LESSON 6

98-361 Software Development Fundamentals

6.1 Understand Relational Database Management Systems

6.2 Understand Database Query Methods

6.3 Understand Database Connection Methods

MTA Software Fundamentals 6 Test

LESSON 6.1

98-361 Software Development Fundamentals

Understand Relational Database Management Systems

Lesson Overview

In this lesson, you will learn about:

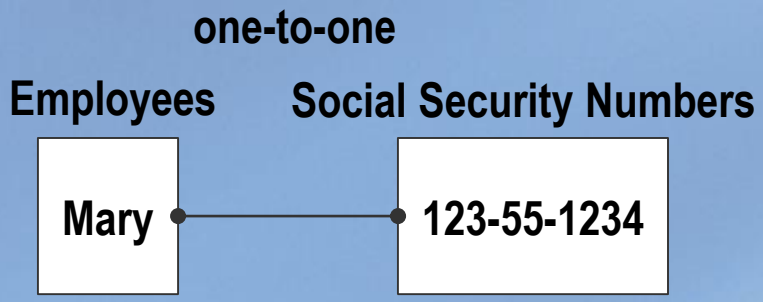
- The characteristics and capabilities of database products
- Database design
- Entity Relationship Diagrams (ERDs)
- Normalization concepts

Characteristics and Capabilities of a Relational Database

- Captures and stores data efficiently
- Reduces data redundancy
- Decreases application processing time by storing data more efficiently
- Provides functions to retrieve, process, and report data

Entity Relationship Diagrams (ERDs)

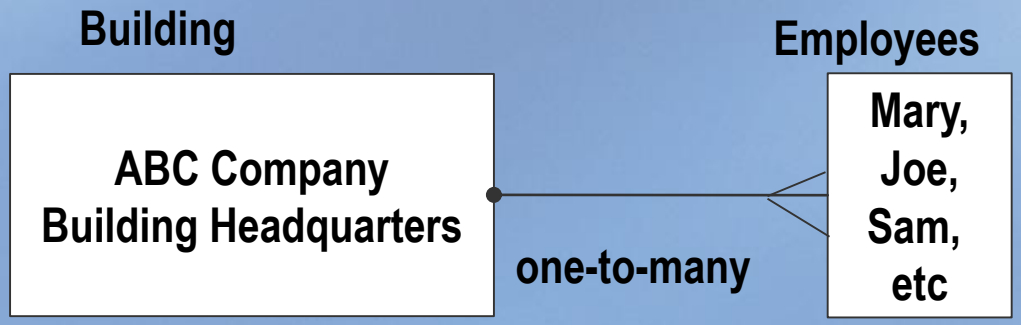
- An entity relationship diagram (ERD) is a graphical representation of the entities, or groups of information, and their relationships.
- There are three types of relationships between entities:
  - One-to-one: one instance of an entity (A) is associated with one other instance of another entity (B). This is an association between two tables in which the primary key value of each record in the primary table corresponds to the value in the matching field or fields of one, and only one, record in the related table.
    - For example, in a database of employees, each employee name (A) is associated with only one Social Security number (B).



Entity Relationship Diagrams (continued)

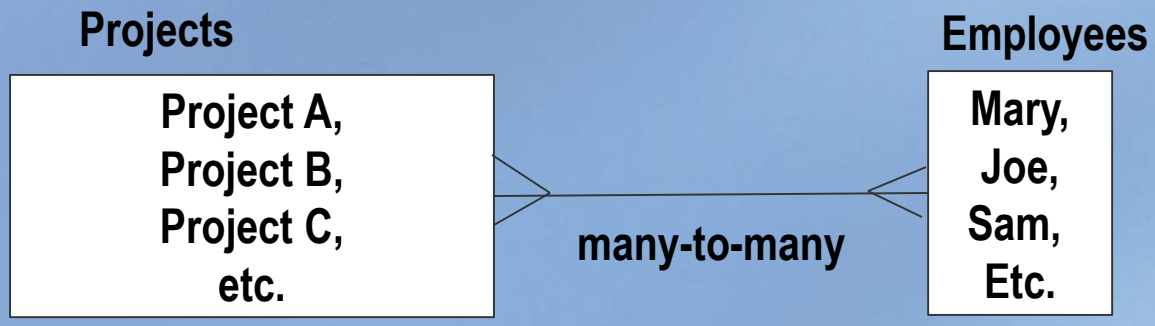
One-to-many: one instance of an entity (A) is associated with zero, one, or many instances of another entity (B), but for one instance of entity B, there is only one instance of entity A.

- For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



Entity Relationship Diagrams (continued)

- Many-to-many: one instance of an entity (A) is associated with one, zero, or many instances of another entity (B), and one instance of entity B is associated with one, zero, or many instances of entity A. This is a complex association between two sets of parameters in which many parameters of each set can relate to many others in the second set.
  - For example, for a company in which all its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



Normalization Concepts

- Used to eliminate redundant data and ensure data dependencies make sense
- The Normal Forms:
  - — The First Normal Form (1NF) eliminates duplicate columns from the same table, creates separate tables for each group of related data, and identifies each row with a unique column or set of columns (the primary key).
  - — The Second Normal Form (2NF) addresses the concepts of removing duplicate data and subsets of data that apply to multiple rows of a table, and placing them in separate tables. It creates relationships between these new tables and their predecessors through the use of foreign keys.
  - — The Third Normal Form (3NF) goes one step further, meeting all requirements for the 2NF and then also removing columns that are not dependent upon the primary key.
  - — The Fourth and Fifth Normal Forms are rarely used.

Keys

- — A primary key defines one or more columns that *uniquely* identify each row in the table.
- — Relationships (or links) between tables are stored as foreign key constraints. A foreign key from one table refers to a primary key in another table; that is, the foreign key, ZIP code, in a demographic table is the primary key in the ZIP code table.

LESSON 6.1

98-361 Software Development Fundamentals

Student Lab 6.1

Microsoft

LESSON 6.2

98-361 Software Development Fundamentals

Understand Database Query Methods

Lesson Overview

In this lesson, you will learn about:

- Structured Query Language (SQL)
- Creating and accessing stored procedures
- Selecting data
- Updating data

Structured Query Language (SQL)

- What is a query?
  - — A specific set of instructions for extracting particular data.
  - — A query is a question to the tables in a database. The syntax of the question depends on the database language.
- SQL commands can be used interactively as a query language or they can be embedded in application programs.

Creating and Accessing Stored Procedures

- A stored procedure is a saved collection of SQL statements or a reference to a Microsoft .NET Framework common language runtime (CLR) method that can take and return user-supplied parameters.
- Procedures can be created for permanent use or for temporary use within a session (called a *local temporary procedure*), or for temporary use within all sessions (called a *global temporary procedure*).

Selecting Data

- Data can be selected from a single table or multiple tables.
- To access data from a single table, the common command is:
  - `SELECT column(s) FROM table`

  (*Note:* This `SELECT` statement does not eliminate duplicates automatically. To eliminate duplicates, you must use the keyword `DISTINCT`.)

  - To add qualifiers to the selection statement, we can add the `WHERE` clause.

Examples of Selecting Data

Precondition: the following database tables are already created and contain data: `STUDENT`, `ENROLLMENT`, and `CLASS`.

The following statement selects all the majors for every student:

```
SELECT major FROM STUDENT
```

Note: This would include duplicates.

The second example selects data from two tables using a subquery to retrieve the name of all students enrolled in Bio 123):

```
SELECT name FROM STUDENT WHERE
STUDENTID in (SELECT studentNumber FROM ENROLLMENT
   WHERE className = "BIO123")
```

Updating Data

- Updating data includes inserting rows, deleting rows, and modifying existing rows.

To insert data:

```
INSERT INTO table VALUES(value1, value2, value3)
```

or to achieve a partial data insert:

```
INSERT INTO table VALUES(value2, value3)
```

To delete data:

```
DELETE table WHERE table.column = value
```

Note: Beware of potential referential integrity problems; that is, deleting rows that other items refer to.

To modify data:

```
UPDATE table SET column= value WHERE column = value
```

Examples of Updating Data

(Note: The same tables are used as earlier in this presentation.)

```
INSERT INTO ENROLLMENT
  VALUES(400, "BIO123",44)
DELETE STUDENT WHERE
  STUDENT.STUDENTID=100
UPDATE ENROLLMENT
  SET positionNumber = 44
  WHERE STUDENTID = 400
```

LESSON 6.2

98-361 Software Development Fundamentals

Student Lab 6.2

LESSON 6.3

98-361 Software Development Fundamentals

Understand Database Connection Methods

Lesson Overview

In this lesson, you will learn about:

- Connecting to various types of data stores such as:
  - Flat files
  - Extensible Markup Language (XML) files
  - In-memory objects
  - Resource optimization

General Information

- To bring data into your application (and send changes back to the data source), some kind of two-way communication needs to be established.
- This two-way communication is typically handled by a connection object (for example, a *SqlConnection*) that is configured with a *connection string*, the information that it needs to connect to the data source.
- The next few slides will discuss specific file types and their uses.

Relational Databases

- Although there are different ways to organize data in a database, a relational database is one of the most effective. In a relational database, data is collected into tables.
- A table represents a class of objects that are important to an organization. For example, a company may have a database with a table for employees, another table for customers, and another for stores. Each table is made of columns and rows.
- When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process called normalization, which ensures that the set of tables you define will organize your data effectively.

How to connect to a flat file

- A flat file database describes any of various means to encode a database model (most commonly a table) as a singular file (such as .txt or .ini).
- A flat file is a file that has no repeating groups. To access these files, you must use an OLE DB to establish a connection between your application and the flat file.
- OLE DB (*Object Linking and Embedding, Database*, sometimes written as OLEDB or OLE-DB) is an application programming interface (API) designed by Microsoft for accessing different types of data stored in a uniform manner.
- OLE DB was designed to access both flat files and database files. It provides an object-oriented interface to data of almost any type.
- OLE DB acts as an interface to access a relational database through Open Database Connectivity (ODBC).

How to connect to an XML file using OLE DB

- XML (Extensible Markup Language) is the emerging Internet standard for data. XML consists of a set of tags that can be used to define the structure of a hypertext document. XML documents can be easily processed by the Hypertext Markup Language (HTML), which is the most important language for displaying Web pages.
- An XML file is typically used for Web design applications.
- There are two separate methods of retrieving XML data from a data source: one uses *CStreamRowset* and the other uses *CXMLAccessor*.

Examples of connections

- ActiveX Data Objects (ADO) is a language-neutral object model that expose data raised by an underlying OLE DB provider. The most commonly used OLE DB provider is the OLE DB provider for ODBC drivers, which exposes ODBC data sources to ADO.
- Examples:

```
Set Cnn = Server.CreateObject("ADODB.Connection")
Cnn.open "PROVIDER=MICROSOFT.JET.OLEDB.4.0;DATA
SOURCE=c:\mydatabase.mdb"
```

```
Set cnn = Server.CreateObject("ADODB.Connection")
cnn.open "PROVIDER=SQLOLEDB;DATA
SOURCE=sqlservername;UID=username;PWD=password;DAT
ABASE=mydatabase " %>
```

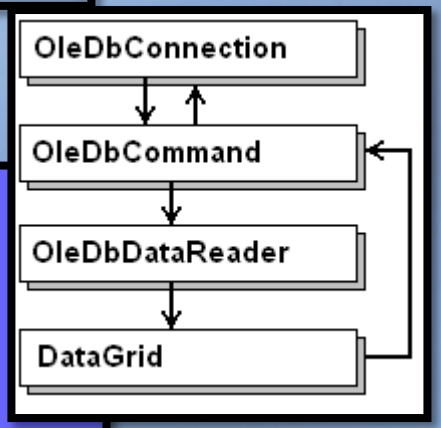
What is an in-memory object?

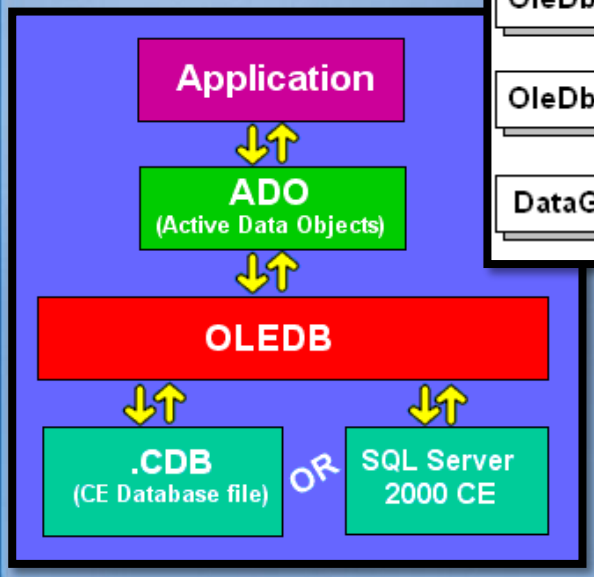
- If your application uses a disconnected data model, you need to store the data in your application temporarily while you work with it; a dataset (an in-memory cache of data) simplifies the process of working with data until you reconnect with your data source.
- Prior to querying your data, you create a dataset to receive the results of a query. The dataset is created with the same shape (schema) of the returned data.
- You typically create new data tables with *TableAdapters* using the TableAdaptorConfiguration Wizard or by dragging database objects from Server Explorer onto the Dataset Designer.
- Data tables are created as a byproduct when you create new *TableAdapters* in the Data Source Configuration Wizard, but they can also be created independently. You can create a stand-alone data table by dragging a *DataTable* object from the DataSet tab of the Toolbox onto the Dataset Designer.

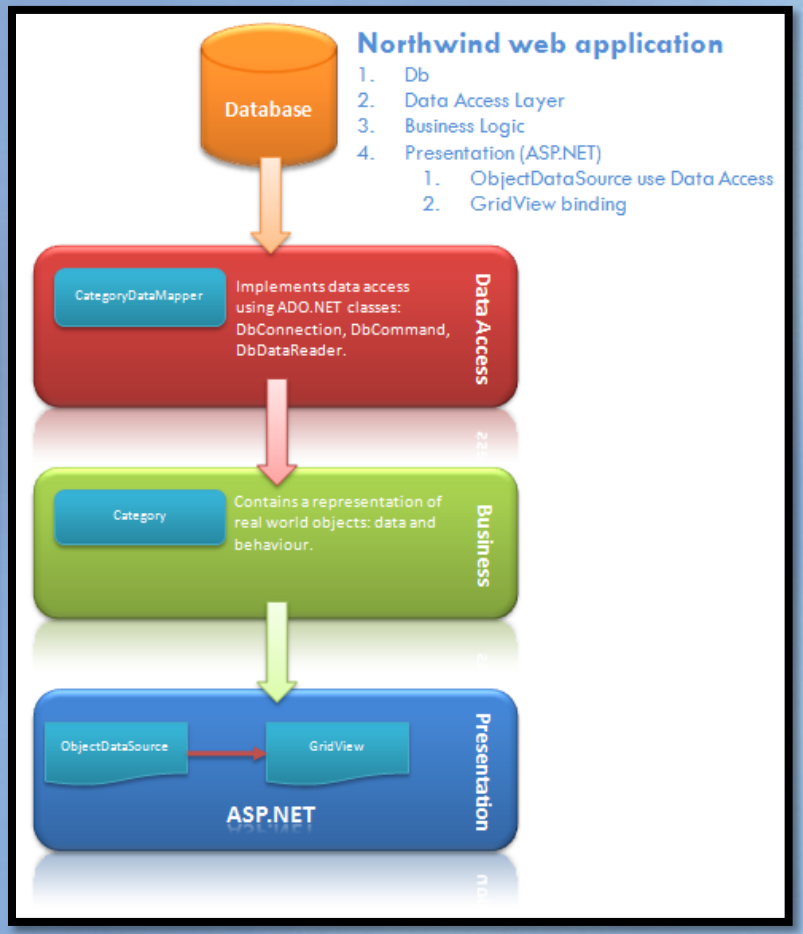
Resource Optimization

- The process of examining data and the data access requirements to optimize the database setup.
- One aspect of resource optimization is the process of normalization, which is discussed in Lesson 6.1.
- Sometimes relations are purposely left un-normalized to improve performance.

98-361 Software Development Fundamentals







LESSON 6.3

98-361 Software Development Fundamentals

No Student Lab 6.3

Complete the QUIA Test

MTA Software Fundamentals 6 Test