CHAPTER 1

Introducing Database Design and Oracle SQL Developer Data Modeler
Database design is the process of producing detailed entity-relationship (ER) diagrams and data flow diagrams (DFDs) in order to produce the data definition language (DDL) scripts that will create the objects needed for the database. Database design consists of requirements analysis, conceptual design, logical design, physical design, and, depending on who you ask, transaction design. (This book will not discuss transaction design.) The process is incremental and iterative, meaning all these phases will be done repeatedly. The backbones of database design are logic theory and relational theory.

Database design is all about the data, namely, how to save the data and how to retrieve it. Data integrity and data quality should always be high priorities when designing a database, and you must consider future needs as well. Even though an application user interface might change every five to ten years, the database behind it must continue to perform well for years to come.

The process of database design is changing as application development processes are getting more agile and iterative. Management demands fast results, so IT projects must be completed faster than ever before. Database designers often do not have time to analyze everything well before starting to design, and sometimes systems are launched into production to be completed later in increments, without having the analysis completed. In fact, sometimes databases are created with no time spent on design and with no thought to the principles of relational theory. Even with the world seemingly getting faster every day, when designing a database, you need to know the full picture of what the database is for. That is what makes database design difficult. The only way to survive is to use a tool that meets all of today’s needs, helping you create databases quickly but with the “big picture” in mind. Without a tool, you cannot be as agile as needed.

Though I’ve mentioned application development processes, I want to be clear that database design is not the same as application design. The database should not be designed as a side product of an application design. When trying to save time and money, people think they will design only either the ER model or the Unified Modeling Language (UML) model and then generate the other. Although it is good they realize they need two models (one for the database and the other for the application), it is not just a question of which notation to choose; the perspectives are very different and so are the goals. For example, let’s look at code tables versus code files. For the application designer, it might be easier to have all the lookup information in files, but the database designer definitely will want them in tables. Why? The database person is also in charge of the data integrity, which cannot be controlled if some of the important data is in files somewhere out of the reach of the database.

I often hear people arguing about which is better, ER or UML. To me this question is irrelevant. ER diagrams are for designing databases, and UML is for designing user interfaces. If you try to design a database with UML, you can get easily distracted and want to start designing the user interface. My recommendation
Chapter 1: Introducing Database Design and Oracle SQL Developer Data Modeler

is that while the database designer designs the database, the application designer designs the application in cooperation with the database designer. And before the database designer moves to the logical design, the database designer and the application designer should sit down and compare their models to be sure that they really have all the requirements implemented in both the designs. There might be information in the UML model that the ER model does not have or should not have. For instance, in the UML model, there might be an attribute named AGE, but in the data model (ER), there might be an attribute called DATE OF BIRTH. There can also be some technical attributes in the data model that do not need to be in the UML model. For instance, every table might have the columns Creator, Created_Date, Modifier, and Modified_Date. The two models (UML and ER) do not have to be the same and actually rarely are, but creating and maintaining both models will guarantee a better result. But this is true only if you have a tool for both purposes; if this work is done without a tool, the dual processes take too much time and money. You want to use a tool to create designs in cooperation and take advantage of everybody's special skills and knowledge.

NOTE

The UML model and the ER model do not need to be the same and rarely are.

The importance of database design increases on agile projects. In that case, the process involves not just designing but also finding the right questions to ask and having pictures (ER models and DFDs) to use when talking to end users. You need as much information as possible from end users and business owners. You need to understand the big picture, lest you get a database totally different than you wanted. It's as simple as thinking before doing.

It's important that you understand the main concepts (entities) of the database and their relationships correctly because it is easy to add entities and attributes later, but it is not easy to divide them later or correct the relationships modeled wrongly. Always design the database for the right purpose and model only what is needed, starting with the most difficult task.

In Figure 1-1, you can see my version of agile database design. It starts with requirements analysis and finding the main concepts and their relationships. Next you try to model the whole conceptual model as well as you can. Then you design the conceptual model for iteration \( n \), making it as detailed as possible; continue to the logical design of iteration \( n \); and finally move to physical design and creating the database objects with the DDL scripts. Then you perform the whole round again for iteration \( n+1 \), and so on. The process is the same as it is in other projects; the only difference in an agile project is that you move from phase to phase faster, and you design in pieces, rather than as a whole.
If a database will hold valuable data, the database must be designed by someone who understands how the database works and knows how the data should be modeled. When designing the database, you may need all different types of subject-matter experts to give you the information you need to make decisions about the design. Additionally, if your deadline is tight, you need even more information to be sure you are making the right decisions; you don’t want to have to change everything later. Prioritization is important for everybody (even the end users). Nothing is more stressful than too much work with too little time to do it. Database design means teamwork, and that’s why you need a tool to do database design right in today’s environment.
Chapter 1: Introducing Database Design and Oracle SQL Developer Data Modeler

When selecting the tool for database design, you’ll want one that supports these features: ER notation, an automatic transformation process from the conceptual design to the logical design, the ability to work in a multiuser environment, version control, reporting capabilities, scripts for generating the database objects automatically (preferably adjustable), and strong documentation tools. It would be a bonus if the tool also has support for the standardization of naming, processes, and design rules; the ability to alter scripts for changing the database to be like the design; and the ability to compare designs to each other and to compare a design to a database.

What Is Oracle SQL Developer Data Modeler?

Oracle SQL Developer Data Modeler (referred to as Data Modeler in this book) is a free tool for designing and documenting databases and data architecture. It supports not only Oracle databases but also DB2 and Microsoft SQL Server databases and, at a certain level, any standards-based database that has a Java Database Connectivity (JDBC) connector. Data Modeler supports all the steps in database design and includes easy forward and backward engineering. After you have designed your database and have a physical model for it, you can export the scripts to create the database objects. Data Modeler also supports different kinds of compares and multidimensional models. Data Modeler helps you keep your databases documented and enables you to be agile. The tool is available as a stand-alone product, but it is also integrated into Oracle SQL Developer, so you can decide which way is the best for you to use the tool. Installing the tool is simple, and support is provided by Oracle if you have a database support contract.

Data Modeler offers the following features for database designers:

- **Database design tools** A collection of metadata about a database is called a *design* in Data Modeler. A design consists of the logical models, multidimensional models, relational models, domains, data type models, process models, business information, and change requests, as well as all the objects those models need. Every object (entity, table, diagram, and so on) is a single Extensible Markup Language (XML) file in a hierarchy that the tool creates automatically. The design itself is saved with the extension .dmd, and the .dmd file contains pointers to individual XML files.

- **Customization** You can tweak Data Modeler to your liking. In Preferences, you can, for instance, define where to keep your working copy of designs.

- **Version control** Data Modeler is integrated into a version control tool called Subversion. This integration allows you to have multiple users...
changing the model at the same time. It also gives you version control functionalities. When working with version control, the latest official version of your design is always on version control, and the one you are working with is in your local saved working copy directory.

- **Documenting existing databases**  You can import designs to Data Modeler from existing databases, from other designing tools (for example, Oracle Designer or ERwin), or from DDL scripts.

- **Reporting capabilities**  Data Modeler has built-in reporting functionalities, but you can also create your own reports and templates and use the Search functionality as a base for a report. It is also possible to use a reporting repository if you want to have reports across all your designs and use SQL to query that information. You can also print the design layouts.

- **Documentation tools, improving quality and efficiency**  Data Modeler helps you standardize the design and data documentation in your company. You can use naming standards, domains, glossaries, and design rules to achieve better quality in your database design. You can also compare models and designs to each other, and you can compare a design to a database. Different compares, transformations, and notations will give you a more cost-efficient working environment with better quality.

## Designing Databases with Oracle SQL Developer Data Modeler

The database design process when using Data Modeler starts with designing a logical model. In the logical model, you define entities, attributes, and relationships. The next step is to create a relational model based on the logical model. You do this simply by clicking the Engineer To Relational Model icon. When you are done with the relational model, it is time to create the physical model. You do this simply by right-clicking Physical Model in the Browser pane and selecting New. When creating a physical model, you must know what product you will use for your database (Oracle, SQL Server, or DB2) as well as its version. All the properties for the physical model depend on the chosen technology. After you have created a physical model, you should define the properties for the physical objects. After you’ve done that, you are ready to generate the DDLs (which are the SQL scripts for creating your database objects). You can create DDLs by selecting File | Export | DDL File. Then just run these DDLs on your database to create the objects. And all this can be done in a multiuser environment and while using version control.
Chapter 1: Introducing Database Design and Oracle SQL Developer Data Modeler

You can also use Data Modeler to document existing databases (Oracle, SQL Server, DB2). You can reverse engineer the documentation from a data dictionary or existing DDLs, or you can import it from another design tool (Oracle Designer, ERwin, or a VAR file). Or, you can combine all these techniques, for example, by bringing some of the descriptions from another design tool and adding it to the physical information from the data dictionary. You can find these features by selecting File | Import.

Since an important part of database design is reporting, you might want to use Data Modeler to create your own templates and create reports based on those templates. You can also create a reporting repository; in addition to the templates, you can use SQL to query the information from there. You can also print the diagrams or use the powerful search functionality to search the information in the report.

It is also important to be able to document all the information related to the database in just one place. In Data Modeler you can document all the information needed for the database design as well as change requests, business information, and much more.

Summary

Going through the database design process is vital if you are storing important data in your database. Database design is the process of producing detailed entity-relationship diagrams and data flow diagrams to produce the DDL scripts for creating the objects for the database. Database design consists of requirements analysis, conceptual design, logical design, physical design, and, depending on who you ask, transaction design. To be able to design a database, especially in an agile system, you need Oracle SQL Developer Data Modeler. It is a free tool that supports all the needs of database designers plus some extra.