ETH Zürich
Chair of Software Engineering

C# Programming in Depth

Prof. Dr. Bertrand Meyer

March 2007 – May 2007

Lecture 10: Database

Lisa (Ling) Liu

Database and Data Representation

- Database Management System (DBMS):
  - provides efficient, convenient, and safe multi-user storage of persistent data
  - provides access to massive amounts of persistent data
  - provides a programming interface that allows a user or program to
    - create new database and specify their struture
    - query and modify the data
- Dominant approach: relational database and SQL

Database and Data Representation

Employee

EmployeeID	Title	ManagerID	VacationHours
1	Production Technician	16	21
2	Marketing Assistant	6	42
3	Engineering Manager	12	2
4	Senior Tool Designer	3	48
Int32	String	Int32	Int16

Database and Data Representation

- A "relation" is a table of data
- The columns are known as "attributes"
- The row are called "tuples"
- It is allowable for some values to be missing
- We can add, remove, or update tuples
- Each attribute has an underlying domain, or data type

SQL Database

- We will generally refer to the relations, attributes, and tuples as tables, columns, and rows
- The structure of a table is referred to as its schema

Employee (HumanResources)

Column Name	Condensed Type	Nullable
EmployeeID	int	No
NationalIDN...	nvarchar(15)	No
ContactID	int	No
LoginID	nvarchar(256)	No
ManagerID	int	Yes
Title	nvarchar(50)	No
BirthDate	datetime	No
MaritalStatus	nchar(1)	No
Gender	nchar(1)	No
HireDate	datetime	No
SalariedFlag	Flag:bit	No
VacationHours	smallint	No
SickLeaveH...	smallint	No
CurrentFlag	Flag:bit	No
rowguid	uniqueidentifier	No
ModifiedDate	datetime	No

SQL Database

Employee

EmployeeID	Title	ManagerID	VacationHours
1	Production Technician	16	21
2	Marketing Assistant	6	42
3	Engineering Manager	12	2
4	Senior Tool Designer	3	48

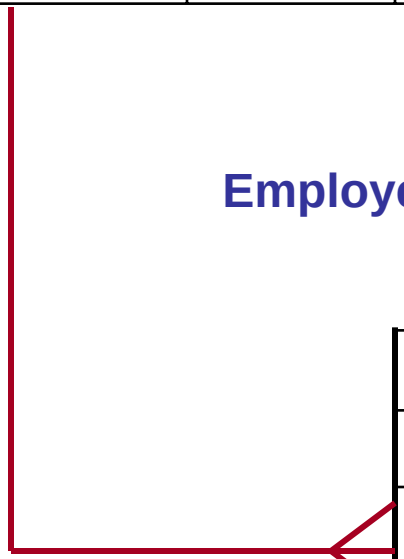
Primary key

Primary key: no two rows can have the same EmployeeID.
EmployeeID cannot be null.

- Assume that we want to add data about employees' salary
  - Assume an employee's salary is changeable, we need to add following two columns.
    - RateChangeDate
    - Rate
- We can't add additional columns for the same employee without violating the primary key constraint. So we use another table.

Employee

EmployeeID	Title	ManagerID	VacationHours
1	PT	16	21
2	MA	6	42
3	EM	12	2
4	STD	3	48



EmployeeID establishs a relationship between the tables.

EmployeePayHistory

EmployeeID	RateChangeDate	Rate
1	31.07.1996	12.4500
2	26.02.1997	13.4615
3	12.12.1997	43.2692
4	05.01.1998	8.6200
4	01.07.2000	23.7200
4	15.01.2002	29.8462

Employee

EmployeeID	Title	ManagerID	VacationHours
1	PT	16	21
2	MA	6	42
3	EM	12	2
4	STD	3	48

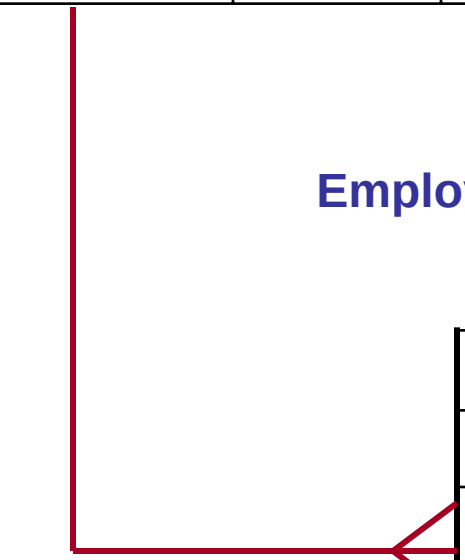
We say that there is a "foreign key constraints" between the tables.

The column referenced in the parent table must be a primary key

Every value in the foreign column must actually appear in the parent table.

EmployeePayHistory

EmployeeID	RateChangeDate	Rate
1	31.07.1996	12.4500
2	26.02.1997	13.4615
3	12.12.1997	43.2692
4	05.01.1998	8.6200
4	01.07.2000	23.7200
4	15.01.2002	29.8462



Relation
one-to-many
parent/child

Foreign key

Simple SQL Queries

- SELECT

```
SELECE  *                          ——   what columns to output
FROM Employee                      ——    what tables are involved
WHERE VacationHours > 20           —— what rows are of interest
```

Employee

EmployeeID	Title	ManagerID	VacationHours
1	PT	16	21
2	MA	6	42
3	EM	12	2
4	STD	3	48

```
SELECT EmployeeID, ManagerID
FROM Employee WHERE
VacationHours > 20
```

EmployeeID	ManagerID
1	16
2	6
4	3

How to interact with data stores?

- ADO.NET

  a set of namespaces defined on .NET platform that understand how to interact with data stores.

  - native support for SQL Server and Oracle
  - support for other databases via older *OleDB* technology
  - requires a knowledge of SQL

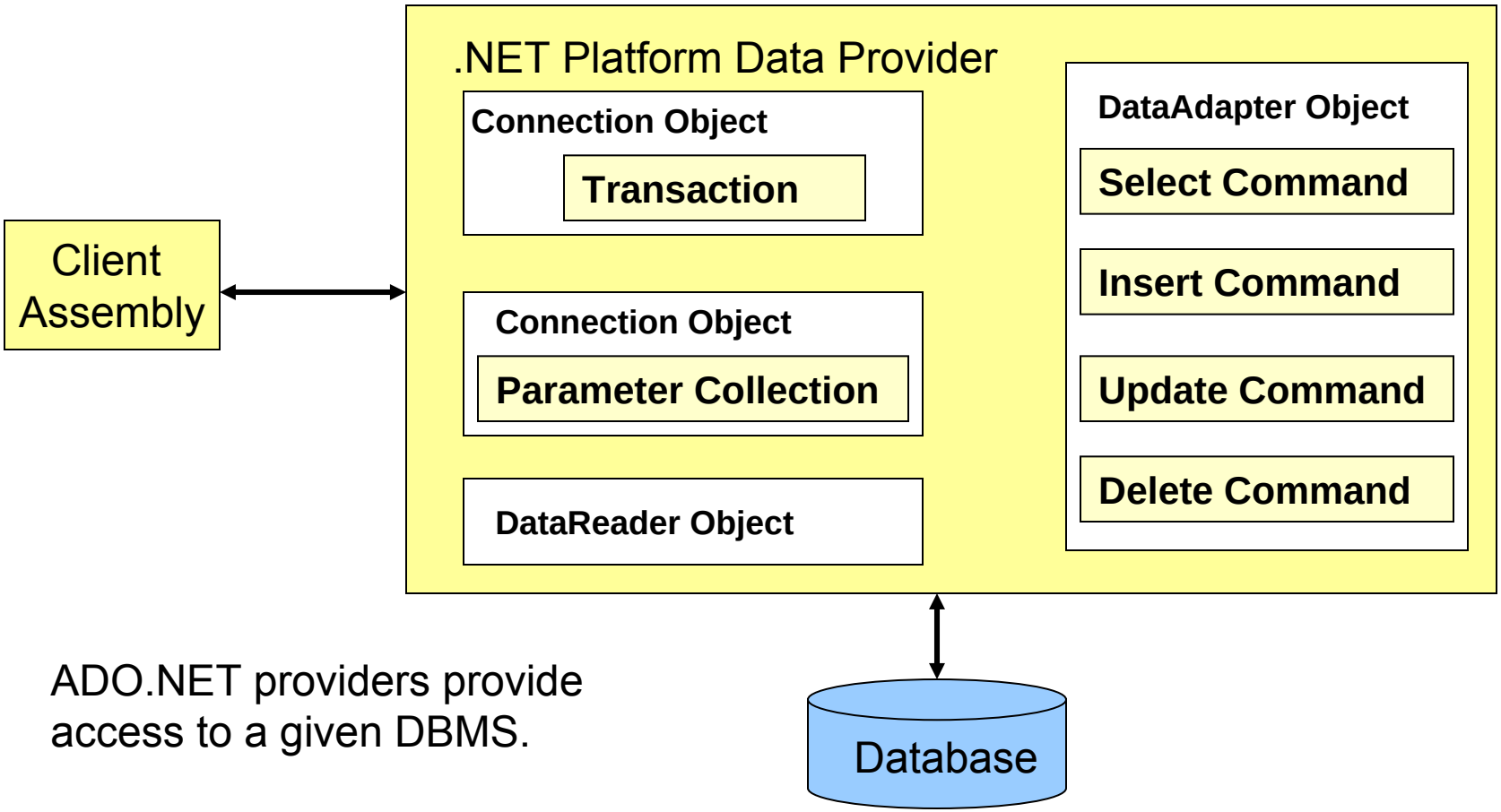
ADO.NET-centric Namespaces

- **Core namespaces:**
  - general:
    `System.Data`
  - SQL Server:
    `System.Data.SqlClient`
  - Oracle:
    `System.Data.OracleClient`
  - OleDB:
    `System.Data.OleDb`

Two manners of accessing database

- Connected manner
  - explicitly connected to and disconnected from the underlying data store
- Disconnected manner
  - using DataSet – a local copy of external data to interact with data stores

Data Providers



ADO.NET providers provide access to a given DBMS.

Overview of database access

- General steps:
  - open connection to database
  - execute SQL to retrieve records / update DB
  - close connection

Database Access (Connect Manner)

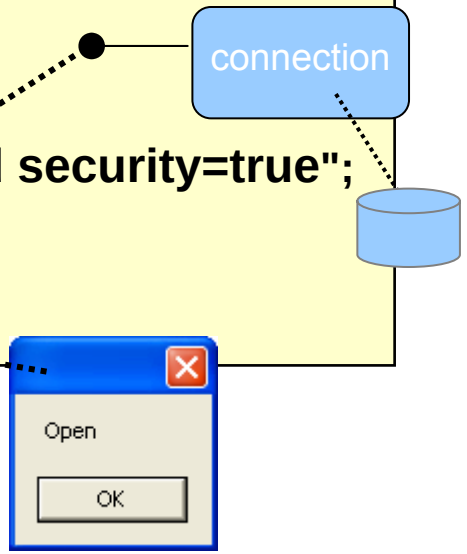
Five steps:

1. Allocate, configure, and open your connection object
2. Allocate and configure a command object
3. Acquire DataReader object
4. Process each record using DataReader object
5. Close connection

Step 1: open connection

- Connections are opened based on *connection string* info
  - here we open a connection to a SQL Server database
  - "AdventureWorks" database must be installed on the local machine.

```
using System.Data;
using System.Data.SqlClient;
...

SqlConnection cn = new SqlConnection();
        cn.ConnectionString =
"server=(local);database=AdventureWorks;integrated security=true";
        cn.Open();

MessageBox.Show( cn.State.ToString() );
```



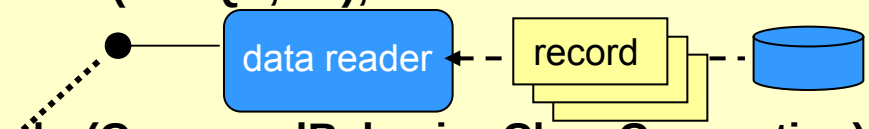
Building connection strings

- Connection strings are vendor-specific, not well-documented
- Where to turn for help?
  - www.connectionstrings.com
  - www.carlprothman.net/Default.aspx?tabid=81

Step 2-4: retrieve records

- Retrieve records via SQL Select query
  - read-only access by database field names

```
string strSQL = "SELECT * FROM HumanResources.Employee";
SqlCommand myCommand = new SqlCommand(strSQL, cn);

SqlDataReader myDataReader;
myDataReader = myCommand.ExecuteReader(CommandBehavior.CloseConnection);

while (myDataReader.Read())
{
  Console.WriteLine("EmployeeID: {0},  Title: {1}, ManagerID: {2}, VacationHours: {3}",
    myDataReader["EmployeeID"].ToString().Trim(),
    myDataReader["Title"].ToString().Trim(),
    myDataReader["ManagerID"].ToString().Trim(),
    myDataReader["VacationHours"].ToString().Trim());
 }
```



Step 5: close connection

- Be sure to close connection…
  - to flush pending updates
  - so others can access DB (connections are limited resources)

```
cn.Close();
```

Guaranteed close?

```
IDbConnection  dbConn = null;

try {
   cn.Open();
   .
   .
   .
}
catch(Exception ex) {
   System.Diagnostics.EventLog.WriteEntry("MyApp", ex.Message);
   System.Diagnostics.EventLog.WriteEntry("MyApp", ex.StackTrace);
   throw ex;
}
finally {
   if ((cn != null) && (cn.State != ConnectionState.Closed))
     cn.Close();
}
```

Updating a database

To update database, execute an SQL Action query

Example:

- delete employee by their id number

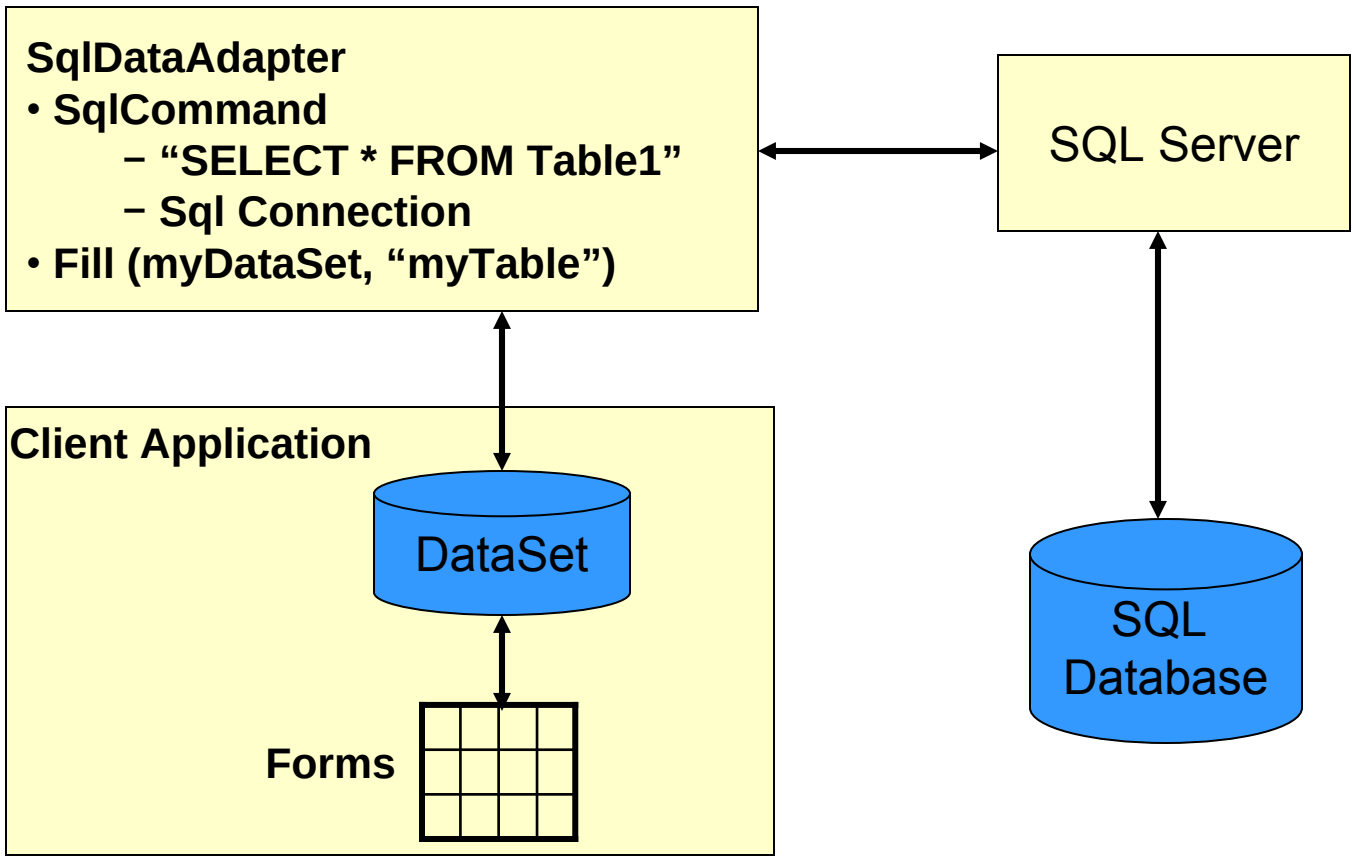
```
string sql = string.Format("DELETE FROM Employee WHERE EmployeeID = '{0}'",
                 employeeID);
SqlCommand cmd = new SqlCommand(sql, cn);

 try
 {
     cmd.ExecuteNonQuery();
  }
 catch
  {
       Console.WriteLine("Sorry! That employ cannot be deleted.");
  }
```

Example of action queries

Insert, update and delete:

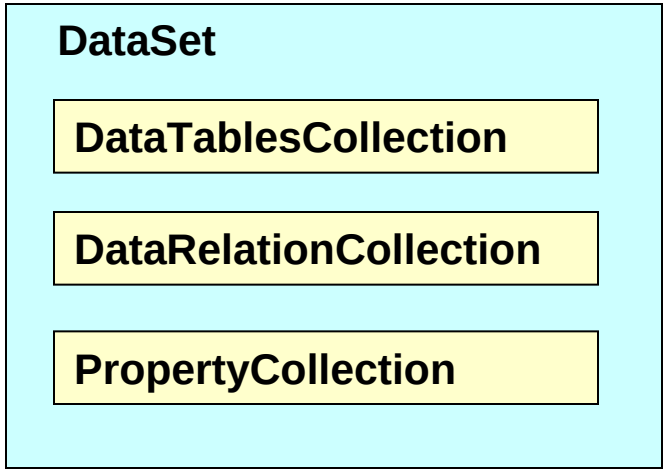
```
Insert Into Customers(CID, FirstName, LastName,
  CreditLimit, Balance) Values(118, 'Jia', 'Zhang',
  10000.0, 0.0);

Update Customers Set CreditLimit = 40000000000.0,
  Balance = 0.0 Where LastName = 'Gates' and
  FirstName = 'Bill';

Delete From Customers Where CID = 666;
```

Database Access (Disconnect Manner)



DataSet

- DataSets are an in-memory, read-write data structure
  - easily filled with data from a database
  - easily displayed in a GUI app



DataAdapter

- make use of DataSet objects to move data between client and data store.
- is used to fill DataSet with DataTable objects
- send modified DataTables back to the database for processing
- take care of connection, hence client don't need to explicitly open and close the connection with DBMS

Steps

1. construct data adapter with a valid connection or connection string and a command object
2. fill DataSet using the internal command within the data adapter
3. operate on the DataSet
4. using data adapter to update data store with the DataSet

Example

Retrieve product info and display in a DataGrid:

```
string sql = "SELECT * FROM Employee";
SqlCommand myCmd = new SqlCommand(sql, cn);
SqlDataAdapter myAdapter = new SqlDataAdapter(myCmd);

 DataSet myDS = new DataSet("HumanResources");
 myAdapter.Fill(myDS, "Employee");

PrintDataSet(myDS);
```

Flushing changes back to database

```
sql = string.Format("INSERT INTO Employee" +
          "(Title, ManagerID, VacationHours) VALUES" +
           "('{0}', '{1}', '{2}')", title, managerID, vacationHours);

SqlCommand insertCmd = new SqlCommand(sql, cn);

myAdapter.InsertCommand = insertCmd;

//Update Employee table with new row
DataRow newEmployee = myDS.Tables["Employee"].NewRow();
newEmployee["Title"] = title;
newEmployee["ManagerID"] = managerID;
newEmployee["VacationHours"] = vacationHours;
myDS.Tables["Employee"].Rows.Add(newEmployee);
myAdapter.Update(myDS.Tables["Employee"]);
```

Untyped DataSets

- Collection of tables
  - Tables are collections of columns and rows
  - Rows hold the data
  - Filling tables does not create relations between them

```
carName = myDS.Tables[“Inventory"].Rows[0][“PetName”]
```

- To use relations between tables in memory, we must write code that builds the relations

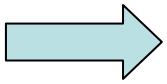
Typed DataSets

- A class derived from DataSet
- Incorporates the schemas for the tables it contains
- Has properties and methods that allow access to tables and columns by name
- Extends DataSet, DataTable, DataRow to provide custom classes
- Table, column, and method are known by names, reducing coding time and erros

```
carName = myDs.Inventory[0].PetName;
```

Generating a typed DataSet

1. Right-click project
2. Add | Add new item.. | Data Set
3. Select a name
4. Find the tables of interest in the Server Explorer and dray them onto the design surface
5. Drag relations onto the child tables and verify the settings
6. Save



- A file with extension .xsd that represents the tables and schemas
- A class derived from DataSet in a .cs file

Questions