johnson        01-JAN-60
Mary            Williams       15-MAR-64
Linda           Anderson       24-OCT-70
Daniel          Robinson       23-NOV-59
Matthew         Garcia         14-APR-71
Helen           Harris         13-JUL-75
Ken             White          22-FEB-58
Donald          Perez          14-MAR-79
Lisa            Lee            15-JUN-63
Carol           Clark          11-AUG-67
Gary            Moore          01-NOV-65
Cynthia         Hall           21-OCT-55
Sandra          Rodriguez      10-MAY-74
Kevin           Lewis          01-JUL-76
George          Taylor         24-DEC-72
Laura           Thomas         26-OCT-81

16 rows selected.

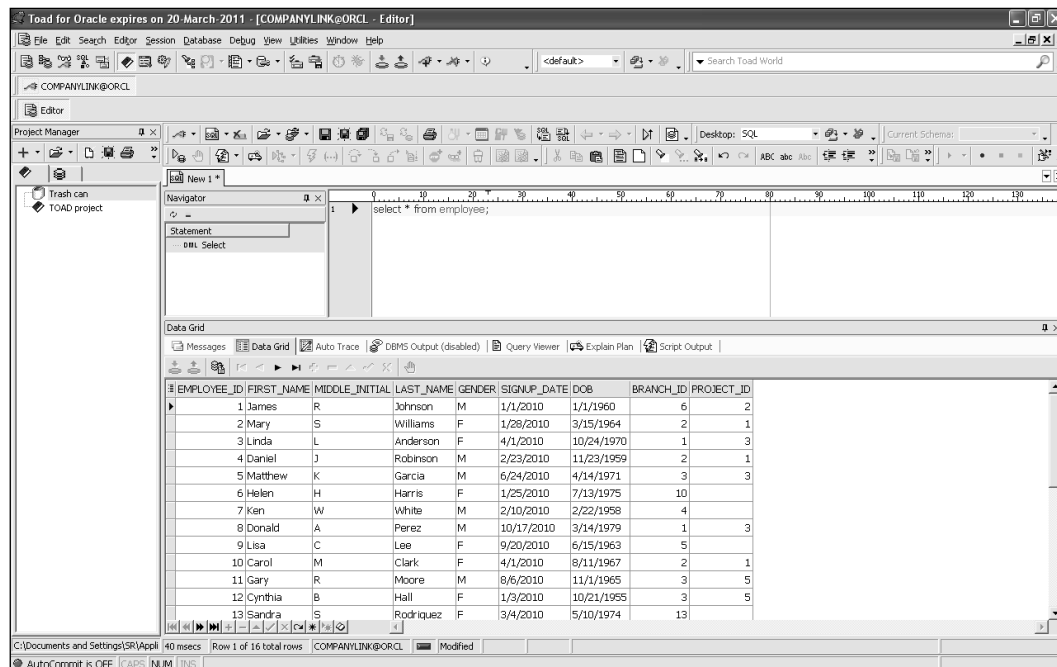
SQL> _

```

TOAD

The **Tool for Oracle Application Developers (TOAD)** is a full-featured development and administration tool for Oracle as well as other relational database systems, including Microsoft SQL Server, Sybase, and IBM's DB2. Originally created by Jim McDaniel for his own use, he later released it as freeware for the Oracle community at large. Eventually, Quest Software acquired the rights for TOAD and began distributing a licensed version, while greatly expanding on the original functionality. TOAD is immensely popular among both DBAs and developers due to its large feature set. For DBAs, it is a complete administration tool, allowing the user to control every major aspect of the database, including storage manipulation, object creation, and security control. For developers, TOAD offers a robust coding interface, including advanced debugging facilities.

TOAD is available for download in both freeware and trial licensed versions. A screenshot is shown as follows:



DBArtisan

DBArtisan (now called **DBArtisan XE**), by Embarcadero Technologies, is another complete suite of database management tools that operates across multiple platforms. DBArtisan is only available as a licensed product, but has extensive administration capabilities, including the ability to do advanced capacity and performance management, all packaged in an attractive and user-friendly GUI frontend. A trial version is available for download from Embarcadero's website.

SQL Worksheet (Enterprise Manager)

The **SQL Worksheet** is not a separate tool in itself; rather, it is a component of the larger Enterprise Manager product. Enterprise Manager is Oracle's flagship, web-based administration suite, comprised of two main components—Database Control and Grid Control. Database Control operates from a single server as a Java process and allows the DBA to manage every aspect of a single database, including storage, object manipulation, security, and performance. Grid Control operates with the same GUI interface, but requires the installation of the Enterprise Manager product on a centralized server. From this central instance of Grid Control, a DBA can manage all the databases to which Grid Control connects, providing the DBA with a web-based interface to the entire environment. **SQL Worksheet**, a link within Database/ Grid Control, provides a basic SQL interface to a database.. The license for both Grid Control and Database Control is included in the license for the Enterprise edition of Oracle, although many of the performance tuning and configuration management features must be separately purchased. A screenshot of SQL Worksheet is shown as follows:

SQL Worksheet : orcl

Enter a SQL statement to execute. If there are multiple statements, the location of the cursor or a highlighted statement determines which will be executed. Statements should be separated with blank lines.

SQL Commands

```
select * from jobs;
```

☐ Use bind variables for execution
☐ Auto commit
☒ Allow only SELECT statements
Format Execute

Last Executed SQL

```
SELECT * FROM jobs
```

Last Execution Details

SQL Repair Advisor SQL Details Schedule SQL Tuning Advisor

Results Statistics Plan

Execution Time (seconds) 0.01

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20080	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
FI_ACCOUNT	Accountant	4200	9000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20080
SA_REP	Sales Representative	6000	12008
PU_MAN	Purchasing Manager	8000	15000
PU_CLERK	Purchasing Clerk	2500	5500
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2008	5000
SH_CLERK	Shipping Clerk	2500	5500
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000
HR_REP	Human Resources Representative	4000	9000
PR_REP	Public Relations Representative	4500	10500

SQL Repair Advisor SQL Details Schedule SQL Tuning Advisor

Related Link

[SQL Worksheet Session Details](#)

Copyright © 1996, 2010, Oracle. All rights reserved.
Oracle, iD Edwards, PeopleSoft, and Relex are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book

PL/SQL Developer

PL/SQL Developer is a full-featured SQL development tool from Allround Automations. Along with many of the other features common to SQL development tools, such as saved connections, data exporting, and table comparisons, PL/SQL Developer places a strong focus on the coding environment. It offers an extensive code editor with an integrated debugger, syntax highlighting, and a code hierarchy that is especially beneficial when working with the PL/SQL language. It also includes a **code beautifier** that formats your code using user-defined rules. PL/SQL Developer can be purchased from Allround Automations or downloaded from their website as a fully-functional, 30-day trial version.

Oracle SQL Developer

Oracle SQL Developer, originally called **Raptor**, is a GUI database interface that takes a somewhat different approach from its competitors. While many of the major licensable GUI administration products have continued to expand their product offering through more and more add-on components, SQL Developer is a much more dedicated tool. It's a streamlined SQL interface to the Oracle database. You can create and manipulate database objects in the GUI interface as well as write and execute SQL statements from a command line. Administration-oriented activities such as storage control are left to Enterprise Manager. SQL Developer aims to be a strong SQL and PL/SQL editor with some GUI functionalities. SQL Developer has gained in popularity in recent years, in large part to several benefits that are listed as follows:

- It is completely free with no mandatory licensable components, although third-party add-ons are available for purchase.
- It is a true cross-platform client-side tool written primarily in Java. While a majority of the commonly-used SQL tools are available only on the Windows platform, SQL Developer runs on Windows, Linux, and even the Mac.
- In many Oracle shops, DBAs have been uncomfortable with the idea of giving developers a tool that can be used to cause massive damage to the database. Because Oracle has separated out most of the administration functions from SQL Developer, it is more of a true development tool.
- SQL Developer supports read-only connections to many popular databases, including SQL Server, Sybase, MySQL, Microsoft Access, DB2, and Teradata.
- Because it is written in Java, it allows for the creation and addition of third-party extensions. If you want a capability that SQL Developer does not have, you can write your own!

- It is provided by Oracle and is now included with any installation of Oracle database. It has essentially replaced SQL*Plus as Oracle's default SQL interface, although SQL*Plus is still available from the command line.

For these reasons, the tool we use in this book for the purposes of demonstration will be SQL Developer. Instructions for downloading the tool are in the foreword. But, before we look at SQL Developer, let's find out a little about the data we'll be using and look at the `Companylink` database.

Working with SQL

Often, the best way to learn something is hands-on. To best facilitate this, we will use a scaled-down set of data that mirrors the type of data used in the real world.

Introducing the Companylink database

This book focuses on two objectives:

1. To prepare you for the 11g SQL Fundamentals I exam (Oracle exam #1Z0-051).
2. To present the knowledge needed for the exam in such a way that you can use it in a real-world setting.

To that end, rather than using the default tables included in Oracle, we will be working with simulated real-world data. The database we will use throughout this book is for the fictional company, `Companylink`. Although most people are aware of the impact of social networking in our private lives, companies are realizing the importance of using it in their industries as well. `Companylink` is a business that focuses on social networking in the corporate setting. The data model that we will use is a small but realistic set of working data that could support a social networking website. The following tables are included in the `Companylink` database, which can be downloaded from Packt support site as well as comments about the business rules that constrain them:

- **Employee:** Information about employees that use the `Companylink` site.
- **Address:** The street address information.
- **Branch:** The corporate branch to which each employee belongs. Each employee belongs to one branch.
- **Division:** It is the corporate division to which each branch belongs. Each division is associated with multiple branches.
- **E-mail:** An employee can store multiple e-mail addresses.

- **Message:** Our fictional Companylink social networking site allows you to send messages to fellow employees. That information is stored here.
- **Website:** Companylink allows users to create their own personal web pages. The URL of these pages is contained in this table.
- **Blog:** In addition to a website, users can optionally create their own blogs. This information is stored in the `BLOG` table.
- **Project:** Each employee is assigned to a single primary project, which is contained here.
- **Award:** Employees can win corporate awards. The list of possible awards is stored here. Employees can win more than one award.
- **Employee_award:** This table is used to relate employees with their awards. Since multiple employees can win the same award and multiple awards can be won by the same employee, this creates a many-to-many table relationship, which, in the relational paradigm, must be avoided. The `employee_award` table divides this many-to-many relationship into two distinct one-to-many relationships.

To create our database, we need to run the downloaded Windows command file. Simply unzip the `companylink.zip` file into a directory and double-click on the `companylink_db.cmd` file. The execution of the file will do the following:

- Make a connection to the database
- Create a user called `companylink` with the password `companylink`
- Create the tables used for the examples in this book
- Populate these tables with data
- Output two log files: `companylink_ddl.txt` and `companylink_data.txt`

If you wish, the log files can be used to verify successful execution of the script. The command file is completely reusable, which is to say that if you break any of the tables or data, you only need to disconnect from the database and double-click the command file again. It will drop the existing data and rebuild the tables from scratch. When you do this, keep in mind that any data you add yourself will be deleted as well. Throughout the book, we will continually be writing SQL statements that access these tables and will even add new ones.

The creation of these tables requires a working installation of the Oracle database software on a machine to which you have access. Fortunately, the Oracle software can be downloaded from <http://www.oracle.com/technology>. There is no purchased license required if you use the software for your own learning purposes.

SQL in the real world



When you're starting out with SQL and Oracle, it's important to get hands-on. Although Oracle makes its software available at no charge for personal use, many aspiring DBAs are hesitant to install it on their personal computers. By using free desktop virtualization software, such as Virtualbox, you can create a virtual machine on your home computer that can be used as your self-contained database server. Whenever you want to work with Oracle, simply start your virtual machine. Whenever you finish, shutdown the virtual machine, and all the resources it used will be released. Virtualization can be a useful solution to isolate your Oracle work from your home use without buying another computer.

An introduction to Oracle SQL Developer

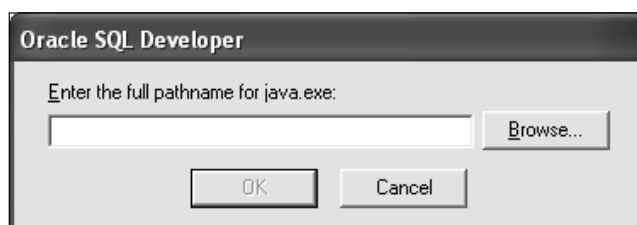
Since SQL Developer is our SQL tool of choice, it's important that we get a good feel for it right from the beginning. In this section, we learn about configuring and running SQL Developer.

Setting up SQL Developer

Let's get started with SQL Developer. If you have Oracle installed, you can launch SQL Developer in Windows XP from the **Start** menu, as shown next:

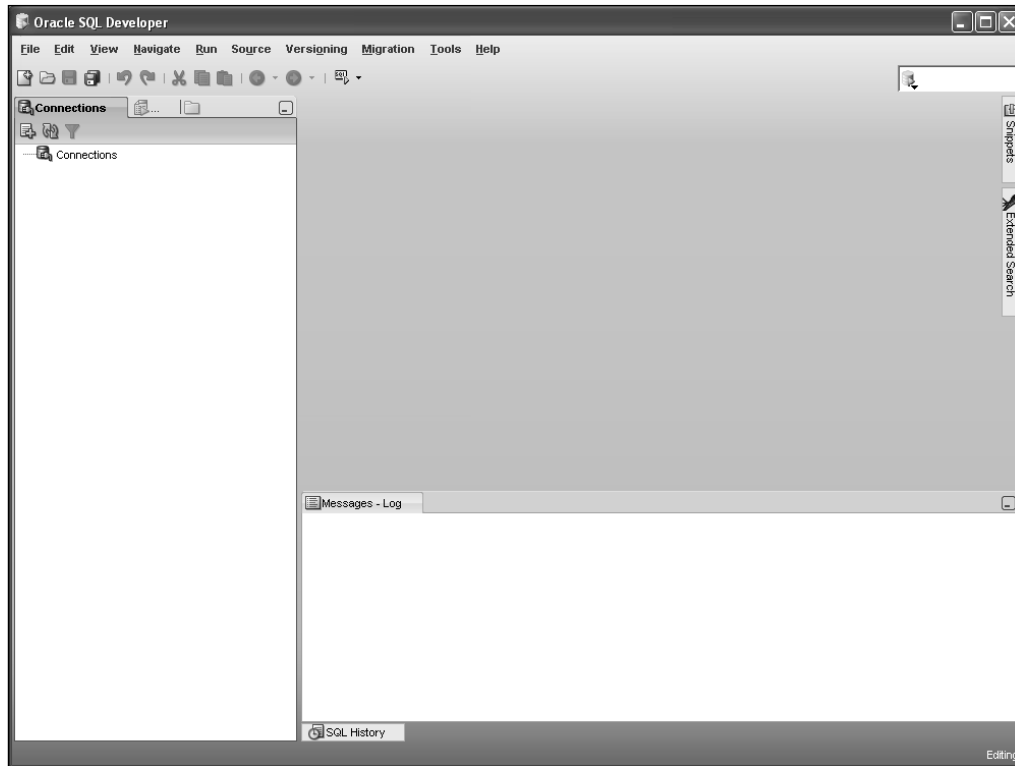
Start | All Programs | Oracle<program group> | Application Development | SQL Developer

SQL Developer runs as a Java application, so it may take a little while to load. The first time you start the application, you may get a message box, such as the one shown in the following screenshot:




If this happens, click on **Browse** and navigate to the `java.exe` file. You do not need a separate installation of Java to run SQL Developer; one is included in the Oracle installation. If you don't know where the `java.exe` is located, simply go to the Oracle installation directory and do a search for `java.exe`. Then, navigate to that path and select it.

Once startup has completed, you will see a **Tip of the Day** screen. Close it and you will be presented with the following screen. It's worth noting that this screen will look the same, irrespective of whether you're running SQL Developer under Windows, Linux, or the Mac OS, due to its cross-platform, Java-based nature.



On the left side, you see a list of connections to databases. At this point, there will be no connections, since we have not created any yet. SQL Developer allows you to maintain multiple connections to various databases. Each one can use any variation of different login names, servers, or database names.



SQL in the real world

In the real world, DBAs and developers run SQL Developer from their desktops and use it to connect to remote databases. Thus, their working environments can run locally, but the databases they connect to can be anywhere in the world!

Before we can use SQL, we need to connect to a database. To do that, we need to create a database connection. Any connection to an Oracle database consists of three pieces of information:

1. The hostname or IP address of the machine to which we're connecting.
2. The port number on which Oracle operates.
3. The name of the database to which we connect.

To set up our connection, we need to click on the **New Connection** button at the top of the left-hand connection frame. This action brings up the **New / Select Database Connection** window. We fill in the information, as listed in the following screenshot. This example connection assumes that you have set up an Oracle database using the standard installation procedure with common defaults. If you're connecting to an existing database, the information you enter will be different:

The screenshot shows the 'New / Select Database Connection' dialog box. It has a title bar with a close button. On the left, there are two tabs: 'Connection N...' and 'Connection D...'. The main area is divided into two sections. The top section contains fields for 'Connection Name' (companylink@orcl), 'Username' (companylink), and 'Password' (masked with asterisks). Below these is a checked checkbox for 'Save Password'. The bottom section is titled 'Oracle' and contains a sub-tab 'Access'. Under 'Access', there are fields for 'Role' (default), 'Connection Type' (Basic), 'OS Authentication' (unchecked), 'Kerberos Authentication' (unchecked), and 'Proxy Connection' (unchecked). Below these are fields for 'Hostname' (myserver), 'Port' (1521), and 'SID' (orcl). The 'Service name' field is empty. At the bottom, there is a 'Status' field and a row of buttons: 'Help', 'Save', 'Clear', 'Test', 'Connect', and 'Cancel'.

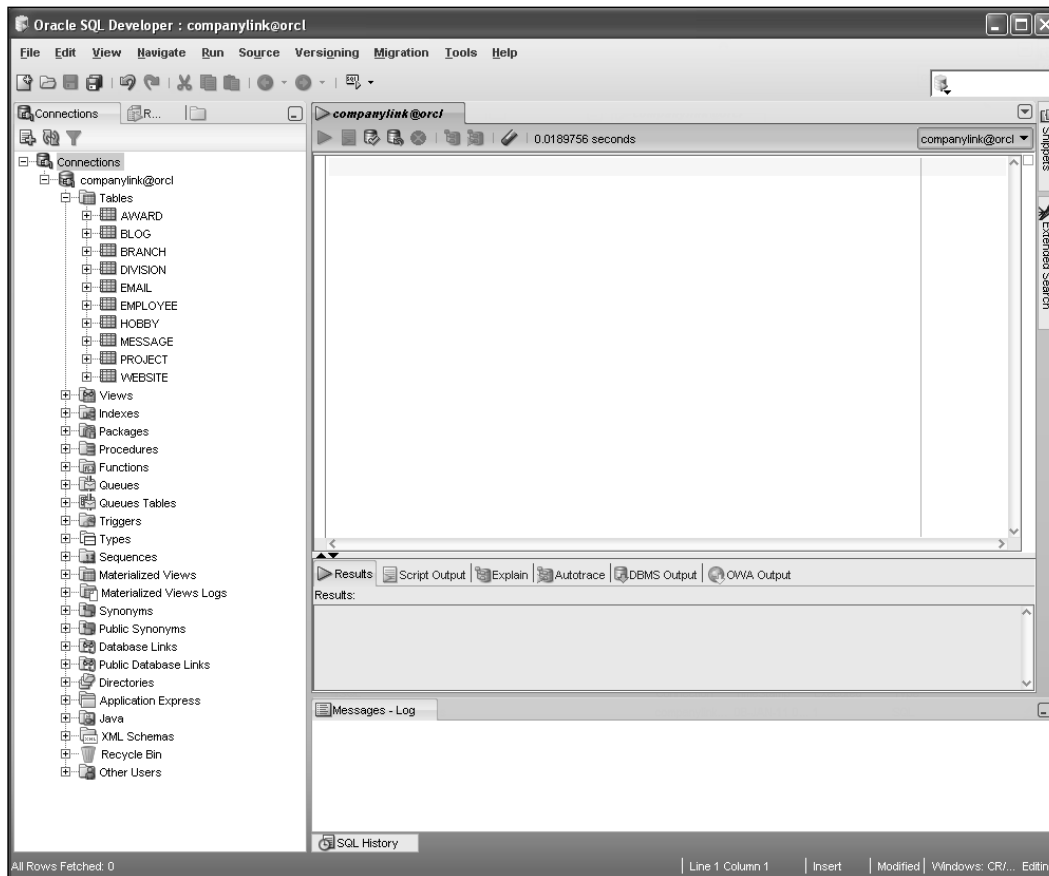
The pieces of information that are relevant to us are as follows:

- **Connection name:** This can be whatever we choose, but it is usually a good idea to make it descriptive of the connection itself. In our example, we choose `companylink@orcl` because it denotes that we are connecting to the `orcl` database as the `companylink` user.
- **Username:** The name of the user we connect as.
- **Password:** The password for the user. The password for our user is `companylink` (non-case sensitive)
- **Save Password:** Select this checkbox to ensure that you don't have to re-enter the password each time you initiate the connection.
- **Hostname:** This will be either the hostname or the IP address of the server that hosts our target database. The example used, `myserver`, will most likely not be the name of your server. Change this to the name relevant to your situation.
- **Port:** This will be the port number that Oracle is running from. Most Oracle databases run from port 1521, although some DBAs change this for security reasons. If you installed Oracle using the default settings, your port number will be 1521.
- **SID:** The SID is the System Identifier for your database, which is the name of the database. In a typical installation of Oracle, the default SID used is `orcl`.

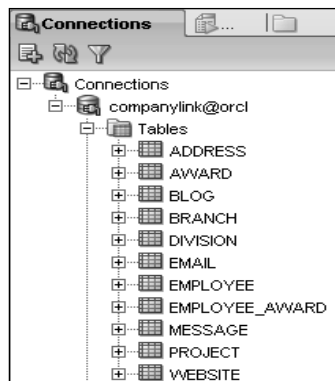
Once the relevant information has been entered, it is always a good idea to click on the **Test** button at the bottom of the window to ensure a connection can be made. If all the information is correct, you should see **Status: Success** on the lower left-hand side of the window. Once we have verified that we can successfully connect, we click on the **Connect** button. Our connection is saved in the **Connections** frame, on the left side of the window, and our connection is established.

Getting around in SQL Developer

Now that we're connected, let's take a look at what SQL Developer has to offer. Click on the plus sign (+) next to your new connection:



On the left side, indented under our connection, is a list of database objects, including **Tables**, **Views**, and **Indexes**. Discussion about some of the other sets of objects is outside the scope of this book, but all are accessible by simply expanding the object group using the plus sign next to it. Click on the + next to **Tables** and your list of tables will be expanded. Your window should now look something similar to what is shown here. These are the tables created, and therefore owned, by the user with whose profile we logged in; in this case, `companylink`. They were created by running `companylink.bat`, earlier in the chapter. The following screenshot shows a list of our `Companylink` tables:



These tables can be expanded to view their characteristics, such as column names and datatypes, but most of this book will focus on how to view and modify tables using only the SQL language instead of GUI tools.

The large portion of the window in the upper right is our SQL working area. This frame will be the area in which we write SQL code. To write SQL in the working area, simply click in the area and begin typing your SQL statements. When you are finished, click on the green arrow in the working area toolbar to execute the statement. Alternatively, you can press `F9` on your keyboard.

Directly below the working area is the **Results** frame. This is the area where we will see the results of our SQL queries. The results will display in columnar format, and the columns can be resized by clicking-and-dragging. The **Results** frame also has several tabs across the top for various other functions, but, for now, we will not concern ourselves with them. Let's try a query and view the output. In the working area, type the SQL query you see in the following screenshot, **select * from employee**, and click on the green execute arrow:

The screenshot shows the Oracle SQL Developer interface. At the top, the title bar reads 'companylink@orcl'. Below it, a toolbar contains icons for running, saving, and other actions, along with a timer showing '0.01024572 seconds'. The main working area contains the SQL query: `select * from employee;`. Below the working area is the **Results** frame, which has tabs for 'Results', 'Script Output', 'Explain', 'Autotrace', 'DBMS Output', and 'OWA Output'. The 'Results' tab is active, displaying a table of 16 rows and 10 columns. The columns are: EMPLOYEE_ID, FIRST_NAME, MIDDLE_INITIAL, LAST_NAME, GENDER, SIGNUP_DATE, DOB, BRANCH_ID, and PROJECT_ID. The data is as follows:

	EMPLOYEE_ID	FIRST_NAME	MIDDLE_INITIAL	LAST_NAME	GENDER	SIGNUP_DATE	DOB	BRANCH_ID	PROJECT_ID
1	1	James	R	Johnson	M	01-JAN-10	01-JAN-60	6	2
2	2	Mary	S	Williams	F	28-JAN-10	15-MAR-64	2	1
3	3	Linda	L	Anderson	F	01-APR-10	24-OCT-70	1	3
4	4	Daniel	J	Robinson	M	23-FEB-10	23-NOV-59	2	1
5	5	Matthew	K	Garcia	M	24-JUN-10	14-APR-71	3	3
6	6	Helen	H	Harris	F	25-JAN-10	13-JUL-75	10	(null)
7	7	Ken	W	White	M	10-FEB-10	22-FEB-58	4	(null)
8	8	Donald	A	Perez	M	17-OCT-10	14-MAR-79	1	3
9	9	Lisa	C	Lee	F	20-SEP-10	15-JUN-63	5	(null)
10	10	Carol	M	Clark	F	01-APR-10	11-AUG-67	2	1
11	11	Gary	R	Moore	M	06-AUG-10	01-NOV-65	3	5
12	12	Cynthia	B	Hall	F	03-JAN-10	21-OCT-55	3	5
13	13	Sandra	S	Rodriguez	F	04-MAR-10	10-MAY-74	13	(null)
14	14	Kevin	L	Lewis	M	09-MAR-10	01-JUL-76	8	4
15	15	George	H	Taylor	M	06-OCT-10	24-DEC-72	12	4
16	16	Laura	I	Thomas	F	07-NOV-10	26-OCT-81	12	5

As we will learn in the next chapter, the SQL query we've placed in the working area uses a wildcard character, '*', to display all the columns and rows from the table called employee. As you can see, this data displays in the **Results** frame, which is listed in columnar format. You have just made your first use of the Structured Query Language.

Below the **Results** frame is the **Messages (Log)** frame. It is used to display the output of certain operations and is not relevant to our concerns. To maximize the areas for the working area and **Results** frame, you can click-and-hold the bar above the **Messages** frame and drag it downward to make it invisible. Similarly, you can click-and-drag the bar between the working area and **Results** frames to change the ratio of space between the two. Many users like to make as much of the **Results** frame visible as possible so as to see more of the resultant data.

The last area we need to point out is the **SQL History** tab just below the **Messages** frame. This tab, when clicked, displays a pop-up of the most recent SQL statements. This can be very useful when trying to remember previous statements. Simply click on the tab, then double-click the statement you want to run, and it will be pasted in the working area. You can then select it and click on **Execute** to run it.

SQL Developer offers a tremendous number of other features that are beyond the scope of this book. If you're interested in more information on SQL Developer, visit <http://www.oracle.com/technology> and view the documentation for it.

Summary

In this chapter, we've gone from the early days of databases to the relational databases that are so prolific today. We've explored the concept of normalization and how it's applied to the relational paradigm. We've looked at tables and how they are structured and introduced the Structured Query Language for relational databases. We've also examined some of the popular SQL tools and created the tables needed for the Companylink database. Finally, we've worked our way around the SQL Developer tool and learned the basics of how to execute queries.

Now that we've learned about relational databases and SQL, we're ready to begin writing SQL statements – the topic of the next chapter.

Test your knowledge

1. What relational term is used to denote any person, place, or thing?
 - a. Attribute
 - b. Entity
 - c. Flat file
 - d. Repeating group
2. What is the name of the process used to transform non-relational data into relational data?
 - a. Normalization
 - b. Transformation
 - c. Repudiation
 - d. Object-oriented

3. A _____ uniquely identifies any single row of data.
 - a. Foreign key
 - b. Attribute
 - c. Primary key
 - d. Column
4. Which of these is NOT a form of entity relationship?
 - a. One-to-many
 - b. One-to-one
 - c. Variant-to-one
 - d. Many-to-many
5. What is the visual representation of a data model called?
 - a. A table
 - b. An entity
 - c. Normalization
 - d. An entity relationship diagram
6. Which of these is NOT required to make a database connection?
 - a. Port number
 - b. Table name
 - c. Database name
 - d. Hostname/IP Address

Where to buy this book

You can buy OCA Oracle Database 11g: SQL Fundamentals I: A Real-World Certification Guide from the Packt Publishing website:

<http://www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book>.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our [shipping policy](#).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



www.PacktPub.com

For More Information:

www.packtpub.com/oca-oracle-database-11g-sql-fundamentals/book