

# Разработка, тестирование и развертывание баз данных в Visual Studio Team System 2010



Дмитрий Андреев

[dmitryan@microsoft.com](mailto:dmitryan@microsoft.com)



# Содержание

- Введение

Проект БД и  
жизненный цикл БД

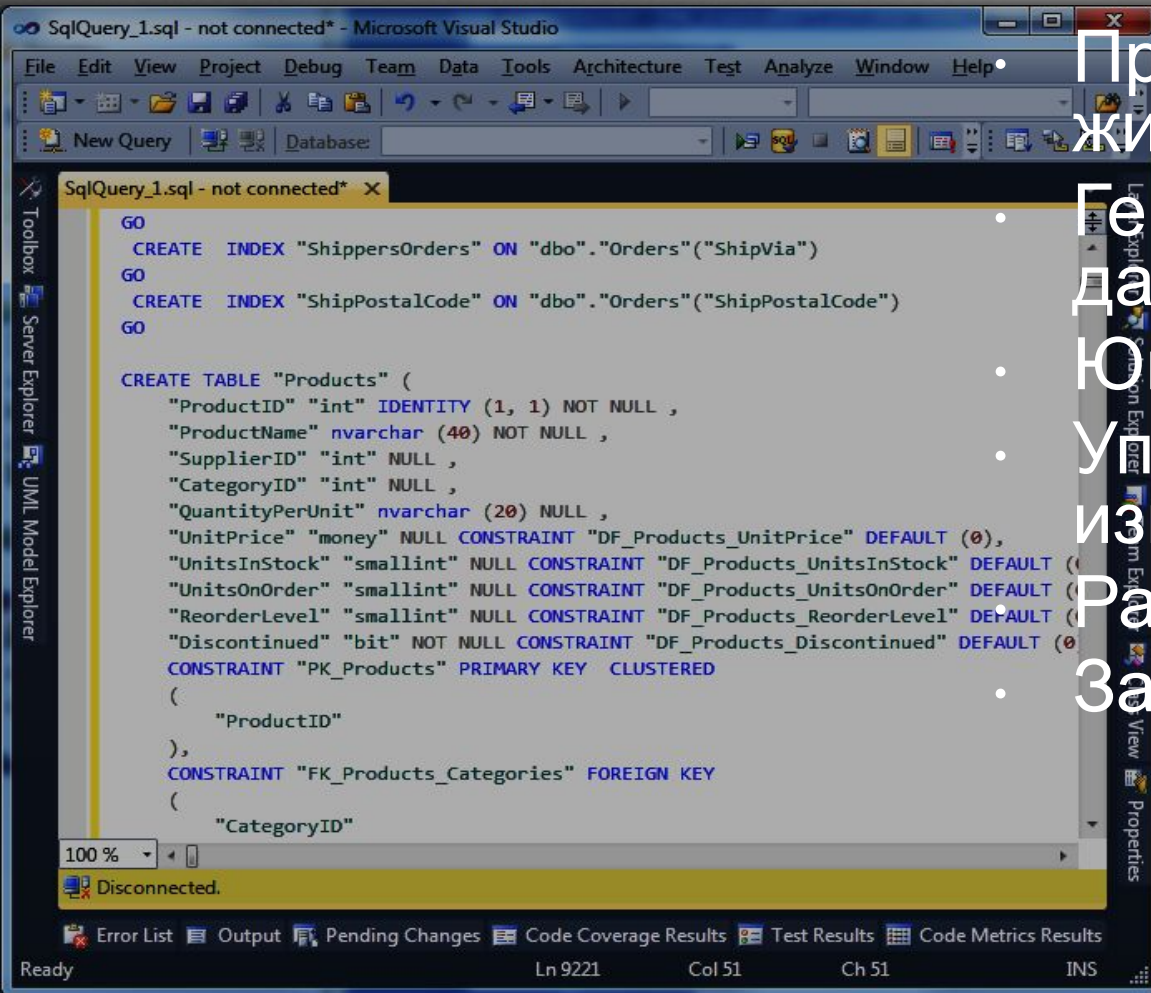
- Генерация тестовых  
данных

- Юнит-тестирование

- Управление  
изменениями

- Развертывание

- Заключение



```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```



# Очень важные вопросы

- Где находится «истинная» схема?
  - Эксплуатационная база?
  - Что насчет исправлений?
  - Что будем делать с следующей версией?
- Как вести версии базы данных?
- Что делать с тестовыми данными?
- Как проверить логику базы данных?
- Какие средства использовать для сравнения схем?
- Как развертывать БД и как делать апгрейд?



# Visual Studio 2010

## Проблема

- Где «истинная» схема?
- Как вести версии?
- Как проводить тестирование?
- Как управлять изменениями?

## Решение

- Где угодно. Схема это исходный проект.
- Так же как и в привычных программных проектах.
- Генерация тестовых данных. Юнит тестирование баз данных.
- Рефакторинг, сравнение схем.



# Правда о «истинной схеме»

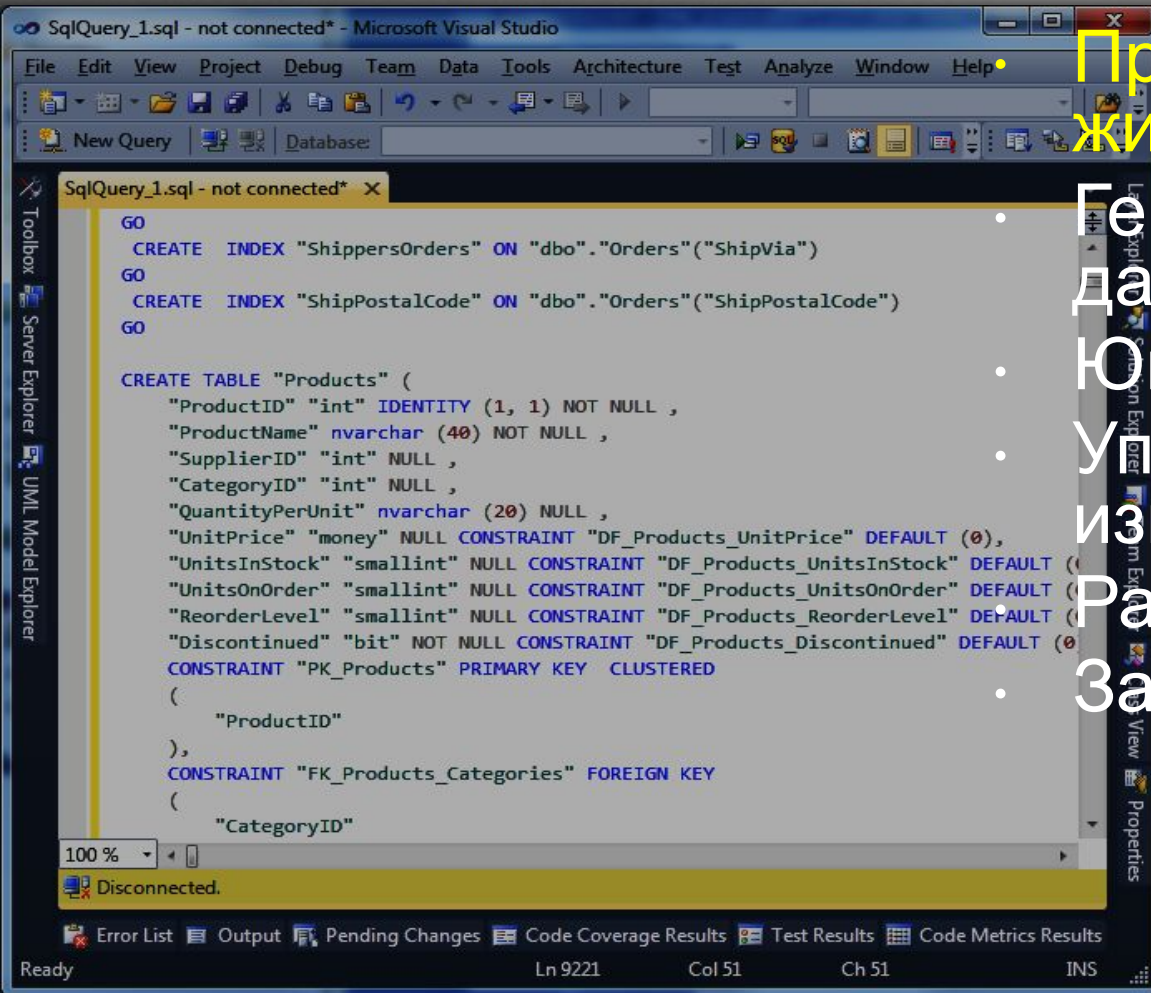
- Эксплуатационная база – единственная верная схема соответствующая версии вашей эксплуатационной системы
- Возможны исправления для этой базы.
- Разработка будущей версии ведется безотносительно к эксплуатационной версии
- Патч для эксплуатационной версии это возникновение:
  - Новой эксплуатационной версии базы
  - Новой версии эксплуатационной системы
- Патч+«Старая версия БД»=релиз билд





# Содержание

- Введение
- Проект БД и жизненный цикл БД
- Генерация тестовых данных
- Юнит-тестирование
- Управление изменениями
- Развертывание
- Заключение



```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```

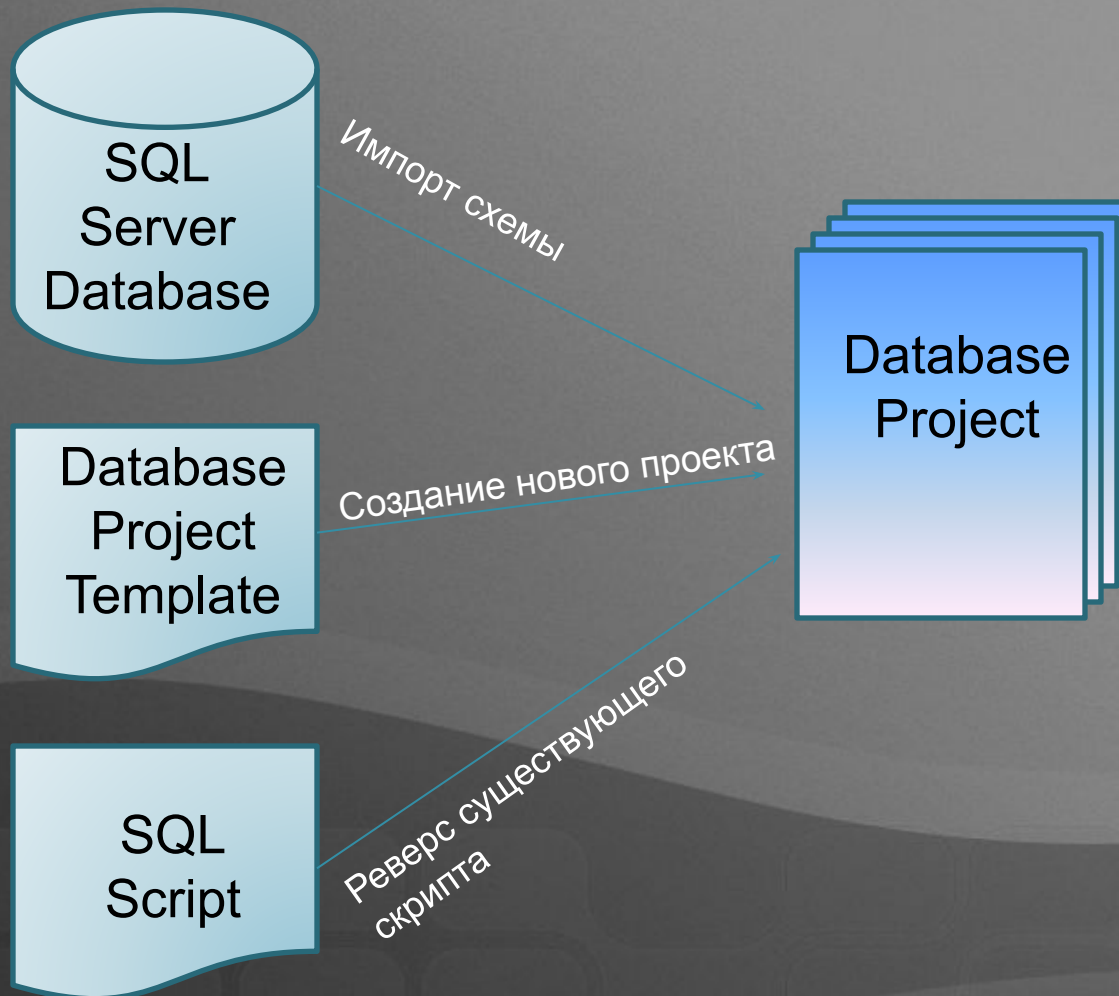


# Проектная модель

- Проект базы данных (Visual Studio Project) отражает эволюционирующую схему
- Проект содержит DDL скрипты (\*.SQL файлы)
- Контроль версий ведется на уровне исходных текстов ЭТИХ скриптов
- Ключ к успеху:
  - Автоматическая генерация тестовых данных
  - Юнит тестирование

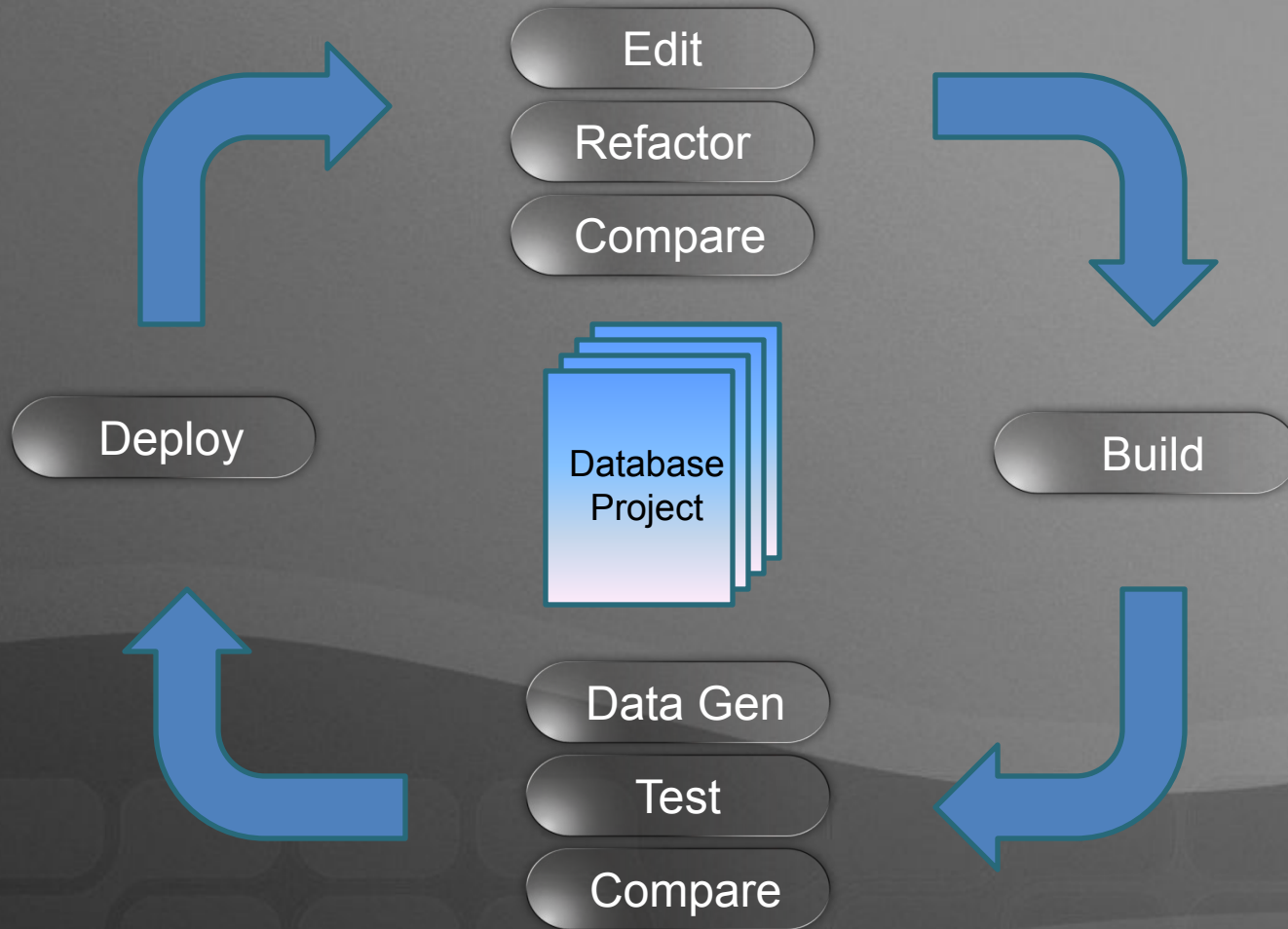


# Жизненный цикл





# Жизненный цикл: классика ALM



# Проблема с контролем версий

```
class class class AuctionApplication
( ( (
  int int int id;
  void void string cacheTitle;
) ) )
  void MethodA();
  void MethodB();
)
```

App



V1

V2

V3

Revision History

```
CREATE ALTER ALTER TABLE dbo.Auction
( WITH WITH CHECK ADD CONSTRAINT
  id AU_PK Au_SK UNIQUE (name)
  name VARCHAR(255) NOT NULL
  start DATETIME NULL
  len INT NULL
)
```

Database



# Ручное ведение версий

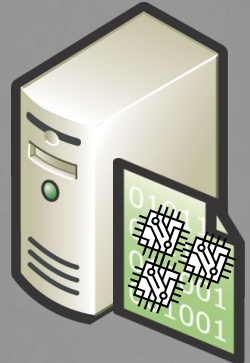
```
-- version 1 Add table dbo.Auction
IF OBJECT_ID (N'dbo.Auction', N'U') IS NULL
BEGIN
CREATE TABLE dbo.Auction
(
    id      INT NOT NULL,
    name   VARCHAR(25) NOT NULL,
    start  DATETIME NULL,
    len    INT NULL
)
END
-- version 2 Add PK Au_PK
IF NOT EXISTS (SELECT * FROM sys.key_constraints WHERE name = 'Au_PK' AND type = 'PK')
BEGIN
    ALTER TABLE Auction
    WITH CHECK ADD CONSTRAINT Au_PK PRIMARY KEY (id)
END
-- version 3 Add UC Au_SK
IF NOT EXISTS (SELECT * FROM sys.key_constraints WHERE name = 'Au_SK' AND type = 'UQ')
BEGIN
    ALTER TABLE Auction
    WITH CHECK ADD CONSTRAINT Au_SK UNIQUE (name)
END
```



# Верный подход к ведению версий

```
class class class AuctionApplication  
( ( ( id;  
int int int  
void void string cacheTitle;  
) void MethodA();  
 ) void MethodB();  
 )
```

App



V1

V2

V3

Revision History

```
CREATE CREATE CREATE TABLE dbo.Auction  
( ( ( id INT NOT NULL PRIMARY KEY,  
id id id  
name name name VARCHAR(25) NOT NULL UNIQUE,  
start start start DATETIME NULL,  
len len len INT NULL  
) ) )
```

Logical  
Database



# Инкрементальное Развертывание

Логическая база



```
CREATE TABLE dbo.Auction
(
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(25) NOT NULL UNIQUE,
  start DATETIME NULL,
  len INT NULL
)
```



Новая система



Эксплуатируемая система



```
CREATE TABLE dbo.Auction
(
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(25) NOT NULL UNIQUE,
  start DATETIME NULL,
  len INT NULL
)
ALTER TABLE dbo.Auction
WITH CHECK ADD CONSTRAINT
Au_SK UNIQUE (name)
```



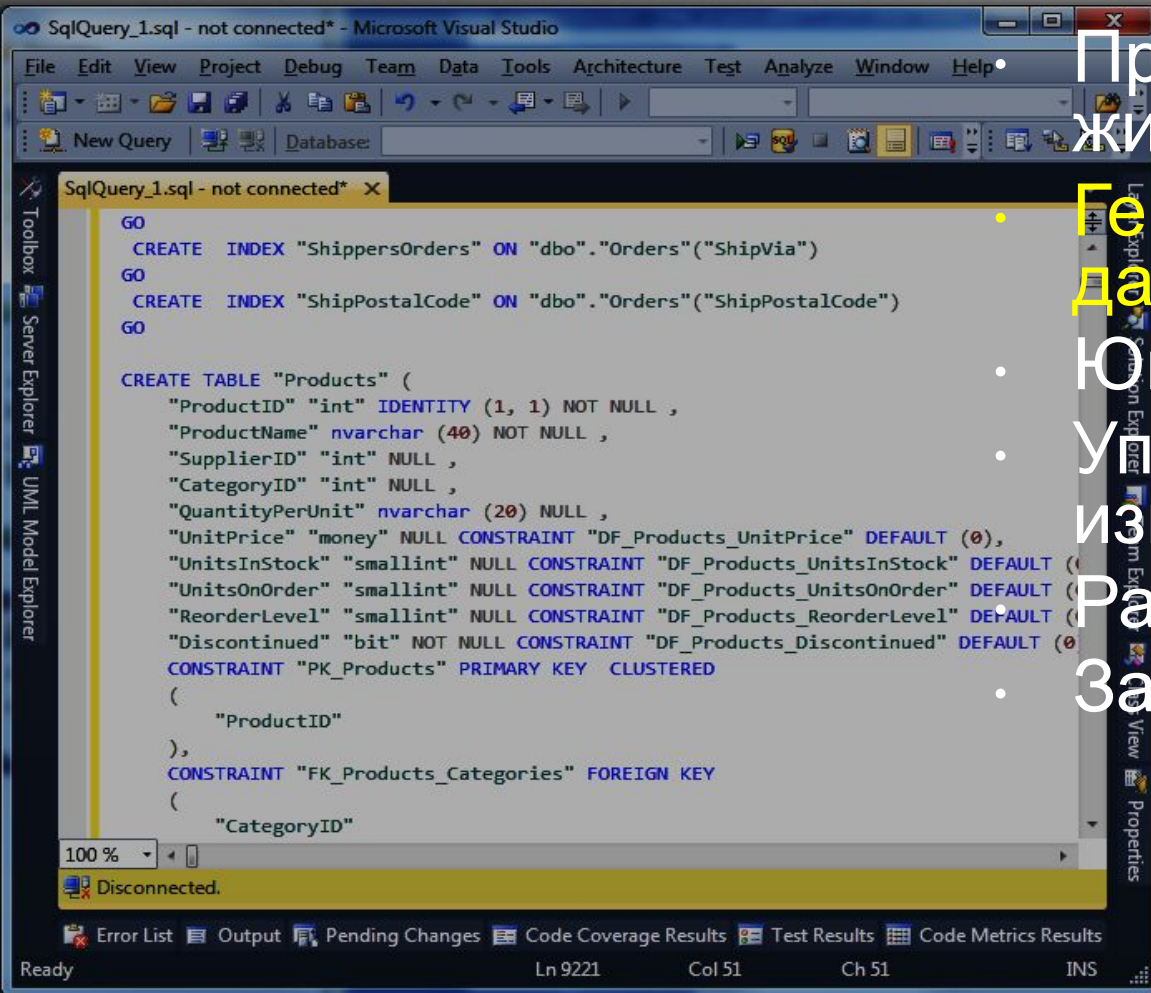


# *Демонстрация*



# Содержание

- Введение
- Проект БД и жизненный цикл БД
- Генерация тестовых данных
- Юнит-тестирование
- Управление изменениями
- Развертывание
- Заключение



```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```



# Тестовые данные и Q&A

- Почему бы не использовать эксплуатационные данные?
  - Они могут быть не верны!
  - Не позволят протестировать «острые углы».
  - Не позволят проверить изменения в схемах данных!
- Какие тестовые данные необходимы?
  - Случайные!
  - Детерминируемые.
  - Распределенные соответствующим образом
    - Количественно (сто первичных ключей -> десять тысяч дочерних записей)
    - Качественно (длина строк, границы числовых значений и дат,...)



# Тестовые данные и Q&A

- Какие отличия необходимы для тестовых данных
  - Функциональные
  - Нагрузочные
- Версионирование
  - Необходимы разные тестовые данные и тесты для разных схем
  - Необходимы даже разные тестовые планы для одной и той же схемы





# Генерация тестовых данных

- Основные инструменты
  - Генерация данных для таблиц
  - Количество записей
- Генераторы для различных типов полей
  - String, RegEx, data bound
  - Можно написать свой собственный генератор
  - Тонкие настройки генераторов



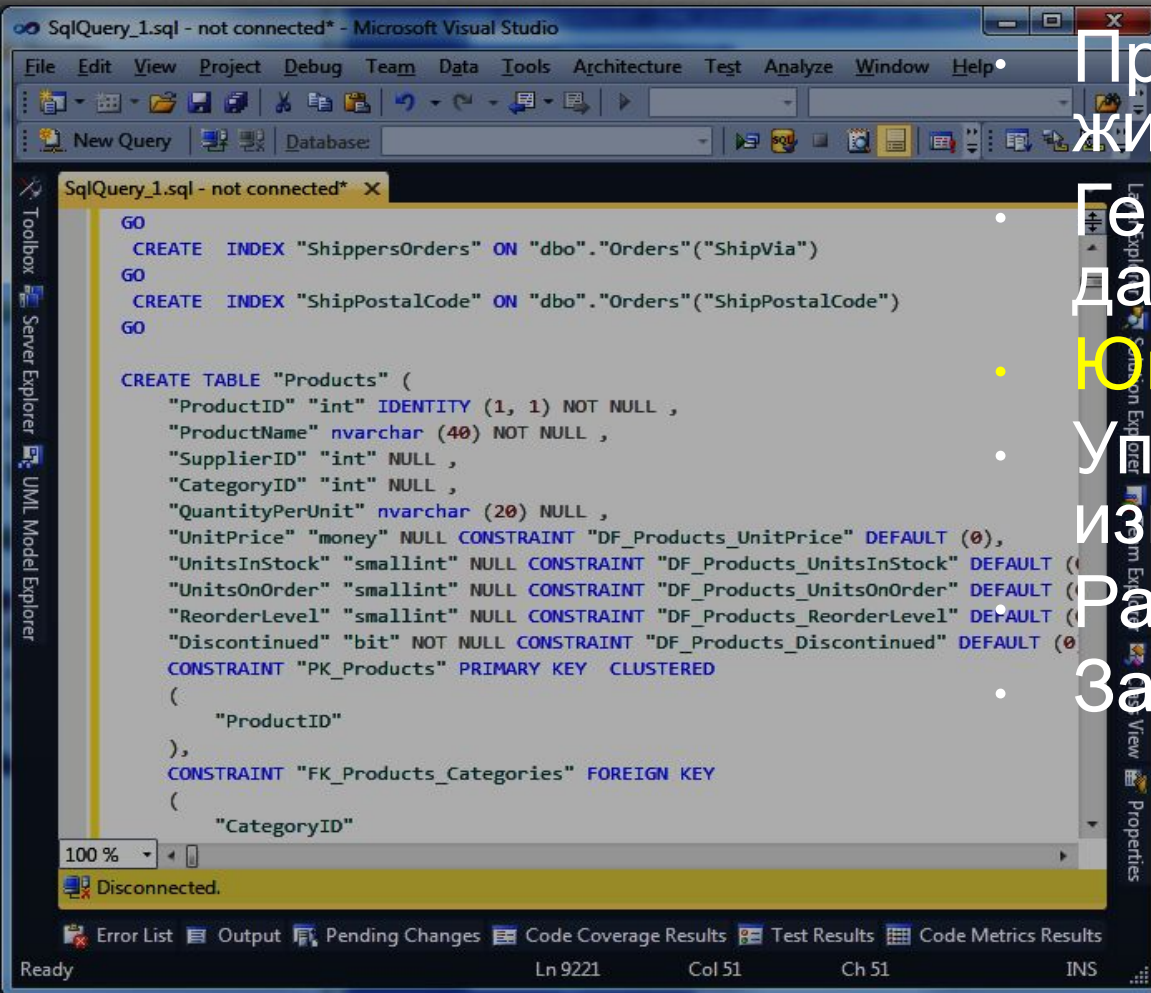


# *Демонстрация*



# Содержание

- Введение
- Проект БД и жизненный цикл БД
- Генерация тестовых данных
- Юнит-тестирование
- Управление изменениями
- Развертывание
- Заключение



The screenshot shows the Microsoft Visual Studio IDE with a SQL query editor. The code defines two indexes on the 'Orders' table and a 'Products' table with various constraints and defaults.

```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```



# Юнит тестирование

- Автоматическая генерация юнит-тестов для
  - Хранимых процедур, Функций, Триггеров
- Валидация результатов тестов (asserts)
  - T-SQL Server based
    - RAISEERROR
  - Ожидаемые значения
    - Не пустые результаты
    - Количество записей
    - Время выполнения
- Предварительные и пост скрипты



# Юнит тестирование

- Автоматизированное развертывание
  - Перед запуском тестов будет сформирована БД
- По соответствующему плану генерации тестовых данных будет создана основа для проверки



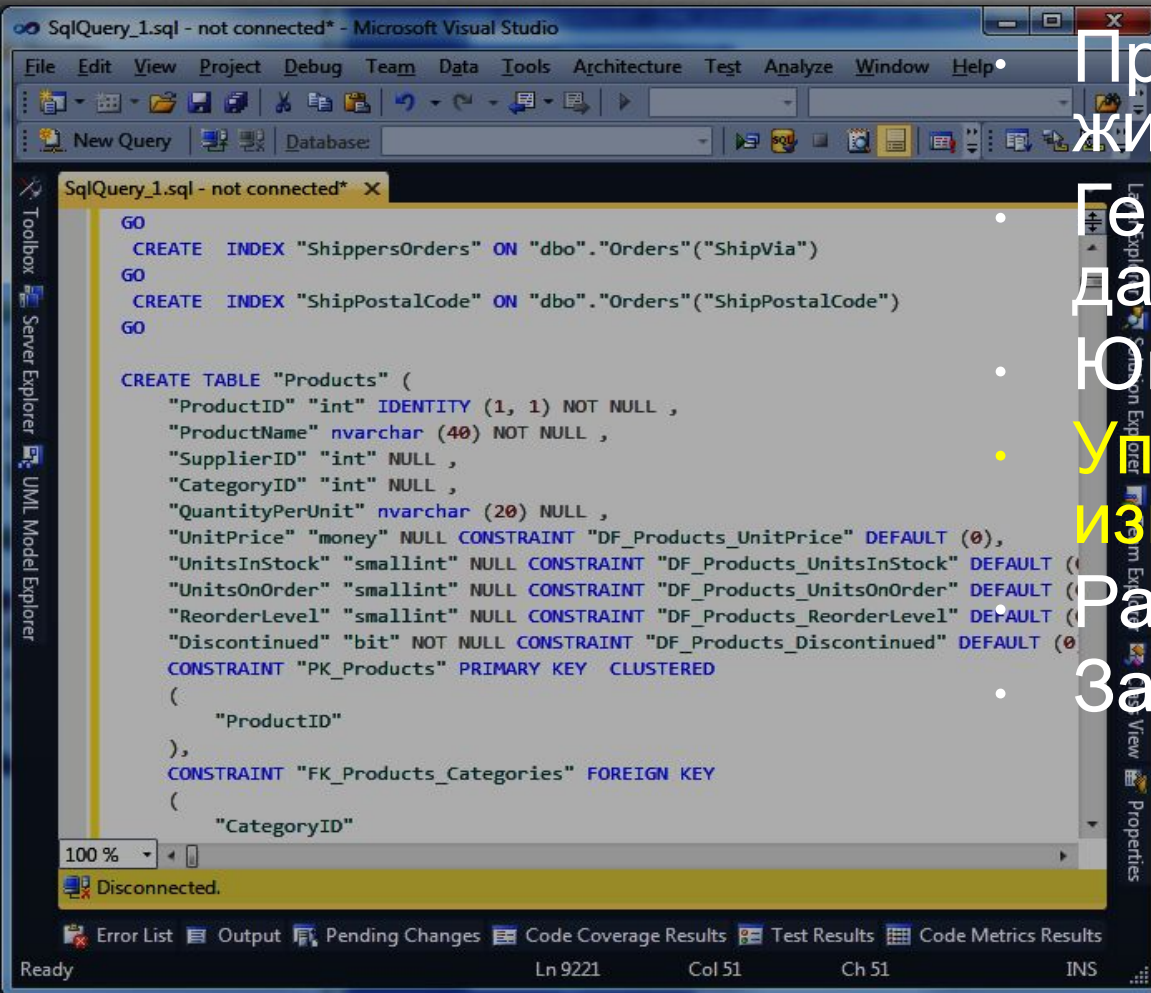
# *Демонстрация*





# Содержание

- Введение
- Проект БД и жизненный цикл БД
- Генерация тестовых данных
- Юнит-тестирование
- **Управление изменениями**
- Развертывание
- Заключение



```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```



# Управление изменениями

- Рефакторинг
- Сравнение схем
- Сравнение данных

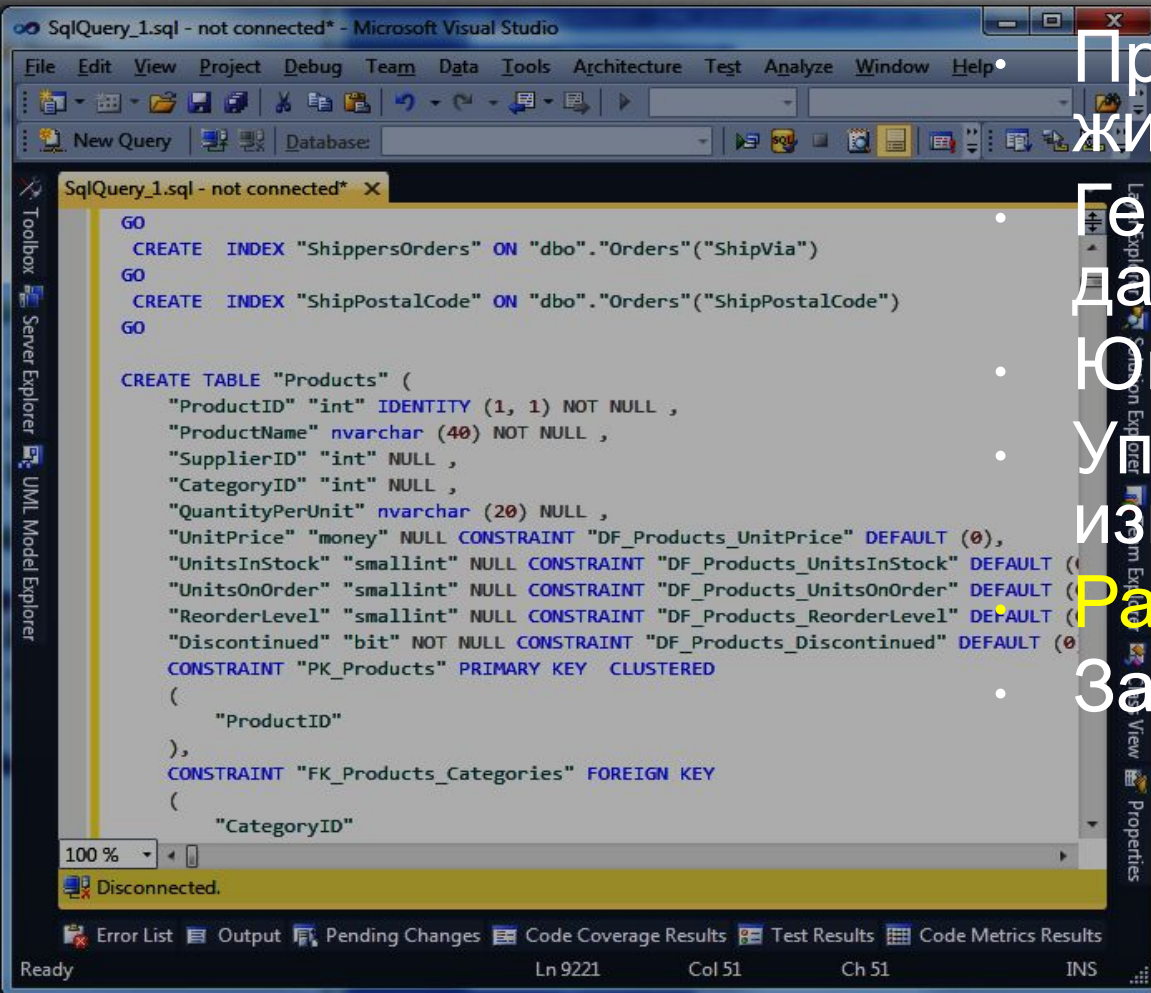


# *Демонстрация*



# Содержание

- Введение
- Проект БД и жизненный цикл БД
- Генерация тестовых данных
- Юнит-тестирование
- Управление изменениями
- **Развертывание**
- Заключение



```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```





# Развертывание

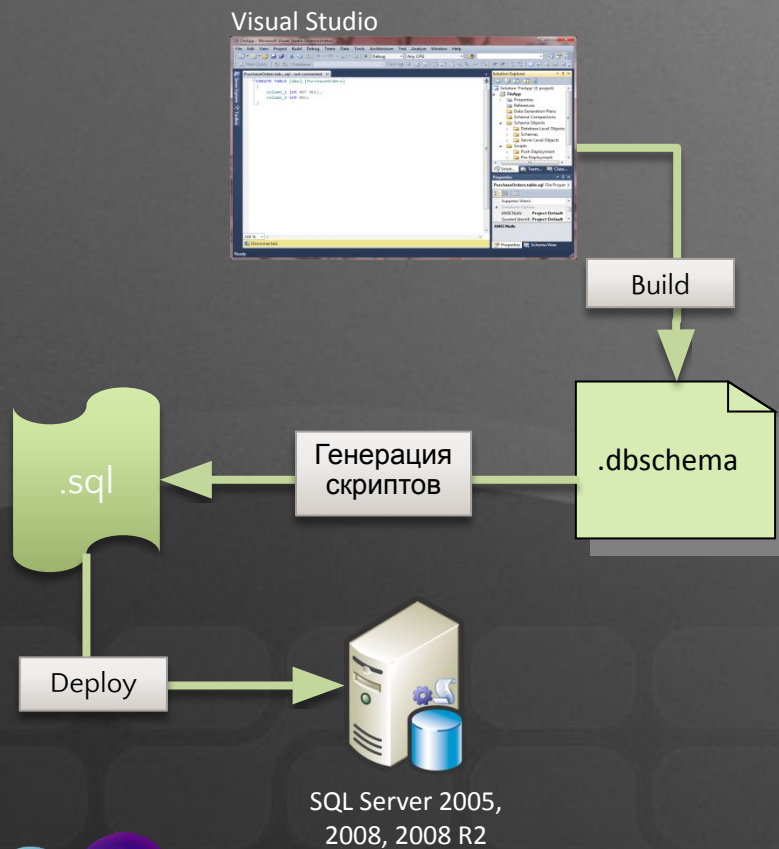
- Стандартный подход
  - Генерация скриптов изменений
    - Через сравнение схемы
  - Взаимодействие с администратором БД
- Новый подход
  - Представляем: *Data Tier Application Project System*



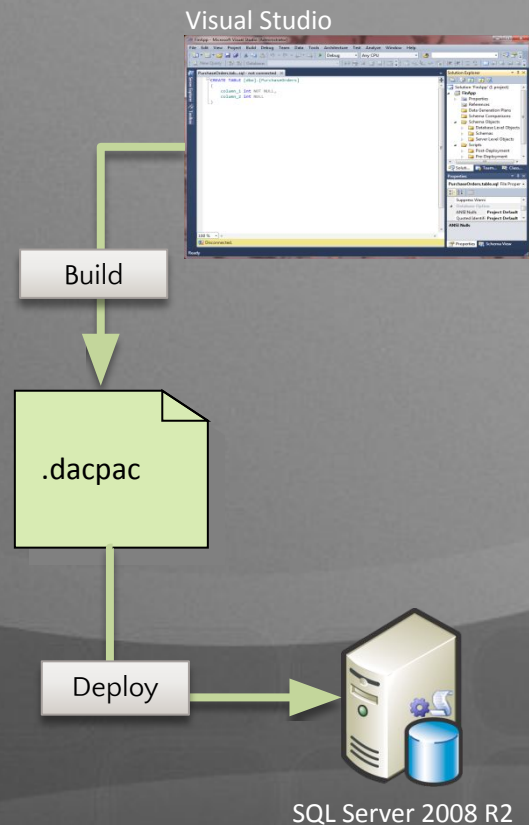


# Database Project vs. Data Tier Project

## Стандартный проект БД



## Data-tier Application Project V1



# Database Project vs. Data Tier Project

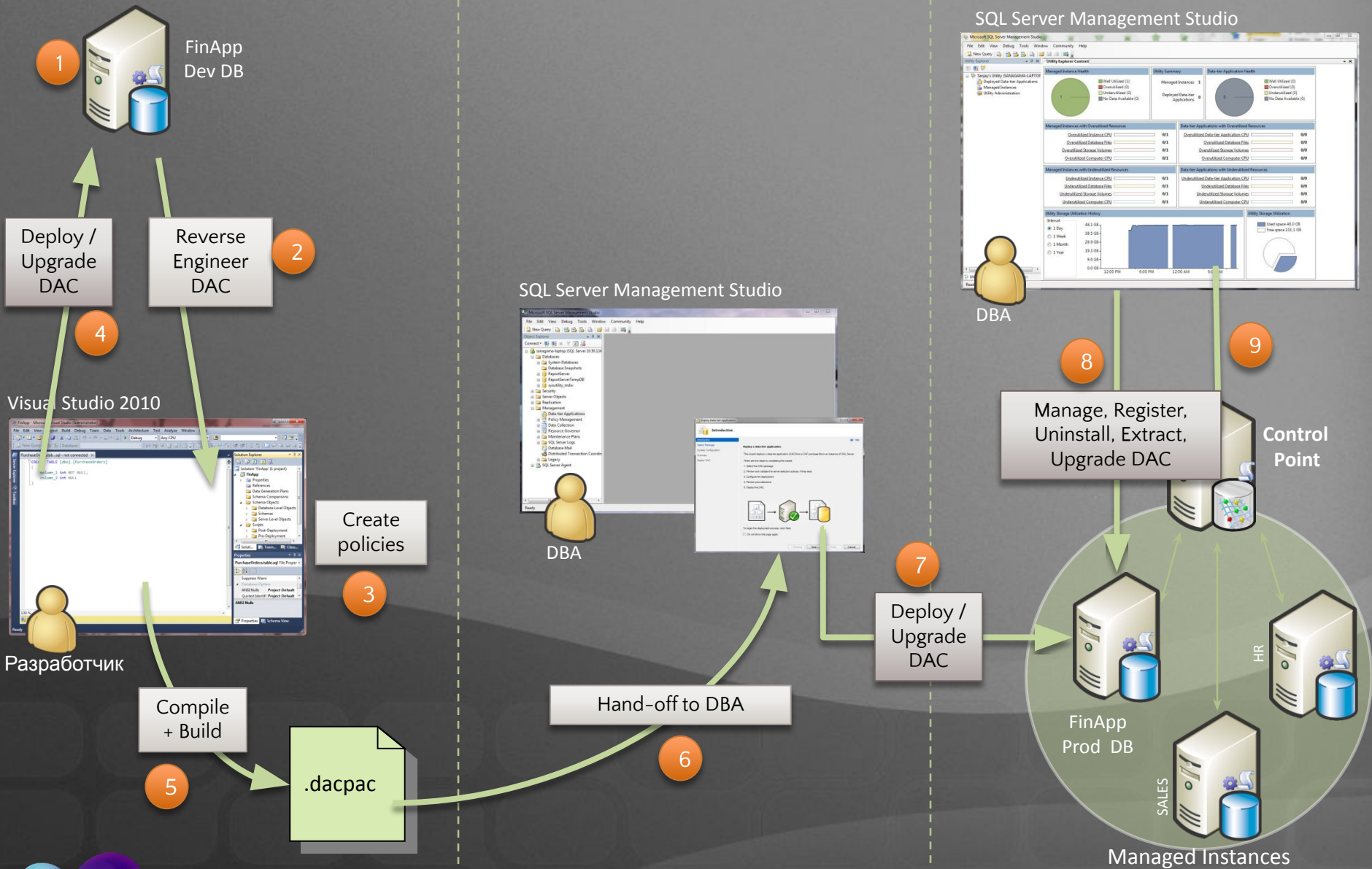
	Стандартный проект БД	Data-tier Application Projects V1
Целевые приложения	Бизнес критические системы	Приложения уровня подразделения
Поддержка БД	SQL 2005, 2008, 2010 и БД третьих производителей	SQL 2008 R2; планируется поддержка SQL 2008.
Сложность схемы	Не ограниченная	До тысячи объектов
Апгрейд (schema + data)	Проект генерирует .sql скрипты которые обновляют схему. Данные остаются на месте или подвергаются миграционным операциям в зависимости от сценариев апгрейда.	.sql скриптов нет. Единый пакет для развертывания или апгрейда содержащий всю необходимую информацию. Данные сохраняются в автоматическом режиме.
Типы поддерживаемых SQL объектов	Полная поддержка	Частичная поддержка
IntelliSense, Debugging, T-SQL Editor	Одинаковые возможности	



# Разработка

# Развертывание

# MANAGE



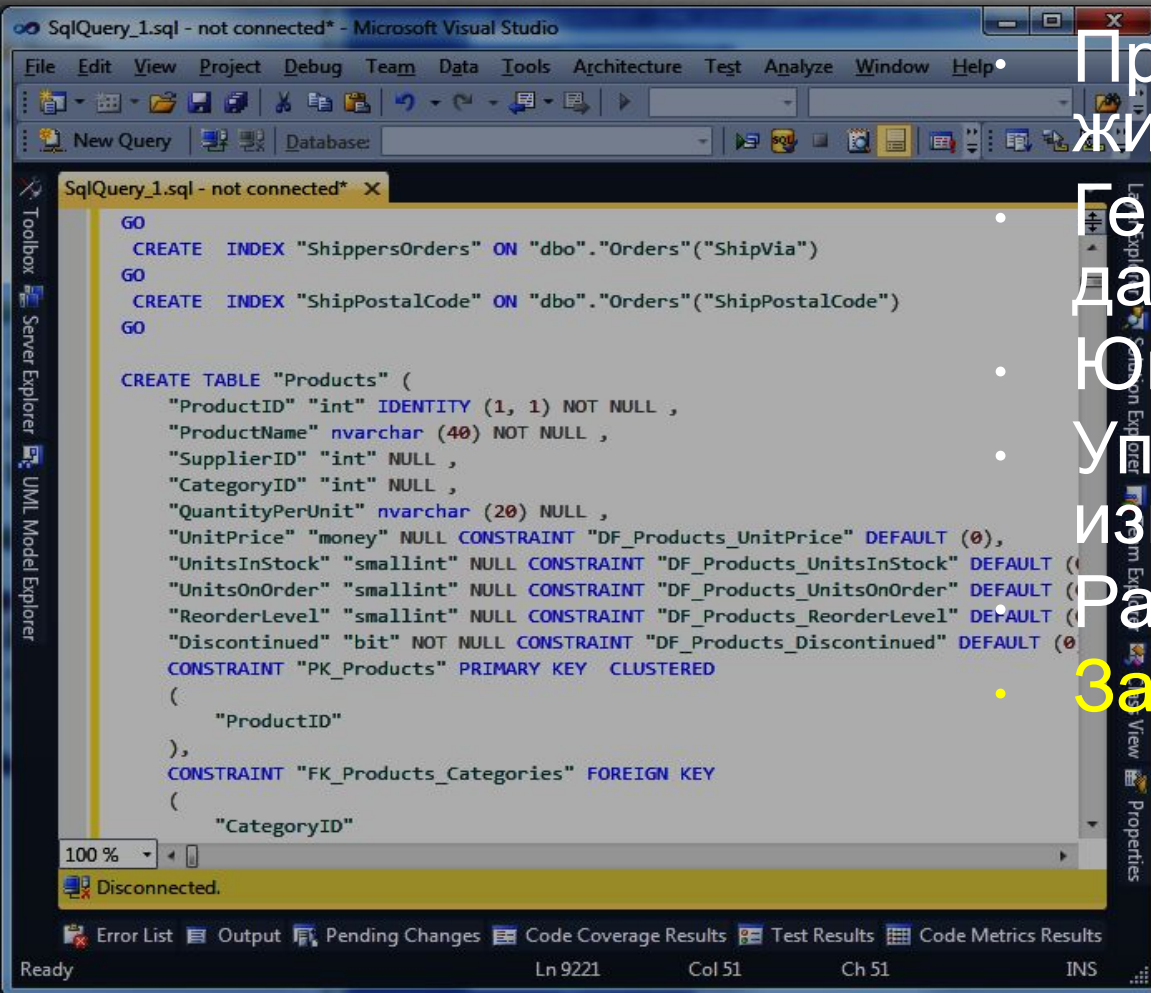
# *Демонстрация*





# Содержание

- Введение
- Проект БД и жизненный цикл БД
- Генерация тестовых данных
- Юнит-тестирование
- Управление изменениями
- Развертывание
- Заключение



The screenshot shows the Microsoft Visual Studio IDE with a SQL query editor. The query contains the following T-SQL code:

```
GO
CREATE INDEX "ShippersOrders" ON "dbo"."Orders"("ShipVia")
GO
CREATE INDEX "ShipPostalCode" ON "dbo"."Orders"("ShipPostalCode")
GO

CREATE TABLE "Products" (
  "ProductID" "int" IDENTITY (1, 1) NOT NULL ,
  "ProductName" "nvarchar" (40) NOT NULL ,
  "SupplierID" "int" NULL ,
  "CategoryID" "int" NULL ,
  "QuantityPerUnit" "nvarchar" (20) NULL ,
  "UnitPrice" "money" NULL CONSTRAINT "DF_Products_UnitPrice" DEFAULT (0),
  "UnitsInStock" "smallint" NULL CONSTRAINT "DF_Products_UnitsInStock" DEFAULT (0),
  "UnitsOnOrder" "smallint" NULL CONSTRAINT "DF_Products_UnitsOnOrder" DEFAULT (0),
  "ReorderLevel" "smallint" NULL CONSTRAINT "DF_Products_ReorderLevel" DEFAULT (0),
  "Discontinued" "bit" NOT NULL CONSTRAINT "DF_Products_Discontinued" DEFAULT (0)
  CONSTRAINT "PK_Products" PRIMARY KEY CLUSTERED
  (
    "ProductID"
  ),
  CONSTRAINT "FK_Products_Categories" FOREIGN KEY
  (
    "CategoryID"
  )
```

The status bar at the bottom indicates the file is "Disconnected" and shows the current cursor position at Ln 9221, Col 51, Ch 51.





# Заключение

- Разработка БД может быть полностью интегрирована в стандартный процесс ALM
- Инструментальные средства позволяют легко создавать объекты БД благодаря IntelliSense, встроенному отладчику
- Гибкие варианты развертывания могут снизить затраты на управление эксплуатационными БД



***Вопросы?***

