JOHN ABBOTT
CEGEP/COLLEGE

(420-PS4-AB)
Data Sources : Part 2 Entity Framework

Aref Mourtada

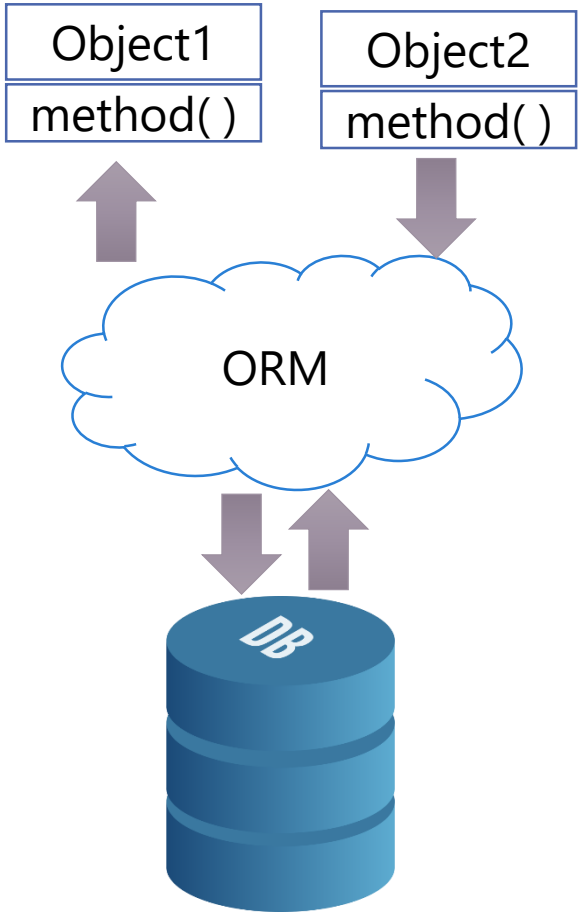
Fall 2017

Entity Framework

- Introduction to Entity framework
- Code First
- When Classes change
- Seeding the database

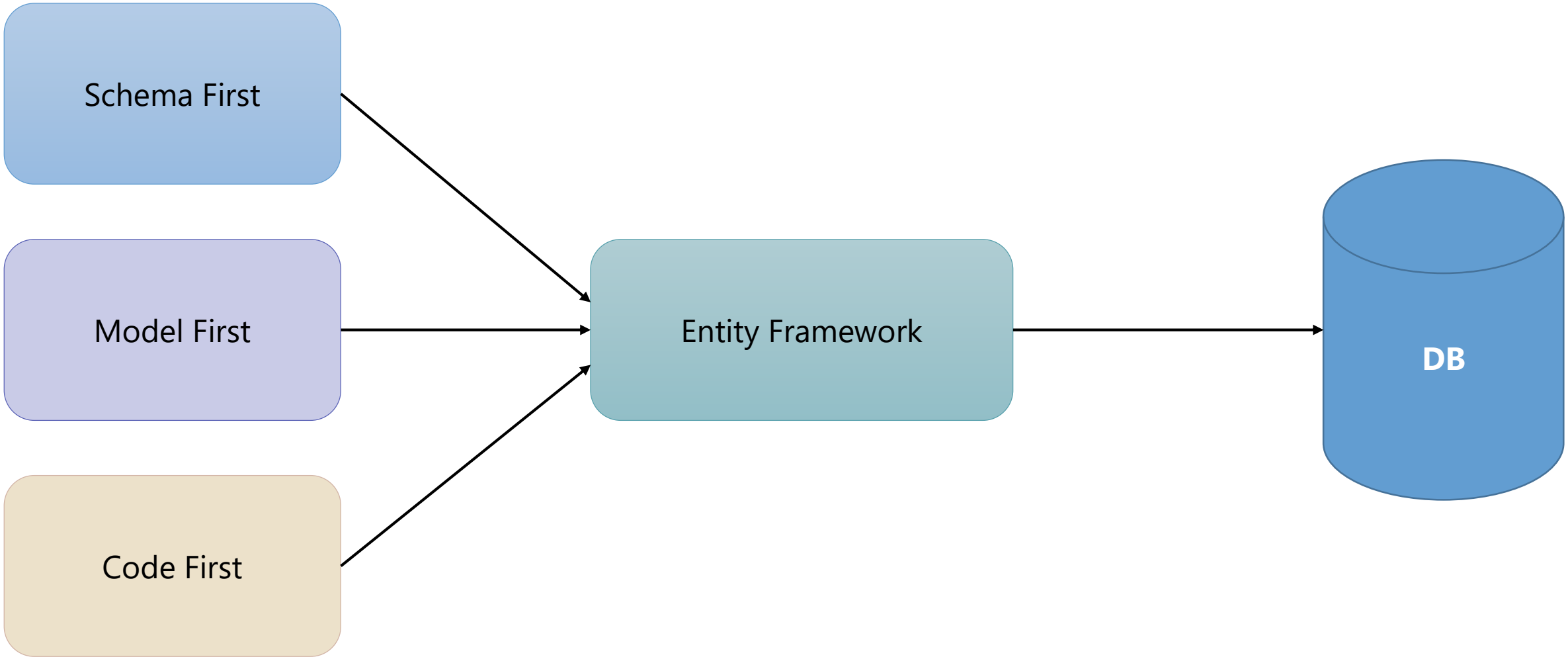
Entity Framework is an ORM

- The .NET Framework provides support for Object Relation Mapping (ORM)



- Features
  - Automatically generate necessary SQL code
  - Map result sets to strongly typed objects
  - Persist object changes back to a database
  - Implicit support for transactions

Entity Framework Feature



Schema First / Model First

1. Add an ADO .NET Data Model into your project
2. Select the database to query from the wizard
3. Select the tables, views and stored procedures
4. Write LINQ to Entities queries that use the ObjectContext

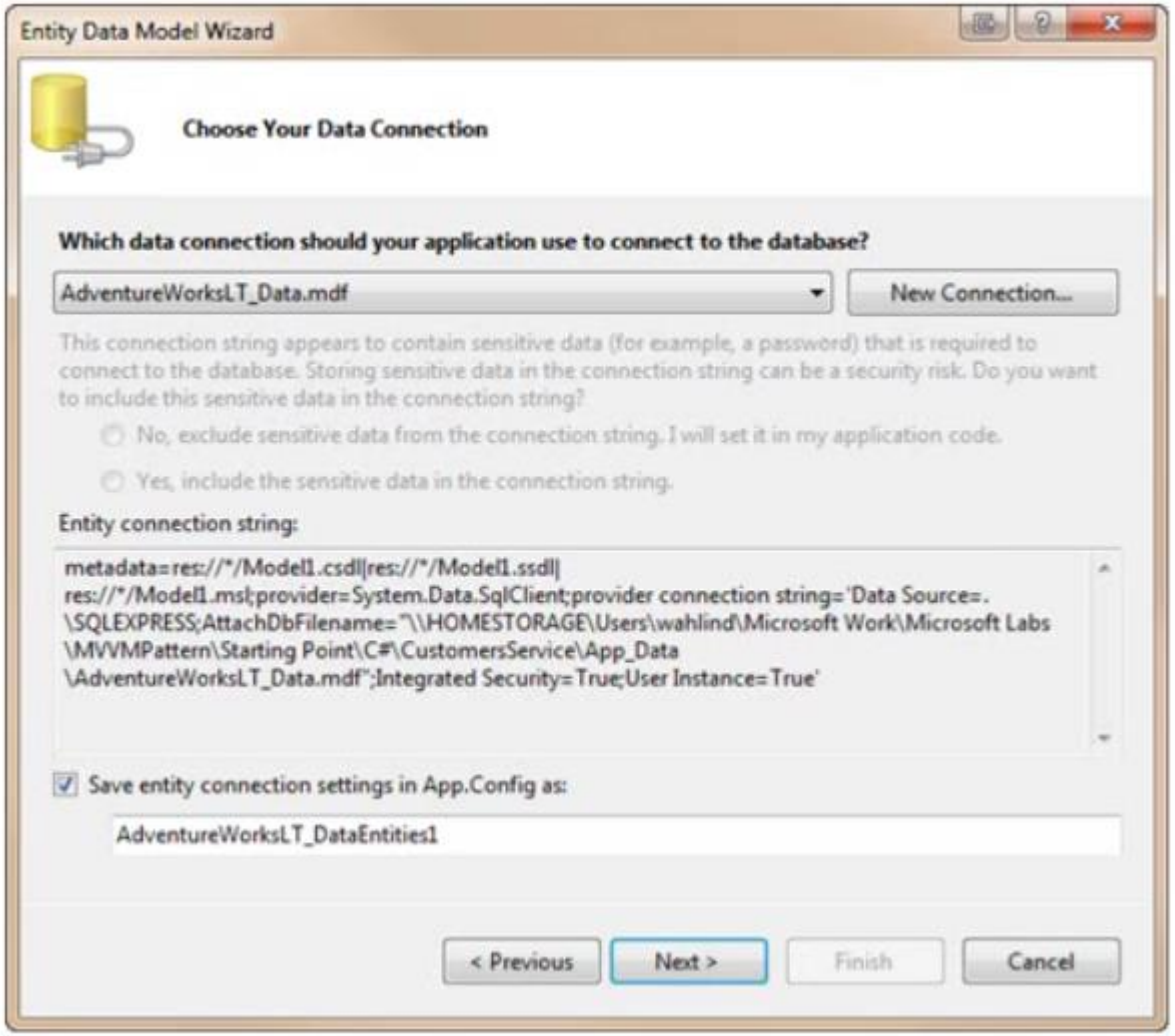
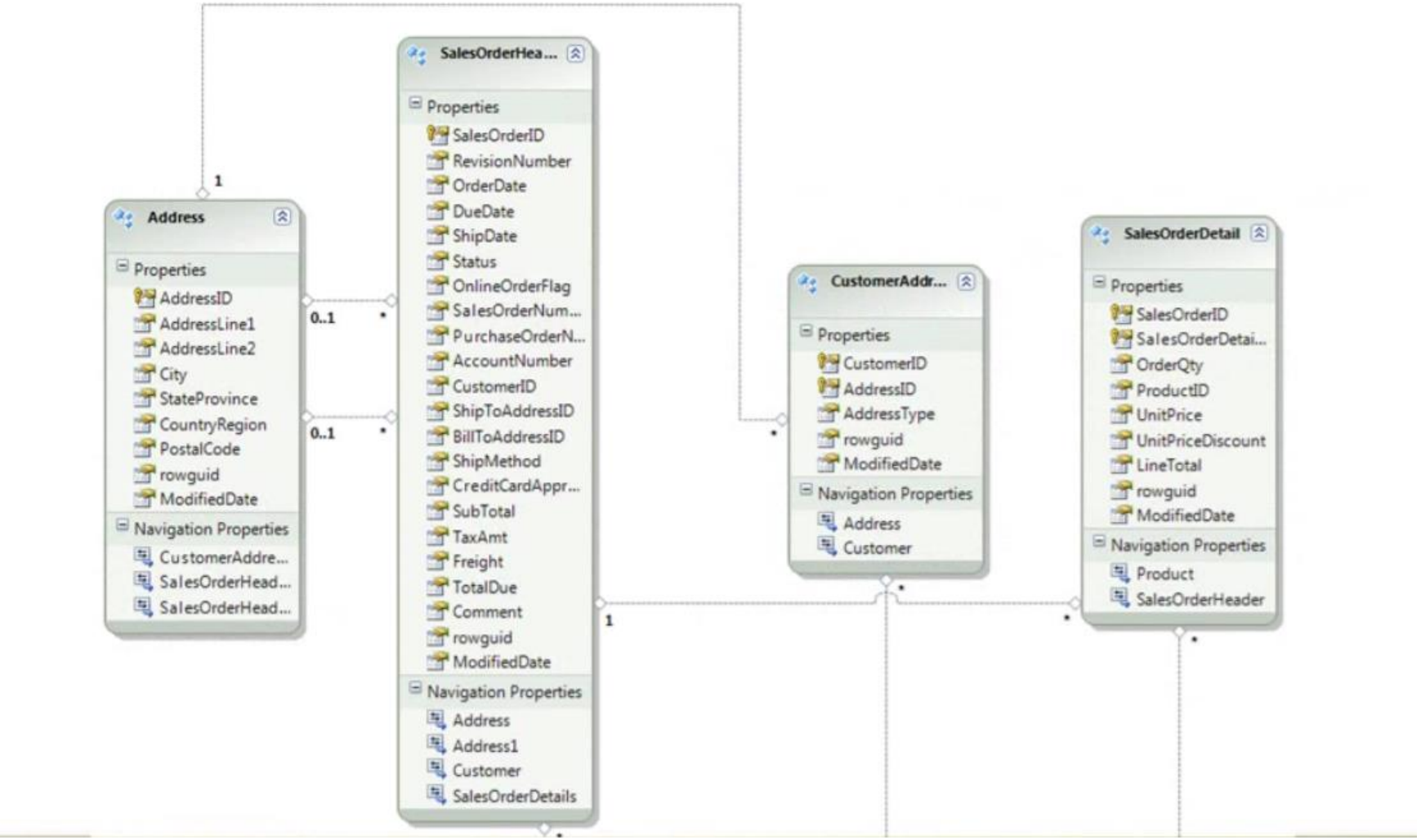
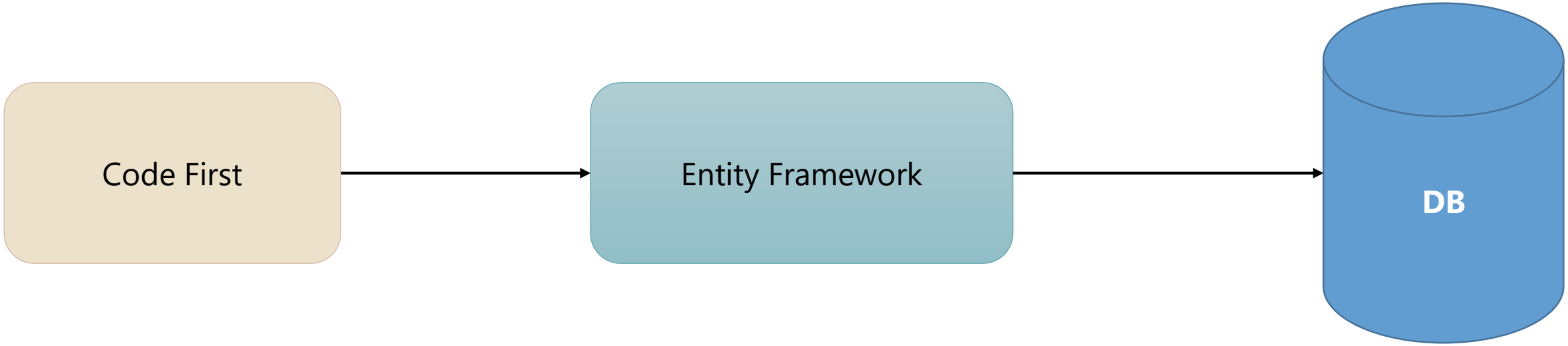


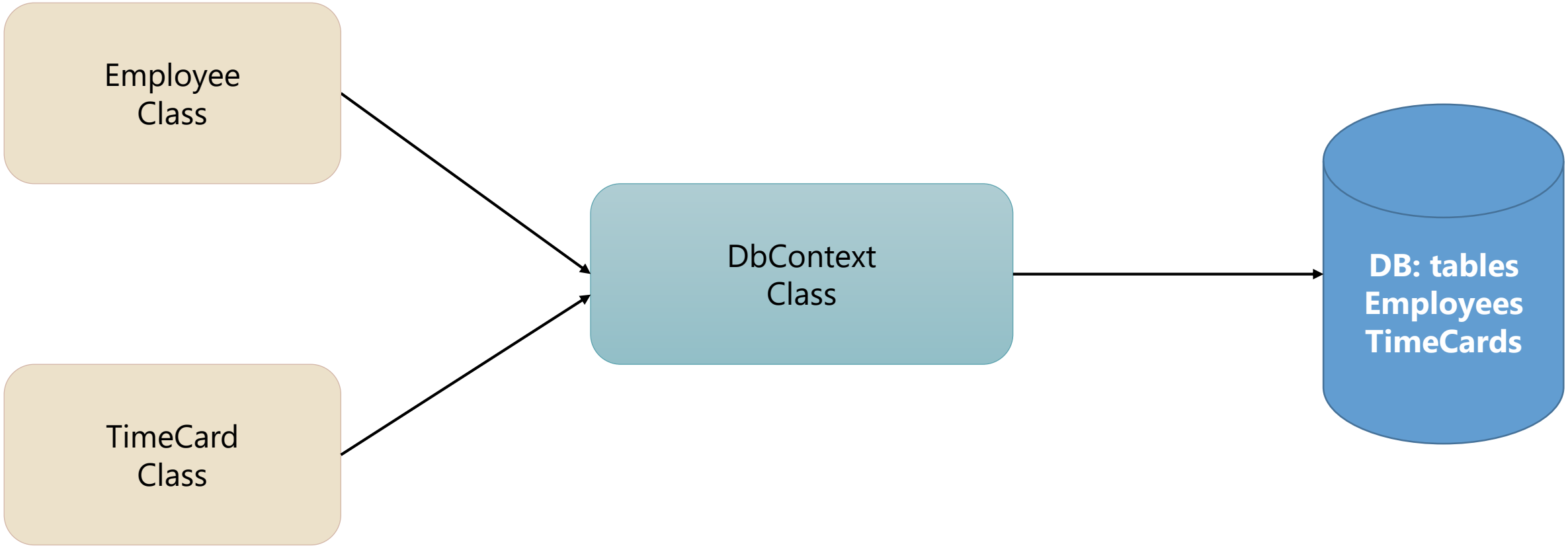
Diagram Outcome



Code First



Code First DbContext



Requirements

- Install EntityFrame from Nuget Manager
- Design & write Classes
- Database Context Class
- Query Class

Requirements

- Install EntityFrame from Nuget Manager
- Design & write Classes
- Database Context Class
- Query Class (repository)
- Add ConnectionString

Demo: TimeTracker

- Write a web application that will use code first function to track time cards for employees in a company.
- For each employee we need to track: ID, First Name, Last Name, Department and all working time cards.
- For each time card we need to know the card: ID, submission data and the hours worked for each day of the week.

Step 1: Employee and TimeCard Classes

```
public class Employee
{
    public int ID { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Department { get; set; }
    public List<TimeCard> timeCards { get; set; }

}
```

```
public class TimeCard
{
    public int ID { get; set; }
    public DateTime submissionDate { get; set; }
    public int MondayHours { get; set; }
    public int TuesdayHours { get; set; }
    public int WednesdayHours { get; set; }
    public int ThurdaydayHours { get; set; }
    public int FridayHours { get; set; }
    public int SaturdayHours { get; set; }
    public int SundayHours { get; set; }
}
```

Step 2: DbContext Class

```
public class TimeTrackerDbContext : DbContext
{
    public DbSet<Employee> Employees { get; set; }
    public DbSet<TimeCard> TimeCards { get; set; }
}
```

Note: do not forget to add the needed name spaces to your created classes.

Step 3: Running Engine – The Repository Class

In this class you write all methods that will retrieve data from you from the DB.

```
public class TimeTrackeRepository
{
    TimeTrackerDbContext _context = new TimeTrackerDbContext();

    /*
     * This methods gets the records of all employees
     * */
    public List<Employee> getAllEmployees()
    {
        List<Employee> allEmps =
            (from data in _context.Employees
             select data).ToList();

        return allEmps;
    }

    public List<TimeCard> getemployeeTimeCard(int empID)
    {
        List<TimeCard> mytimeCards =
            (from data in _context.Employees
```

Note: do not forget to add the needed name spaces to your created classes.

Step 4: Adding a Connection String in web.config

- Connection string

```
<connectionStrings>
    <add name"Data Source=BH210-12;
        Integrated Security=SSPI;
        database=TimeTracker2"
    providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Step 5: Design your Web Application

localhost:60598/Default.aspx

Welcome to my TimeTracker Application

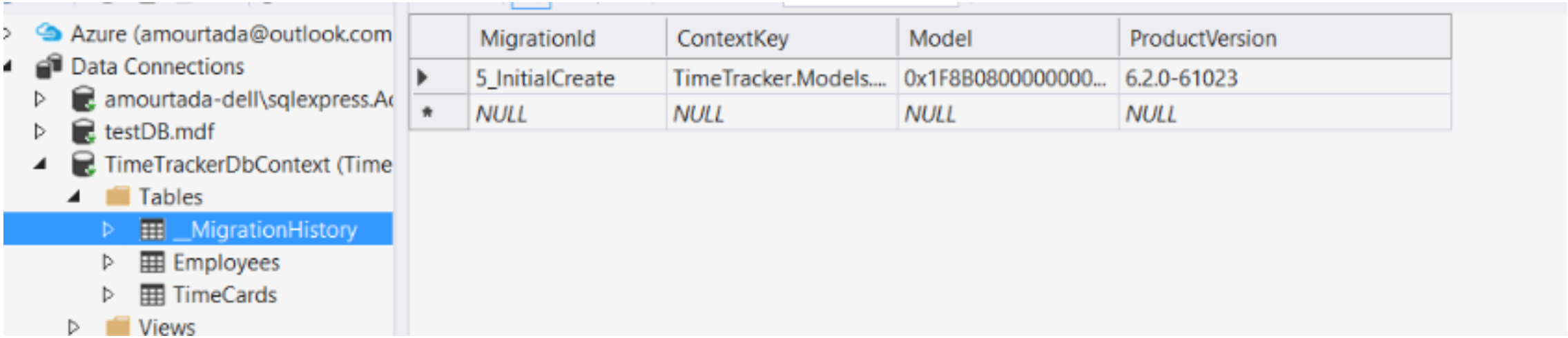
	ID	First Name	Last Name	Department
Select	1	Barry	Allen	IT
Select	2	Tom	Alex	HR
Select	3	Tim	Horton	Sales
Select	4	Alice	Wonder	IT
Select	5	Aref	Mour	IT
Select	6	Thomas	Adison	Engineering

Changes in Class

- Need to change to classes
- Classes are connected to the database table
- How can we handle changes.
- Seeding options

Tracking Changes in Class Design

- _MigrationHistory table is used to track any change in the entity framework model.



Demo: Handling Changes

- Add an initializer class to drop and create database when the model changes. (can be used to seed you DB)

```
public class TimeTrackerDbContextInitializer : DropCreateDatabaseIfModelChanges<TimeTrackerDbContext>
{
    protected override void Seed(TimeTrackerDbContext context)
    {
        Employee tempEmployee = new Employee();

        tempEmployee.ID = 1;
        tempEmployee.FirstName = "Barry";
                                .
                                .
        context.Employees.Add(tempEmployee);
        base.Seed(context);
    }
```

Note: do not forget to add the needed name spaces to your created classes.

Q & A

JOHN ABBOTT
CEGEP/COLLEGE