Базы данных

[X] ADO.NET с 2001, работает с любыми СУБД, отсоединенные данные.

Недостатки.

1. Запросы к СУБД в текстовом виде “” \
2. Никакой проверки типов данных
3. Никакой проверки синтаксиса
4. Необходимость работать с ячейками в строках столбцах
5. Unboxing потому что обычно все упаковано object
6. Разработчикам в памяти приходится держать имена таблиц, полей, и их типы.

В 200x появилась новая технология ORM – Object Relational Mapper

* Большой фреймворк который создает м/у БД и разработчиком дополнительную прослойку (ООП) – где таблицы из базы конвертируются в объекты / сущности / ООП.

Разработчики работают с объектами, классами, проверка синтаксиса, проверка типов данных, интел подсказки, Система ORM транслирует действия разработчика в запросы SQL(авто режим)

Работа с базой происходит почти как с обычной бизнес логикой

ORM система есть для всех популярных ЯП.

C# => Microsoft Entity Framework, Dapper (stackoverflow – ASP.NET)

JAVA =>

NHibernate

Php => Doctrine, Propel, …

JS, NodeJS => Sequelise

Python – SQLAlchemy

EF => в 2007 в .NET Framework 3.5 вместе с WPF, Linq, WCF.

1. Classic – Microsoft Entity Framework 6.5 - .NET Framework (Windows) – для настольный ИС
2. Fresh – EF 3 - .NET Core (Linux, Mac) – Web много

Для EF есть два подхода

1. Классический – работа через Model First (Создается графическая модель сущности) Работа с базой происходит через модель(С 2007)
2. Современный подход – Code First (с 2010) – Разработчикам нужно руками писать классы для сущности.

В этом только Model First, в след (веб-теч) – Code First.

Оба эти подхода используются в 2-х вариантах

1. Если базы данных еще нет, то она генерируется из Модели или Нашего набора классов.
2. Если база уже есть, то из нее генерируется или Модель или набор классов для Code First.

Модель ADO.NET EDM нужна

1. Красивая модель – диаграмма (курсовая-проектирование)
2. SQL – скрипт для генерации базы
3. Набор классов на C# для всех