

БАЗЫ ДАННЫХ

Лекция 1
Введение в курс баз данных

Перед тем, как начать

- Лектор – Блинова Евгения Александровна,
eugenia.blinova@gmail.com
- Доклады
- Выступления на конференциях, олимпиадах и проекты
- Книги, ПО и задачи
- <https://diskstation.belstu.by:5001> student fitfit
- /Для_студентов_ФИТ_БГТУ/ПРЕПОДАВАТЕЛИ/
Блинова/Базы данных - 2 курс
- Посещение
- Сдача лабораторных работ

Описание курса

- 36 часов лекций, 36 часов лабораторных работ
- Задания на лабораторные работы - лабораторный практикум
- Microsoft SQL Server 2012
- Самостоятельная работа
- 2 контрольные работы и тестирование
- Экзамен – 2 теоретических вопроса и задача (ИСИТ, ПОИМБС)
- Для ПОИТ – экзамен в следующем семестре по материалу обоих семестров

База данных

- База данных – это совокупность взаимосвязанных данных

Требования к информации в БД

- Полезность - уменьшает информационную энтропию системы
- Полнота информации - информации должно быть достаточно, чтобы осуществить качественное управление
- Точность
- Достоверность - заведомо ошибочные данные не должны храниться в базе данных
- Непротиворечивость
- Актуальность

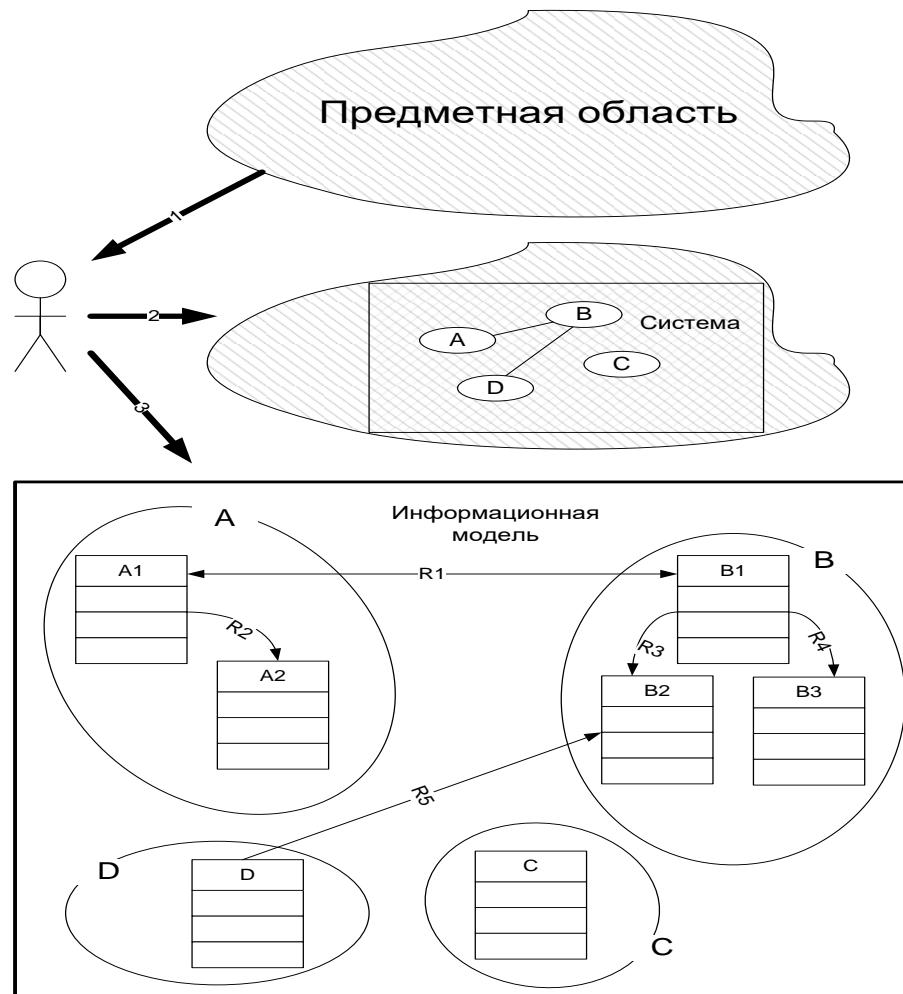
Проектирование БД

- Определение границ исследуемой области – предметной области
- Системный анализ – определение объектов и связей между ними
- Построение логической схемы базы данных в соответствии с определенными правилами – моделью данных
- Реализация базы данных – описание ее в терминах некоторой СУБД

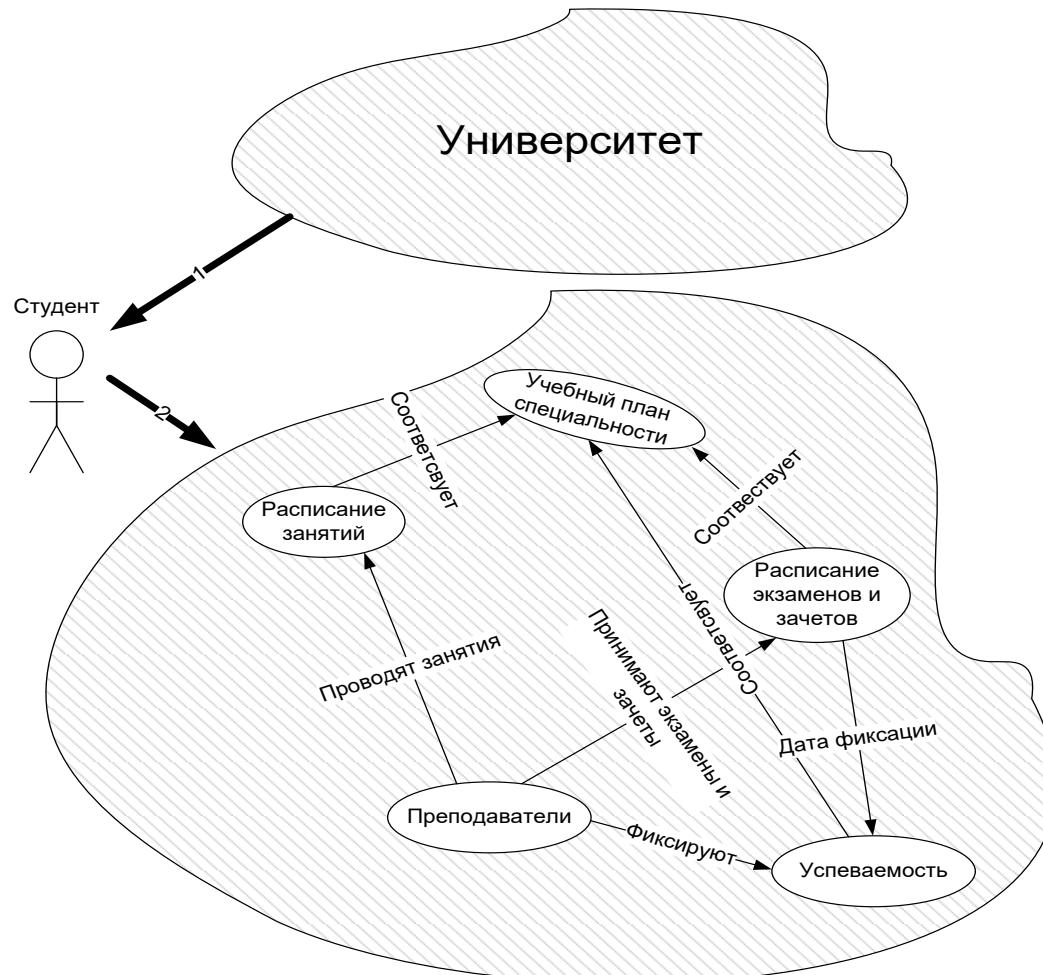
Проектирование БД

- **Предметная область** – часть реального мира, подлежащая изучению, с целью описания и управления
- **Предметная область** – это множество объектов и связей между этими объектами
- **Модель данных** – структурированное представление данных и связей между ними

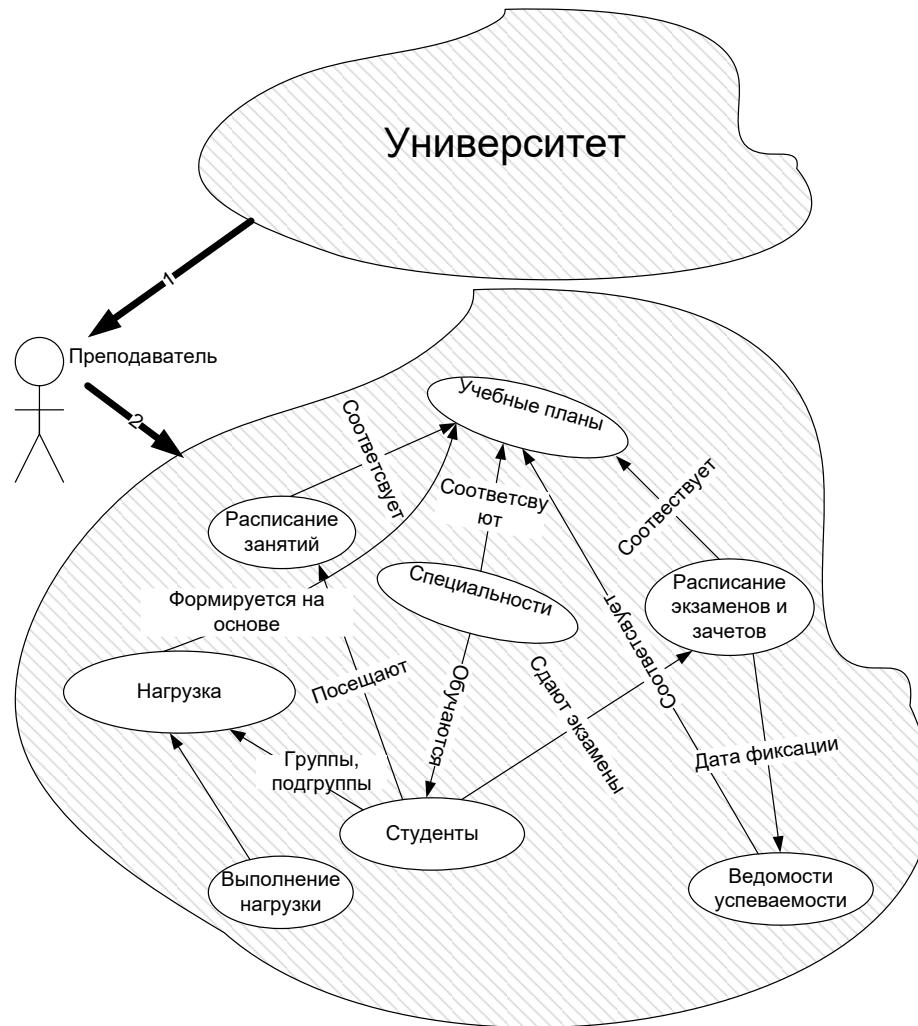
Проектирование БД



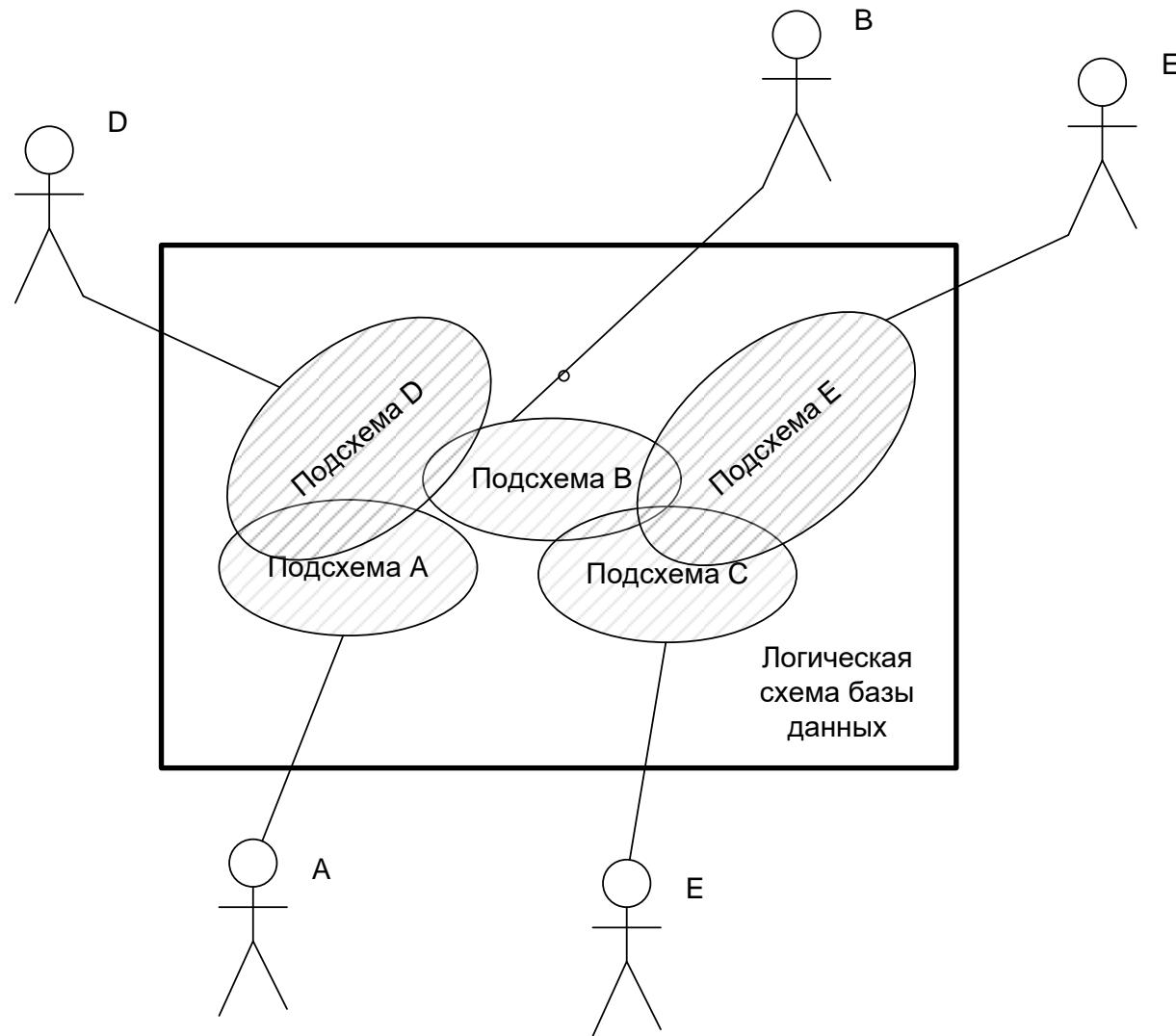
Пример



Пример



Логическая схема данных



База данных

- Хранилище динамически обновляемой информации
- Информация отражает состояние некоторой предметной области (объекта) и должна быть полезной, точной, актуальной и непротиворечивой
- Информация представлена в виде:
 - метаданных (описание модели данных)
 - данных
- Каждый пользователь базы данных знает только о существовании данных, необходимых для решения его задач
- Совокупность всех представлений - это логическая схема данных

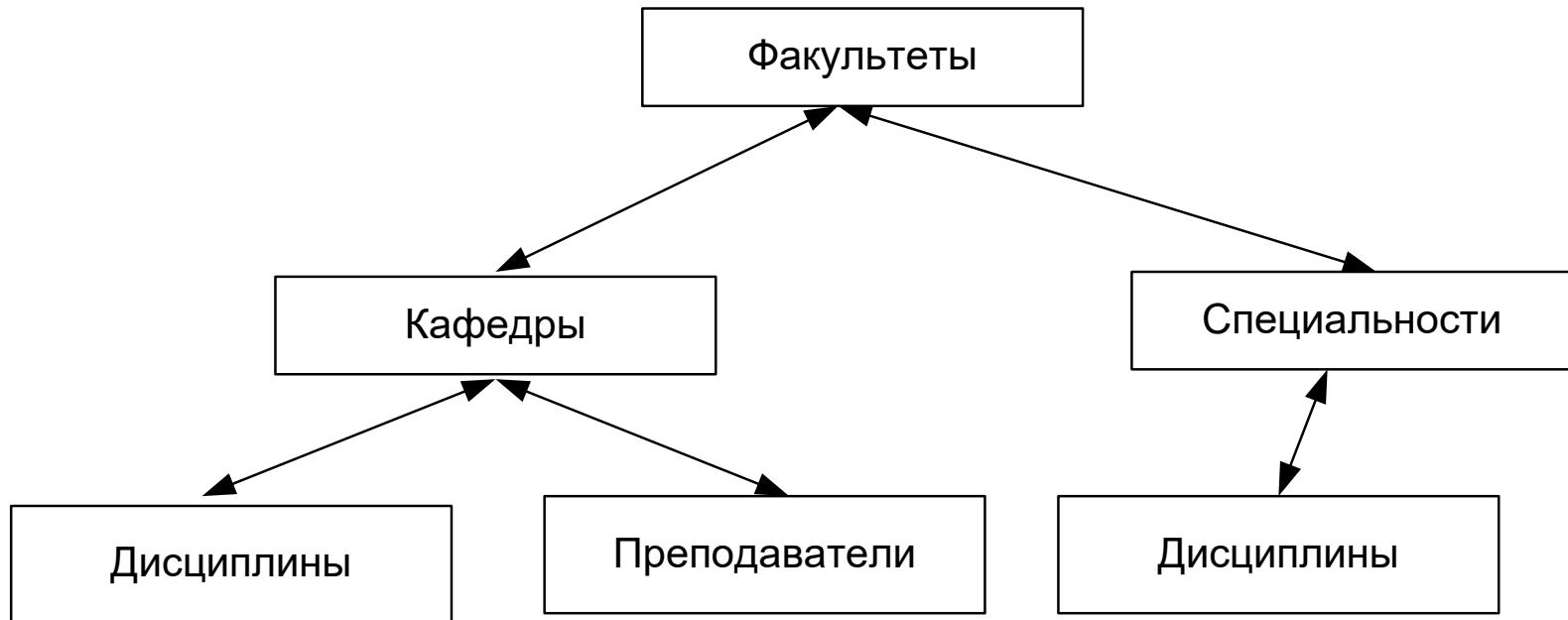
Система управления базами данных

- Программная реализация **технологии** хранения, извлечения, обновления и обработки данных в базе данных

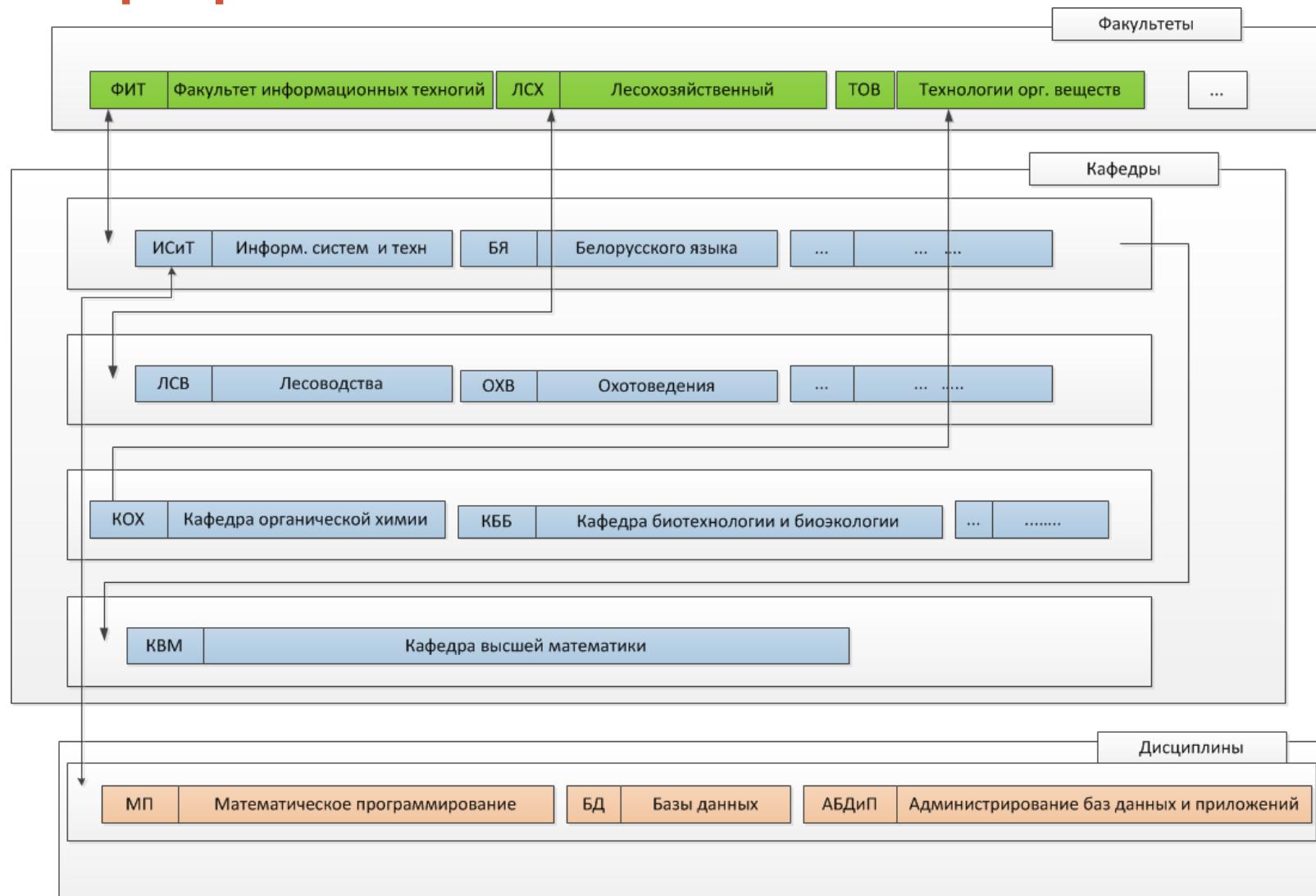
Модели данных

- Иерархическая
- Сетевая
- Реляционная

Иерархическая модель данных



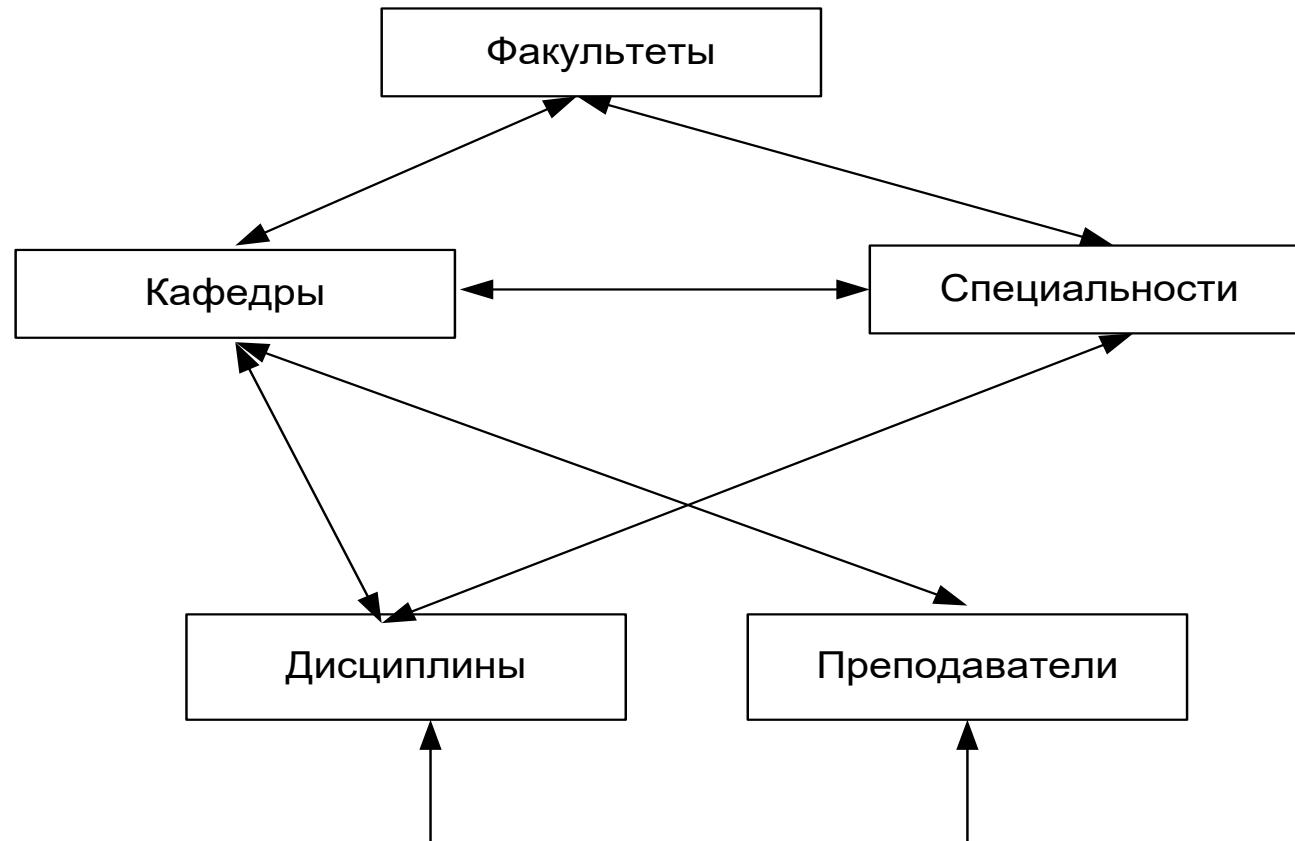
Иерархическая модель данных



Иерархическая модель данных

- Типичный представитель – IBM Information Management System

Сетевая модель данных



Сетевая модель данных

- Integrated Data Store от General Electric
- DMS от UNIVAC

Реляционная модель данных

- Основана на теории множеств
- Реляционная алгебра – Эдгар Франк Кодд (1923-2003)
- Данные имеют собственную природу, независимую от способа их использования
- Определения:
 - - **домен**: множество;
 - - **таблица**: отношение;
 - - **атрибут**: имя столбца таблицы (имя домена);
 - - **заголовок таблицы**: множество всех атрибутов;
 - - **кортеж**: элемент отношения или строка таблицы

Операции реляционной алгебры

- **UNION** (объединение)
- **INTERSECT** (пересечение)
- **MINUS** (разность)
- **TIMES** (декартово произведение)
- **WHERE** (ограничение)
- **PROJECT** (проекция)
- **JOIN** (соединение)
- **DIVIDE BY** (реляционное деление)
- **RENAME** (переименование)
- **:=** (присваивание).

Реляционная модель данных

- Relation – отношение
- Отношение может быть представлено в виде двумерной таблицы
- Реляционная база данных представляет собой набор взаимосвязанных таблиц
- Все объекты разделяются на типы
- Объекты одного и того же типа имеют свой набор атрибутов
- Один из атрибутов однозначно идентифицирует объект в таблице – первичный ключ

Реляционная модель данных

- Структурный аспект — данные в базе данных представляют собой набор отношений
- Аспект целостности — отношения (таблицы) отвечают определенным условиям целостности
- РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных
- Аспект обработки — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление)

Нормализация данных

- **Нормализация данных** – процесс преобразования таблиц базы данных к нормальной форме
- **Шесть нормальных форм** – 1NF, 2NF,...6NF.
- Широкое практическое применение имеют формы 1NF, 2NF, 3NF

Первая нормальная форма

- Таблица не должна содержать повторяющихся групп данных
- Атомарность – каждый столбец должен содержать одно неделимое значение
- Пример:
- ФИО – Адрес (город, улица, дом, квартира)
- Фильм – Исполнители (список актеров)

Первая нормальная форма

- УстраниТЬ повторяющиЕСЯ группы в отдельных таблицах
- Создать отдельную таблицу для каждого набора связанных данных
- Идентифицировать каждый набор связанных данных с помощью первичного ключа

Вторая нормальная форма

- Таблица находится в первой нормальной форме
- Каждый неключевой атрибут полностью функционально зависит от каждого возможного ключа
- Простой и составной ключ
- Пример:
- Студент – Университет – Средний балл – Стипендия

Вторая нормальная форма

- Создать отдельные таблицы для наборов значений, относящихся к нескольким записям
- Связать эти таблицы с помощью внешнего ключа

Третья нормальная форма

- Таблица находится во второй нормальной форме
- Отсутствуют транзитивные зависимости
- Пример:
- Студент – Группа – Факультет – Университет

SQL

- Язык **SQL** (**Structured Query Language**, язык структурированных запросов) – специализированный язык, предназначенный для написания запросов к реляционной БД

SQL

- 1986 – первый вариант стандарта
- 1989 – доработан стандарт
- 1992 – внесены значительные изменения (SQL2)
- 1999 – (SQL3) добавлены:
 - поддержка регулярных выражений
 - рекурсивных запросов
 - поддержка триггеров
 - базовые процедурные расширения
 - нескалярные типы данных
 - некоторые объектно-ориентированные возможности
- 2003 - поддержка XML
- 2006 - возможность совместно использовать в запросах SQL и XQuery
- 2008 - улучшены возможности оконных функций

Операторы SQL

- **DDL** - Data Definition Language - язык определения данных
- **DML** - Data Manipulation Language - язык манипулирования данными
- **TCL** - Transaction Control Language - язык управления транзакциями
- **DCL** - Data Control Language - язык управления данными

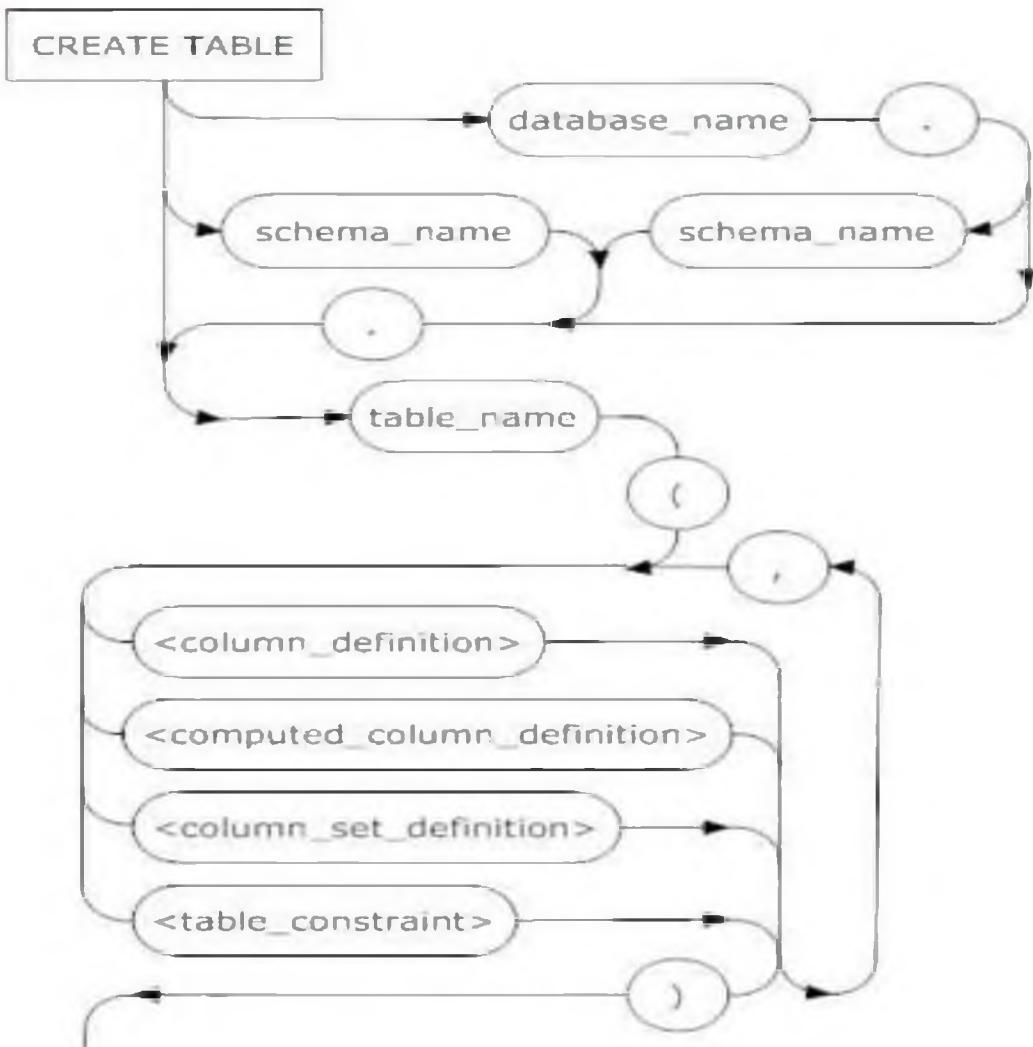
Операторы DDL

- Операторы DDL предназначены для **создания, удаления и изменения** объектов БД или сервера СУБД
- DDL включает операторы:
 - CREATE
 - ALTER
 - DROP

```
create тип имя дополнение
```

```
CREATE TABLE STUDENT (NAME nvarchar(50), GROUP_NUM int)
```

SQL Railroad Diagram



Операторы DML

- Операторы DML предназначены для **работы со строками таблиц**
- DML включает операторы:
 - **SELECT**
 - **INSERT**
 - **DELETE**
 - **UPDATE**

```
select список дополнение
```

SELECT * FROM STUDENT

SELECT NAME, GROUP_NUM FROM STUDENT

Операторы TCL

- Операторы TCL предназначены для **управления транзакциями**
- Транзакция – это несколько DML-операторов, которые либо **все** выполняются, либо все не выполняются
- TCL SQL включает операторы:
 - **BEGIN TRAN**
 - **SAVE TRAN**
 - **COMMIT TRAN**
 - **ROLLBACK TRAN**

```
begin tran дополнение
```

Операторы DCL

- Операторы DCL предназначены для **управления процессом авторизации**
- **Авторизация** – это процедура проверки разрешений на выполнение определенных операций
- **Принципал** – это объект сервера или БД, которому может быть выдано разрешение на выполнение операции, а также отобрано или запрещено разрешение
- DCL включает в себя операторы:
 - **GRANT**
 - **REVOKE**
 - **DENY**

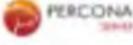
grant список on объект to принципал

Microsoft SQL Server

- **Microsoft SQL Server** — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft
 - Входит в «большую тройку»: Oracle, DB2
-
- Имеет свой диалект языка запросов SQL
 - Имеет свое процедурное расширение языка запросов Transact-SQL

#	Название
1 0	 MySQL <u>MySQL</u>
2 +1	 PostgreSQL <u>PostgreSQL</u>
3 -1	 MS SQL Server <u>MS SQL Server</u>
4 +1	 MongoDB <u>MongoDB</u>
5 +1	 SQLite <u>SQLite</u>
6 -2	 Oracle Database <u>Oracle Database</u>

<u>Год</u>	<u>SQL</u>	<u>Разработчик</u>
1995	✓	Oracle
1995	✓	сообщество
1988	✓	Microsoft
2009	✗	MongoDB
2000	✓	Hwaci, сообщество
1979	✓	Oracle

7 +1	 Firebird	Firebird	2000	▼
8 0	 CouchDB	CouchDB	2005	▶
8 -1	 DB2	DB2	1995	▼
9 new	 MariaDB	MariaDB	2009	▼
10 -1	 RavenDB	RavenDB	2009	▶
10 new	 redis	Redis	2009	▶
10 -1	 SAP	SAP ASE (ex: Sybase)	1988	▼
11 new	 Percona Server	Percona Server	2006	▼

сообщество

Apache

IBM

MariaDB Corporation Ab,
MariaDB Foundation, сообщество

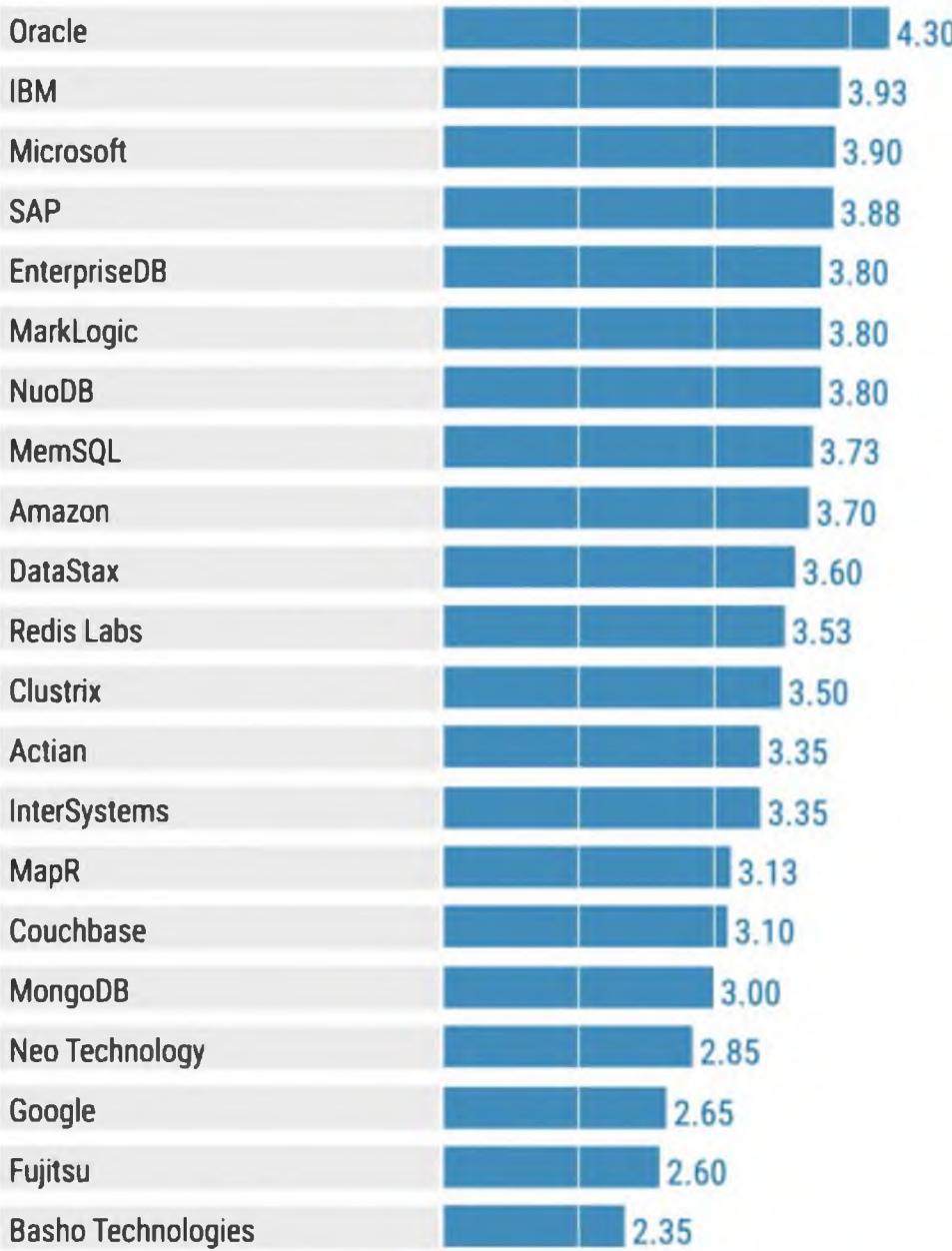
Hibernating Rhinos

Redis Labs, сообщество

SAP AG

Percona

Product or Service Scores for Traditional Transactions



1 2 3 4 5

As of August 2016

Rank			DBMS	Database Model	Score		
Sep 2018	Aug 2018	Sep 2017			Sep 2018	Aug 2018	Sep 2017
1.	1.	1.	Oracle	Relational DBMS	1309.12	-2.91	-49.97
2.	2.	2.	MySQL	Relational DBMS	1180.48	-26.33	-132.13
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1051.28	-21.37	-161.26
4.	4.	4.	PostgreSQL	Relational DBMS	406.43	-11.07	+34.07
5.	5.	5.	MongoDB	Document store	358.79	+7.81	+26.06
6.	6.	6.	DB2	Relational DBMS	181.06	-0.78	-17.28
7.	↑ 8.	↑ 10.	Elasticsearch	Search engine	142.61	+4.49	+22.61
8.	↓ 7.	↑ 9.	Redis	Key-value store	140.94	+2.37	+20.54
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	133.39	+4.30	+4.58
10.	10.	↓ 8.	Cassandra	Wide column store	119.55	-0.02	-6.65
11.	11.	11.	SQLite	Relational DBMS	115.46	+1.73	+3.42
12.	12.	12.	Teradata	Relational DBMS	77.38	-0.02	-3.52
13.	13.	↑ 16.	Splunk	Search engine	74.03	+3.53	+11.45
14.	14.	↑ 18.	MariaDB	Relational DBMS	70.64	+2.34	+15.17
15.	15.	↓ 13.	Solr	Search engine	60.20	-1.69	-9.71
16.	↑ 18.	↑ 19.	Hive	Relational DBMS	59.63	+1.69	+11.02
17.	17.	↓ 15.	HBase	Wide column store	58.47	-0.33	-5.87
18.	↓ 16.	↓ 14.	SAP Adaptive Server	Relational DBMS	58.04	-2.39	-8.71
19.	19.	↓ 17.	FileMaker	Relational DBMS	55.30	-0.75	-5.69
20.	↑ 21.	↑ 22.	Amazon DynamoDB	Multi-model	53.34	+1.69	+15.52
21.	↓ 20.	↓ 20.	SAP HANA	Relational DBMS	52.73	+0.80	+4.40
22.	22.	↓ 21.	Neo4j	Graph DBMS	40.10	-0.83	+1.67
23.	23.	23.	Couchbase	Document store	34.55	+1.59	+1.44
24.	24.	24.	Memcached	Key-value store	31.54	-1.38	+2.60
25.	25.	↑ 27.	Microsoft Azure SQL Database	Relational DBMS	25.25	-0.85	+3.65

CHALLENGERS

MarkLogic ●
MongoDB ●
EnterpriseDB ●
InterSystems ●
Redis Labs ●
DataStax ● ●

Couchbase ●
Clustrix ●
Basho Technologies ●
NuoDB ●
Google ●
MemSQL ●
Neo Technology ●

Actian ●

Fujitsu ●

NICHE PLAYERS

ABILITY TO EXECUTE →
COMPLETENESS OF VISION

LEADERS

- Microsoft
- Oracle
- SAP
- Amazon Web Services
- IBM

VISIONARIES

As of October 2016



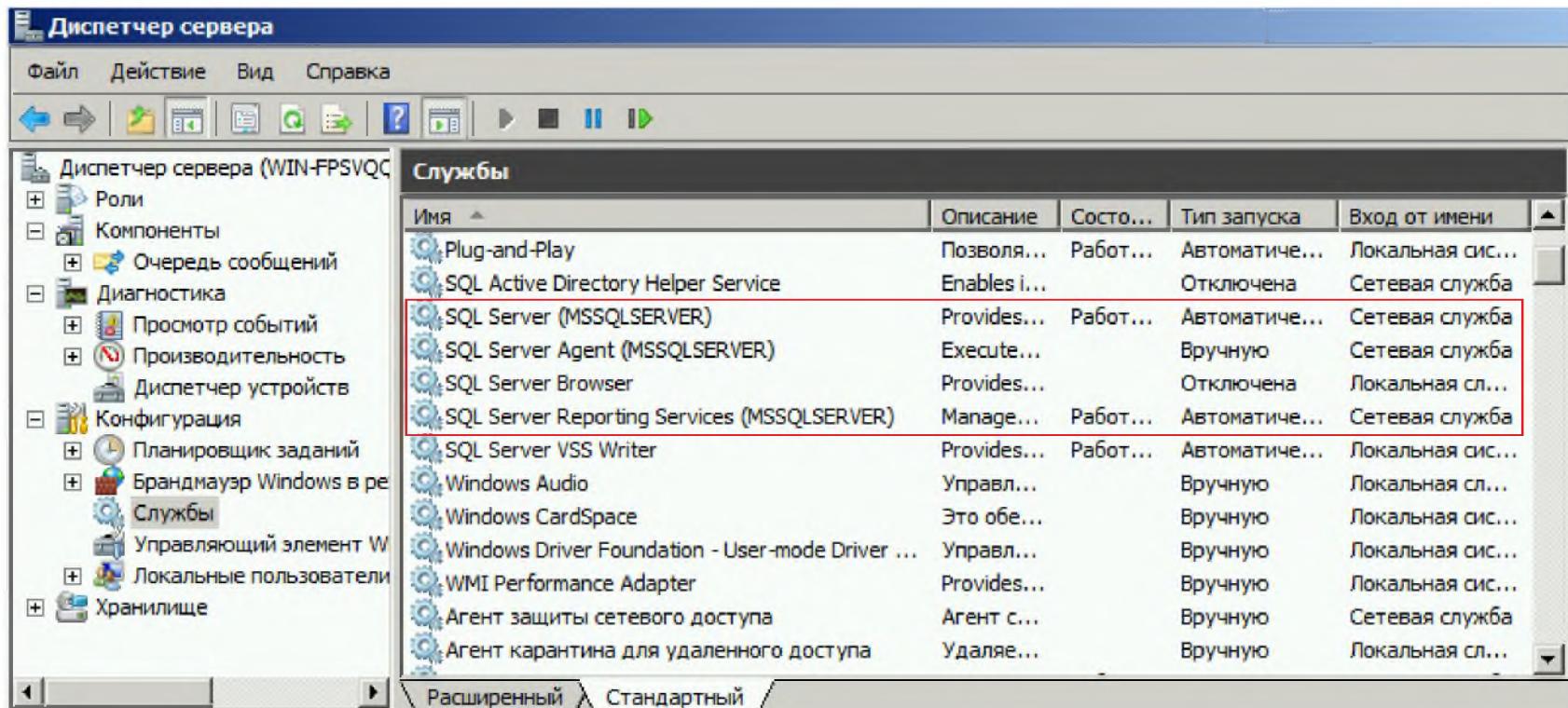
Microsoft SQL Server

Версия	Год	Название
1.0	1989	SQL Server 1.0 (16 bit)
1.1	1991	SQL Server 1.1 (16 bit)
4.21	1993	SQL Server 4.21
6.0	1995	SQL Server 6.0
6.5	1996	SQL Server 6.5
7.0	1998	SQL Server 7.0
8.0	2000	SQL Server 2000
8.0	2003	SQL Server 2000 64-bit
9.0	2005	SQL Server 2005
10.0	2008	SQL Server 2008
10.25	2010	Azure SQL DB
10.50	2010	SQL Server 2008 R2
11.0	2012	SQL Server 2012
12.0	2014	SQL Server In-Memory OLTP
13.0	2016	SQL Server 2016
14.0	2017	SQL Server vNext

Редакции Microsoft SQL Server 2012

- **SQL Server 2012 Enterprise Edition**
 - Не имеет ограничений по количеству поддерживаемых ядер
 - Не имеет ограничений по максимальному объему используемой памяти.
 - Максимальный размер реляционной базы данных — 524 Пб.
- **SQL Server 2012 Business Intelligence Edition**
 - Имеет ограничение — 4 процессора или 16 ядер на экземпляр;
 - Максимальный объем используемой памяти — 64 Гб;
 - Максимальный размер реляционной базы данных — 524 Пб.
- **SQL Server 2012 Standard Edition**
 - Имеет ограничение — 4 процессора или 16 ядер на экземпляр;
 - Максимальный объем используемой памяти — 64 Гб;
 - Максимальный размер реляционной базы данных — 524 Пб.
- **SQL Server 2012 Web Edition**
- **SQL Server 2012 Developer Edition .**
- **SQL Server 2012 Express Edition**
 - Имеет ограничение — 1 процессор (до 4 ядер);
 - Максимальный объем используемой памяти — 1 Гбайт;
 - Максимальный размер реляционной базы данных — 10 Гб.

Microsoft SQL Server



Службы Microsoft SQL Server

Наименование службы	Назначение
Database Engine	Управление реляционными БД
Analysis Services	Управление OLAP-кубами и интеллектуальный анализ данных
Integration Services	Поддержка решений по извлечению, преобразованию и загрузке данных
Reporting Services	Управление отчетами, построенными на основе SQL-запросов к реляционным БД
Full-Text Search	Управление полнотекстовым поиском
SQL Server Agent	Автоматизация административных задач
SQL Server Browser	Управление соединениями

Database Engine

- **Database Engine** является ядром системы управления реляционной БД
- Может быть установлено **несколько** экземпляров службы **Database Engine**
- При этом **только один** экземпляр может быть службой по умолчанию (с именем **MS SQL SERVER**), другие экземпляры должны иметь уникальные имена
- Каждый экземпляр службы **Database Engine** требует отдельной инсталляции, конфигурации и настройки безопасности
- Один **Database Engine** может обеспечить доступ к нескольким БД

Системные базы данных

Системная база данных	Назначение
master	Хранит все системные данные Database Engine, а также информацию о других БД.
msdb	Используется службами SQL Server Agent (выполнение заданий по расписанию), Database Mail (формирование уведомлений по электронной почте), а также хранит информацию о резервном копировании БД.
tempdb	Пространство для временных объектов Database Engine и пользовательских временных таблиц. База данных пересоздается при каждой перезагрузке
model	Шаблон, используемый при создании всех БД, управляемых экземпляром Database Engine.
resource	БД, используемая только для чтения. Содержит системные объекты экземпляра Database Engine. Файлы БД являются скрытыми и не отображаются в MSMS.

Утилиты Microsoft SQL Server

- SQL Server Management Studio
- SQL Server Books Online
- SQLCMD
- Microsoft SQL Configuration Manager

Вопросы?

БАЗЫ ДАННЫХ

Лекция 2
Основные операторы SQL

Пример

Счет № 1723423 Дата: 10.02.2017

Покупатель № : 392

Фамилия: Рыбаков

Имя: Евгений Николаевич

Телефон: (029)555 66 76, (017) 322 45 12

Адрес: г. Минск, ул. Калинина, 52а - 13

Артикул	Наименование	Цена	Количество	Стоимость
223355	Рамка для фото 20x30	25	3	75
338566	Альбом для фото 13x18	33	2	66
767111	Рамка для фото 13x18	12	2	24
655443	Фотобумага Lomond, 100 листов	20	2	40
Итого				205

Пример

- Первая нормальная форма?
- Вторая нормальная форма?
- Третья нормальная форма?

Первая нормальная форма

Счет №	Дата	Покупатель номер	Покупатель	Покупатель адрес	Телефон	Артикул	Наименование	Цена	Количество	Стоймость
17234 23	10.02.2 017	392	Рыбаков	г. Минск, ул. Калинина, 52а - 13	(029)555 66 76, (017) 322 45 12	223355	Рамка для фото 20x30	25	3	75
17234 23	10.02.2 017	392	Рыбаков	г. Минск, ул. Калинина, 52а - 13	(029)555 66 76, (017) 322 45 12	338566	Альбом для фото 13x18	33	2	66
17234 23	10.02.2 017	392	Рыбаков	г. Минск, ул. Калинина, 52а - 13	(029)555 66 76, (017) 322 45 12	767111	Рамка для фото 13x18	12	2	24
17234 23	10.02.2 017	392	Рыбаков	г. Минск, ул. Калинина, 52а - 13	(029)555 66 76, (017) 322 45 12	655443	Фотобумага Lomond, 100 листов	20	2	40

Вторая нормальная форма

- Товар

Артикул	Наименование	Цена
223355	Рамка для фото 20x30	25
338566	Альбом для фото 13x18	33
767111	Рамка для фото 13x18	12
655443	Фотобумага Lomond, 100 листов	20

- Заказ

Номер	Дата	Номер покупателя	Фамилия	Адрес	Телефон
1723423	10.02.2017	392	Рыбаков

- Заказано

Заказ	Товар	Количество
1723423	223355	3
1723423	338566	2
1723423	767111	2
1723423	655443	2

Третья нормальная форма

Артикул	Наименование	Цена
223355	Рамка для фото 20x30	25
338566	Альбом для фото 13x18	33
767111	Рамка для фото 13x18	12
655443	Фотобумага Lomond, 100 листов	20

Номер	Дата	Покупатель	Номер покупателя	Фамилия	Адрес	Телефон
1723423	10.02.2017	392	392	Рыбаков

Заказ	Товар	Количество
1723423	223355	3
1723423	338566	2
1723423	767111	2
1723423	655443	2

Таблицы

Покупатель
Номер покупателя
Фамилия
...

Товар
Артикул
Наименование
Цена

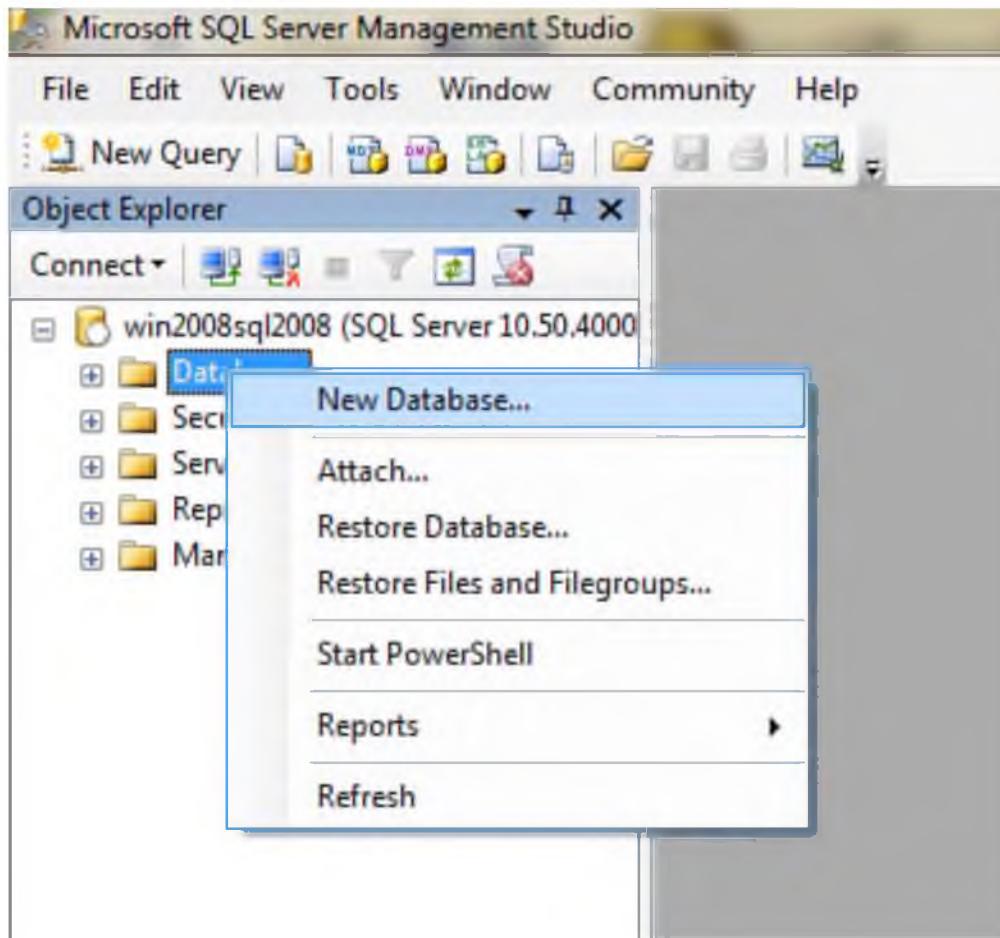
Заказ
Номер заказа
Дата
Покупатель

Заказано
Номер заказа
Товар
Количество

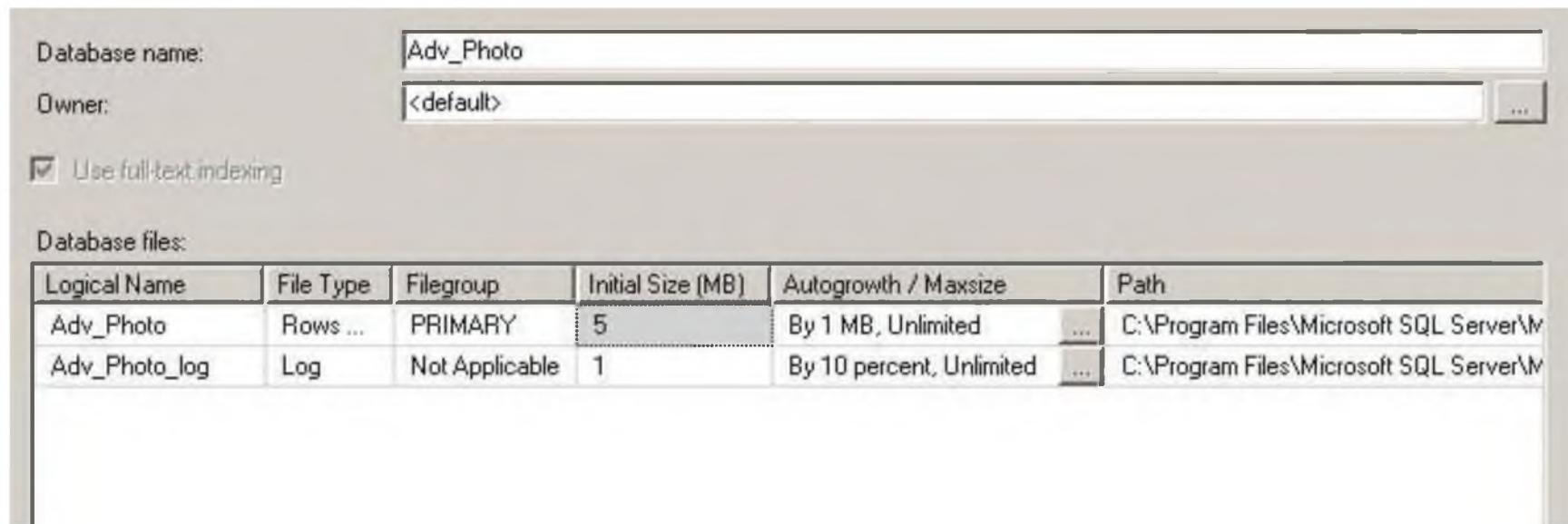
Таблицы



Создание базы данных



Создание базы данных



Создание базы данных

```
CREATE DATABASE [Adv_Photo]
CONTAINMENT = NONE
ON PRIMARY
(
    NAME = N'Adv_Photo',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\Adv_Photo.mdf' ,
    SIZE = 5120KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
(
    NAME = N'Adv_Photo_log',
    FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\Adv_Photo_log.ldf' ,
    SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
```

Создание таблиц

```
CREATE TABLE Покупатель(
    НомерПокупателя int NOT NULL,
    Фамилия nvarchar(50) NOT NULL,
    Имя nvarchar(50) NOT NULL,
    Отчество nvarchar(50) NULL,
    Адрес nvarchar(50) NULL,
    Телефон nvarchar(50) NULL,
    CONSTRAINT PK_Покупатель PRIMARY KEY (НомерПокупателя);
```

```
CREATE TABLE Товар(
    Артикул int NOT NULL,
    Наименование nvarchar(50) NOT NULL,
    Цена int NULL,
    CONSTRAINT PK_Товар PRIMARY KEY (Артикул)
```

Создание таблиц

```
CREATE TABLE Заказ(
    НомерЗаказа int NOT NULL,
    ДатаЗаказа date NOT NULL,
    Покупатель int NOT NULL,
    CONSTRAINT PK_Заказ PRIMARY KEY (НомерЗаказа));
```

```
CREATE TABLE Заказано(
    КодЗаказано int NOT NULL,
    НомерЗаказа int NOT NULL,
    Товар int NOT NULL,
    Количество int NOT NULL,
    CONSTRAINT PK_Заказано PRIMARY KEY (КодЗаказано));
```

Удаление таблицы

```
☒ DROP TABLE dbo.Покупатель
```

Изменение структуры таблицы

```
└─ ALTER TABLE Покупатель ADD Улица nvarchar(50);
```

```
└─ ALTER TABLE Покупатель DROP COLUMN Улица;
```

Добавление данных INSERT

```
INSERT INTO Покупатель(
    НомерПокупателя,
    Фамилия,
    Имя,
    Отчество,
    Адрес,
    Телефон)
VALUES (392, 'Рыбаков', 'Евгений', 'Николаевич',
    'г.Минск, ул.Калинина, 52а-13', '(029) 555 6676, (017) 322 45 12');
```

INSERT

```
INSERT INTO Товар (Артикул, Наименование, Цена)
VALUES (223355, 'Рамка для фото 20x30', 25);

INSERT INTO Товар (Артикул, Наименование, Цена)
VALUES (338566, 'Альбом для фото 13x18', 33),
       (767111, 'Рамка для фото 13x18', 12),
       (655443, 'Фотобумага Lomond, 100 листов', 20)
```

Обновление данных UPDATE

```
UPDATE Покупатель SET Отчество = 'Витальевич';
```

	НомерПокупат...	Фамилия	Имя	Отчество	Адрес	Телефон
	392	Рыбаков	Евгений	Витальевич	г.Минск, ул.Ка...	(029) 555 6676,,
	NULL	NULL	NULL	NULL	NULL	NULL

Удаление данных DELETE

```
| [DELETE Заказ WHERE [НомерЗаказа]=393;]
```

Выборка SELECT

The screenshot shows a SQL query window with the following content:

```
SELECT * FROM Товар;
```

The results pane displays a table with four rows of data:

	Артикул	Наименование	Цена
1	223355	Рамка для фото 20x30	25
2	338566	Альбом для фото 13x18	33
3	655443	Фотобумага Lomond, 100 листов	20
4	767111	Рамка для фото 13x18	12

SELECT

```
SELECT Артикул, Наименование FROM Товар;
```

100 %

Results

Messages

	Артикул	Наименование
1	223355	Рамка для фото 20x30
2	338566	Альбом для фото 13x18
3	655443	Фотобумага Lomond, 100 листов
4	767111	Рамка для фото 13x18

SELECT

```
SELECT count(*) FROM Товар;
```

100 %

Results

Messages

(No column name)

1

4

```
SELECT count(*) [Товаров в ассортименте] FROM Товар;
```

100 %

Results

Messages

Товаров в ассортименте

1

4

SELECT

```
| SELECT Артикул, Наименование FROM Товар WHERE Артикул = 223355;|
```

00 %

Results Messages

	Артикул	Наименование
1	223355	Рамка для фото 20x30

```
| SELECT Артикул, Наименование, Цена FROM Товар WHERE Артикул = 223355 OR Цена = 20;
```

00 %

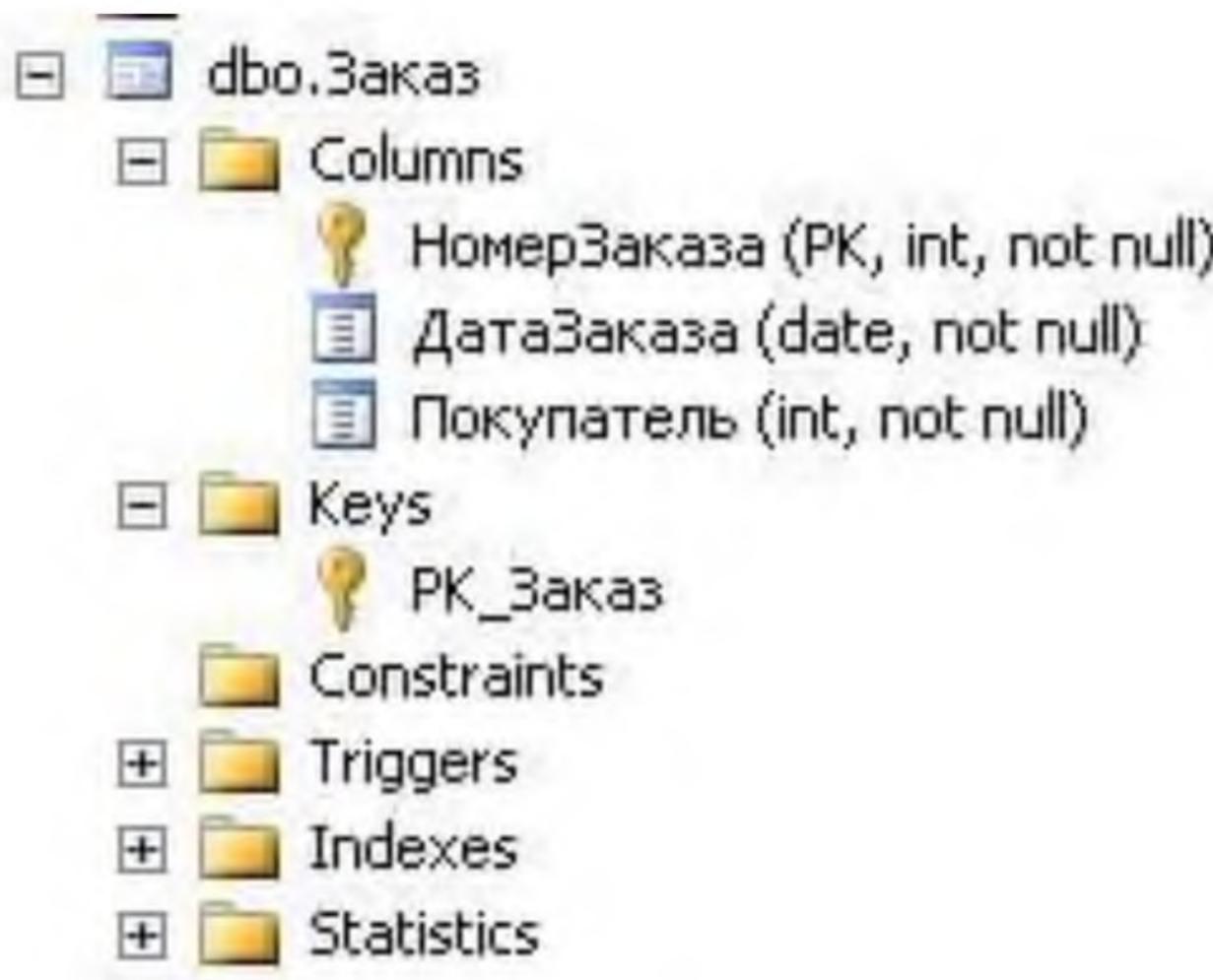
Results Messages

	Артикул	Наименование	Цена
1	223355	Рамка для фото 20x30	25
2	655443	Фотобумага Lomond, 100 листов	20

Первичный ключ

- Первичный ключ — в один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию)
- Уникальность
- Минимальность
- Простой
- Составной
- Естественный
- Суррогатный

Первичный ключ



Первичный ключ

-  dbo.Заказано

-  Columns

 КодЗаказано (PK, int, not null)

 НомерЗаказа (int, not null)

 Товар (int, not null)

 Количество (int, not null)

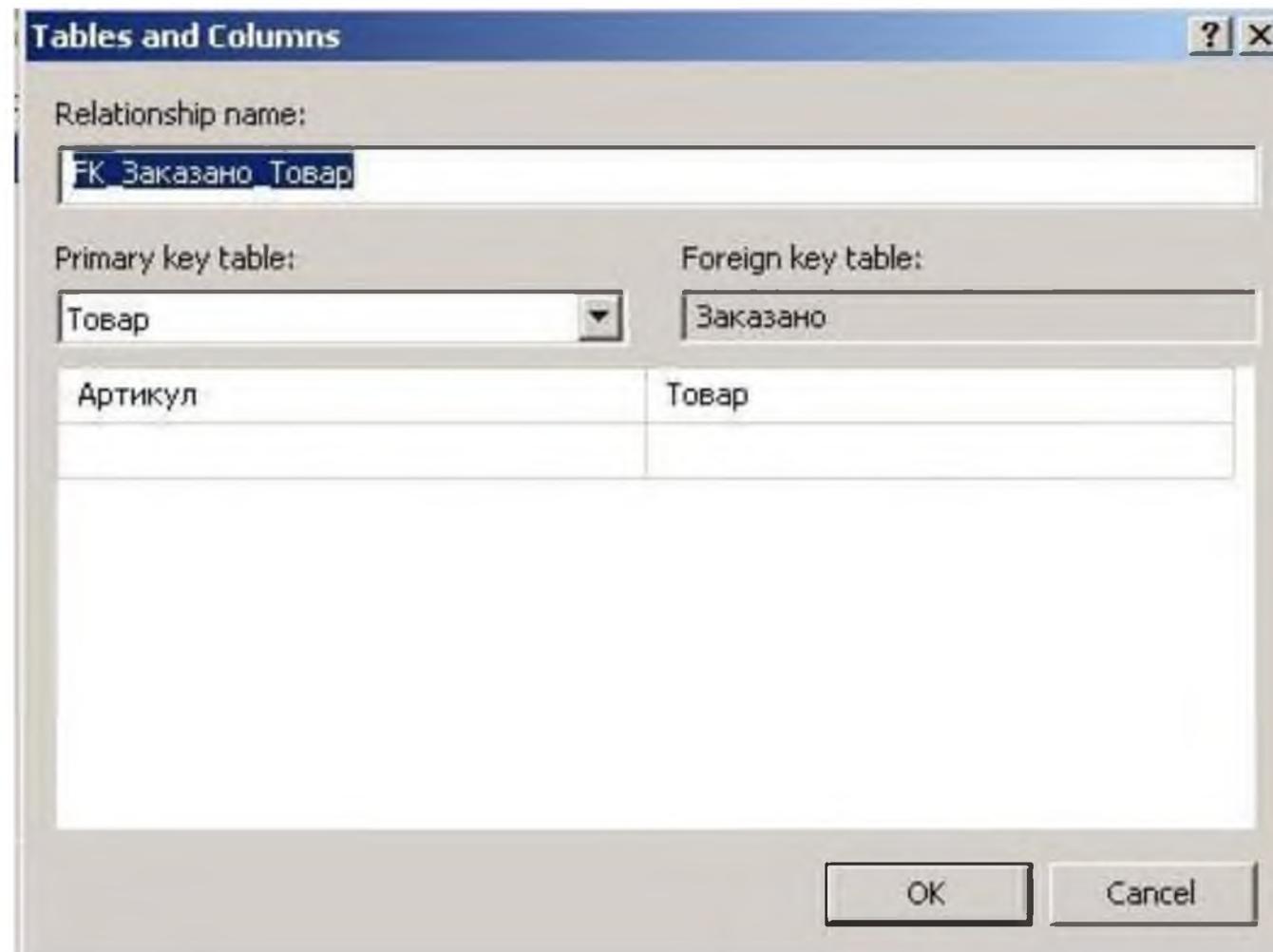
-  Keys

 PK_Заказано

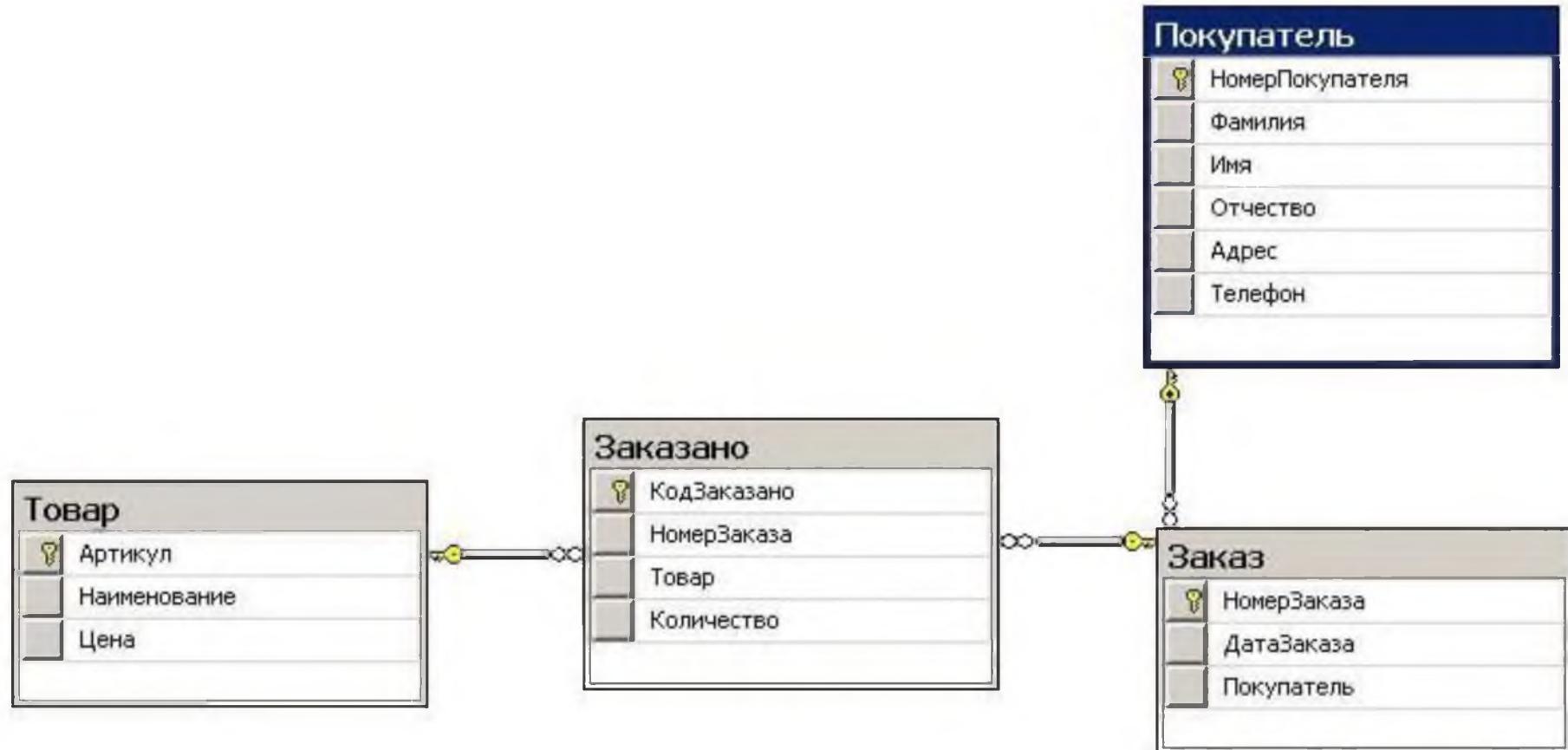
Внешний ключ

- Внешний ключ – подмножество атрибутов некоторой переменной отношения R_2 , значения которых должны совпадать со значениями некоторого потенциального ключа некоторой переменной отношения R_1 .
- Соблюдение ссылочной целостности

Внешний ключ



Внешний ключ



Внешний ключ

The screenshot shows the Object Explorer tree for the 'Заказ' table in the 'dbo' schema:

- dbo.Заказ** (Selected item)
 - Columns**
 - НомерЗаказа (PK, int, not null)
 - ДатаЗаказа (date, not null)
 - Покупатель (FK, int, not null)
 - Keys**
 - PK_Заказ
 - FK_Заказ_Покупатель
 - Constraints**
 - Triggers**
 - Indexes**
 - Statistics**

Внешний ключ

The screenshot shows the object browser for a database table named 'Заказано'. The table has the following structure:

- Columns**:
 - КодЗаказано (PK, int, not null)
 - НомерЗаказа (FK, int, not null)
 - Товар (FK, int, not null)
 - Количество (int, not null)
- Keys**:
 - PK_Заказано
 - FK_Заказано_Заказ
 - FK_Заказано_Товар
- Constraints**
- Triggers**
- Indexes**
- Statistics**

Внешний ключ

dbo.Покупатель
Columns
НомерПокупателя (PK, int, not null)
Фамилия (nvarchar(50), not null)
Имя (nvarchar(50), not null)
Отчество (nvarchar(50), null)
Адрес (nvarchar(50), null)
Телефон (nvarchar(50), null)
Keys
PK_Покупатель
Constraints
Triggers
Indexes
Statistics
dbo.Товар
Columns
Артикул (PK, int, not null)
Наименование (nvarchar(50), г)
Цена (int, null)
Keys
PK_Товар

NULL

```
INSERT INTO Товар (Артикул, Наименование, Цена)
VALUES (223395, null, null);
```

100 %

Messages

Msg 515, Level 16, State 2, Line 1
Cannot insert the value NULL into column 'Наименование', table 'Adv_Photo.d'
The statement has been terminated.

```
INSERT INTO Товар (Артикул, Наименование, Цена)
VALUES ('article', 'article', 20);
```

00 %

Messages

Msg 245, Level 16, State 1, Line 1
Conversion failed when converting the varchar value 'article' to data type i

Внешний ключ

```
INSERT INTO Заказ(НомерЗаказа, ДатаЗаказа, Покупатель)
VALUES
    (1723423, '10/02/2017', 392)
```

GO

The screenshot shows a SQL query window in SSMS. The query is:

```
SELECT * FROM Заказ;
```

The results pane displays one row of data:

	НомерЗаказа	ДатаЗаказа	Покупатель
1	1723423	2017-10-02	392

Внешний ключ

```
INSERT INTO Заказ(НомерЗаказа, ДатаЗаказа, Покупатель)
VALUES
    (1723424, '10/02/2017', 393)
GO
```

0 %

Messages

Msg 547, Level 16, State 0, Line 1

The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Заказ_Покупатель".

The statement has been terminated.

Вопросы?

БАЗЫ ДАННЫХ

Лекция 3 Типы данных
Ограничения целостности

Пример

- Преподаватели пишут учебники по предметам
- Преподаватель – Предмет – Учебник

Пример

- Преподаватель
 - IdTeacher
 - Name
 - Birthday
- Предмет
 - IdSubject
 - SubjectName
 - SubjectType
- Авторы
 - IdBookAuthor
 - IdTeacher
 - IdBook
- Учебник
 - IdBook
 - BookName
 - PublicationYear
 - NumberOfPages
 - IdSubject

Пример

- Студент – Группа – Староста

Пример

- Студент
- IdStudent
- StudentName
- Birthday
- Address
- **IdGroup**
- Группа
- IdGroup
- GroupNum
- Profession
- Faculty
- **Head**

Типы данных

- Числовые
- Денежные
- Символьные
- Дата и время
- Прочие

Числовые типы данных

- Точные
- Приближенные

Точные числовые типы данных

Тип данных	Диапазон значений	Количество байт
tinyint	0 – 255	1
smallint	-32768 – 32768	2
int	$-2^{31} – (2^{31}-1)$	4
bigint	$-2^{63} – (2^{63}-1)$	8
bit	0 или 1	1
decimal(p, s) numeric(p, s) $1 \leq p \leq 38,$ $0 \leq s < p$	$(-10^{38}+1) – (10^{38}+1)$	5 – 17 <p>p - максим. количество цифр (точность) s - количество цифр после точки (масштаб)</p>

Точность и масштаб

- Точность представляет собой количество десятичных знаков в числе
- Масштаб представляет собой количество десятичных знаков справа от десятичного разделителя
- Например:
- число 153,411
- точность 6
- масштаб 3

Точность и масштаб

Операция	Точность результата	Масштаб результата
$e_1 + e_2$	$\max(s_1, s_2) + \max(p_1 - s_1, p_2 - s_2) + 1$	$\max(s_1, s_2)$
$e_1 - e_2$	$\max(s_1, s_2) + \max(p_1 - s_1, p_2 - s_2) + 1$	$\max(s_1, s_2)$
$e_1 * e_2$	$p_1 + p_2 + 1$	$s_1 + s_2$
e_1 / e_2	$p_1 - s_1 + s_2 + \max(6, s_1 + p_2 + 1)$	$\max(6, s_1 + p_2 + 1)$
$e_1 \% e_2$	$\min(p_1 - s_1, p_2 - s_2) + \max(s_1, s_2)$	$\max(s_1, s_2)$

Числовые типы данных

```
use TEMPDB
go
create table EXACT_NUMBER
(
    N_BIT      bit,
    N_TINYINT   tinyint,
    N_SMALLINT  smallint,
    N_INT       int,
    N_BIGINT    bigint,
    N_DECIMAL   decimal(10,3)
);
go
insert into EXACT_NUMBER values (1, 123, 12345, 1234567890
insert into EXACT_NUMBER values (-1, 123, -12345, -1234567890
insert into EXACT_NUMBER values (0, 123, 12345, 1234567890
go
select * from EXACT_NUMBER;
go
```

N_BIT	N_TINYINT	N_SMALLINT	N_INT	N_BIGINT	N_DECIMAL
1	123	12345	1234567890	123456789012345678	1234567.890
1	123	-12345	-1234567890	-123456789012345678	-1234567.123
0	123	12345	1234567890	123456789012345678	1234567.123

Приближенные числовые типы данных

Тип данных	Диапазон значений	Количество байт
float(p) $1 \leq p \leq 53$	$-1.79 \times 10^{308} - 2.23 \times 10^{-308}$, 0, $2.23 \times 10^{-308} - 1.79 \times 10^{308}$	4 ($p \leq 24$) 8 ($p > 24$)
real float(24)	$-3.4 \times 10^{38} - 1.18 \times 10^{-38}$, 0, $1.18 \times 10^{-38} - 3.4 \times 10^{38}$	4 REAL является синонимом типа FLOAT(24).

Приближенные числовые типы данных

```
use TEMPDB
go
create table APPROXIMATE_NUMBER
(
    N_FLOAT      float(53) ,
    N_REAL       real
) ;
go

insert into APPROXIMATE_NUMBER values (1.1,2.2);
insert into APPROXIMATE_NUMBER values (1.12345678901
insert into APPROXIMATE NUMBER values (112.345678901
```

Денежные типы данных

Тип данных	Диапазон	Хранение
money	От -922 337 203 685 477,5808 до 922 337 203 685 477,5807	8 байт
smallmoney	От -214 748,3648 до 214 748,3647	4 байта

Символьные типы данных

Тип данных	Размер в символах	Количество байт
char(n)	1 – 8000	n
varchar(n)	1 – 4000	кол. символов + 2
varchar(max)	1 – $(2^{31}-1)$	кол. символов + 2
nchar(n)	1 – 8000	2n
nvarchar(n)	1 – 4000	2 кол. символов + 2
nvarchar(max)	1 – $(2^{30}-1)$	2хкол. символов + 2

Символьные типы данных

```
use TEMPDB
go

create table CHARACTER_DATA
(
    C_CHAR          char(100),
    C_VARCHAR       varchar(200),
    C_VARCHARM      varchar(max),
    C_NCHAR          nchar(100),
    C_NVARCHAR      nvarchar(200),
    C_NVARCHARM     nvarchar(max)
) ;
go
```

Типы данных для даты и времени

Тип данных	Диапазон, точность, формат	Количество байт
date	01.01.1753 – 31.12.9999, 1 день, YYYYMMDD	3
time(p) $0 \leq p \leq 7$	00:00:00.0000000 – 23:59:59.9999999, 100 нс, hh:mm:ss.nnnnnnnn	3 – 5
smalldatetime	01.01.1900 00:00 – 06.06.2079 23:59, 1 мин, YYYYMMDD hh:mm	4

Типы данных для даты и времени

Тип данных	Диапазон, точность, формат	Количество байт
datetime	01.01.1753 00:00:00.000 – 31.12.9999 23:59:59.999, 0.003 с, YYYYMMDD hh:mm:ss.nnn	8
datetime2(p) 0 ≤ p ≤ 7	01.01.0001 00:00:00.0000000 – 31.12.9999 23:59:59.9999999, 100 нс, YYYYMMDD hh:mm:ss.nnnnnnnn	6 – 8
datetimeoffset(p) 0 ≤ p ≤ 7	01.01.0001.00:00:00:0000000 +00:00 – 31.12.9999.23:59:59:9999999 +23:59, 100 нс, YYYYMMDDhh:mm:ss:nnnnnnnn±hh:mm	8 – 10

Типы данных для даты и времени

- SELECT
 - `CAST('2017-05-08 12:35:29. 1234567 +12:15' AS time(7)) AS 'time',`
 - `CAST('2017-05-08 12:35:29. 1234567 +12:15' AS date) AS 'date',`
 - `CAST('2017-05-08 12:35:29.123' AS smalldatetime) AS 'smalldatetime',`
 - `CAST('2017-05-08 12:35:29.123' AS datetime) AS 'datetime',`
 - `CAST('2017-05-08 12:35:29. 1234567 +12:15' AS datetime2(7)) AS 'datetime2',`
 - `CAST('2017-05-08 12:35:29.1234567 +12:15' AS datetimeoffset(7)) AS 'datetimeoffset';`

Типы данных для даты и времени

Тип данных	Вывод
time	12:35:29. 1234567
date	2017-05-08
smalldatetime	2017-05-08 12:35:00
datetime	2017-05-08 12:35:29.123
datetime2	2017-05-08 12:35:29. 1234567
datetimeoffset	2017-05-08 12:35:29.1234567 +12:15

Функции CAST и CONVERT

- CAST (expression AS data_type)
- CONVERT (data_type, expression[, style])

Без века (гг)	С веком (гггг)	Стандартная схема	Стиль
-	0 или 100	default	мес дд гггг чч:мм
1	101	США	1 = мм/дд/гг 101 = мм/дд/гггг
2	102	ANSI	2 = гг.мм.дд 102 = гггг.мм.дд
10	110	США	10 = мм-дд-гг 110 = мм-дд-гггг
12	112	ISO	12 = ггммдд 112 = ггггммдд

ФУНКЦИИ CAST И CONVERT

```
select cast('-123.123' as numeric(7,3)),    -- СИМВОЛЫ --> ЧИСЛО
       convert(numeric(7,3), '-123.123')
select cast( -123.123  as varchar(12)),   -- ЧИСЛО --> СИМВОЛЫ
       convert(varchar(12), -123.123)
select cast(getdate() as char(30)),          -- ДАТА --> СИМВОЛЫ
       convert(char(30), getdate()),
       convert(char(30), getdate(), 112) -- 112 - СТИЛЬ
select cast('20140107 22:00:01.123' as datetime), -- СИМВОЛЫ --> ДАТА
       convert(datetime, '20140107 22:00:01.123')
```

Двоичные типы данных

- Хранится последовательность битов
- Применяются для хранения изображений, звука, видео
- Можно хранить любые данные

Двоичные типы данных

Тип данных	Размер в байтах	Количество байт
binary(n)	1 – 8000	n
varbinary(n)	1 – 8000	кол. символов + 2
varbinary(max)	1 – $(2^{31}-1)$	кол. символов + 2

Двоичные типы данных

```
use TEMPDB
go
create table BINARY_DATA
(
    B_BINARY      binary(400),
    B_VARBINARY   varbinary(400),
    B_VARBINARYM  varbinary(max)
) ;
go
insert into BINARY_DATA
values (
    0x123456789012345,
    0x123456789012345,
```

Прочие типы данных

- TIMESTAMP
- UNIQUEIDENTIFIER
- XML
- HIERARCHYID
- GEOGRAPHY, GEOMETRY
- FILESTREAM
- SQLVARIANT
- TEXT, NTEXT, IMAGE

TIMESTAMP

- **ROWVERSION** - синоним **TIMESTAMP**
- Необходимо установить хронологию изменения данных
- занимает 8 байт
- Значения могут вводиться и изменяться **только сервером**

TIMESTAMP

```
use TEMPDB
go
create table TIMESTAMP_DATA1
(
    ID      int,
    timestamp        -- имя по умолчанию TIME
);
go
create table TIMESTAMP_DATA2
(
    ID          int,
    T_ROWVERSION  rowversion  -- имя обязательнс
);
go
insert into TIMESTAMP_DATA1 (ID)      values (1);
insert into TIMESTAMP_DATA2 (ID)      values (1);
```

ID	timestamp
1	0x0000000000007EF
ID	T_ROWVERSION
1	0x0000000000007FD

UNIQUEIDENTIFIER

- 16-байтовый идентификатор GUID
- Главная особенность – способность генерировать уникальные значения, которые с очень малой вероятностью могут быть независимо получены еще раз.
- Могут быть получены при помощи встроенной функции **NEWID**.
- Могут быть преобразованы из строковой константы в формате `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, где каждый символ x представляет шестнадцатеричную цифру в диапазоне 0–9 или a–f
- Пример:
- `8F1719F1-8B37-D821-B52D-00C04FC964FF`

UNIQUEIDENTIFIER

```
use TEMPDB
go
create table ID_DATA
(
    ID      uniqueidentifier,
    V       varchar(10)
);
go

insert into ID_DATA values(newid(), 'ABCD')

select *, datalength(ID) [datalength(ID)] from ID_DATA
```

ID	V	datalength(ID)
0E947E7A-CEDC-4253-8732-4A52A9024CF7	ABCD	16

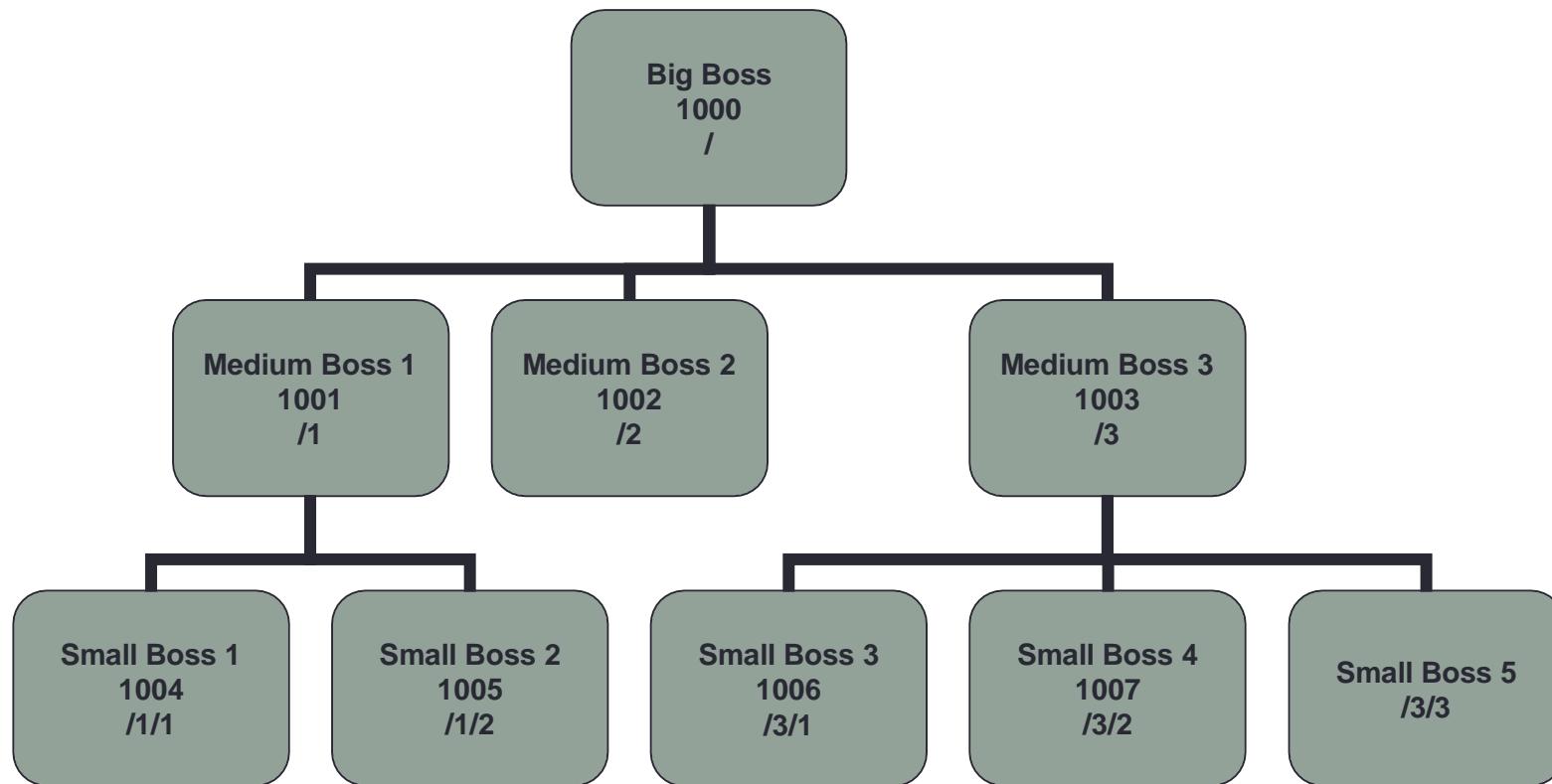
XML

- Тип данных, в котором хранятся XML-данные

```
<?xml version="1.0"?>
<journal>
    <title>Very Useful Journal</title>
    <contacts>
        <address>sdsds</address>
        <tel>8-3232-121212</tel>
        <tel>8-3232-121212</tel>
        <email>j@j.ru</email>
        <url>www.j.ru</url>
    </contacts>
    <issues-list>
        <issue index="2">
            <title>XML today</title>
            <date>12.09.98</date>
            <about>XML</about>
            <home-url>www.j.ru/issues/</home-url>
            <articles>
                <article ID="3">
                    <title>Issue overview</title>
                    <url>/article1</url>
                    <hotkeys>
                        <hotkey>language</hotkey>
                        <hotkey>markup</hotkey>
                    </hotkeys>
                </article>
            </articles>
        </issue>
    </issues-list>
</journal>
```

HIERARCHYID

- Системный тип данных переменной длины
- Используется для представления положения в иерархии



Пространственные типы

- **GEOGRAPHY**
- **GEOMETRY**
- **geography** хранит эллиптические данные, такие как координаты широты и долготы GPS
- **geometry** представляет данные в евклидовом пространстве (плоской системе координат)

FILESTREAM

- FILESTREAM размещает данные больших двоичных объектов (BLOB) типа **varbinary(max)** в файловой системе в виде файлов
- Можно вставлять, обновлять, запрашивать, выполнять поиск и выполнять резервное копирование данных FILESTREAM
- Следует использовать в следующих случаях:
 - средний размер сохраняемых объектов превышает 1 МБ;
 - важен быстрый доступ для чтения;

SQLVARIANT

- Тип данных, хранящий значения различных типов данных
- Максимальная длина значения типа **sql_variant** составляет 8016 байт
- Сюда включены структура и значение базового типа
- Максимальная длина значения соответствующего базового типа составляет 8 000 байт

SQLVARIANT

- Типы значений, которые не могут храниться в типе данных **sql_variant**

varchar(max)	varbinary(max)
nvarchar(max)	xml
text	ntext
image	rowversion (timestamp)
sql_variant	geography
hierarchyid	geometry
Определяемые пользователем типы	datetimeoffset

Приоритет

- Тип данных с меньшим приоритетом будет преобразован в тип данных с большим приоритетом
- Если неявное преобразование не поддерживается, возвращается ошибка

Приоритет

1. определяемые пользователем типы данных (высший приоритет);
2. sql_variant;
3. xml;
4. datetimeoffset;
5. datetime2;
6. datetime;
7. smalldatetime;
8. date;
9. time;
10. float;
11. real;
12. decimal;
13. money;
14. smallmoney;
15. bigint;
16. int;
17. smallint;
18. tinyint;
19. bit;
20. ntext;
21. text;
22. image;
23. timestamp;
24. uniqueidentifier;
25. nvarchar (включая nvarchar(max));
26. nchar;
27. varchar (включая varchar(max));
28. char;
29. varbinary (включая varbinary(max));
30. binary (низший приоритет).

Ограничения целостности

Условное обозначение	Действие ограничения целостности
data type тип данных	предотвращает появление в столбце значений, не соответствующих типу данных
not null запрет значений null	предотвращает появление в столбце значений null
default Значение по умолчанию	устанавливает значение в столбце по умолчанию при выполнении операции INSERT
primary key первичный ключ	предотвращает появление в столбце (группе столбцов) повторяющихся значений и пустого значения
foreign key внешний ключ	устанавливает связь между таблицей со столбцом, имеющим свойство foreign key (FK внешний ключ) и таблицей, имеющей столбец со свойством primary key (PK – первичный ключ); предотвращает несогласованные операции между PK и FK
unique уникальное значение	аналогично primary key, но допускает пустые значения и не может быть использован для связи с foreign key
check проверка значений	предотвращает появление в столбце значения, не удовлетворяющего логическому условию, записанному после check

Ограничения целостности

- Для ограничений целостности
 - PRIMARY KEY
 - FOREIGN KEY
 - UNIQUE
 - CHECK
- может быть задано **имя**
- Если это имя не задано, при создании таблицы сервер назначает ограничениям собственные имена

PRIMARY KEY

- Столбец или группа столбцов, имеющие уникальные значения для каждой строки, называется **ключом**
- **Create table FACULTY** -- факультет

(**FACULTY** char(10), -- идентификатор
FACULTY_NAME varchar(50)); -- полное имя

```
insert into FACULTY (FACULTY, FACULTY_NAME )
    values ('ХТИТ', 'Химическая технология и техника');
insert into FACULTY (FACULTY, FACULTY_NAME )
    values ('ХТИТ', 'Химическая технология и техника');
select * from FACULTY;
```

FACULTY	FACULTY_NAME
ХТИТ	Химическая технология и техника
ХТИТ	Химическая технология и техника

PRIMARY KEY

- Create table **FACULTY** --факультет
- (**FACULTY** **char(10)** **primary key**, --идентификатор
- **FACULTY_NAME** **varchar(50)** --полное имя
-);

```
insert into FACULTY    (FACULTY,    FACULTY_NAME )
    values ('ХТИТ', 'Химическая технология и техника');
insert into FACULTY    (FACULTY,    FACULTY_NAME )
    values ('ХТИТ', 'Химическая технология и техника');
select * from FACULTY;
```

Msg 2627, Level 14, State 1, Line 4

Violation of PRIMARY KEY constraint 'PK_FACULTY_81E78829173876EA'. Cannot insert duplicate key in object 'dbo.FACULTY'.

The statement has been terminated.

PRIMARY KEY

```
Create table FACULTY  
( FACULTY char(10)  
    constraint PK_FACULTY_FACULTY  
    primary key,  
    FACULTY_NAME varchar(50));
```

```
Create table FACULTY  
( FACULTY char(10),  
    FACULTY_NAME varchar(50),  
    constraint PK_FACULTY_FACULTY  
    primary key (FACULTY));
```

PRIMARY KEY

- **Create table SCHEDULE_TEACHER** -- расписание преподавателей
- (**CLASSDATE smalldatetime**, -- дата и время занятий
- **TEACHER char(10)**, -- преподаватель
- **SUBJECT char(10)**, -- дисциплина
- **AUDITORIUM char(10)**, -- аудитория
- **constraint PK_S_TEACHER primary key**
 (CLASSDATE, TEACHER));

NOT NULL

```
insert into FACULTY (FACULTY, FACULTY_NAME)
    values ('ИДиП', 'Издательское дело и полиграфия');
insert into FACULTY (FACULTY)
    values ('ХТИТ'); -- неявный ввод значения NULL
insert into FACULTY (FACULTY, FACULTY_NAME)
    values ('ЛХФ', NULL); -- явный ввод значения NULL
```

```
select * from FACULTY;
```

FACULTY	FACULTY_NAME
ИДиП	Издательское дело и полиграфия
ЛХФ	NULL
ХТИТ	NULL

```
update FACULTY set FACULTY_NAME = NULL where FACULTY = 'ИДиП';
select * from FACULTY;
```

DEFAULT

```
create table FACULTY -- факультеты
(
    FACULTY      char(10),                      -- идентификатор факультета
    FACULTY_NAME varchar(50) default '???',    -- полное наименование
    constraint FACULTY_PK primary key(FACULTY)
);
go

insert into FACULTY (FACULTY, FACULTY_NAME)
    values ('ИДиП', 'Издательское дело и полиграфия');
insert into FACULTY (FACULTY)
    values ('ХТиТ');                         -- неявный ввод значения default
insert into FACULTY (FACULTY, FACULTY_NAME)
    values ('ЛХФ', NULL);                    -- явный ввод значения NULL
insert into FACULTY (FACULTY, FACULTY_NAME)
    values ('ТТЛП', NULL);                  -- явный ввод значения NULL |
insert into FACULTY (FACULTY, FACULTY_NAME)
    values ('ИЭФ', default);                -- явный ввод значения default
update FACULTY set FACULTY_NAME = NULL where FACULTY = 'ИДиП';
update FACULTY set FACULTY_NAME = default where FACULTY = 'ЛХФ';
select * from FACULTY;
```

FACULTY	FACULTY_NAME
ИДиП	NULL
ИЭФ	???
ЛХФ	???
ТТЛП	NULL
ХТиТ	???

DEFAULT

```
create table FACULTY -- факультеты
{
    FACULTY      char(10),                      -- идентификатор факуль:
    FACULTY_NAME varchar(50) not null default '???' , -- полное наименование
    constraint FACULTY_PK primary key(FACULTY)
};
```

FOREGN KEY

- **Внешний ключ** – ограничение целостности, основанное на связи, установленной между двумя таблицами БД
- Виды связей:
 - 1:1 – каждому экземпляру **одной** таблицы соответствует в точности **один** экземпляр второй и **наоборот**
 - 1:n – может существовать экземпляр **одной** таблицы, который соответствует **нескольким** экземплярам другой, и **обратное не допускается**
 - m:n – экземпляр **одной** таблицы соответствует **нескольким** экземплярам другой таблицы и **наоборот**

FOREGN KEY

```
create table PULPIT -- кафедра
(
    PULPIT      char(20)          -- идентификатор кафедры
                    constraint PULPIT_PK primary key,
    PULPIT_NAME varchar(100)       -- полное наименование кафедры
                    default '??',
    FACULTY     char(10)           -- идентификатор факультета
                    constraint PULPIT_FACULTY_FK foreign key
                    references FACULTY(FACULTY)
);
```

FACULTY	FACULTY_NAME
ИДиП	Издательское дело и полиграфия
ИЭФ	Инженерно-экономический факультет
ЛХФ	Лесохозяйственный факультет
ТОВ	Технология органических веществ
ТТЛП	Технология и техника лесной промышленности
ХтиТ	Химическая технология и техника

FOREGN KEY – INSERT

```
insert into PULPIT(PULPIT, PULPIT_NAME, FACULTY) -- OK
    values('ИСиГ', 'Информационных систем и технологий ', 'ИДиП' );
insert into PULPIT(PULPIT, PULPIT_NAME, FACULTY) -- OK
    values('ПСиСОИ', 'Полиграфического оборудования и систем обработки информации ', NULL);
insert into PULPIT(PULPIT, PULPIT_NAME, FACULTY) -- OK
    values('ЛУ', 'Лесоустройства', 'ЛХФ');
insert into PULPIT(PULPIT, PULPIT_NAME, FACULTY) -- error 547
    values('ЛВ', 'Лесоводства', 'XXXX');
insert into PULPIT(PULPIT, PULPIT_NAME, FACULTY) -- OK
    values('ЛВ', 'Лесоводства', 'ЛХФ');
insert into PULPIT(PULPIT, PULPIT_NAME, FACULTY) -- OK
    values ('ТЛ', 'Транспорта леса',NULL);
select * from PULPIT order by FACULTY
```

FOREGN KEY – INSERT

```
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
Msg 547, Level 16, State 0, Line 7
The INSERT statement conflicted with the FOREIGN KEY constraint "PULPIT_FACULTY_FK".
The conflict occurred in database "BSTU", table "dbo.FACULTY", column 'FACULTY'.
The statement has been terminated.
(1 row(s) affected)
(1 row(s) affected)
(5 row(s) affected)
```

PULPIT	PULPIT_NAME	FACULTY
ПОиСОИ	Полиграфического оборудования и систем обработки...	NULL
ТЛ	Транспорта леса	NULL
ИСиТ	Информационных систем и технологий	ИДиП
ЛВ	Лесоводства	ЛХФ
ЛУ	Лесоустройства	ЛХФ

FOREGN KEY – UPDATE

```
update PULPIT set FACULTY = 'ИДиП' where PULPIT = 'ПОиСОИ'; -- OK
update PULPIT set FACULTY = 'ЛХФ' where FACULTY is NULL; -- OK
update PULPIT set FACULTY = NULL where PULPIT = 'ИСиТ'; -- OK
update PULPIT set FACULTY = 'XXX' where FACULTY = 'ЛХФ'; -- error 547

select * from PULPIT order by FACULTY
```

FOREGN KEY – UPDATE

```
(1 row(s) affected)
```

```
(1 row(s) affected)
```

```
(1 row(s) affected)
```

Msg 547, Level 16, State 0, Line 4

The UPDATE statement conflicted with the FOREIGN KEY constraint "PULPIT_FACULTY_FK".

The conflict occurred in database "BSTU", table "dbo.FACULTY", column 'FACULTY'.

The statement has been terminated.

PULPIT	PULPIT_NAME	FACULTY
ИСиТ	Информационных систем и технологий	NULL
ПОиСОИ	Полиграфического оборудования и систем обработки...	ИДиП
ТЛ	Транспорта леса	ЛХФ
ЛВ	Лесоводства	ЛХФ
ЛУ	Лесоустройства	ЛХФ

FOREGN KEY – DELETE

```
select count(*) 'количество строк до DELETE' from PULPIT;  
delete PULPIT; -- удаление всех строк PULPIT  
select count(*) 'количество строк после DELETE' from PULPIT;
```

количество строк до DELETE

5

количество строк после DELETE

0

CHECK

CHECK

```
insert into TEACHER      (TEACHER,    TEACHER_NAME, PULPIT )          -- OK
                  values ('СМЛВ',      'Смелов Владиславович', 'ИСиТ');
insert into TEACHER      (TEACHER,    TEACHER_NAME, PULPIT, GENDER )   -- OK
                  values ('АКНВЧ', 'Акунович Станислав Иванович', 'ИСиТ', 'м');
insert into TEACHER      (TEACHER,    TEACHER_NAME, PULPIT, GENDER )   -- OK
                  values ('БРС',      'Брусянцева Татьяна Палладьевна', 'ИСиТ', 'м');
insert into TEACHER      (TEACHER,    TEACHER_NAME, PULPIT, GENDER )   -- OK
                  values ('МРЗ',      'Мороз Елена Станиславовна',     'ИСиТ', 'ж');
insert into TEACHER      (TEACHER,    TEACHER_NAME, PULPIT, GENDER )   -- error 547
                  values ('РМНК',      'Романенко Дмитрий Михайлович', 'ИСиТ', 'х');

update TEACHER set GENDER = 'х' where TEACHER = 'БРС';           -- OK
update TEACHER set GENDER = NULL where TEACHER = 'МРЗ';          -- OK
update TEACHER set GENDER = 'у' where TEACHER = 'СМЛВ';          -- error 547

select * from TEACHER
```

CHECK

```
create table TEACHER
(
    TEACHER      char(10)  constraint TEACHER_PK primary key,
    TEACHER_NAME varchar(100),
    GENDER        char(1)
                      not null
                      default 'M'
                      constraint TEACHER_GENDER_CH check (GENDER in ('M', 'X')),
    PULPIT       char(20)  constraint TEACHER_PULPIT_FK foreign key
                          references PULPIT(PULPIT)
);
```

UNIQUE

```
create table PROFESSION  -- специальность
(
    PROFESSION      char(20)      -- специальность
                      constraint PROFESSION_PK primary key,
    FACULTY         char(10)       -- факультет
                      constraint PROFESSION_FACULTY_FK
                      foreign key references FACULTY(FACULTY),
    PROFESSION_NAME varchar(100)   -- наименование специальности
                      constraint PROFESSION_NAME_UQ unique,
    QUALIFICATION   varchar(50)    -- квалификация
);
go
```

UNIQUE

```
insert into PROFESSION(FACULTY, PROFESSION,PROFESSION_NAME, QUALIFICATION) --OK
    values ('ИДиП','1-40 01 02', 'Информационные системы и технологии',
            'инженер-программист-системотехник' );
insert into PROFESSION(FACULTY, PROFESSION,PROFESSION_NAME, QUALIFICATION) --OK
    values ('ИДиП','1-47 01 01','Издательское дело', 'редактор-технолог' );
insert into PROFESSION(FACULTY, PROFESSION,PROFESSION_NAME, QUALIFICATION) -- error 2627
    values ('ИДиП', '1-36 06 01','Издательское дело', 'инженер-электромеханик' );
insert into PROFESSION(FACULTY, PROFESSION,PROFESSION_NAME, QUALIFICATION) --OK
    values ('ХТиТ', '1-36 01 08',NULL,'инженер-механик' );
insert into PROFESSION(FACULTY, PROFESSION,PROFESSION_NAME, QUALIFICATION) -- error 2627
    values ('ХТиТ', '1-36 07 01',NULL, 'инженер-механик' );
update PROFESSION set PROFESSION_NAME = 'Информационные системы и технологии' -- error 26
    where PROFESSION = '1-36 01 08';
select * from PROFESSION;
```

IDENTITY

```
create table GROUPS --  учебные группы
(
    IDGROUP      int          -- идентификатор учебной группы
                      identity(1,1)
                      constraint GROUP_PK primary key,
    FACULTY      char(10)     -- факультет
                      constraint GROUPS_FACULTY_FK foreign key
                      references FACULTY(FACULTY),
    PROFESSION   char(20)     -- специальность
                      constraint GROUPS_PROFESSION_FK foreign key
                      references PROFESSION(PROFESSION),
    YEAR_FIRST   smallint     -- год поступления
                      check (YEAR_FIRST<=YEAR(GETDATE()) ),
) ;
go
```

IDENTITY

```
insert into GROUPS  (FACULTY,  PROFESSION,  YEAR_FIRST ) -- OK
    values ('ИДиП','1-40 01 02', 2013),
            ('ИДиП','1-40 01 02', 2012),
            ('ИДиП','1-40 01 02', 2011),
            ('ИДиП','1-40 01 02', 2010),
            ('ИДиП','1-47 01 01', 2013);
insert into GROUPS  ( IDGROUP, FACULTY,  PROFESSION,  YEAR_FIRST) -- error 544
    values (99, 'ИДиП','1-47 01 01', 2013);
select * from GROUPS;
```

Вопросы?

БАЗЫ ДАННЫХ

Лекция 4
Архитектура базы данных Microsoft SQL Server

Возможности СУБД

- разнообразные пользовательские интерфейсы
- физическая независимость данных
- логическая независимость данных
- оптимизация запросов
- целостность данных
- управление параллелизмом
- резервное копирование и восстановление
- безопасность баз данных

Основные понятия безопасности БД

- Аутентификация – процесс проверки подлинности учетных данных пользователя, чтобы не допустить использования системы несанкционированными пользователями.
- Авторизация – процесс, применяемый к пользователям, уже получившим доступ к системе, чтобы определить их права на использование определенных ресурсов.

Системные требования

- Память и дисковое пространство (min 512 Мб)
- ОС Windows 32 или 64 разряда
- Требования к сети

Версии SQL Server

- Express Edition
- Workgroup Edition
- Standard Edition
- Web Edition
- Enterprise Edition
- Developer Edition
- Datacenter Edition
- Parallel Data Warehouse Edition

Версии SQL Server

- Express Edition
 - Облегченная версия SQL Server
 - Предназначена для разработчиков приложений
 - По этой причине продукт содержит базовую программу
 - Поддерживает интеграцию общеязыковой среды выполнения CLR и собственный язык XML
 - SQL Server Management Express для SQL Server Express,
<http://msdn.microsoft.com/express>

Версии SQL Server

- Workgroup Edition
 - Предназначена для малого бизнеса и для использования на уровне отделов предприятия
 - Отсутствуют средства бизнес-аналитики (БА) и возможности обеспечения высокого уровня доступности
 - Поддерживает системы с двумя процессорами и максимум 2 Гб оперативной памяти

Версии SQL Server

- Standard Edition
 - Версия предназначена для малого и среднего бизнеса
 - Содержит весь диапазон возможностей бизнес-аналитики, включая службы Analysis Services, Reporting Services и Integration Services.
 - Не содержит многих возможностей из версии Enterprise Edition
 - Поддерживает до 4x процессоров и 2 Тб оперативной памяти

Версии SQL Server

- Web Edition
 - Версия предназначена для поставщиков веб-хостинга
 - Содержит службы отчетности Reporting Services
 - Поддерживается до 4x процессоров и 2 Тб оперативной памяти

Версии SQL Server

- Enterprise Edition
 - Предназначена для приложений, критичных по времени и с большим количеством пользователей
 - Поддерживает секционирование данных, возможность получения мгновенных снимков состояния базы данных и онлайновую поддержку баз данных

Версии SQL Server

- Developer Edition
 - Содержит всю функциональность версии Enterprise Edition
 - Лицензия разрешает использование только для разработки, тестирования и демонстрации на необходимом количестве систем
 - Для использования другими разработчиками необходимо приобрести дополнительные лицензии

Версии SQL Server

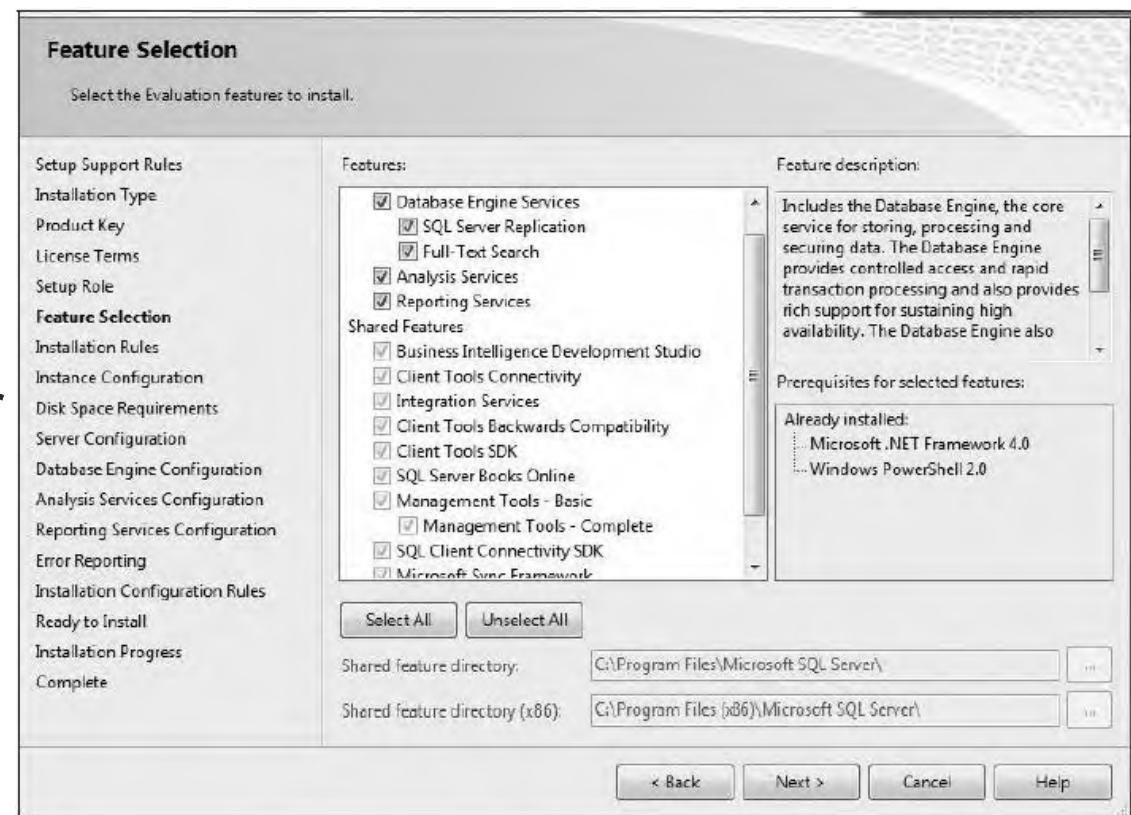
- Datacenter Edition
 - Предназначена для поддержки масштабирования наивысшего уровня
 - Нет ограничений по памяти
 - Можно создавать до 25 экземпляров сервера
 - Обеспечивается поддержка до 256 логических процессоров

Версии SQL Server

- Parallel Data Warehouse Edition
 - Версия специализирована для хранилищ данных и поддерживает базы данных хранилищ данных размером от 10 Тб до 1 Пб (петабайт, 1 Пб = 1024 Тб)
 - Используется архитектура массово-параллельной обработки (МПО с возможностями высокопроизводительных вычислений HPC (High Performance Computing))

Компоненты SQL Server

- Database Engine
- Analysis Services
- Reporting Services
- Integration Services
- Full-Text Search
- SQL Server Agent
- SQL Server Browser



Экземпляр сервера Bd

- по умолчанию (default)
- именованные (named)

Режим проверки подлинности

- Режим проверки подлинности Windows
- Режим проверки подлинности SQL Server

Базы данных

- Системные
- Пользовательские

Системные Бд

- Master
 - информация системного уровня: параметры настройки, о других базах данных и их файлах, учетные записи и др.
- Msdb
 - для планирования работы заданий
- Tempdb
 - временные объекты (таблицы, процедуры, курсоры)
- Model
 - шаблон базы данных
- Resource
 - содержит системные объекты, которые входят в состав SQL Server

Системные Бд

- создаются при инсталляции сервера
- используются сервером в процессе работы

Хранение объектов баз данных

- страницы
- экстенты
- файлы
- файловые группы

Хранение объектов баз данных

- Основной единицей хранилища данных является страница
- Размер страницы постоянен и составляет 8 КБ
- Каждая страница имеет заголовок размером в 96 байтов для системная информации
- Строки данных размещаются на странице сразу же после заголовка
- Виды страниц:
 - страницы данных
 - страницы индексов

Хранение объектов баз данных

- Содержимое базы данных хранится в одном или нескольких файлах
- Каждый файл разделен на несколько страниц
- Каждую страницу таблицы или индекса можно однозначно идентифицировать, используя
 - идентификатор базы данных
 - идентификатор файла базы данных
 - номер страницы

Хранение объектов баз данных

- Физическая единица дискового пространства, используемая для выделения памяти для таблиц и индексов, называется экстентом
- Размер экстента составляет 8 последовательно расположенных страниц или иначе 64 Кбайт

Хранение объектов баз данных

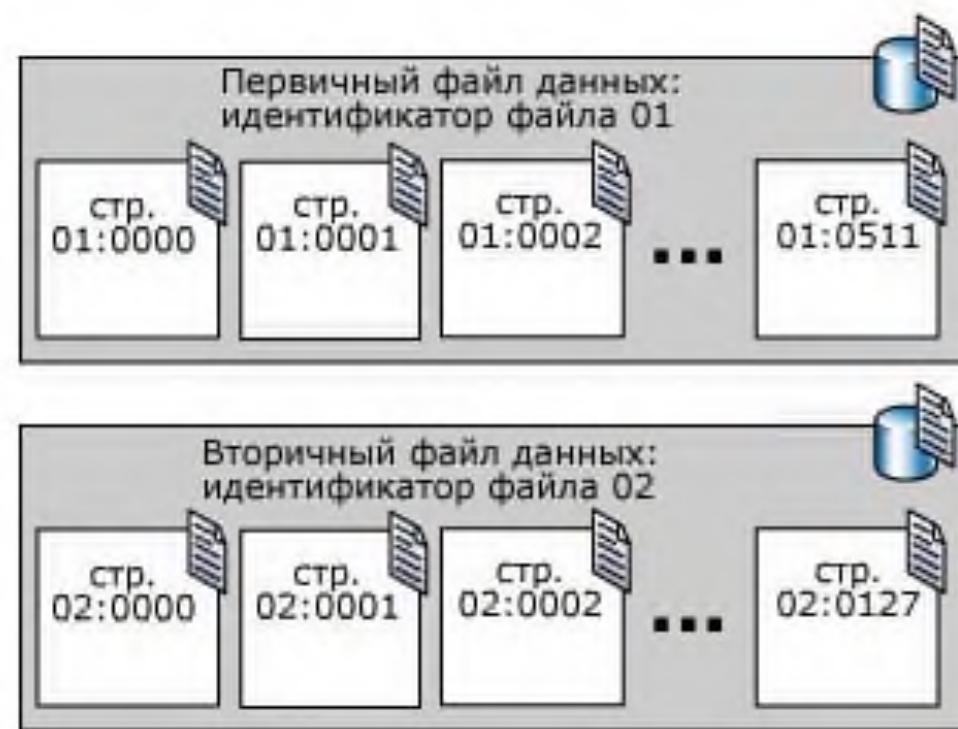
- Все страницы данных имеют фиксированный размер (8 Кб) и состоят из следующих частей:
 - заголовка страницы (96 байтов)
 - пространства для данных
 - таблицы смещений строк

Хранение объектов баз данных



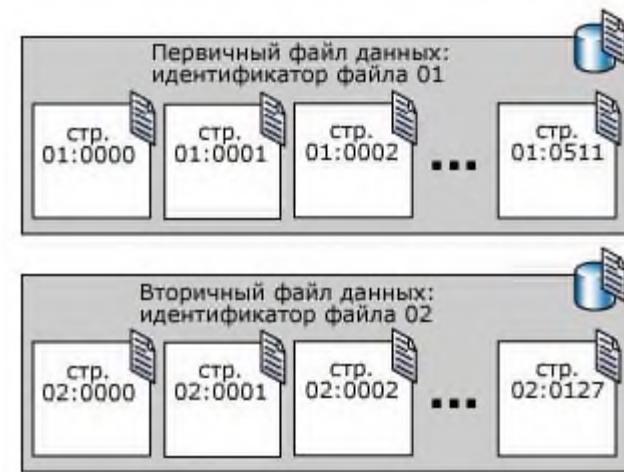
Хранение данных на диске

- страницы
- экстенты
- файлы
- файловые группы



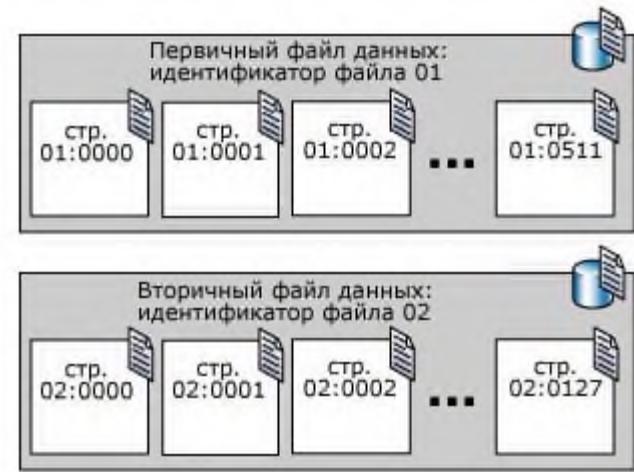
Страницы

- Основной единицей хранилища данных является страница.
- Размер страницы постоянен и составляет 8 Кбайт.
- Каждая страница имеет заголовок 96 байтов, в котором хранится системная информация.
- Строки данных размещаются на странице сразу же после заголовка.
- Виды страниц:
 - страницы данных;
 - страницы индексов.



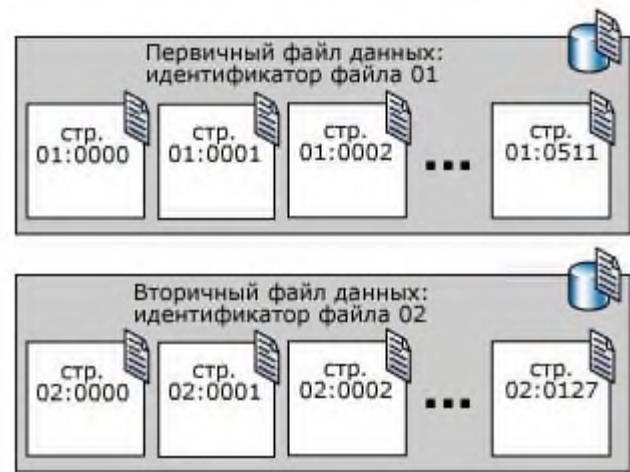
Экстенты

- Физическая единица дискового пространства, используемая для выделения памяти для таблиц и индексов, называется экстентом.
- Размер экстента составляет восемь последовательно расположенных страниц или 64 Кбайт.
- Существует два следующих типа экстентов:
 - однородные экстенты;
 - смешанные экстенты.



Экстенты

- Однородные экстенты содержат данные одной таблицы или индекса
- Смешанные экстенты могут содержать данные до восьми таблиц или индексов.



Свойства страниц данных

- Все типы страниц данных имеют фиксированный размер (8 Кбайт) и состоят из следующих частей:
 - заголовка страницы;
 - пространства для данных;
 - таблицы смещений строк.



Пространство для данных

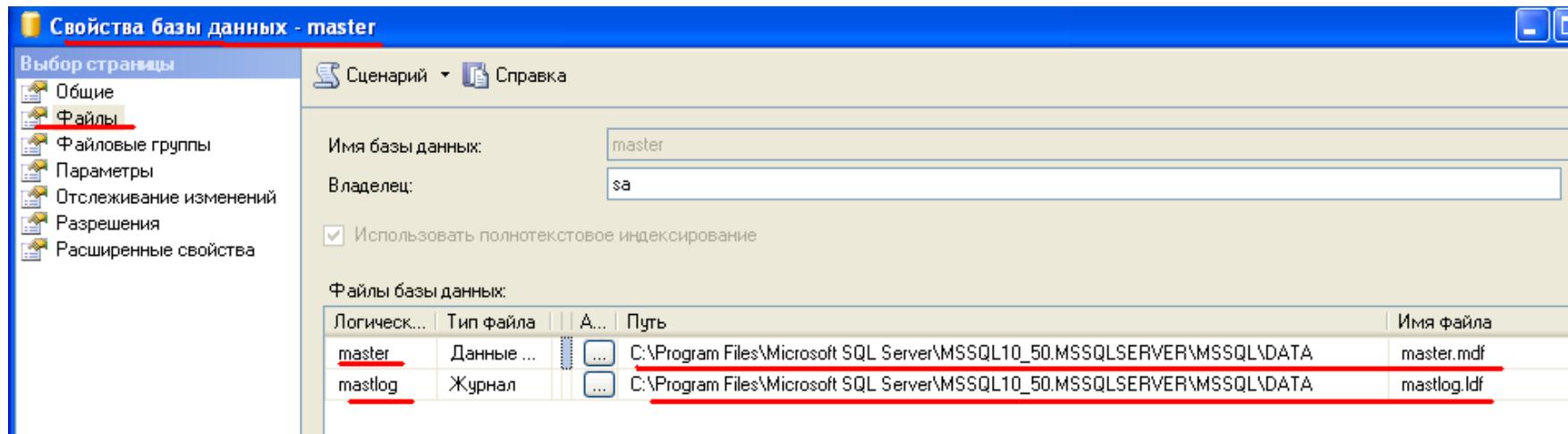
- Для каждой строки страницы создается
 - запись данных
 - запись в таблице смещения
- Стока данных не может выходить за пределы одной страницы
- Строки сохраняются в последовательном порядке

Пространство для данных

- Если в странице нет места для новой строки той же таблицы, эта строка сохраняется в следующей странице цепочки страниц.
- Для таблиц, которые имеют только столбцы фиксированного размера, в каждой странице сохраняется одинаковое количество строк.
- Если в таблице есть хотя бы один столбец переменной длины, количество строк в странице может колебаться, и в странице сохраняется столько строк, сколько поместится.

Файлы базы данных

- первичный файл (.mdf)
- вторичные файлы (.ndf)
- файлы журнала транзакций (.log)



Файловые группы

- Файловые группы – поименованный набор файлов БД
- Все файлы БД, кроме файлов журнала транзакций, распределены по файловым группам
 - Первичные
 - Вторичные
- Используются для упрощения администрирования
 - Секционирование
 - Резервное копирование и восстановление

Создание базы данных

- CREATE database BSTU
- ON PRIMARY -- первичная файловая группа
- (name = N'BSTU_mdf', -- логическое имя файла
- filename = N'D:\BD\BSTU_mdf.mdf', -- имя файла в ОС
- size = 10240Kb, -- первоначальный размер файла
- maxsize = UNLIMITED, -- максимальный размер файла
- filegrowth = 1024Kb) -- приращение
- LOG ON -- журнал транзакций
- (name = N'BSTU_log', -- логическое имя файла
- filename = N'D:\BD\BSTU_log.ldf', -- физический файл
- size = 10240Kb, -- первоначальный размер
- maxsize = 2048Gb, -- максимальный размер
- filegrowth = 10%) -- приращение
-

Журнал транзакций

- Сначала запись идет в журнал транзакций, потом в файл данных
- Файл журнала транзакций используется для восстановления данных БД в случае аварийного завершения работы сервера

Файловые группы

- Во вторичных файловых группах могут располагаться только вторичные файлы.
- В первичной файловой группе помимо обязательного первичного файла тоже могут быть расположены вторичные файлы.
- При создании таблиц и индексов дисковая память для них автоматически отводится в файловой группе по умолчанию.
- Для размещения в другой файловой группе следует явно указывать ее имя в операторе CREATE, создающем таблицу или индекс.

Пример

- CREATE database BSTU **ON PRIMARY**
- (name = N'BSTU_1', filename = N'D:\BD\BSTU_1.mdf',
(size = 10240Kb, maxsize = UNLIMITED, filegrowth = 1024Kb),
(name = N'BSTU_2', filename = N'D:\BD\BSTU_2.ndf', size =
10240KB, maxsize = 1Gb, filegrowth = 25%),
- **FILEGROUP FG1**
- (name = N'BSTU_fg1_1', filename = N'D:\BD\BSTU_3.ndf',
size = 10240Kb, maxsize = 1Gb, filegrowth = 25%),
(name = N'BSTU_fg1_2', filename = N'D:\BD\BSTU_4.ndf',
size = 10240Kb, maxsize = 1Gb, filegrowth = 25%)
- **LOG ON**
- (name = N'BSTU_log', filename = N'D:\BD\BSTU_log.ldf',
size = 10240Kb, maxsize = 2048Gb, filegrowth = 10%)

Изменение файловых групп

- alter database BSTU add filegroup G1;
- alter database BSTU add file
 - (name = N'BSTU1', filename = N'C:\BSTU1.ndf',
 - size = 3072KB, maxsize = unlimited, filegrowth = 1024KB
 -) to filegroup G1;
- alter database BSTU modify filegroup G1 default;

Параметры БД

- параметры автоматических действий,
- параметры курсора,
- параметры внешних действий,
- параметры восстановления,
- параметры SQL,
- параметры моментальных снимков,
- параметры компонента Service Broker,
- параметры доступности БД.

Параметры БД

- **ALTER DATABASE ... SET**

```
use MASTER
```

```
go
```

```
alter database BSTU set AUTO_CLOSE off          -- автоматическое закрытие БД  
alter database BSTU set AUTO_CREATE_STATISTICS on -- автоматический сбор статистики  
alter database BSTU set AUTO_SHRINK on           -- автоматическое сжатие файлов  
go
```

Параметры БД

- AUTO_CLOSE – закрытие БД при отсутствии с ней соединений.
- AUTO_CREATE_STATISTICS – включить или отключить автоматический сбор статистики для оптимизатора запросов
- AUTO_SHRINK – необходимость периодического сжатия файлов БД

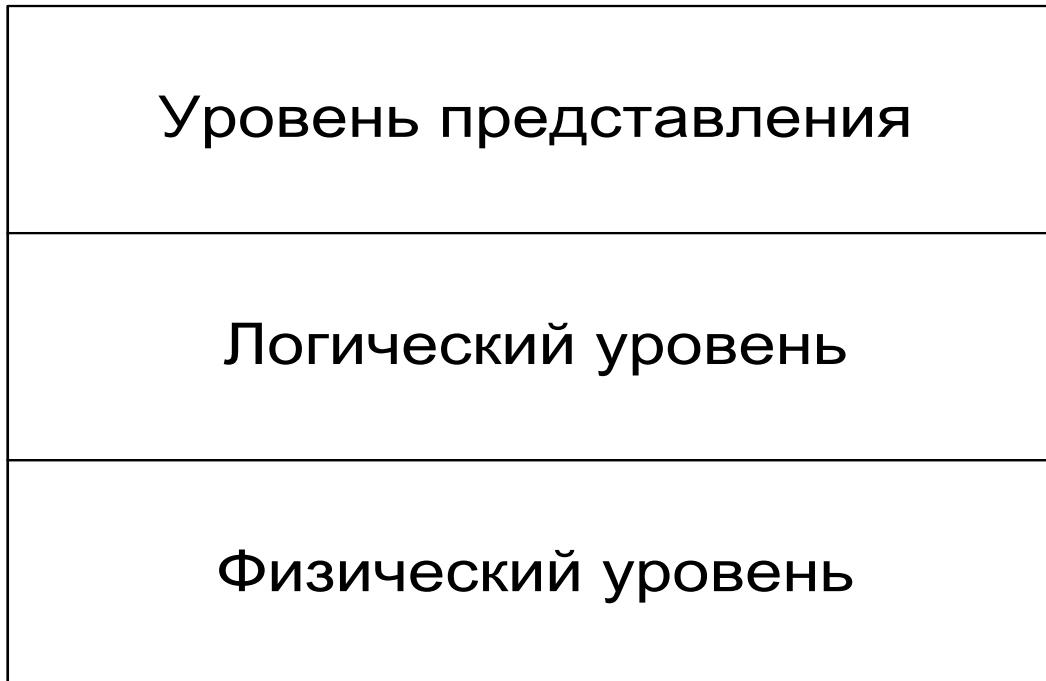
Параметры БД

```
use MASTER
go

select case DATABASEPROPERTYEX('BSTU', 'IsAutoClose')
        when 0 then 'off'          -- если = 0, то вернуть 'off'
        else 'on'                  -- иначе вернуть 'on'
    end IsAutoClose,
       case DATABASEPROPERTYEX('BSTU', 'IsAutoShrink')
        when 0 then 'off'
        else 'on'
    end IsAutoShrink,
       case DATABASEPROPERTYEX('BSTU', 'IsAutoCreateStatistics')
        when 0 then 'off'
        else 'on'
    end IsAutoCreateStatistics
go
```

IsAutoClose	IsAutoShrink	IsAutoCreateStatistics
off	on	on

Принципы проектирования логической схемы базы данных



Пользователи

Проектировщики

Системные
администраторы

Проектирование логической схемы базы данных

- **Системный администратор:** БД – это набор файлов операционной системы, имеющих определенную физическую структуру, формат и организацию.
- **Проектировщик:** БД – это **логическая схема данных**, представляющая собой набор структурированных данных с заданными ограничениями и связями.
- **Пользователь:** БД представлена в виде подмножества **таблиц** или **представлений**, набора **процедур** и функций на языке **T-SQL** и пр.

Проектирование логической схемы базы данных

- Теория реляционных БД предполагает независимость трех уровней друг от друга.
- Физическое размещение данных никак не должно ограничивать возможности проектирования логической схемы данных.
- Представления пользователей о БД не зависят от структуры логической схемы.
- В реальности – частичная независимость уровней друг от друга.

Размещение таблиц в файловой группе

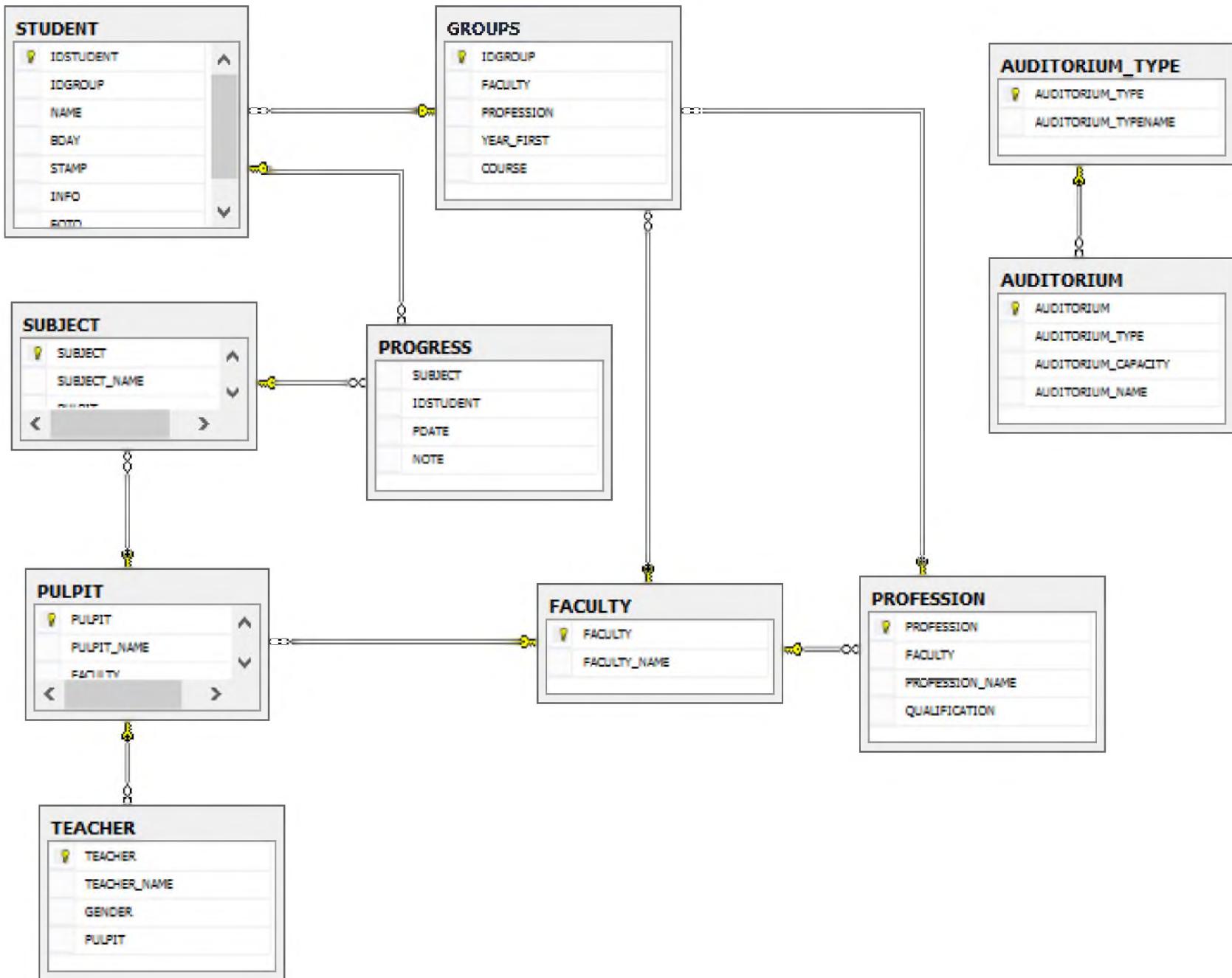
```
create table AUDITORIUM -- аудитории вуза
(
    AUDITORIUM           char(20)   -- идентификатор аудитории (304-1, 401-4, ...)
                                constraint AUDITORIUM_PK primary key,
    AUDITORIUM_TYPE      char(10)   -- тип аудитории (ЛК, ЛБ-К, ...)
                                constraint AUDITORIUM_AUDITORIUM_TYPE_FK foreign key
                                references AUDITORIUM_TYPE(AUDITORIUM_TYPE),
    AUDITORIUM_CAPACITY int       -- вместимость (макс. кол. слушателей)
                                constraint AUDITORIUM_CAPACITY_CHECK default 1
                                check (AUDITORIUM_CAPACITY between 1 and 300),
    AUDITORIUM_NAME      varchar(50) -- текстовое наименование аудитории (комментарий)
) on FG1; -- в файловой группе FG1
```

Размещение таблиц в файловой группе

```
create table SHEDULE_TEACHER      -- расписание преподавателя
(
    [DATETIME]    smalldatetime,      -- дата и время занятий
    TEACHER        char(10),          -- преподаватель
    [SUBJECT]      char(10),          -- дисциплина
    AUDITORIUM    char(10),          -- аудитория
    constraint PK_SHEDULE_TEACHER primary key([DATETIME],TEACHER)
) on FG2
```

Удаление таблиц

- **DROP TABLE ...**
-
- Проблема удаления таблицы:
 - Пользователь не имеет достаточных прав на удаление
 - Таблица заблокирована транзакцией другого сеанса
 - На первичный ключ удаляемой таблицы ссылается внешний ключ другой таблицы



Изменение структуры таблицы

- Добавление или удаление столбцов
- Добавление или удаление ограничений
- Изменение точности или типа числовых данных, размерности символьных данных, ...

Просмотр структуры таблицы

```
use BSTU  
go  
exec SP_HELP TEACHER;
```

Просмотр структуры таблицы

Name	Owner	Type	Created_datetime
TEACHER	dbo	user table	2014-01-09 15:24:54.093

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource
TEACHER	char	no	10			no	no	no
TEACHER_NAME	varchar	no	100			yes	no	yes
GENDER	char	no	1			yes	no	yes
PULPIT	char	no	20			yes	no	yes

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
CHECK on column GENDER	TEACHER_GENDER_CH	(n/a)	(n/a)	Enabled	Is_For_Replication	([GENDER]=x'OR [GENDER]=m')
PRIMARY KEY (clustered)	TEACHER_PK	(n/a)	(n/a)	(n/a)	(n/a)	TEACHER
FOREIGN KEY	TEACHER_PULPIT_FK	No Action	No Action	Enabled	Is_For_Replication	PULPIT
						REFERENCES BSTU.dbo.PULPIT

Изменение структуры таблицы

```
use BSTU
go
select *, len(TEACHER_NAME) 'len(TEACHER_NAME)' from TEACHER order by TEACHER
```

TEACHER	TEACHER_NAME	GENDER	PULPIT	len(TEACHER_NAME)
АКНВЧ	Акунович Станислав Иванович	NULL	ИСиТ	27
АРС	Арсентьев Виталий Арсентьевич	NULL	ПОиСОИ	29
АТР	Атрощенко Олег Александрович	NULL	ПУ	28
БЗБРДВ	Безбородов Владимир Степанович	NULL	ОХ	30
БПВ	Белоев Валерий Павлович	NULL	ПОиСОИ	23

```
use BSTU
go
alter table TEACHER -- изменение столбца
    alter column TEACHER_NAME varchar(50) not null;
alter table TEACHER -- добавление нового столбца
    add BDATE date not null default '19600101'
```

Изменение структуры таблицы

```
use BSTU
go
select *, len(TEACHER_NAME) 'len(TEACHER_NAME)' from TEACHER order by TEACHER
```

TEACHER	TEACHER_NAME	GENDER	PULPIT	BDATE	len(TEACHER_NAME)
АКНВЧ	Акунович Станислав Иванович	NULL	ИСиТ	1960-01-01	27
АРС	Арсентьев Виталий Арсентьевич	NULL	ПОиСОИ	1960-01-01	29
АТР	Атрощенко Олег Александрович	NULL	ЛУ	1960-01-01	28

Системный каталог

- Системный каталог состоит из таблиц, описывающих структуру объектов базы данных
- К системным базовым таблицам нельзя обращаться напрямую



Представления каталога

- sys.objects
- sys.columns
- sys.database_principals

```
USE sample;
SELECT object_id, principal_id, type
FROM sys.objects
WHERE name = 'employee';
```

Динамические административные представления и функции

- Возвращают информацию о состоянии сервера, которую можно применить для отслеживания и настройки производительности системы
 - ◆ `sys.dm_db_*` —
 - ◆ `sys.dm_tran_*` —
 - ◆ `sys.dm_io_*` —
 - ◆ `sys.dm_exec_*` —

Информационная схема

- Состоит из представлений, доступных только для чтения, которые предоставляют информацию обо всех таблицах, представлениях и столбцах компонента Database Engine, к которым имеется доступ.
 - `information_schema.tables`
 - `information_schema.columns`

Системные хранимые процедуры

- Системные хранимые процедуры применяются для выполнения административных и пользовательских задач:
- переименование объектов базы данных,
- идентификация пользователей
- мониторинг авторизации и ресурсов

- sp_help
- sp_configure

Системные функции

- OBJECT_ID(object_name)
- OBJECT_NAME(object_id)
- USER_ID([user_name])
- USER_NAME([user_id])
- DB_ID([db_name])
- DB_NAME([db_id])

ФУНКЦИИ СВОЙСТВ

- Функции свойств возвращают свойства объектов баз данных, типов данных и файлов.
 - Функции свойств поддерживают десятки свойств в виде параметров, которые можно явно указывать.
 - Почти все функции свойств возвращают одно из следующих значений: 0, 1 или NULL.
-
- OBJECTPROPERTY(id, property)
 - COLUMNPROPERTY(id, column, property)
 - FILEPROPERTY(filename, property)
 - TYPEPROPERTY(type, property)

Вопросы?

БАЗЫ ДАННЫХ

Лекция 5
Оператор SELECT

Оператор SELECT

- Основная инструкция для выборки информации — инструкция SELECT
- Результаты выполнения инструкции SELECT помещаются в **результатирующий набор** (result set)

```
drop table AUDITORIUM_TYPE
create table AUDITORIUM_TYPE
(
    AUDITORIUM_TYPE      char(10) primary key, -- код типа аудитории
    AUDITORIUM_TYPENAME  varchar(30)           -- тип аудитории
)

drop table AUDITORIUM
create table AUDITORIUM
(
    AUDITORIUM           char(10) primary key, -- код аудитории
    AUDITORIUM_NAME       varchar(200),          -- аудитория
    AUDITORIUM_CAPACITY   integer,              -- вместимость
    AUDITORIUM_TYPE        char(10) not null     -- тип аудитории
    references AUDITORIUM_TYPE(AUDITORIUM_TYPE)
)
```

AUDITORIUM_TYPE

	Имя столбца	Сжатый тип	Допускает значения NULL	Тип данных
PK	AUDITORIUM_TYPE	char(10)	Нет	char(10)
	AUDITORIUM_TYPENAME	varchar(30)	Да	varchar(30)



AUDITORIUM *

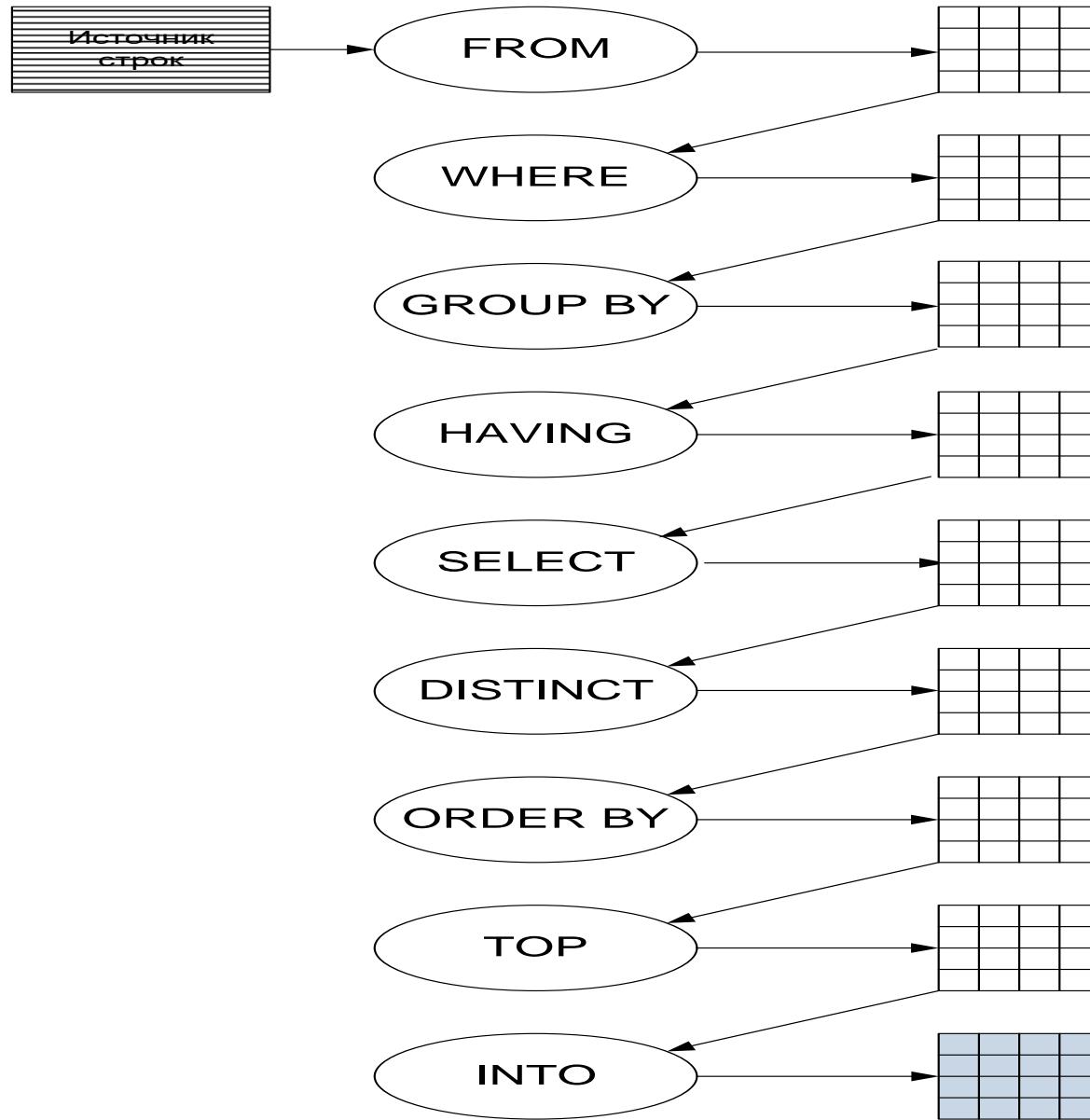
	Имя столбца	Сжатый тип	Допускает значения NULL	Тип данных
PK	AUDITORIUM	char(10)	Нет	char(10)
	AUDITORIUM_NAME	varchar(2...	Да	varchar(2...
	AUDITORIUM_CA...	int	Да	int
	AUDITORIUM_TYPE	char(10)	Нет	char(10)

```
delete AUDITORIUM_TYPE;
insert into AUDITORIUM_TYPE (AUDITORIUM_TYPE, AUDITORIUM_TYPENAME )
    values ('ЛК', 'Лекционная');
insert into AUDITORIUM_TYPE (AUDITORIUM_TYPE, AUDITORIUM_TYPENAME )
    values ('ЛБ-К', 'Компьютерный класс');
insert into AUDITORIUM_TYPE (AUDITORIUM_TYPE, AUDITORIUM_TYPENAME )
    values ('ЛК-К', 'Лекционная с уст. компьютерами');
insert into AUDITORIUM_TYPE (AUDITORIUM_TYPE, AUDITORIUM_TYPENAME )
    values ('ЛБ-Х', 'Химическая лаборатория');
insert into AUDITORIUM_TYPE (AUDITORIUM_TYPE, AUDITORIUM_TYPENAME )
    values ('ЛБ-СК', 'Спец. компьютерный класс');
```

```
delete AUDITORIUM;
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('206-1', '206-1', 'ЛВ-К', 15);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY)
    values ('301-1', '301-1', 'ЛВ-К', 15);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('236-1', '236-1', 'ЛК', 60);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('313-1', '313-1', 'ЛК', 60);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('324-1', '324-1', 'ЛК', 50);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('413-1', '413-1', 'ЛВ-К', 15);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('423-1', '423-1', 'ЛВ-К', 90);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('408-2', '408-2', 'ЛК', 90);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('103-4', '103-4', 'ЛК', 90);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('105-4', '105-4', 'ЛК', 90);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('107-4', '107-4', 'ЛК', 90);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('110-4', '110-4', 'ЛК', 30);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('111-4', '111-4', 'ЛК', 30);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('114-4', '114-4', 'ЛК-К', 90 );
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('132-4', '132-4', 'ЛК', 90);
insert into AUDITORIUM (AUDITORIUM, AUDITORIUM_NAME, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY )
    values ('026-4', '026-4', 'ЛК', 90);
```

Синтаксис

```
SELECT select_list
      [INTO new_table]
      FROM table
      [WHERE search_condition]
      [GROUP BY group_by_expression]
      [HAVING search_condition]
      [ORDER BY order_expression [ASC | DESC]];
```



FROM

```
|  
| SELECT 'HELLO, WORLD!';  
| SELECT SYSDATETIME();
```

A screenshot of a Microsoft SQL Server Management Studio (SSMS) window. The title bar shows the connection details: 'sa - [local]' and 'master'. The window has a toolbar with a magnifying glass icon and a dropdown arrow. Below the toolbar is a tab bar with 'Results' and 'Messages'. The 'Results' tab is selected. The results grid contains two rows of data:

	(No column name)
	HELLO, WORLD!

Below the results grid is another row of data:

	(No column name)
	2017-03-04 00:27:45.9913489

FROM

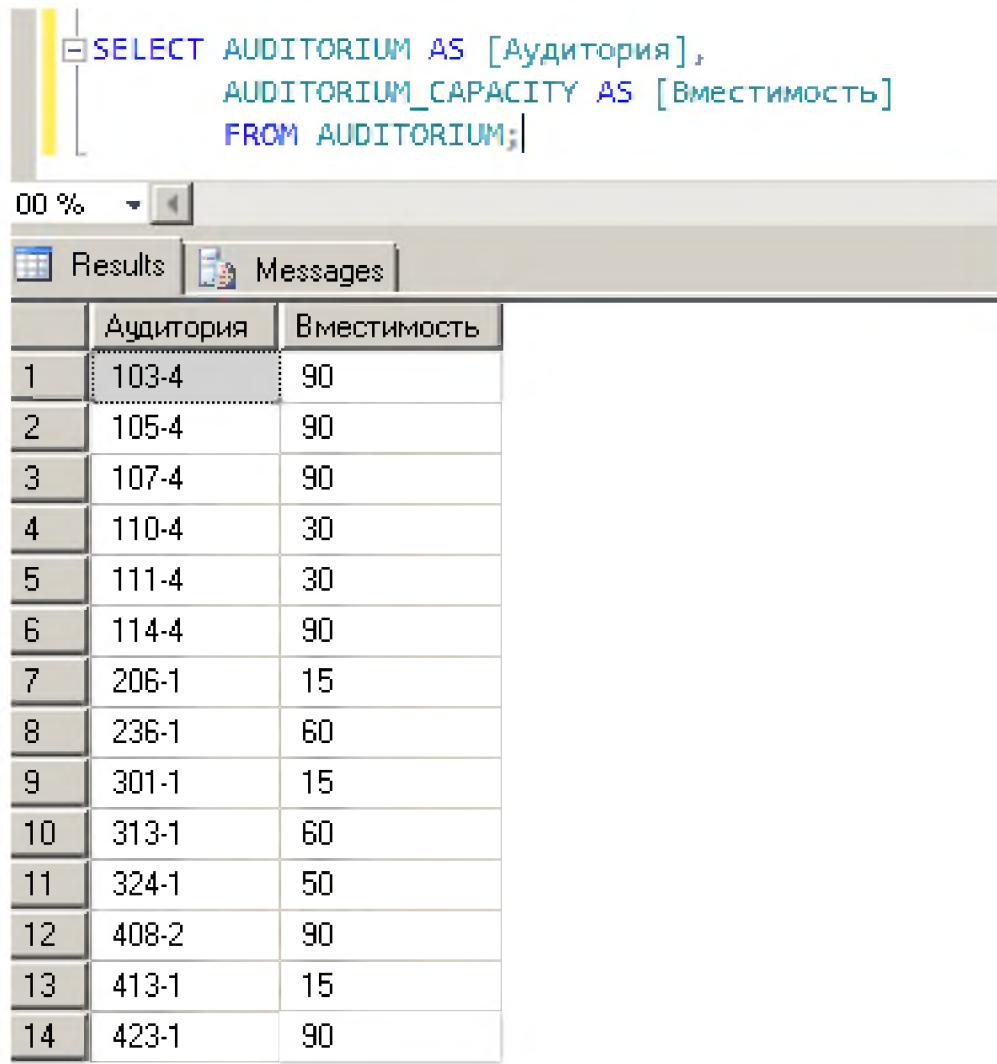
```
SELECT * FROM AUDITORIUM;  
SELECT AUDITORIUM, AUDITORIUM_CAPACITY FROM AUDITORIUM;
```

100 %

Results Messages

	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
1	103-4	ЛК	90	103-4
2	105-4	ЛК	90	105-4
3	107-4	ЛК	90	107-4
4	110-4	ЛК	30	110-4
5	111-4	ЛК	30	111-4
6	114-4	ЛК-К	90	114-4
7	206-1	ЛБ-К	15	206-1
8	236-1	ЛК	60	236-1
9	301-1	ЛБ-К	15	301-1
10	313-1	ЛК-К	60	313-1
11	324-1	ЛК-К	50	324-1
12	408-2	ЛК	90	408-2
13	413-1	ЛБ-К	15	413-1
14	423-1	ЛБ-К	90	423-1

FROM



The screenshot shows a SQL query being run in a database environment. The query is:

```
SELECT AUDITORIUM AS [Аудитория],  
       AUDITORIUM_CAPACITY AS [Вместимость]  
  FROM AUDITORIUM;
```

The results are displayed in a table with two columns: 'Аудитория' (Auditorium) and 'Вместимость' (Capacity). The data is as follows:

	Аудитория	Вместимость
1	103-4	90
2	105-4	90
3	107-4	90
4	110-4	30
5	111-4	30
6	114-4	90
7	206-1	15
8	236-1	60
9	301-1	15
10	313-1	60
11	324-1	50
12	408-2	90
13	413-1	15
14	423-1	90

FROM

```
1 SELECT AUDITORIUM_TYPENAME FROM
2   (SELECT * FROM AUDITORIUM_TYPE WHERE AUDITORIUM_TYPE = 'ЛК') as t1;
```

% ▾

Results Messages

AUDITORIUM_TYPENAME
Лекционная

WHERE

```
select * from AUDITORIUM WHERE AUDITORIUM_CAPACITY < 60
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
206-1	206-1	15	ЛБ-К
301-1	301-1	15	ЛБ-К
324-1	324-1	50	ЛК
413-1	413-1	15	ЛБ-К

WHERE

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY < 60 and  
AUDITORIUM_CAPACITY > 15
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
324-1	324-1	50	ЛК

WHERE

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY < 60 or  
AUDITORIUM_CAPACITY > 15
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
026-4	026-4	90	ЛК
103-4	103-4	90	ЛК
105-4	105-4	90	ЛК
107-4	107-4	90	ЛК
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
114-4	114-4	90	ЛК-К
132-4	132-4	90	ЛК
206-1	206-1	15	ЛБ-К
229-4	229-4	90	ЛК
236-1	236-1	60	ЛК
301-1	301-1	15	ЛБ-К
304-4	304-4	90	ЛБ-К
313-1	313-1	60	ЛК
314-4	314-4	90	ЛК
320-4	320-4	90	ЛК
324-1	324-1	50	ЛК
408-2	408-2	90	ЛК
413-1	413-1	15	ЛБ-К
423-1	423-1	90	ЛБ-К
429-4	429-4	90	ЛК

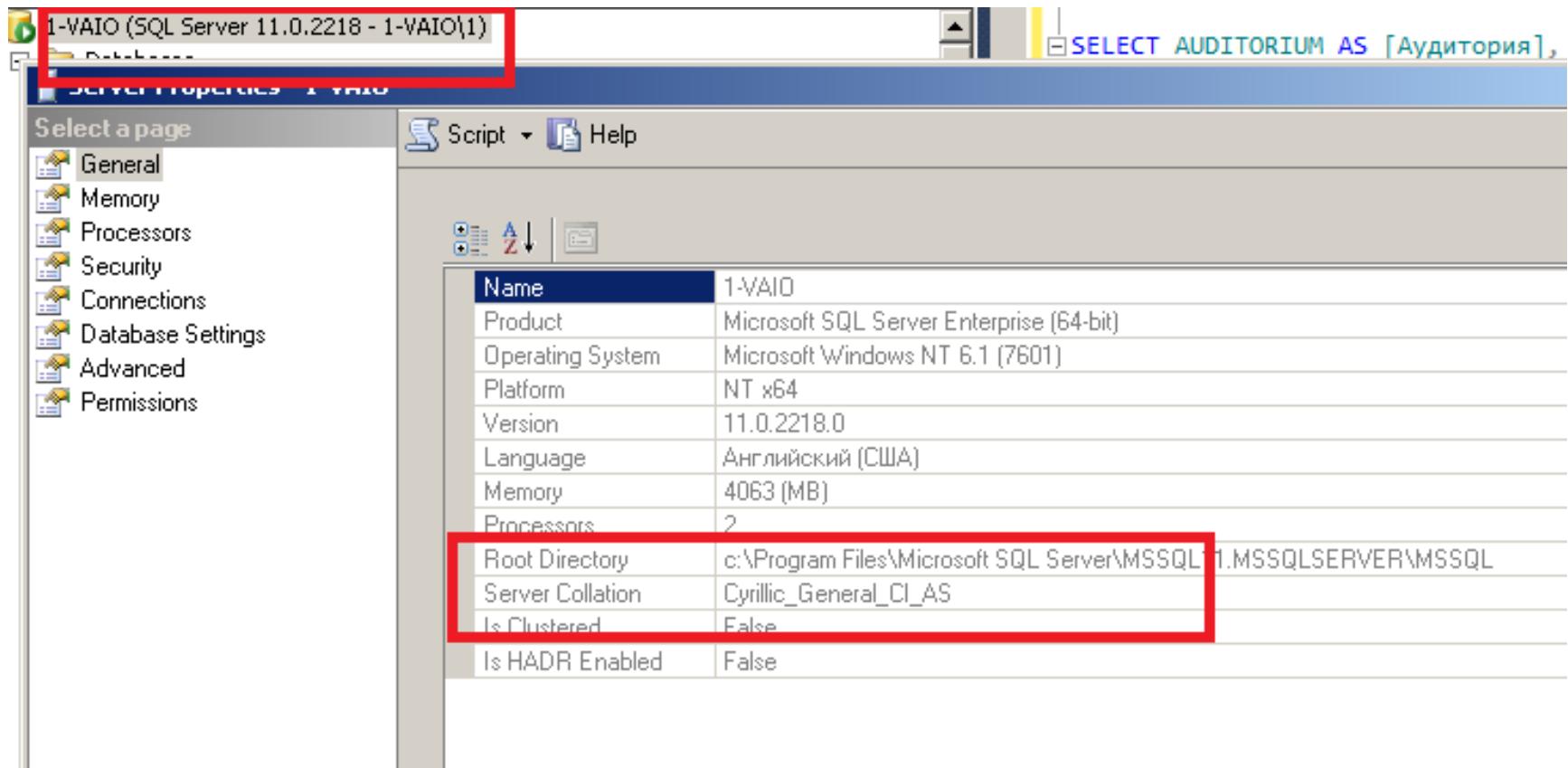
WHERE

\neq (или $!=$)	не равно
$<$	меньше чем
$>$	больше чем
\geq	больше чем или равно
\leq	меньше чем или равно
$\not\geq$	не больше чем
$\not\leq$	не меньше чем

WHERE

- Сравнение строк (CHAR, VARCHAR, NCHAR и NVARCHAR) выполняется в **действующем порядке сортировки**
- При сравнении строк сравниваются соответствующие символы каждой строки
- Старшинство символа определяется его позицией в кодовой таблице: символ, чей код стоит в таблице раньше, считается меньше
- При сравнении строк разной длины, более короткая строка дополняется в конце пробелами до длины более длинной строки

WHERE



- CI или CS
- AI или AS

WHERE

```
SELECT * FROM AUDITORIUM WHERE AUDITORIUM_TYPE > 'ЛК';
```

100 %

Results | Messages

	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
1	114-4	ЛК-К	90	114-4
2	313-1	ЛК-К	60	313-1
3	324-1	ЛК-К	50	324-1

WHERE

```
SELECT * FROM AUDITORIUM WHERE AUDITORIUM_TYPE >= 'ЛК';
```

0 %

Results | Messages

	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
1	103-4	ЛК	90	103-4
2	105-4	ЛК	90	105-4
3	107-4	ЛК	90	107-4
4	110-4	ЛК	30	110-4
5	111-4	ЛК	30	111-4
6	114-4	ЛК-К	90	114-4
7	236-1	ЛК	60	236-1
8	313-1	ЛК-К	60	313-1
9	324-1	ЛК-К	50	324-1
10	408-2	ЛК	90	408-2

WHERE

```
SELECT * FROM AUDITORIUM WHERE AUDITORIUM_TYPE > '1k';
```

100 %

Results Messages

	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
1	103-4	ЛК	90	103-4
2	105-4	ЛК	90	105-4
3	107-4	ЛК	90	107-4
4	110-4	ЛК	30	110-4
5	111-4	ЛК	30	111-4
6	114-4	ЛКК	90	114-4
7	206-1	ЛБ-К	15	206-1
8	236-1	ЛК	60	236-1
9	301-1	ЛБ-К	15	301-1
10	313-1	ЛКК	60	313-1
11	324-1	ЛКК	50	324-1
12	408-2	ЛК	90	408-2
13	413-1	ЛБ-К	15	413-1
14	423-1	ЛБ-К	90	423-1

WHERE

- Приоритет выполнения:
 - оператор NOT
 - оператор AND
 - оператор OR

```
SELECT * FROM AUDITORIUM  
    WHERE AUDITORIUM_TYPE = 'ЛК' AND AUDITORIUM_CAPACITY = 30  
        OR  
            AUDITORIUM_CAPACITY = 90 | AND AUDITORIUM = '103-4';
```

%

Results Messages

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
103-4	ЛК	90	103-4
110-4	ЛК	30	110-4
111-4	ЛК	30	111-4

```
SELECT * FROM AUDITORIUM  
    WHERE((AUDITORIUM_TYPE = 'ЛК' AND AUDITORIUM_CAPACITY = 30 )  
        OR  
            AUDITORIUM_CAPACITY = 90 ) AND AUDITORIUM = '103-4';
```

%

Results Messages

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
103-4	ЛК	90	103-4

WHERE NOT

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY not between 15 and 60
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
103-4	103-4	90	ЛК
105-4	105-4	90	ЛК
107-4	107-4	90	ЛК
114-4	114-4	90	ЛК-К
132-4	132-4	90	ЛК
229-4	229-4	90	ЛК
304-4	304-4	90	ЛБ-К
314-4	314-4	90	ЛК
320-4	320-4	90	ЛК
408-2	408-2	90	ЛК
423-1	423-1	90	ЛБ-К
429-4	429-4	90	ЛК

WHERE

- IN
- BETWEEN



WHERE BETWEEN

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY between 15 and 60
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
206-1	206-1	15	ЛБ-К
236-1	236-1	60	ЛК
301-1	301-1	15	ЛБ-К
313-1	313-1	60	ЛК
324-1	324-1	50	ЛК
413-1	413-1	15	ЛБ-К

WHERE IN

```
SELECT * FROM AUDITORIUM  
WHERE AUDITORIUM_CAPACITY IN (30, 90);
```

%

Results Messages

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
103-4	ЛК	90	103-4
105-4	ЛК	90	105-4
107-4	ЛК	90	107-4
110-4	ЛК	30	110-4
111-4	ЛК	30	111-4
114-4	ЛК-К	90	114-4
408-2	ЛК	90	408-2
423-1	ЛБ-К	90	423-1

WHERE IS NULL

```
| update AUDITORIUM set AUDITORIUM_NAME = NULL  
|   where AUDITORIUM_CAPACITY between 15 and 60  
  
select * from AUDITORIUM where AUDITORIUM_NAME IS NULL
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
110-4	NULL	30	ЛК
111-4	NULL	30	ЛК
206-1	NULL	15	ЛБ-К
236-1	NULL	60	ЛК
301-1	NULL	15	ЛБ-К
313-1	NULL	60	ЛК
324-1	NULL	50	ЛК
413-1	NULL	15	ЛБ-К

WHERE IS NULL

```
update AUDITORIUM set AUDITORIUM_NAME = AUDITORIUM  
where AUDITORIUM_NAME IS NULL
```

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY between 15 and 60
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
206-1	206-1	15	ЛБ-К
236-1	236-1	60	ЛК
301-1	301-1	15	ЛБ-К
313-1	313-1	60	ЛК
324-1	324-1	50	ЛК
413-1	413-1	15	ЛБ-К

WHERE IS NULL

```
|  
| SELECT * FROM AUDITORIUM  
| WHERE AUDITORIUM <> NULL;  
|
```

A screenshot of a SQL query window. The query is:

```
SELECT * FROM AUDITORIUM  
WHERE AUDITORIUM <> NULL;
```

The results pane shows the following columns:

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME

WHERE LIKE

```
select * from AUDITORIUM_TYPE
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
ЛБ-Х	Химическая лаборатория
ЛБ-К	Компьютерный класс
ЛБ-СК	Спец. компьютерный класс
ЛК	Лекционная
ЛК-К	Лекционная с уст. компьютерами

```
select * from AUDITORIUM_TYPE where AUDITORIUM_TYPENAME like '%комп%'
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
ЛБ-К	Компьютерный класс
ЛБ-СК	Спец. компьютерный класс
ЛК-К	Лекционная с уст. компьютерами

```
select * from AUDITORIUM_TYPE where AUDITORIUM_TYPENAME like '%комп%кл%'
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
ЛБ-К	Компьютерный класс
ЛБ-СК	Спец. компьютерный класс

WHERE LIKE

```
select * from AUDITORIUM_TYPE
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
ЛБ-Х	Химическая лаборатория
ЛБ-К	Компьютерный класс
ЛБ-СК	Спец. компьютерный класс
ЛК	Лекционная
ЛК-К	Лекционная с уст. компьютерами

```
SELECT * FROM AUDITORIUM_TYPE  
WHERE AUDITORIUM_TYPENAME LIKE 'Лекционна%';
```

% ▾

Results Messages

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
ЛК	Лекционная
ЛК-К	Лекционная с уст. проектором

```
SELECT * FROM AUDITORIUM_TYPE  
WHERE AUDITORIUM_TYPENAME LIKE 'Лекционна_';
```

6 ▾

Results Messages

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
ЛК	Лекционная

WHERE LIKE

```
INSERT INTO AUDITORIUM_TYPE(AUDITORIUM_TYPE,AUDITORIUM_TYPENAME)
VALUES ('%%%', 'Неописуемый класс');

SELECT * FROM AUDITORIUM_TYPE
WHERE AUDITORIUM_TYPE LIKE '%[%]%';
```

0 %

Results

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
%%%	Неописуемый класс

(1 row(s) affected)

```
SELECT * FROM AUDITORIUM_TYPE
WHERE AUDITORIUM_TYPE LIKE '[%]';
```

%

Results

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME

(0 row(s) affected)

WHERE - UPDATE

```
| UPDATE AUDITORIUM SET AUDITORIUM_CAPACITY = AUDITORIUM_CAPACITY * 1.1  
| WHERE AUDITORIUM_CAPACITY >= 90;  
| SELECT AUDITORIUM, AUDITORIUM_CAPACITY FROM AUDITORIUM;
```

The screenshot shows a Windows-style application window titled 'Results' with a tab bar below it. The 'Results' tab is selected, and the 'Messages' tab is visible to its right. The main area displays a table with two columns: 'AUDITORIUM' and 'AUDITORIUM_CAPACITY'. The table contains 15 rows of data, each representing an auditorium with its capacity multiplied by 1.1.

AUDITORIUM	AUDITORIUM_CAPACITY
103-4	99
105-4	99
107-4	99
110-4	30
111-4	30
114-4	99
206-1	15
236-1	60
301-1	15
313-1	60
324-1	50
408-2	99
413-1	15
423-1	99

WHERE - DELETE

```
DELETE FROM AUDITORIUM WHERE AUDITORIUM_CAPACITY = 50;
```

```
SELECT AUDITORIUM, AUDITORIUM_CAPACITY FROM AUDITORIUM;
```

The screenshot shows a database query results window. At the top, there are two tabs: "Results" (which is selected) and "Messages". Below the tabs is a table with two columns: "AUDITORIUM" and "AUDITORIUM_CAPACITY". The table contains 14 rows of data. The first row, which has the value "103-4" in the "AUDITORIUM" column, is highlighted with a dashed border.

AUDITORIUM	AUDITORIUM_CAPACITY
103-4	90
105-4	90
107-4	90
110-4	30
111-4	30
114-4	90
206-1	15
236-1	60
301-1	15
313-1	60
408-2	90
413-1	15
423-1	90

GROUP BY

```
select AUDITORIUM_TYPE from AUDITORIUM group by AUDITORIUM_TYPE
```

AUDITORIUM_TYPE
ЛБ-К
ЛК
ЛК-К

```
select AUDITORIUM_TYPE, AUDITORIUM_CAPACITY from AUDITORIUM  
group by AUDITORIUM_TYPE, AUDITORIUM_CAPACITY
```

AUDITORIUM_TYPE	AUDITORIUM_CAPACITY
ЛБ-К	15
ЛБ-К	90
ЛК	30
ЛК	50
ЛК	60
ЛК	90
ЛК-К	90

GROUP BY

- Каждый столбец в списке выборки запроса также должен присутствовать в предложении GROUP BY
- Не распространяется на константы и столбцы, являющиеся частью агрегатной функции
 - MIN
 - MAX
 - SUM
 - AVG
 - COUNT
- Последовательность имен столбцов в GROUP BY не обязательно должна быть такой же, как SELECT

GROUP BY

```
SELECT AUDITORIUM_TYPE,
       AUDITORIUM_NAME,
       SUM(AUDITORIUM_CAPACITY) AS [Суммарная вместимость]
  FROM AUDITORIUM
 GROUP BY AUDITORIUM_TYPE;
```

% < >

Messages

Msg 8120, Level 16, State 1, Line 3
Column 'AUDITORIUM.AUDITORIUM_NAME' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

GROUP BY

```
select AUDITORIUM_TYPE, COUNT(*) from AUDITORIUM  
group by AUDITORIUM_TYPE
```

AUDITORIUM_TYPE	(Отсутствует имя столбца)
ЛБ-К	5
ЛК	16
ЛК-К	1

```
select AUDITORIUM_CAPACITY, COUNT(*) from AUDITORIUM  
group by AUDITORIUM_CAPACITY
```

AUDITORIUM_CAPACITY	(Отсутствует имя столбца)
15	3
30	2
50	1
60	2
90	14

GROUP BY

```
select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM
```

(Отсутствует имя столбца)

1535

```
SELECT AUDITORIUM_TYPE,  
       SUM(AUDITORIUM_CAPACITY) AS [Суммарная вместимость]  
  FROM AUDITORIUM  
 GROUP BY AUDITORIUM_TYPE;
```

%

Results | Messages

AUDITORIUM_TYPE	Суммарная вместимость
ЛБ-К	135
ЛК	480
ЛК-К	200

GROUP BY

```
select SUM(AUDITORIUM_CAPACITY) 'сумма' ,  
MIN(AUDITORIUM_CAPACITY) 'min' ,  
MAX(AUDITORIUM_CAPACITY) 'max' from AUDITORIUM
```

	сумма	min	max
	1535	15	90

```
select AUDITORIUM_TYPE, SUM(AUDITORIUM_CAPACITY) 'сумма' ,  
MIN(AUDITORIUM_CAPACITY) 'min' ,  
MAX(AUDITORIUM_CAPACITY) 'max'  
from AUDITORIUM  
group by AUDITORIUM_TYPE
```

AUDITORIUM_TYPE	сумма	min	max
ЛБ-К	225	15	90
ЛК	1220	30	90
ЛК-К	90	90	90

GROUP BY

```
SELECT AUDITORIUM_TYPE,
       MIN(AUDITORIUM_CAPACITY) AS [Минимальная вместимость]
  FROM AUDITORIUM
 WHERE AUDITORIUM_CAPACITY >30
 GROUP BY AUDITORIUM_TYPE;
```

The screenshot shows a database query window with a toolbar at the top and two tabs: 'Results' and 'Messages'. The 'Results' tab is selected, displaying a table with three rows. The table has two columns: 'AUDITORIUM_TYPE' and 'Минимальная вместимость' (Minimum capacity). The data is as follows:

AUDITORIUM_TYPE	Минимальная вместимость
ЛБ-К	90
ЛК	60
ЛК-К	50

HAVING

- В предложении HAVING определяется условие, которое применяется к группе строк.
- Синтаксис:
- HAVING *condition*

```
SELECT AUDITORIUM_TYPE,
       COUNT(AUDITORIUM_CAPACITY) AS [Количество]
  FROM AUDITORIUM
 GROUP BY AUDITORIUM_TYPE;
```

AUDITORIUM_TYPE	Количество
ЛБ-К	4
ЛК	7
ЛК-К	3

```
] SELECT AUDITORIUM_TYPE,
       COUNT(AUDITORIUM_CAPACITY) AS [Количество]
  FROM AUDITORIUM
 GROUP BY AUDITORIUM_TYPE
 HAVING COUNT(AUDITORIUM_CAPACITY) > 3;
```

AUDITORIUM_TYPE	Количество
ЛБ-К	4
ЛК	7

HAVING

```
SELECT AUDITORIUM_TYPE  
FROM AUDITORIUM  
GROUP BY AUDITORIUM_TYPE  
HAVING AUDITORIUM_TYPE = 'ЛБ-К';
```

The screenshot shows a database query window with a tree view on the left and a results grid on the right. The results grid has two tabs: 'Results' and 'Messages'. The 'Results' tab is selected, displaying a single row with one column labeled 'AUDITORIUM_TYPE' containing the value 'ЛБ-К'.

AUDITORIUM_TYPE
ЛБ-К



DISTINCT

```
select AUDITORIUM_CAPACITY from AUDITORIUM
```

AUDITORIUM_CAPACITY
90
90
90
90
90
30
30
90
90
15
90
60
15
90
60
90
90
50
90

```
select distinct AUDITORIUM_CAPACITY from AUDITORIUM
```

AUDITORIUM_CAPACITY
15
30
50
60
90

ORDER BY

```
select * from AUDITORIUM order by AUDITORIUM asc
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
?	???	90	ЛК
026-4	026-4	90	ЛК
103-4	103-4	90	ЛК
105-4	105-4	90	ЛК
107-4	107-4	90	ЛК
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
114-4	114-4	90	ЛК-К
132-4	132-4	90	ЛК
206-1	206-1	15	ЛБ-К
229-4	229-4	90	ЛК
236-1	236-1	60	ЛК
301-1	301-1	15	ЛБ-К
304-4	304-4	90	ЛБ-К
313-1	313-1	60	ЛК
314-4	314-4	90	ЛК
320-4	320-4	90	ЛК
324-1	324-1	50	ЛК
408-2	408-2	90	ЛК
413-1	413-1	15	ЛБ-К

ORDER BY

```
select * from AUDITORIUM order by AUDITORIUM desc
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
429-4	429-4	90	ЛК
423-1	423-1	90	ЛБ-К
413-1	413-1	15	ЛБ-К
408-2	408-2	90	ЛК
324-1	324-1	50	ЛК
320-4	320-4	90	ЛК
314-4	314-4	90	ЛК
313-1	313-1	60	ЛК
304-4	304-4	90	ЛБ-К
301-1	301-1	15	ЛБ-К
236-1	236-1	60	ЛК
229-4	229-4	90	ЛК
206-1	206-1	15	ЛБ-К
132-4	132-4	90	ЛК
114-4	114-4	90	ЛК-К
111-4	111-4	30	ЛК
110-4	110-4	30	ЛК
107-4	107-4	90	ЛК
105-4	105-4	90	ЛК
103-4	103-4	90	ПК

ORDER BY

```
select * from AUDITORIUM order by AUDITORIUM_CAPACITY desc,  
AUDITORIUM_TYPE asc
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
304-4	304-4	90	ЛБ-К
423-1	423-1	90	ЛБ-К
429-4	429-4	90	ЛК
408-2	408-2	90	ЛК
314-4	314-4	90	ЛК
320-4	320-4	90	ЛК
?	???	90	ЛК
026-4	026-4	90	ЛК
103-4	103-4	90	ЛК
105-4	105-4	90	ЛК
107-4	107-4	90	ЛК
132-4	132-4	90	ЛК
229-4	229-4	90	ЛК
114-4	114-4	90	ЛК-К
236-1	236-1	60	ЛК
313-1	313-1	60	ЛК
324-1	324-1	50	ЛК
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК

TOP

```
select top 5 * from AUDITORIUM
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
?	???	90	ЛК
026-4	026-4	90	ЛК
103-4	103-4	90	ЛК
105-4	105-4	90	ЛК
107-4	107-4	90	ЛК

```
SELECT TOP 5 AUDITORIUM FROM AUDITORIUM  
ORDER BY AUDITORIUM DESC;
```

The screenshot shows a software interface for running SQL queries. At the top, there is a text input field containing the query. Below it, there are two tabs: 'Results' (which is currently selected) and 'Messages'. The 'Results' tab displays a vertical list of values, likely the output of the query, which consists of five rows of auditorium numbers.

AUDITORIUM
423-1
413-1
408-2
324-1
313-1

INTO

```
SELECT  
    AUDITORIUM_TYPE,  
    AUDITORIUM_TYPENAME  
INTO AUD_TYPE  
FROM AUDITORIUM_TYPE  
WHERE AUDITORIUM_TYPE = 'ЛК';
```



```
(1 row(s) affected)  
  
SELECT * FROM AUD_TYPE;
```

A screenshot of a database interface showing the 'Results' tab. The tab is highlighted in blue, indicating it is active. Below the tab, there is a table with two columns: 'AUDITORIUM_TYPE' and 'AUDITORIUM_TYPENAME'. The data row shows 'ЛК' in the first column and 'Лекционная' in the second column.

	AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
	ЛК	Лекционная

Временные таблицы

- Временные таблицы создаются для временного хранения результатов SELECT-запросов
- Локальные временные таблицы
 - имена, начинаются с символа #
 - доступны только пользователю, ее создавшему
 - автоматически удаляется при отключении пользователя
- Глобальные временные таблицы
 - имена, начинаются с символов ##,
 - доступны всем пользователям, подключенными к серверу

Временные таблицы

```
create table #TEACHER
(
    TEACHER      char(10)
                constraint [I_TEACHER_PK] primary key,
    TEACHER_NAME varchar(100) default '???',
    GENDER        char(1) default 'м'
                constraint [I_TEACHER_GENDER_CH] check (GENDER in ('м', 'ж')),
    PULPIT       char(20)
);
go
insert into #TEACHER (TEACHER, TEACHER_NAME, PULPIT )
values ('СМЛВ', 'Смелов Владимир Владиславович', 'ИСиТ'),
       ('АКНВЧ', default, 'ИСиТ'),
       ('КЛСНВ', null, 'ИСиТ'),
       ('ГРМН', 'Герман Олег Витольдович', 'ИСиТ'),
       ('ЛЩНК', 'Лашенко Анатолий Павлович', 'ИСиТ');
update #TEACHER set TEACHER_NAME = TEACHER where TEACHER_NAME = '???';
delete #TEACHER where TEACHER_NAME is null;
select * from #TEACHER;
drop table #TEACHER;
```

Вопросы?

БАЗЫ ДАННЫХ

Лекция 6
Соединение таблиц

Оператор соединения

- **JOIN** - позволяет извлекать данные более чем из одной таблицы

Варианты синтаксиса соединения

- явный синтаксис соединения (ANSI SQL:1992)
- неявный синтаксис соединения (старого стиля)

Пример

- Employee
- empid
- name
- firstname
- dep_no
- Department
- dep_no
- department

Пример

empid	firstname	name	dep_no
18316	John	Barrimore	d1
29346	James	James	d2
25348	Matthew	Smith	d3
10102	Ann	Jones	d3
28559	Sybill	Moser	d1
9031	Elsa	Bertoni	d2

Пример

dep_no	department
d1	Research
d2	Accounting
d3	Marketing

ЯВНЫЙ СИНТАКСИС

SELECT

name,

firstname,

department

FROM employee

INNER JOIN department ON

employee.dep_no =

department.dep_no;

ЯВНЫЙ СИНТАКСИС

- CROSS JOIN;
- [INNER] JOIN;
- LEFT [OUTER] JOIN;
- RIGHT [OUTER] JOIN;
- FULL [OUTER] JOIN.

Неявный синтаксис

SELECT

name,

firstname,

department

FROM employee, department

WHERE

employee.dep_no =

department.dep_no;

Соединение таблиц

empid	firstname	name	dep_no	dep_no	department
18316	John	Barrimore	d1	d1	Research
18316	John	Barrimore	d1	d2	Accounting
18316	John	Barrimore	d1	d3	Marketing
29346	James	James	d2	d1	Research
29346	James	James	d2	d2	Accounting
29346	James	James	d2	d3	Marketing

Соединение таблиц

empid	firstname	name	dep_no	dep_no	department
25348	Matthew	Smith	d3	d1	Research
25348	Matthew	Smith	d3	d2	Accounting
25348	Matthew	Smith	d3	d3	Marketing
10102	Ann	Jones	d3	d1	Research
10102	Ann	Jones	d3	d2	Accounting
10102	Ann	Jones	d3	d3	Marketing

Соединение таблиц

empid	firstname	name	dep_n o	dep_n o	department
28559	Sybill	Moser	d1	d1	Research
28559	Sybill	Moser	d1	d2	Accounting
28559	Sybill	Moser	d1	d3	Marketing
9031	Elsa	Bertoni	d2	d1	Research
9031	Elsa	Bertoni	d2	d2	Accounting
9031	Elsa	Bertoni	d2	d1	Research

Соединение таблиц

- Соединяется каждый с каждым
- Остаются только совпадающие
- Реализуются разными стратегиями
- Неявный синтаксис устарел

Соединение более чем двух таблиц

- Ограничение - 64 таблицы
- SELECT name, firstname
- FROM works_on
- JOIN employee ON
works_on.empid=employee.empid
- JOIN department ON
employee.dep_no=department.dep_no

Виды соединений

- Естественное соединение
- Декартово произведение
(перекрестное соединение)
- Внешнее соединение
- Тета-соединение, самосоединение и
полусоединение

Естественное соединение

- Внутреннее соединение содержит только те строки одной таблицы, для которых имеются соответствующие строки в другой таблице
 - Соединяемые таблицы
 - Условие соединения
 - Столбцы соединения

Естественное соединение

AUDITORIUM_TYPE				
	Имя столбца	Сжатый тип	Допускает значения NULL	Тип данных
◆	AUDITORIUM_TYPE	char(10)	Нет	char(10)
	AUDITORIUM_TYPENAME	varchar(30)	Да	varchar(30)



AUDITORIUM *				
	Имя столбца	Сжатый тип	Допускает значения NULL	Тип данных
◆	AUDITORIUM	char(10)	Нет	char(10)
	AUDITORIUM_NAME	varchar(2...	Да	varchar(2...
	AUDITORIUM_CA...	int	Да	int
	AUDITORIUM_TYPE	char(10)	Нет	char(10)

```
select aa.AUDITORIUM, aa.AUDITORIUM_CAPACITY, at.AUDITORIUM_TYPENAME  
from AUDITORIUM_TYPE at inner join AUDITORIUM aa  
on at.AUDITORIUM_TYPE = aa.AUDITORIUM_TYPE
```

AUDITORIUM	AUDITORIUM_CAPACITY	AUDITORIUM_TYPENAME
?	90	Лекционная
026-4	90	Лекционная
103-4	90	Лекционная
105-4	90	Лекционная
107-4	90	Лекционная
110-4	30	Лекционная
111-4	30	Лекционная
114-4	90	Лекционная с уст. компьютерами
132-4	90	Лекционная
206-1	15	Компьютерный класс
229-4	90	Лекционная
236-1	60	Лекционная
301-1	15	Компьютерный класс
304-4	90	Компьютерный класс
313-1	60	Лекционная
314-4	90	Лекционная
320-4	90	Лекционная
324-1	50	Лекционная
408-2	90	Лекционная
413-1	15	Компьютерный класс
423-1	90	Компьютерный класс
429-4	90	Лекционная

Естественное соединение

```
select aa.AUDITORIUM, aa.AUDITORIUM_CAPACITY, at.AUDITORIUM_TYPENAME  
      from AUDITORIUM_TYPE at join AUDITORIUM aa  
      on at.AUDITORIUM_TYPE = aa.AUDITORIUM_TYPE
```

Декартово произведение

- Ортогональное соединение, перекрестное соединение
- CROSS JOIN
- Декартово произведение соединяет каждую строку первой таблицы с каждой строкой второй таблицы
- Количество строк $n \times m$

Декартово произведение

```
select * from AUDITORIUM CROSS JOIN AUDITORIUM_TYPE;
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE	AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
?	???	90	ЛК	ЛБ-Х	Химическая лаборатория
026-4	026-4	90	ЛК	ЛБ-Х	Химическая лаборатория
103-4	103-4	90	ЛК	ЛБ-Х	Химическая лаборатория
105-4	105-4	90	ЛК	ЛБ-Х	Химическая лаборатория
107-4	107-4	90	ЛК	ЛБ-Х	Химическая лаборатория
110-4	110-4	30	ЛК	ЛБ-Х	Химическая лаборатория
111-4	111-4	30	ЛК	ЛБ-Х	Химическая лаборатория
114-4	114-4	90	ЛК-К	ЛБ-Х	Химическая лаборатория
132-4	132-4	90	ЛК	ЛБ-Х	Химическая лаборатория
229-4	229-4	90	ЛК	ЛБ-Х	Химическая лаборатория
236-1	236-1	60	ЛК	ЛБ-Х	Химическая лаборатория
301-1	301-1	15	ЛБ-К	ЛБ-Х	Химическая лаборатория
304-4	304-4	90	ЛБ-К	ЛБ-Х	Химическая лаборатория
313-1	313-1	60	ЛК	ЛБ-Х	Химическая лаборатория
314-4	314-4	90	ЛК	ЛБ-Х	Химическая лаборатория

Декартово произведение

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM CROSS JOIN AUDITORIUM_TYPE) 'к-во строк в CROSS JOIN'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в CROSS JOIN
21	5	105

Внешнее соединение

- При внешнем соединении результирующий набор содержит все строки одной таблицы и те из второй таблицы, для которых имеются соответствующие строки в первой таблице.

LEFT OUTER JOIN

```
insert into AUDITORIUM_TYPE (AUDITORIUM_TYPE, AUDITORIUM_TYPENAME )
    values ('XX', 'XXXXX');
```

```
select * from AUDITORIUM_TYPE at LEFT OUTER JOIN AUDITORIUM aa
    on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE;
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME	AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
XX	XXXXX	NULL	NULL	NULL	NULL
ЛБ-Х	Химическая лаборатория	NULL	NULL	NULL	NULL
ЛБ-К	Компьютерный класс	301-1	301-1	15	ЛБ-К
ЛБ-К	Компьютерный класс	304-4	304-4	90	ЛБ-К
ЛБ-К	Компьютерный класс	413-1	413-1	15	ЛБ-К
ЛБ-К	Компьютерный класс	423-1	423-1	90	ЛБ-К
ЛБ-СК	Спец. компьютерный класс	NULL	NULL	NULL	NULL
ЛК	Лекционная	?	???	90	ЛК
ЛК	Лекционная	026-4	026-4	90	ЛК
ЛК	Лекционная	103-4	103-4	90	ЛК
ЛК	Лекционная	105-4	105-4	90	ЛК
ЛК	Лекционная	107-4	107-4	90	ЛК
ЛК	Лекционная	110-4	110-4	30	ЛК
ЛК	Лекционная	111-4	111-4	30	ЛК
ЛК	Лекционная	132-4	132-4	90	ЛК

LEFT OUTER JOIN

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE at LEFT OUTER JOIN AUDITORIUM aa
            on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) 'к-во строк в LEFT OUTER JOIN'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в LEFT OUTER JOIN
21	6	24

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM aa LEFT OUTER JOIN AUDITORIUM_TYPE at
            on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) 'к-во строк в LEFT OUTER JOIN'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в LEFT OUTER JOIN
21	6	21

RIGHT OUTER JOIN

```
select * from AUDITORIUM_TYPE at RIGHT OUTER JOIN AUDITORIUM aa  
on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE;
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME	AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
ЛК	Лекционная	?	???	90	ЛК
ЛК	Лекционная	026-4	026-4	90	ЛК
ЛК	Лекционная	103-4	103-4	90	ЛК
ЛК	Лекционная	105-4	105-4	90	ЛК
ЛК	Лекционная	107-4	107-4	90	ЛК
ЛК	Лекционная	110-4	110-4	30	ЛК
ЛК	Лекционная	111-4	111-4	30	ЛК
ЛК-К	Лекционная с уст. комп...	114-4	114-4	90	ЛК-К
ЛК	Лекционная	132-4	132-4	90	ЛК
ЛК	Лекционная	229-4	229-4	90	ЛК
ЛК	Лекционная	236-1	236-1	60	ЛК
ЛБ-К	Компьютерный класс	301-1	301-1	15	ЛБ-К
ЛБ-К	Компьютерный класс	304-4	304-4	90	ЛБ-К
ЛК	Лекционная	313-1	313-1	60	ЛК
ПК	Практическая	314-4	314-4	90	ПК

RIGHT OUTER JOIN

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE at RIGHT OUTER JOIN AUDITORIUM aa
            on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) 'к-во строк'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в RIGHT OUTER JOIN
21	6	21

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM aa RIGHT OUTER JOIN AUDITORIUM_TYPE at
            on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) 'к-во строк'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в RIGHT OUTER JOIN
21	6	24

RIGHT OUTER JOIN

```
select * from AUDITORIUM aa RIGHT OUTER JOIN AUDITORIUM_TYPE at  
on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE  
order by AUDITORIUM;
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE	AUDITORIUM_TYPE	AUDITORIUM_TYPENAME
NULL	NULL	NULL	NULL	ЛБ-СК	Спец. компьютерный класс
NULL	NULL	NULL	NULL	ХХ	ХХХХХ
NULL	NULL	NULL	NULL	ЛБ-Х	Химическая лаборатория
?	???	90	ЛК	ЛК	Лекционная
026-4	026-4	90	ЛК	ЛК	Лекционная
103-4	103-4	90	ЛК	ЛК	Лекционная
105-4	105-4	90	ЛК	ЛК	Лекционная
107-4	107-4	90	ЛК	ЛК	Лекционная
110-4	110-4	30	ЛК	ЛК	Лекционная
111-4	111-4	30	ЛК	ЛК	Лекционная
114-4	114-4	90	ЛК-К	ЛК-К	Лекционная с уст. компь...

FULL OUTER JOIN

```
select * from AUDITORIUM_TYPE at FULL OUTER JOIN AUDITORIUM aa  
on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE  
order by aa.AUDITORIUM, at.AUDITORIUM_TYPE
```

AUDITORIUM_TYPE	AUDITORIUM_TYPENAME	AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
ХХ	XXXXX	NULL	NULL	NULL	NULL
ЛБ-Х	Химическая лаборатория	NULL	NULL	NULL	NULL
ЛБ-СК	Спец. компьютерный класс	NULL	NULL	NULL	NULL
ЛК	Лекционная	?	???	90	ЛК
ЛК	Лекционная	02б-4	02б-4	90	ЛК
ЛК	Лекционная	103-4	103-4	90	ЛК
ЛК	Лекционная	105-4	105-4	90	ЛК
ЛК	Лекционная	107-4	107-4	90	ЛК
ЛК	Лекционная	110-4	110-4	30	ЛК
ЛК	Лекционная	111-4	111-4	30	ЛК
ЛК-К	Лекционная с уст. компьютерами	114-4	114-4	90	ЛК-К
ЛК	Лекционная	132-4	132-4	90	ЛК
ЛК	Лекционная	229-4	229-4	90	ЛК

FULL OUTER JOIN

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE at FULL OUTER JOIN AUDITORIUM as
          on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) 'к-во строк в FULL OUTER JOIN'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в FULL OUTER JOIN
21	6	24

```
select (select COUNT(*) from AUDITORIUM) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM_TYPE) 'к-во строк в AUDITORIUM',
       (select COUNT(*) from AUDITORIUM aa FULL OUTER JOIN AUDITORIUM_TYPE at
          on aa.AUDITORIUM_TYPE = at.AUDITORIUM_TYPE) 'к-во строк в RIGHT OUTER JOIN'
```

к-во строк в AUDITORIUM	к-во строк в AUDITORIUM	к-во строк в RIGHT OUTER JOIN
21	6	24

Виды соединений

- Тета-соединение
- Самосоединение
- Полусоединение

Тета-соединение

- Соединение, в котором используется общее условие сравнения столбцов соединения, называется тета-соединением
- `SELECT name, firstname,
user_location, dept_location`
- `FROM employee JOIN department`
- `ON user_location < dept_location;`

Самосоединение

- При самосоединении таблица соединяется сама с собой, столбец таблицы сравнивается сам с собой
- `SELECT t1.dep_no, t1.dep_name,
t1.location`
- `FROM department t1 JOIN department t2`
- `ON t1.location = t2.location`
- `WHERE t1.dep_no <> t2.dep_no;`

Самосоединение

- Иерархия



Самосоединение

Results Messages

	id_employee	employee_name	department	head
1	1	John Dow	1	NULL
2	2	Jane Smith	1	1
3	3	Judith Perks	1	1

```
select e1.employee_name, e2.employee_name  
from Employee e1 inner join Employee e2  
on e1.head=e2.id_employee;
```

0 %

Results Messages

	employee_name	employee_name
	Jane Smith	John Dow
	Judith Perks	John Dow

Полусоединение

- Полусоединение возвращает набор всех строк из одной таблицы, для которой в другой таблице есть одно или несколько совпадений
- `SELECT empid, name, e.dep_no`
- `FROM employee e JOIN department d`
- `ON e.dep_no = d.dep_no`

Операция соединения

- Коммутативность операций соединения?

Вопросы?

БАЗЫ ДАННЫХ

Лекция 7 Подзапросы
Группировка данных

Лекция 7

- Подзапросы
- Операции над множествами
- Группировки
- DML

Подзапросы

- Конструкция SELECT, которая содержится в другой инструкции SELECT, называется подзапросом
- Первая инструкция SELECT подзапроса называется внешним запросом
- Внутренняя инструкция (или инструкции) SELECT, называется вложенным запросом
- Первым выполняется вложенный запрос, а его результат передается внешнему запросу

Подзапросы

```
CREATE TABLE FACULTY
```

```
(  
    FACULTY      CHAR(10)      NOT NULL,  
    FACULTY_NAME VARCHAR(50),  
    CONSTRAINT PK_FACULTY PRIMARY KEY(FACULTY)  
) ;
```

```
CREATE TABLE PROFESSION
```

```
(  
    PROFESSION    CHAR(20),  
    FACULTY       CHAR(10),  
    PROFESSION_NAME VARCHAR(100),  
    QUALIFICATION  VARCHAR(50),  
    CONSTRAINT PK_PROFESSION PRIMARY KEY(PROFESSION),  
    CONSTRAINT FK_PROFESSION FOREIGN KEY (FACULTY)  
        REFERENCES FACULTY (FACULTY)  
) ;
```

Подзапросы

```
select * from faculty;
```

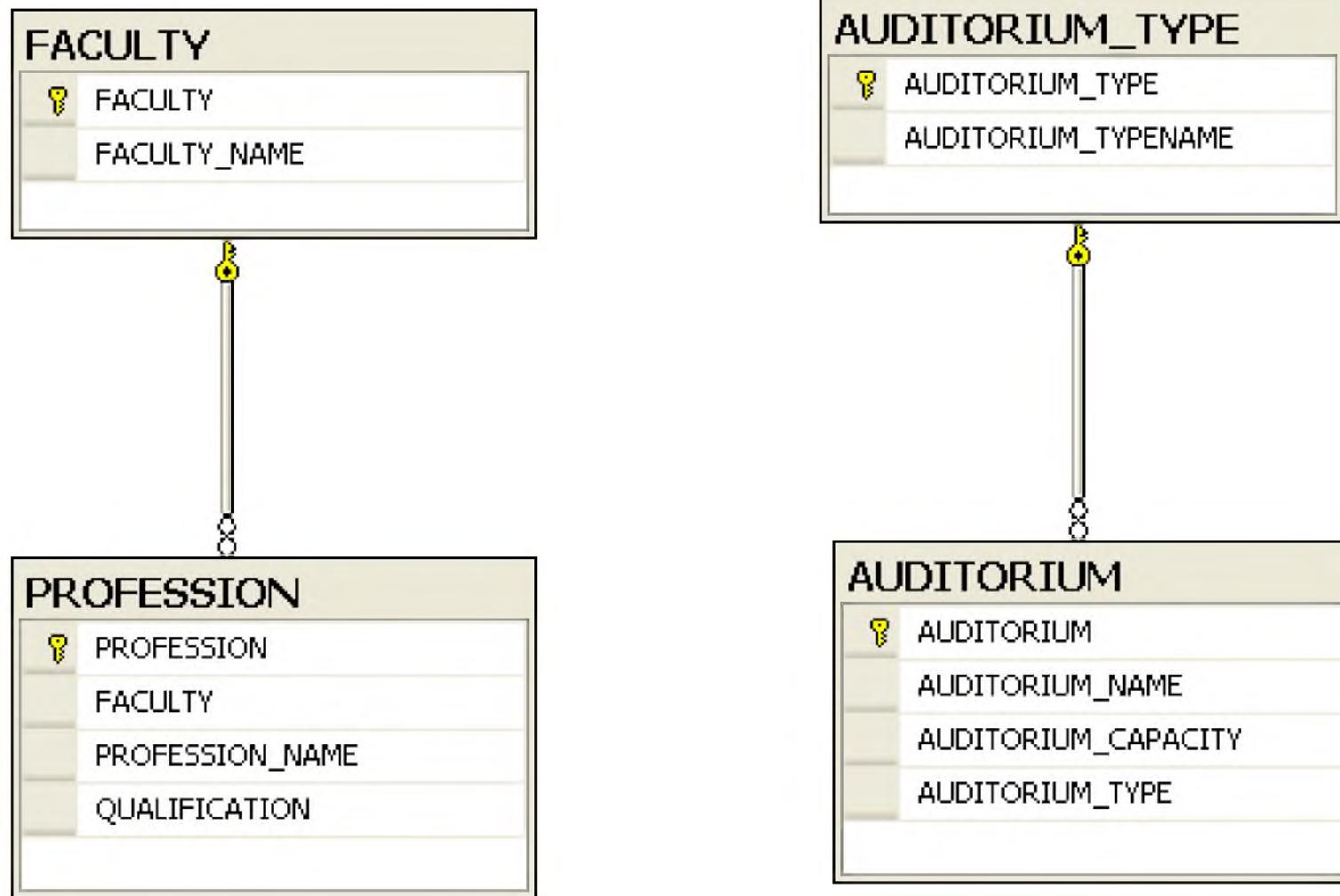
FACULTY	FACULTY_NAME
ИДиП	Издательское дело и полиграфия
ИЭФ	Инженерно-экономический факультет
ЛХФ	Лесохозяйственный факультет
ТОВ	Технология органических веществ
ТТЛП	Технология и техника лесной промышленности
ХТИТ	Химическая технология и техника

Подзапросы

```
select * from profession
```

PROFESSION	FACULTY	PROFESSION_NAME	QUALIFICATION
1-25 01 07	ИЭФ	Экономика и управление на предприятиях	экономист-менеджер
1-25 01 08	ИЭФ	Бухгалтерский учет, анализ и аудит	экономист
1-36 01 08	ХТиТ	Конструирование и производство изделий из компози...	инженер-механик
1-36 05 01	ТТЛП	Машины и оборудование лесного комплекса	инженер-механик
1-36 07 01	ХТиТ	Машины и аппараты химических производств и предп...	инженер-механик
1-40 01 02	ИДиП	Информационные системы и технологии	инженер-программист-системотехник
1-46 01 01	ТТЛП	Лесоинженерное дело	инженер-технолог
1-47 01 01	ИДиП	Издательское дело	редактор-технолог
1-48 01 02	ТОВ	Химическая технология органических веществ, матер...	инженер-химик-технолог
1-48 01 05	ТОВ	Химическая технология переработки древесины	инженер-химик-технолог
1-54 01 03	ТОВ	Физико-химические методы и приборы контроля каче...	инженер по сертификации
1-75 01 01	ЛХФ	Лесное хозяйство	инженер лесного хозяйства
1-75 02 01	ЛХФ	Садово-парковое строительство	инженер садово-паркового строительства
1-89 02 02	ЛХФ	Туризм и природопользование	специалист в сфере туризма

Подзапросы



Подзапросы

Найти специальности, которые
получают на
Лесохозяйственном
факультете?

Подзапросы

- Найти код лесохозяйственного факультета
- Найти специальности по коду факультета

Подзапросы

```
SELECT FACULTY FROM FACULTY  
WHERE FACULTY_NAME = 'Лесохозяйственный факультет';
```

% ▾

Results | Messages |

FACULTY
ЛХФ

```
SELECT QUALIFICATION FROM PROFESSION  
WHERE FACULTY = 'ЛХФ';
```

% ▾

Results | Messages |

QUALIFICATION
инженер лесного хозяйства
инженер садово-паркового строительства
специалист в сфере туризма

Подзапросы

```
SELECT QUALIFICATION FROM PROFESSION
    WHERE FACULTY = (SELECT FACULTY FROM FACULTY
    WHERE FACULTY_NAME = 'Лесохозяйственный факультет');
```

The screenshot shows a SQL Server Management Studio interface. At the top, there's a toolbar with a dropdown arrow and a refresh button. Below it is a tab bar with 'Results' selected and 'Messages' available. The main area displays the results of the executed query:

QUALIFICATION
инженер лесного хозяйства
инженер садово-паркового строительства
специалист в сфере туризма

Подзапросы

- FROM
- WHERE
- SELECT

Подзапросы

- *Коррелируемый* подзапрос зависит от внешнего запроса и выполняется для каждой строки результирующего набора.
- *Независимый* подзапрос не зависит от внешнего запроса и выполняется только один раз, но результат его выполнения подставляется в каждую строку результирующего набора.
- Допускается применять только такие подзапросы, которые формируют скалярный результирующий набор

Подзапросы

Найти студентов, оценки которых по предмету меньше средней оценки по этому предмету

Подзапросы

```
SELECT * FROM PROGRESS;
```

SUBJECT	IDSTUDENT	PDATE	NOTE
ОАиП	1000	2013-01-10	6
ОАиП	1001	2013-01-10	8
ОАиП	1002	2013-01-10	7
ОАиП	1003	2013-01-10	5
ОАиП	1005	2013-01-10	4
СУБД	1014	2013-01-12	5
СУБД	1015	2013-01-12	9
СУБД	1016	2013-01-12	5
СУБД	1017	2013-01-12	4
КГ	1018	2013-06-05	4
КГ	1019	2013-06-05	7
КГ	1020	2013-06-05	7
КГ	1021	2013-06-05	9
КГ	1022	2013-06-05	5
КГ	1023	2013-06-05	6
ОХ	1064	2013-01-01	6
ОХ	1065	2013-01-01	4
пх	1066	2013-01-01	9

```
SELECT * FROM STUDENT;
```

IDSTUDENT	IDGROUP	NAME
1000	1	Хартанович Екатерина Александровна
1001	1	Горбач Елизавета Юрьевна
1002	1	Зыкова Кристина Дмитриевна
1003	1	Борисевич Ольга Анатольевна
1004	1	Медведева Мария Андреевна
1005	1	Шенец Екатерина Сергеевна
1006	1	Шитик Алина Игоревна
1007	2	Силюк Валерия Ивановна
1008	2	Сергель Виолетта Николаевна
1009	2	Добродей Ольга Анатольевна
1010	2	Подоляк Мария Сергеевна
1011	2	Никитенко Екатерина Дмитриевна
1012	3	Яцкевич Галина Иосифовна

Подзапросы

```
SELECT SUBJECT, AVG(NOTE) FROM PROGRESS GROUP BY SUBJECT;
```

The screenshot shows a database query results window. At the top, there are two tabs: 'Results' (which is selected) and 'Messages'. Below the tabs is a table with five rows. The table has two columns: 'SUBJECT' and '(No column name)'. The data is as follows:

SUBJECT	(No column name)
КГ	6
ОАиП	6
ОХ	6
СУБД	5
ЭТ	6

```
SELECT NAME, SUBJECT, NOTE
  FROM STUDENT S1 INNER JOIN PROGRESS P1
    ON S1.IDSTUDENT = P1.IDSTUDENT
 WHERE NOTE < (SELECT AVG(NOTE) FROM PROGRESS P2
                  WHERE P1.SUBJECT = P2.SUBJECT
                  GROUP BY SUBJECT)
 ORDER BY NAME;
```

The screenshot shows a Windows application window titled 'Results' containing a table of student data. The table has three columns: 'NAME', 'SUBJECT', and 'NOTE'. The data is as follows:

NAME	SUBJECT	NOTE
Абрамов Денис Дмитриевич	ЭТ	5
Борисевич Ольга Анатольевна	ОАиП	5
Бороховский Виталий Петрович	СУБД	2
Буянова Мария Александровна	КГ	4
Власик Евгения Викторовна	ЭТ	4
Горбач Елизавета Юрьевна	СУБД	3
Даниленко Максим Васильевич	ОХ	5
Кузмичева Анна Андреевна	ОХ	4
Мацкевич Надежда Валерьевна	СУБД	2
Мацкевич Надежда Валерьевна	КГ	5
Плешкун Милана Анатольевна	ОАиП	4
Плешкун Милана Анатольевна	СУБД	4
Шарапо Мария Владимировна	ОХ	4
Шенец Екатерина Сергеевна	ОАиП	4
Шенец Екатерина Сергеевна	СУБД	4

Подзапросы

Может применяться со следующими операторами:

- операторами сравнения и EXISTS;
- оператором IN;
- операторами ANY и ALL;

Подзапросы

```
select * from AUDITORIUM  
|  
| select AUDITORIUM_TYPENAME  
from AUDITORIUM_TYPE at  
where exists (  
    | select * from AUDITORIUM aa  
    where at.AUDITORIUM_TYPE = aa.AUDITORIUM_TYPE  
        and aa.AUDITORIUM_CAPACITY > 60  
    )
```

AUDITORIUM_TYPENAME

Компьютерный класс

Лекционная

Лекционная с уст. компьютерами

Подзапросы

```
select AUDITORIUM_TYPENAME  
from AUDITORIUM_TYPE at  
where not exists (  
    _____  
        select * from AUDITORIUM aa  
        where at.AUDITORIUM_TYPE = aa.AUDITORIUM_TYPE  
        and aa.AUDITORIUM_CAPACITY > 60  
    )
```

AUDITORIUM_TYPENAME

XXXXX

Химическая лаборатория

Спец. компьютерный класс

Подзапросы

```
select f.FACULTY_NAME, p.QUALIFICATION, p.PROFESSION
from FACULTY f join (
    — — —
        select FACULTY, QUALIFICATION, PROFESSION
        from PROFESSION
        where (QUALIFICATION like '%технолог%')
    ) p
    on f.FACULTY = p.FACULTY
```

FACULTY_NAME	QUALIFICATION	PROFESSION
Технология и техника лесной промышленности	инженер-технолог	1-46 01 01
Издательское дело и полиграфия	редактор-технолог	1-47 01 01
Технология органических веществ	инженер-химик-технолог	1-48 01 02
Технология органических веществ	инженер-химик-технолог	1-48 01 05

Подзапросы с IN

```
select FACULTY_NAME  
from FACULTY  
where FACULTY in (select FACULTY from PROFESSION where (QUALIFICATION like '%технолог%'))
```

FACULTY_NAME

Издательское дело и полиграфия

Технология органических веществ

Технология и техника лесной промышленности

```
select FACULTY_NAME  
from FACULTY  
where FACULTY not in (select FACULTY from PROFESSION where (QUALIFICATION like '%технолог%'))
```

FACULTY_NAME

Инженерно-экономический факультет

Лесохозяйственный факультет

Химическая технология и техника

Подзапросы с ANY и ALL

column_name operator [ANY|ALL] query

Оператор ANY возвращает значение TRUE, если результат вложенного запроса содержит хотя бы одну строку, удовлетворяющую условию сравнения.

Подзапросы с ANY и ALL

```
select AUDITORIUM_TYPENAME  
from AUDITORIUM_TYPE at  
where 60 >= ANY(  
    _____ select AUDITORIUM_CAPACITY from AUDITORIUM aa  
    where at.AUDITORIUM_TYPE = aa.AUDITORIUM_TYPE  
)
```

AUDITORIUM_TYPENAME

Компьютерный класс

Лекционная

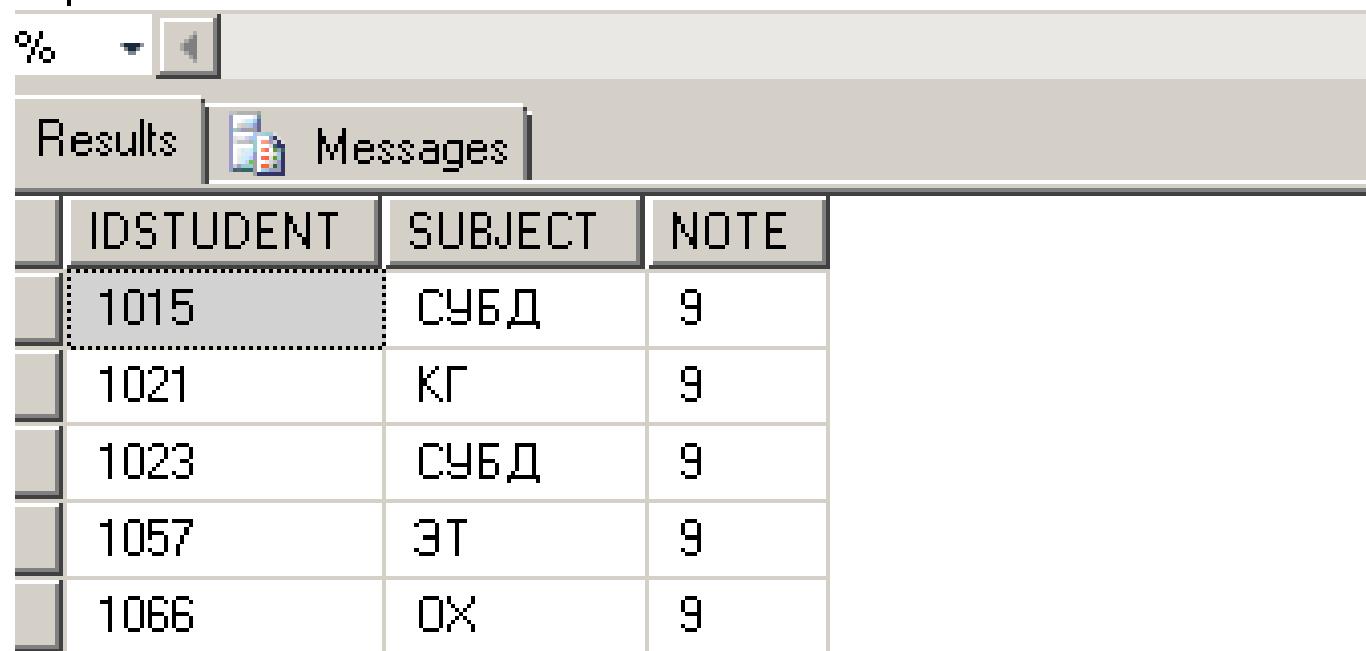
Подзапросы с ANY и ALL

column_name operator [ANY|ALL] query

Оператор ALL возвращает значение TRUE, если результат вложенного запроса возвращает все значения, обрабатываемого столбца, удовлетворяющие условию сравнения.

Подзапросы с ANY и ALL

```
SELECT IDSTUDENT, SUBJECT, NOTE  
FROM PROGRESS  
WHERE NOTE >= ALL(SELECT NOTE FROM PROGRESS)  
ORDER BY IDSTUDENT;
```



The screenshot shows the SQL Server Management Studio interface with the following details:

- Query Editor:** The top pane contains the T-SQL query.
- Results Pane:** The bottom pane displays the query results in a tabular format. The results show five rows of data with columns: IDSTUDENT, SUBJECT, and NOTE. The data is as follows:

IDSTUDENT	SUBJECT	NOTE
1015	СУБД	9
1021	КГ	9
1023	СУБД	9
1057	ЭТ	9
1066	ОХ	9

Соединения или подзапросы?

Соединения или подзапросы?

- Подзапросы - когда требуется вычислить агрегатное значение "на лету" и использовать его в другом запросе для сравнения.
- SELECT emp_no, enter_date
- FROM works_on
- WHERE enter_date = (SELECT min(enter_date) FROM works_on);

Соединения или подзапросы?

- Соединения - когда список выбора инструкции SELECT в запросе содержит столбцы более чем из одной таблицы.
- SELECT employee.empid, name, job
- FROM employee INNER JOIN works_on
- ON employee.empid = works_on.empid
- AND enter_date = '11.03.2017';

Подзапросы

- Список выбора внутреннего подзапроса, которому предшествует операция сравнения, может содержать только одно выражение или название столбца, и подзапрос должен возвращать единственный результат.
- Тип данных столбца, указанного в конструкции `where` внешнего оператора, должен быть совместим с типом данных в столбце, указанным в списке выбора подзапроса

Подзапросы

- Количество вложенных уровней для подзапросов не должно превышать 16
- Максимальное число подзапросов на каждой стороне объединения не больше 16

Операции над множествами

- UNION
- UNION ALL
- INTERSECT
- EXCEPT

UNION

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY < 30
union
select * from AUDITORIUM where AUDITORIUM_CAPACITY > 60
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
206-1	206-1	15	ЛБ-К
301-1	301-1	15	ЛБ-К
413-1	413-1	15	ЛБ-К
?	???	90	ЛК
026-4	026-4	90	ЛК
103-4	103-4	90	ЛК
105-4	105-4	90	ЛК
107-4	107-4	90	ЛК
114-4	114-4	90	ЛК-К
132-4	132-4	90	ЛК
229-4	229-4	90	ЛК
304-4	304-4	90	ЛБ-К
314-4	314-4	90	ЛК
320-4	320-4	90	ЛК
408-2	408-2	90	ЛК
423-1	423-1	90	ЛБ-К
429-4	429-4	90	ЛК

UNION ALL

```
select * from AUDITORIUM where AUDITORIUM_CAPACITY < 30
union all
select * from AUDITORIUM where AUDITORIUM_CAPACITY < 60
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
206-1	206-1	15	ЛБ-К
301-1	301-1	15	ЛБ-К
413-1	413-1	15	ЛБ-К
110-4	110-4	30	ЛК
111-4	111-4	30	ЛК
206-1	206-1	15	ЛБ-К
301-1	301-1	15	ЛБ-К
324-1	324-1	50	ЛК
413-1	413-1	15	ЛБ-К

INTERSECT

```
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY < 30  
intersect  
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY > 60
```

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY
------------	-----------------	---------------------

```
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY > 30  
intersect  
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY < 90
```

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY
236-1	ЛК	60
313-1	ЛК	60
324-1	ЛК	50

EXCEPT

```
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY < 50  
except  
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY < 60
```

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY
------------	-----------------	---------------------

```
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY < 60  
except  
select AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY  
      from AUDITORIUM where AUDITORIUM_CAPACITY < 50
```

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY
324-1	ЛК	50

Операции над множествами

- Приоритет:
 - INTERSECT
 - UNION и EXCEPT

Группировки

- GROUP BY
- GROUP BY HAVING
- GROUP BY WITH CUBE
- GROUP BY WITH ROLLUP

GROUP BY

```
select AUDITORIUM_TYPE, COUNT(*) as 'количество'  
from AUDITORIUM group by AUDITORIUM_TYPE;
```

AUDITORIUM_TYPE	количество
ЛБ-К	5
ЛК	16
ЛК-К	1

GROUP BY

```
select AUDITORIUM_TYPE, COUNT(*) as 'количество'  
from AUDITORIUM  
where AUDITORIUM_CAPACITY > 15  
group by AUDITORIUM_TYPE
```

AUDITORIUM_TYPE	количество
ЛБ-К	2
ЛК	16
ЛК-К	1

```
select AUDITORIUM_CAPACITY, COUNT(*) as 'количество'  
from AUDITORIUM  
where AUDITORIUM_TYPE != 'ЛБ-К'  
group by AUDITORIUM_CAPACITY
```

AUDITORIUM_CAPACITY	количество
30	2
50	1
60	2
90	12

GROUP BY HAVING

```
select AUDITORIUM_CAPACITY, COUNT(*)
  from AUDITORIUM
 group by AUDITORIUM_CAPACITY
 having AUDITORIUM_CAPACITY > 15
```

AUDITORIUM_CAPACITY	(Отсутствует имя столбца)
30	2
50	1
60	2
90	14

GROUP BY HAVING

```
select AUDITORIUM_CAPACITY, COUNT(*), SUM(AUDITORIUM_CAPACITY)
  from AUDITORIUM
 group by AUDITORIUM_CAPACITY
```

AUDITORIUM_CAPACITY	(Отсутствует имя столбца)	(Отсутствует имя столбца)
15	3	45
30	2	60
50	1	50
60	2	120
90	14	1260

```
select AUDITORIUM_CAPACITY, COUNT(*)
  from AUDITORIUM
 group by AUDITORIUM_CAPACITY
 having SUM(AUDITORIUM_CAPACITY) >= 60
```

AUDITORIUM_CAPACITY	(Отсутствует имя столбца)
30	2
60	2
90	14

GROUP BY WITH CUBE

```
select AUDITORIUM_TYPE, COUNT(*) as 'количество'  
from AUDITORIUM group by AUDITORIUM_TYPE;
```

AUDITORIUM_TYPE	количество
ЛБ-К	5
ЛК	16
ЛК-К	1

```
select AUDITORIUM_TYPE, COUNT(*) as 'количество'  
from AUDITORIUM group by AUDITORIUM_TYPE with cube
```

AUDITORIUM_TYPE	количество
ЛБ-К	5
ЛК	16
ЛК-К	1
NULL	22

GROUP BY WITH CUBE

```
select AUDITORIUM_TYPE, AUDITORIUM_CAPACITY, COUNT(*) as 'количество'  
from AUDITORIUM  
group by AUDITORIUM_TYPE, AUDITORIUM_CAPACITY
```

AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	количество
ЛБ-К	15	3
ЛК	30	2
ЛК	50	1
ЛК	60	2
ЛБ-К	90	2
ЛК	90	11
ЛК-К	90	1

GROUP BY WITH CUBE

```
select AUDITORIUM_TYPE, AUDITORIUM_CAPACITY, COUNT(*) as 'количество'  
from AUDITORIUM  
group by AUDITORIUM_TYPE, AUDITORIUM_CAPACITY with cube .
```

AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	количество
ЛБ-К	15	3
NULL	15	3
ЛК	30	2
NULL	30	2
ЛК	50	1
NULL	50	1
ЛК	60	2
NULL	60	2
ЛБ-К	90	2
ЛК	90	11
ЛК-К	90	1
NULL	90	14
NULL	NULL	22
ЛБ-К	NULL	5
ЛК	NULL	16
ЛК-К	NULL	1

GROUP BY WITH ROLLUP

```
select AUDITORIUM_TYPE, COUNT(*) as 'количество'  
from AUDITORIUM group by AUDITORIUM_TYPE;
```

AUDITORIUM_TYPE	количество
ЛБ-К	5
ЛК	16
ЛК-К	1

```
select AUDITORIUM_TYPE, COUNT(*) as 'количество'  
from AUDITORIUM group by AUDITORIUM_TYPE with rollup;
```

AUDITORIUM_TYPE	количество
ЛБ-К	5
ЛК	16
ЛК-К	1
NULL	22

GROUP BY WITH ROLLUP

```
select AUDITORIUM_TYPE, AUDITORIUM_CAPACITY, COUNT(*) as 'количество'
from AUDITORIUM
group by AUDITORIUM_TYPE, AUDITORIUM_CAPACITY with rollup
```

AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	количество
ЛБ-К	15	3
ЛБ-К	90	2
ЛБ-К	NULL	5
ЛК	30	2
ЛК	50	1
ЛК	60	2
ЛК	90	11
ЛК	NULL	16
ЛК-К	90	1
ЛК-К	NULL	1
NULL	NULL	22

INSERT

```
exec SP_HELP [SUBJECT]
```

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource
SUBJECT	char	no	10			no	no	no
SUBJECT_NAME	varchar	no	100			yes	no	yes
PULPIT	char	no	20			yes	no	yes

```
insert into SUBJECT values ('СУБД',      'Системы управления базами данных', 'ИСиТ');  
insert into SUBJECT values ('БД',        'Базы данных', 'ИСиТ'),  
                           ('Инф',       'Информационные технологии', 'ИСиТ'),  
                           ('САиП',     'Основы алгоритмизации и программирования', 'ИСиТ'),  
                           ('ПЗ',        'Представление знаний в компьютерных системах', 'ИСиТ');
```

INSERT

```
exec SP_HELP GROUPS
```

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource
IDGROUP	int	no	4	10	0	no	(n/a)	(n/a)
FACULTY	char	no	10			yes	no	yes
PROFESSION	char	no	20			yes	no	yes
YEAR_FIRST	smallint	no	2	5	0	yes	(n/a)	(n/a)
COURSE	int	yes	4	10	0	yes	(n/a)	(n/a)

Identity	Seed	Increment	Not For Replication
IDGROUP	1	1	0

INSERT

```
INSERT INTO AUDITORIUM (AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY, AUDITORIUM_NAME)
VALUES ('311-4', '%%%', NULL, '311-4');

INSERT INTO AUDITORIUM (AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY, AUDITORIUM_NAME)
VALUES ('311-3', '%%%', DEFAULT, '311-3');

SELECT * FROM AUDITORIUM WHERE AUDITORIUM LIKE '311%';
```

The screenshot shows a database query results window. At the top, there are tabs for 'Results' and 'Messages'. The 'Results' tab is selected, displaying a table with four columns: AUDITORIUM, AUDITORIUM_TYPE, AUDITORIUM_CAPACITY, and AUDITORIUM_NAME. There are two rows of data:

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
311-3	%%%	1	311-3
311-4	%%%	NULL	311-4

INSERT

```
select st.NAME, pr.PDATE, sb.SUBJECT_NAME, pr.NOTE  
from STUDENT st join PROGRESS pr on st.IDSTUDENT = pr.IDSTUDENT  
join [SUBJECT] sb on pr.[SUBJECT] = sb.[SUBJECT]
```

NAME	PDATE	SUBJECT_NAME	NOTE
Манакова Анастасия Владимировна	2013-01-10	Основы алгоритмизации и программирования	6
Хартанович Екатерина Александровна	2013-01-10	Основы алгоритмизации и программирования	8
Горбач Елизавета Юрьевна	2013-01-10	Основы алгоритмизации и программирования	7
Зыкова Кристина Дмитриевна	2013-01-10	Основы алгоритмизации и программирования	5

```
insert [#Результаты экзамена] ([оценка], [дисциплина], [дата экзамена], [Ф.И.О. студента])  
select pr.NOTE, sb.SUBJECT_NAME, pr.PDATE, st.NAME  
from STUDENT st join PROGRESS pr on st.IDSTUDENT = pr.IDSTUDENT  
join [SUBJECT] sb on pr.[SUBJECT] = sb.[SUBJECT];
```

INSERT

```
use BSTU
go
create table [#Журнал]
(
    [Идентификатор] bigint identity (1,1),
    [Операция] char(3),
    [Дата и время] datetime2,
    [Ключ] varchar(100)
)
go
insert into TEACHER (TEACHER, TEACHER_NAME, PULPIT)
    output 'INSERT', SYSDATETIME(), inserted.TEACHER into [#Журнал]
values ('АКНВЧ', 'Акунович Станислав Иванович', 'ИСИТ'),
       ('КЛСНВ', 'Колесников Виталий Леонидович', 'ИСИТ'),
       ('ППРМС', 'Приходько Ольга Романовна', 'ИСИТ')
```

INSERT

```
use BSTU  
go  
select * from [#Журнал];
```

Идентификатор	Операция	Дата и время	Ключ
1	INSERT	2014-01-12 01:29:30.9693593	АКНВЧ
2	INSERT	2014-01-12 01:29:30.9693593	КЛСНВ
3	INSERT	2014-01-12 01:29:30.9693593	ГРМН

DELETE

```
select count(*) [количество строк в таблице PROGRESS] from PROGRESS;  
delete from PROGRESS; -- удаление всех строк  
select count(*) [количество строк в таблице PROGRESS] from PROGRESS;
```

количество строк в таблице PROGRESS

34

количество строк в таблице PROGRESS

0

DELETE

select

```
(select count(*) from PROGRESS) [кол. строк в PROGRESS],  
(select count(*)  
from PROGRESS p join STUDENT s on p.IDSTUDENT = s.IDSTUDENT  
                join GROUPS g on s.IDGROUP      = g.IDGROUP  
                and g.FACULTY = 'ИДиП') [кол. строк соединилось]
```

Кол. строк в PROGRESS кол. строк соединилось

34	23
----	----

```
select count(*) [количество строк в таблице PROGRESS] from PROGRESS;  
delete PROGRESS  
from PROGRESS p join STUDENT s on p.IDSTUDENT = s.IDSTUDENT  
                join GROUPS g on s.IDGROUP      = g.IDGROUP  
                and FACULTY = 'ИДиП';  
select count(*) [количество строк в таблице PROGRESS] from PROGRESS;
```

количество строк в таблице PROGRESS

34

количество строк в таблице PROGRESS

11

DELETE

```
select top(0)
    '1234567890' [оператор],
    sysdatetime() [дата и время],
    * into #LOG_PROGRESS from PROGRESS; -- создание временной таблицы

delete PROGRESS
output 'DELETE' [оператор], sysdatetime() [дата удаления], deleted.* into #LOG_PROGRESS
from PROGRESS p join STUDENT s on p.IDSTUDENT = s.IDSTUDENT
                    join GROUPS g on s.IDGROUP      = g.IDGROUP
where g.FACULTY = 'ИЭФ';
select * from #LOG_PROGRESS;
```

оператор	дата и время	SUBJECT	IDSTUDENT	PDATE	NOTE
DELETE	2014-02-16 23:33:46.6569140	ЭТ	1155	2013-01-15	7
DELETE	2014-02-16 23:33:46.6569140	ЭТ	1156	2013-01-15	8
DELETE	2014-02-16 23:33:46.6569140	ЭТ	1157	2013-01-15	9
DELETE	2014-02-16 23:33:46.6569140	ЭТ	1158	2013-01-15	4
DELETE	2014-02-16 23:33:46.6569140	ЭТ	1159	2013-01-15	5

TRUNCATE

```
select count(*) [кол. строк в TEACHER] from TEACHER;  
truncate table TEACHER;  
select count(*) [кол. строк в TEACHER] from TEACHER;
```

кол. строк в TEACHER

39

кол. строк в TEACHER

0

Запись в журнал транзакций:

- DELETE
- TRANCATE

UPDATE

```
select top(5) * from AUDITORIUM;  
update AUDITORIUM set AUDITORIUM_CAPACITY = 100;  
select top(5) * from AUDITORIUM;
```

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
026-4	ЛК	90	026-4
103-4	ЛК	90	103-4
105-4	ЛК	90	105-4
107-4	ЛК	90	107-4
110-4	ЛК	30	110-4

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
026-4	ЛК	100	026-4
103-4	ЛК	100	103-4
105-4	ЛК	100	105-4
107-4	ЛК	100	107-4
110-4	ЛК	100	110-4

UPDATE

Операция	Назначение
=	значение
= default	значение по умолчанию
+=	сумма (конкатенация для символьных данных)
-=	разница
*=	произведение
/=	частное
%=	остаток от деления
&=	результат побитового логического AND
^=	результат побитового логического XOR
=	результат побитового логического OR

UPDATE

```
select top(3) * from AUDITORIUM;  
update AUDITORIUM set AUDITORIUM_CAPACITY *= 1.2,  
                      AUDITORIUM_NAME += ' +20%'  
select top(3) * from AUDITORIUM;
```

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
02Б-4	ЛК	90	02Б-4
103-4	ЛК	90	103-4
105-4	ЛК	90	105-4

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
02Б-4	ЛК	108	02Б-4 +20%
103-4	ЛК	108	103-4 +20%
105-4	ЛК	108	105-4 +20%

UPDATE

```
select * from TEACHER  
update TEACHER set TEACHER_NAME += (' '+GENDER),  
PULPIT = default, GENDER = default;  
select * from TEACHER;
```

TEACHER	TEACHER_NAME	GENDER	PULPIT
ГРН	Гурин Николай Иванович	м	ИСиТ
ДЛН	Долинина Татьяна Николаевна	ж	СБУАиА
ДМДК	Демидко Марина Николаевна	м	ЛПиСПС
ДТК	Дятко Александр Аркадьевич	м	ИСиТ
ЕЩНК	Ещенко Людмила Семеновна	ж	ТНВиО...
ЖЛК	Жилляк Надежда Александровна	ж	ИСиТ

TEACHER	TEACHER_NAME	GENDER	PULPIT
ГРН	Гурин Николай Иванович м	м	NULL
ДЛН	Долинина Татьяна Николаевна ж	м	NULL
ДМДК	Демидко Марина Николаевна м	м	NULL
ДТК	Дятко Александр Аркадьевич м	м	NULL
ЕЩНК	Ещенко Людмила Семеновна ж	м	NULL
ЖЛК	Жилляк Надежда Александровна ж	м	NULL

UPDATE

```
select distinct pr.IDSTUDENT, pr.[SUBJECT], pr.FDATE
into [#Механики_с_оценкой_<_10]
from STUDENT s join GROUPS g      on s.IDGROUP = g.IDGROUP
                  join PROGRESS pr  on pr.IDSTUDENT = s.IDSTUDENT
                  join PROFESSION p  on p.PROFESSION = g.PROFESSION
where p.QUALIFICATION like '%механик%' and pr.NOTE < 10;

select t.IDSTUDENT, t.[SUBJECT], t.FDATE, pr.NOTE
from PROGRESS pr join [#Механики_с_оценкой_<_10] t
                  on pr.IDSTUDENT = t.IDSTUDENT and
                     pr.[SUBJECT] = t.[SUBJECT] and
                     pr.FDATE = t.FDATE
```

IDSTUDENT	SUBJECT	FDATE	NOTE
1064	OX	2013-01-19	6
1065	OX	2013-01-19	4
1066	OX	2013-01-19	9
1067	OX	2013-01-19	5
1068	OX	2013-01-19	8
1069	OX	2013-01-19	4

UPDATE

```
update PROGRESS set NOTE+=1
from PROGRESS pr join [#Механики_с_оценкой_<_10] t
    on pr.IDSTUDENT = t.IDSTUDENT and
    pr.[SUBJECT] = t.[SUBJECT] and
    pr.PDATE = t.PDATE;

select t.IDSTUDENT, t.[SUBJECT], t.PDATE, pr.NOTE
from PROGRESS pr join [#Механики_с_оценкой_<_10] t
    on pr.IDSTUDENT = t.IDSTUDENT and
    pr.[SUBJECT] = t.[SUBJECT] and
    pr.PDATE = t.PDATE;
```

IDSTUDENT	SUBJECT	PDATE	NOTE
1064	ОХ	2013-01-19	7
1065	ОХ	2013-01-19	5
1066	ОХ	2013-01-19	10
1067	ОХ	2013-01-19	6
1068	ОХ	2013-01-19	9
1069	ОХ	2013-01-19	5

UPDATE

```
select top(0) 'XXXXXXX' [операция],  
       AUDITORIUM [аудитория] ,  
       AUDITORIUM_CAPACITY [вместимость до],  
       AUDITORIUM_CAPACITY [вместимость после]  
into [#Регистрация изменений] -- создать таблицу  
from AUDITORIUM;  
  
update AUDITORIUM set AUDITORIUM_CAPACITY*=1.2  
output 'UPDATE' [операция], inserted.AUDITORIUM [аудитория],  
           deleted.AUDITORIUM_CAPACITY [до],  
           inserted.AUDITORIUM_CAPACITY [после] into [#Регистрация изменений]  
where AUDITORIUM_TYPE = 'ЛБ-К';  
select * from [#Регистрация изменений];
```

операция	аудитория	вместимость до	вместимость после
UPDATE	206-1	15	18
UPDATE	301-1	15	18
UPDATE	304-4	90	108
UPDATE	413-1	15	18
UPDATE	423-1	90	108

Вопросы?

БАЗЫ ДАННЫХ

Лекция 8 Представления

Представления

- Поименованный SELECT-запрос

Представления

```
drop view vAUDITORIM
```

```
create view vAUDITORIM
```

```
as select AUDITORIUM AU, AUDITORIUM TYPE AT  
from AUDITORIUM
```

```
select * from vAUDITORIM
```

AU	AT
026-4	ЛК
103-4	ЛК
105-4	ЛК
107-4	ЛК
110-4	ЛК
111-4	ЛК
114-4	ЛК-К
132-4	ЛК
229-4	ЛК
236-1	ЛК
301-1	ЛБ-К
304-4	ЛБ-К
313-1	ЛК
314-4	ЛК
320-4	ЛК
324-1	ЛК



Представления

```
drop view vPROFESSION_FACULTY
```

```
create view vPROFESSION_FACULTY
```

```
as select f.FACULTY_NAME FN, p.PROFESSION_NAME PN, p.QUALIFICATION QU  
from FACULTY f join PROFESSION p on f.FACULTY = p.FACULTY
```

```
select * from vPROFESSION_FACULTY order by FN
```

FN	PN	QU
Издательское дело и полиграфия	Информационные системы и технологии	инженер-программист-системотехник
Издательское дело и полиграфия	Издательское дело	редактор-технолог
Инженерно-экономический факультет	Экономика и управление на предприятиях	экономист-менеджер
Инженерно-экономический факультет	Бухгалтерский учет, анализ и аудит	экономист
Лесохозяйственный факультет	Лесное хозяйство	инженер лесного хозяйства
Лесохозяйственный факультет	Садово-парковое строительство	инженер садово-паркового строительства
Лесохозяйственный факультет	Туризм и природопользование	специалист в сфере туризма
Технология и техника лесной промышленности	Машины и оборудование лесного комплекса	инженер-механик
Технология и техника лесной промышленности	Лесоинженерное дело	инженер-технолог
Технология органических веществ	Химическая технология органических веществ, мате...	инженер-химик-технолог
Технология органических веществ	Химическая технология переработки древесины	инженер-химик-технолог
Технология органических веществ	Физико-химические методы и приборы контроля кач...	инженер по сертификации
Химическая технология и техника	Конструирование и производство изделий из композ...	инженер-механик
Химическая технология и техника	Машины и аппараты химических производств и предп...	инженер-механик

Представления

- Упрощение
- Механизм безопасности
- Обеспечение обратной совместимости

Представления

- Допустимо выполнение DML-операций INSERT, DELETE и UPDATE?

Представления INSERT

```
insert into vAUDITORIM(AU, AT) values('137-4', 'ЛК');
insert into vAUDITORIM(AU, AT) values('310a-1', 'ЛБ-К');
select * from vAUDITORIM where AU = '137-4' or AU = '310a-1';
```

AU	AT
137-4	ЛК
310a-1	ЛБ-К

Представления

```
select * from AUDITORIUM where AUDITORIUM = '137-4' or AUDITORIUM = '310a-1'
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
137-4	NULL	NULL	ЛК
310a-1	NULL	NULL	ЛБ-К

Представления

- Любая операция выполняется над базовой таблицей
- Все существующие **ограничения целостности** базовой таблицы наследуются представлением

Представления INSERT

```
create view vgAUDITORIUM  
as select AUDITORIUM_TYPE AT, COUNT(*) ATC from AUDITORIUM group by AUDITORIUM_TYPE  
  
select * from vgAUDITORIUM
```

AT	ATC
ЛБ-К	5
ЛК	17
ЛК-К	1

```
insert into vgAUDITORIUM(AT, ATC)values('hhh',99);
```

Сообщение 4406, уровень 16, состоянне 1, строка 1

Ошибка обновления, или вставки представления или функции "vgAUDITORIUM" из-за отсутствия производного или постоянно

Представления

```
drop view vAUDITORIUM50  
create view vAUDITORIUM50  
    as select AUDITORIUM AU, AUDITORIUM_TYPE AT, AUDITORIUM_CAPACITY AC  
        from AUDITORIUM where AUDITORIUM_CAPACITY > 50  
select * from vAUDITORIUM50
```

AU	AT	AC
?	ЛК	90
02Б-4	ЛК	90
103-4	ЛК	90
105-4	ЛК	90
107-4	ЛК	90
114-4	ЛК-К	90
132-4	ЛК	90
229-4	ЛК	90
236-1	ЛК	60
304-4	ЛБ-К	90
313-1	ЛК	60
314-4	ЛК	90
320-4	ЛК	90
408-2	ЛК	90
423-1	ЛБ-К	90
429-4	ЛК	90

Представления

```
insert into vAUDITORIUM50(AU, AT, AC) values ('131-4', 'ЛБ-К', 20);
select * from vAUDITORIUM50
```

AU	AT	AC
?	ЛК	90
02Б-4	ЛК	90
103-4	ЛК	90
105-4	ЛК	90
107-4	ЛК	90
114-4	ЛК-К	90
132-4	ЛК	90
229-4	ЛК	90
236-1	ЛК	60
304-4	ЛБ-К	90
313-1	ЛК	60
314-4	ЛК	90
320-4	ЛК	90
408-2	ЛК	90
423-1	ЛБ-К	90
429-4	ЛК	90

Представления

```
select COUNT(*) from VAUDITORIUM50 where AU = '131-4'
```

(Отсутствует имя столбца)

0

```
update VAUDITORIUM50 set AC = 75 where AU = '131-4'
```

(строк обработано: 0)

```
delete VAUDITORIUM50 where AU = '131-4'
```

(строк обработано: 0)

```
insert into VAUDITORIUM50(AU, AT, AC)values ('131-4', 'ЛБ-К', 30);
```

Нарушение "PK_AUDITORI_537260101CF15040" ограничения PRIMARY KEY.

Невозможно вставить повторяющийся ключ в объект
"dbo.AUDITORIUM".

Представления

```
select * from AUDITORIUM where AUDITORIUM = '131-4'
```

AUDITORIUM	AUDITORIUM_NAME	AUDITORIUM_CAPACITY	AUDITORIUM_TYPE
131-4	NULL	20	ЛБ-К

Представления UPDATE

```
update AUDITORIUM set AUDITORIUM_CAPACITY = 75 where AUDITORIUM = '131-4'  
select * from vAUDITORIUM50
```

AU	AT	AC
026-4	ЛК	90
103-4	ЛК	90
105-4	ЛК	90
107-4	ЛК	90
114-4	ЛК-К	90
131-4	ЛБ-К	75
132-4	ЛК	90
229-4	ЛК	90
236-1	ЛК	60
304-4	ЛБ-К	90
313-1	ЛК	60
314-4	ЛК	90
320-4	ЛК	90
408-2	ЛК	90
423-1	ЛБ-К	90
429-4	ЛК	90

Представления

```
delete vAUDITORIUM50 where AU = '131-4'  
select (  
    case  
        when exists (select * from vAUDITORIUM50 where AU = '131-4') then 'есть 131-4'  
        else 'нет 131-4'  
    end  
) vAUDITORIUM50,  
(  
    case  
        when exists (select * from AUDITORIUM where AUDITORIUM = '131-4') then 'есть 131-4'  
        else 'нет 131-4'  
    end  
) AUDITORIUM
```

vAUDITORIUM50	AUDITORIUM
нет 131-4	нет 131-4

Представления

```
drop view vAUDITORIUM50
create view vAUDITORIUM50
    as select AUDITORIUM AU, AUDITORIUM_TYPE AT, AUDITORIUM_CAPACITY AC
        from AUDITORIUM where AUDITORIUM_CAPACITY > 50
with check option

select * from vAUDITORIUM50
```

AU	AT	AC
?	ЛК	90
026-4	ЛК	90
103-4	ЛК	90
105-4	ЛК	90
107-4	ЛК	90
114-4	Л...	90
132-4	ЛК	90
229-4	ЛК	90
236-1	ЛК	60
304-4	Л...	90
313-1	ЛК	60
314-4	ЛК	90
320-4	ЛК	90
408-2	ЛК	90
423-1	Л...	90
429-4	ЛК	90

Представления

```
insert into vAUDITORIUM50(AU, AT, AC)values ('131-4','ЛБ-К', 20);
```

Ошибка при попытке вставки или обновления, поскольку целевое представление либо указывает WITH CHECK OPTION, .

```
insert into vAUDITORIUM50(AU, AT, AC)values ('131-4','ЛБ-К', 75);
select (
    case
        when exists (select * from vAUDITORIUM50 where AU = '131-4') then 'есть 131-4'
        else 'нет 131-4'
    end
) vAUDITORIUM50,
(
    case
        when exists (select * from AUDITORIUM where AUDITORIUM = '131-4') then 'есть 131-4'
        else 'нет 131-4'
    end
) AUDITORIUM
```

vAUDITORIUM50	AUDITORIUM
есть 131-4	есть 131-4

Представления

```
update AUDITORIUM set AUDITORIUM_CAPACITY = 20 where AUDITORIUM = '131-4'  
select (-----  
    case  
        when exists (select * from VAUDITORIUM50 where AU = '131-4') then 'есть 131-4'  
        else 'нет 131-4'  
    end  
) VAUDITORIUM50,  
(-----  
    case  
        when exists (select * from AUDITORIUM where AUDITORIUM = '131-4') then 'есть 131-4'  
        else 'нет 131-4'  
    end  
) AUDITORIUM
```

VAUDITORIUM50	AUDITORIUM
нет 131-4	есть 131-4

Представления ограничения

- В операторе CREATE после ключевого слова **VIEW** следует **имя представления** и далее после ключевого слова **AS** текст SELECT-запроса, лежащего в основе представления.
- При этом к SELECT-запросу представления предъявляется следующие **требования**:
- секцию ORDER BY можно использовать только совместно с опцией TOP;
- не допускается применение секции INTO;
- все столбцы результирующего набора, формируемого SELECT-запросом, должны быть **поименованы**.

SCHEMABINDING

- Запрещение на операции с таблицами и представлениями, используемыми в SELECT-запросе, на котором основано представление
- Требуется указывать схему БД, к которой принадлежит таблица или представление
- Схема – это поименованный контейнер объектов БД, позволяющий разграничить объекты с одинаковыми именами

Представления ограничения

- При создании представлений, позволяющих выполнять операции INSERT, DELETE и UPDATE, базовый SELECT-запрос должен удовлетворять следующим **правилам**.
- Запрос не должен содержать секцию группировки GROUP BY и агрегатные функции.
- Запрос не должен использовать опции DISTINCT и TOP.
- Запрос не должен использовать операторы UNION, INTERSECT и EXCEPT.
- В SELECT-списке запроса не должно быть вычисляемых значений.
- В секции FROM запроса должна указываться только **одна** таблица.

Представления каталога

- sys.objects
- sys.columns
- sys.database_principals

```
USE sample;
SELECT object_id, principal_id, type
FROM sys.objects
WHERE name = 'employee';
```

Вопросы?

БАЗЫ ДАННЫХ

Лекция 9 Основы T-SQL

Пакет

- Пакет - это группа операторов T-SQL, которая обрабатывается сервером СУБД вместе

```
---- пакет 1 -----
use TEMPDB
go
---- пакет 2 -----
create table #A ([Номер] int identity (1,1));
go
---- пакет 3 -----
insert into #A default values;
go 3           -- повторить 3 раза
---- пакет 4 -----
select * from #A;
go
---- пакет 5 -----
drop table #A;
```

Номер
1
2
3

DECLARE

- DECLARE - объявление используемых переменных
- Для каждой переменной указывается имя и тип

DECLARE

```
declare @x int = 1,
        @y decimal(7,2), -- не инициализирована
        @z float(4) = 2.1E-3;
declare @a char(10),      -- не инициализирована
        @b varchar(50) = 'Hello World!',
        @c nvarchar(50) = N'Привет мир!';
declare @d date = '20140107',
        @e datetime = getdate(),
        @f time;           -- не инициализирована
declare @g uniqueidentifier = newid();
declare @h table(
    num int identity(1,1),
    fil varchar(30) default 'XXX'
);
insert @h default values; -- добавление одной строки
select @x x, @y y, @z z;
select @a a, @b b, @c c;
select @d d, @e e, @f f;
select @g g;
select * from @h;
```

x	y	z
1	NULL	0.0021

a	b	c
NULL	Hello World!	Привет мир!

d	e	f	t
2014-01-07	2014-03-04 02:39:34.530	NULL	

g
A341C7D3-B1B3-447F-B52C-E3EC6C2C9783

num	fil
1	XXX

Область видимости переменной

- Областью видимости переменной являются все инструкции между ее объявлением и концом пакета или хранимой процедурой, где она объявлена.

Операторы присваивания

- инициализировать в DECLARE
- присвоить значение SET
- присвоить значение SELECT

Операторы присваивания

```
declare @y decimal(7,2) = 0.0,  
        @a char(10) ;  
set @y = 2.55;  
set @a = 'Hello';  
select @y y, @a a;  
go
```

y	a
2.55	Hello

```
declare @y decimal(7,2) = 0.0,  
        @a char(10) ;  
select @y = 2.55, @a = 'Hello';  
select @y y, @a a;  
go
```

y	a
2.55	Hello

Оператор PRINT

```
declare  
    @StudentsCount    int    = (select count(*) from STUDENT),  
    @AuditoriumCount int    = (select count(*) from AUDITORIUM),  
    @TeacherCount     int    = (select count(*) from TEACHER);  
  
select AUDITORIUM_TYPE [Тип аудитории],  
       count(*)          [Количество аудиторий]  
  from AUDITORIUM group by AUDITORIUM_TYPE;  
select PULPIT           [Кафедра],  
       count(*)          [Количество дисциплин]  
  from SUBJECT group by PULPIT;  
print 'Количество студентов : ' + cast (@StudentsCount as varchar(10));  
print 'Количество аудиторий : ' + cast (@AuditoriumCount as varchar(10));  
print 'Количество преподавателей : ' + cast (@TeacherCount as varchar(10));  
go
```

Тип аудитории	Количество аудиторий
ХХХХ	1
ЛБ-К	5
ПК	13

```
Количество студентов : 178  
Количество аудиторий : 24  
Количество преподавателей : 24
```

Кафедра	Количество дисциплин
ИСиТ	14
ЛЗиДВ	1
ПМиПЗ	1

CAST и CONVERT

- `CAST(<выражение> AS <тип данных> [(длина)])`
- `CAST для NULL = NULL`
- `CONVERT (<тип данных> [(длина)] ,<выражение> [, style])`

CAST

Исходный тип	Целевой тип	Поведение
numeric	numeric	Округление
numeric	int	Усечение
numeric	money	Округление
money	int	Округление
money	numeric	Округление
float	int	Усечение
float	numeric	Округление
float	datetime	Округление
datetime	int	Округление

CAST

K:	binary	varbinary	char	varchar	nchar	nvarchar	datetime	smalldatetime	decimal	numeric	float	real	bignat	int(INT4)	smallint(INT2)	tinyint(INT1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	ntext	text	sql_variant	xml
От:	binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
binary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
varbinary	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
char	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
varchar	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
nchar	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
nvarchar	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
datetime	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
smalldatetime	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
decimal	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
numeric	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
float	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
real	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
bignat	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
int(INT4)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
smallint(INT2)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
tinyint(INT1)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
money	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
smallmoney	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
bit	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
timestamp	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
uniqueidentifier	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
image	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
ntext	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
text	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
sql_variant	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
xml	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	

- Явное преобразование
- Неявное преобразование
- Преобразование недопустимо
- * Требуется явное CAST, чтобы предотвратить потерю при неявном преобразовании
- Неявные преобразования между типами XML-данных

CAST

```
-- numeric(p,s)      1<=p<38,  0<=s<=p
```

```
| select 14/24 '-i-',
|         CAST (14/24 as numeric(5,3)) '-ii-',
|         CAST (14      as numeric(5,3))/CAST (24 as numeric(5,3)) '-iii-'
```

-i-	-ii-	-iii-
0	0.000	0.583333333

CAST

```
declare @y1 numeric (8,3) = (select CAST(SUM(AUDITORIUM CAPACITY) as numeric(8,3)) from AUDITORIUM)
       @y2 numeric (8,3), @y3 numeric (8,3), @y4 numeric (8,3)
if @y1 > 200
begin
    print 'общее количество мест больше 200'
    select @y2 = CAST((select COUNT(*)
                        from AUDITORIUM)as numeric(8,3)),
           @y3 = (select CAST(AVG(AUDITORIUM_CAPACITY) as numeric(8,3))
                   from AUDITORIUM),
           @y4 = (select CAST(COUNT(*) as numeric(8,3))
                   from AUDITORIUM where AUDITORIUM_CAPACITY > @y3)

    select @y1 '@y1 ', @y2 '@y2 ', @y3 '@y3 ', @y4 '@y4 ', @y4/@y2*100 '%'
end
else if @y1 > 100
    print 'общее количество мест от 100 до 200'
else if @y1 > 50
    print 'общее количество мест от 50 до 100'
else
    print 'общее количество мест меньше 50'
```

@y1	@y2	@y3	@y4	%
1564.000	24.000	71.000	0.000	0.00000000000000

CAST

```
declare @y1 numeric (8,3) = (select CAST(SUM(AUDITORIUM_CAPACITY) as numeric(8,3)) from AUDITORIUM,
@y2 numeric (8,3), @y3 numeric (8,3), @y4 numeric (8,3)
if @y1 > 200
begin
print 'общее количество мест больше 200'
select @y2 = CAST((select COUNT(*)
                    from AUDITORIUM)as numeric(8,3)),
@y3 = (select CAST(AVG(AUDITORIUM_CAPACITY) as numeric(8,3))
        from AUDITORIUM)

set @y4 = (select CAST(COUNT(*) as numeric(8,3))
            from AUDITORIUM where AUDITORIUM_CAPACITY > @y3)

select @y1 '@y1 ', @y2 '@y2 ', @y3 '@y3 ', @y4 '@y4 ', @y4/@y2*100 '%'
end
else if @y1 > 100
print 'общее количество мест от 100 до 200'
else if @y1 > 50
print 'общее количество мест от 50 до 100'
else
print 'общее количество мест меньше 50'
```

@y1	@y2	@y3	@y4	%
1564.000	24.000	71.000	14.000	58.333333333300

Арифметические операции

```
declare @x int = 0, @x1 int = 1,  
        @y int = 1, @y1 int = 2,  
        @z numeric(7,2) = 2.0, @z1 numeric(5,1) = 33.0,  
        @w float(8) = 3.0, @w1 float(4) = 5.1E3;
```

```
set @x+=1;   print '@x-=1 = ' + cast(@y as varchar(10));  
set @y-=1;   print '@y-=1 = ' + cast(@y as varchar(10));  
set @z*=3.0; print '@z*=3.0 = ' + cast(@z as varchar(10));  
set @w/=2.0; print '@w/=2.0 = ' + cast(@w as varchar(10));  
set @z%=4;   print '@z/=4    = ' + cast(@z as varchar(10));  
----- @x-=1 = 1  
----- @y-=1 = 0  
----- @z*=3.0 = 6.00  
----- @w/=2.0 = 1.5  
----- @z/=4    = 2.00
```

Арифметические операции

```
set @z=1233.2 + 15.01;      print '@z=1233.2 + 15.01 = '+ cast(@z as varchar(10));
set @w=12.33E3 - 15.01E2;    print '@w=12.33E3 - 15.01E2 = '+ cast(@w as varchar(10));
set @x=1233 * 15;           print '@x=1233 * 15 = '+ cast(@x as varchar(10));
set @y=1233.33 / 15;         print '@y=1233.33 / 15 = '+ cast(@y as varchar(10));
set @x=-100%11;              print '@x=100%10 = '+ cast(@x as varchar(10));
```

@z=1233.2 + 15.01 = 1248.21

@w=12.33E3 - 15.01E2 = -137770

@x=1233 * 15 = 18495

@y=1233.33 / 15 = 82

@x=100%10 = -1

Арифметические операции

```
select @x1 = @x+@y, @y1 = @x-@y;
print '@x1 = @x+@y = '+ cast(@x1 as varchar(10))+'; @y1 = @x-@y = '+cast(@y1 as varchar(10));
select @z1 = @x*@z1, @w1 = @w1/@y;
print '@z1 = @x*@z1 = '+ cast(@z1 as varchar(10))+'; @w1 = @w1/@y = '+cast(@w1 as varchar(10));
select @w1 = @z%@x;   print '@w1 = @z%@x = '+ cast(@w1 as varchar(10));
```

```
@x1 = @x+@y = 81; @y1 = @x-@y = -83
```

```
@z1 = @x*@z1 = -33.0; @w1 = @w1/@y = 62.1951
```

```
@w1 = @z%@x = 0.21
```

Встроенные функции

```
print 'Округление' : '+ cast(round(12345.12345, 2) as varchar(12));  
print 'Нижнее целое' : '+ cast(floor(24.5) as varchar(12));  
print 'Верхнее целое' : '+ cast(ceiling(24.5) as varchar(12));  
print 'Возведение в степень' : '+ cast(power(12.0, 2) as varchar(12));  
print 'Логарифм по exp' : '+ cast(log(144.0) as varchar(12));  
print 'Корень квадратный' : '+ cast(sqrt(144.0) as varchar(12));  
print 'Экспонента' : '+ cast(exp(4.96981) as varchar(12));  
print 'Абсолютное значение' : '+ cast(abs(-5) as varchar(12));  
print 'Радианы' : '+ cast(radians(180.0) as varchar(20));  
print 'Число пи' : '+ cast(pi() as varchar(12));  
print 'Градусы' : '+ cast(degrees(pi()) as varchar(20));  
print 'Синус' : '+ cast(sin(pi()) as varchar(12));  
print 'Косинус' : '+ cast(cos(pi()) as varchar(12));
```

Встроенные функции

Округление	: 12345.12000
Нижнее целое	: 24
Верхнее целое	: 25
Возведение в степень	: 144.0
Логарифм по exp	: 4.96981
Корень квадратный	: 12
Экспонента	: 144
Абсолютное значение	: 5
Радианы	: 3.141592653589793100
Число пи	: 3.14159
Градусы	: 180
Синус	: 1.22465e-016

Конкатенация

```
declare @y1 numeric (8,3) = (select CAST(SUM(AUDITORIUM_CAPACITY) as numeric(8,3)) from AUDITORIUM),
        @y2 numeric (8,3), @y3 numeric (8,3), @y4 numeric (8,3)
if @y1 > 200
begin
    print 'общее количество мест больше 200'
    select @y2 = CAST((select COUNT(*)
                        from AUDITORIUM)as numeric(8,3)),
           @y3 = (select CAST(AVG(AUDITORIUM_CAPACITY) as numeric(8,3))
                   from AUDITORIUM)

    set @y4 = (select CAST(COUNT(*) as numeric(8,3))
               from AUDITORIUM where AUDITORIUM_CAPACITY > @y3)

    print ' @y1 = ' + CAST(@y1 as varchar(20))+
          ' @y2 = ' + CAST(@y2 as varchar(20))+
          ' @y3 = ' + CAST(@y3 as varchar(20))+
          ' @y4 = ' + CAST(@y4 as varchar(20))+
          ' % = ' + CAST(@y4/@y2*100 as varchar(20))
end
else if @y1 > 100
    print 'общее количество мест от 100 до 200'
else if @y1 > 50
    print 'общее количество мест от 50 до 100'
else
    print 'общее количество мест меньше 50'

@y1 = 1564.000 @y2 = 24.000 @y3 = 71.000 @y4 = 14.000   % = 58.33333333300
```

Строковые функции

```
print 'Подстрока          : '+ substring('1234567890', 3,2);
print 'Удалить пробелы слева : '+ ltrim('      67890');
print 'Удалить пробелы справа : '+ rtrim('12345      ') +'X';
print 'Нижний регистр       : '+ lower ('ВЕРХНИЙ РЕГИСТР');
print 'Верхний регистр       : '+ upper ('нижний регистр');
print 'Заменить              : '+ replace('1234512345', '5', 'X');
print 'Строка пробелов      : '+ 'X'+ space(5) +'X';
print 'Повторить строку      : '+ replicate('12', 5);
print 'Найти по шаблону      : '+ cast (patindex ('%Y_Y%', '123456YxY7890') as varchar(5));
```

Подстрока	: 34
Удалить пробелы слева	: X67890
Удалить пробелы справа	: 12345X
Нижний регистр	: верхний регистр
Верхний регистр	: НИЖНИЙ РЕГИСТР
Заменить	: 1234X1234X
Строка пробелов	: X X
Повторить строку	: 1212121212
Найти по шаблону	: 7

ФУНКЦИИ ДАТЫ И ВРЕМЕНИ

```
declare @dt datetime = getdate() , -- текущая дата
        @ed datetime = dateadd(d, 200, getdate()); -- текущая дата +200 дней
print 'Текущая дата           : '+ convert(varchar(12), @dt, 103 );
print 'Текущая дата+200 дней : '+ convert(varchar(12), @ed, 103 );
print 'День                  : '+ convert(varchar(12), day(@dt));
print 'День недели (1-вс)    : '+ convert(varchar(12), datepart(dw, @dt));
print 'Неделя                : '+ convert(varchar(12), datepart(wk, @dt));
print 'Месяц                 : '+ convert(varchar(12), month(@dt));
print 'Квартал               : '+ convert(varchar(12), datepart(q, @dt));
print 'Год                   : '+ convert(varchar(12), year(@dt));
print '+1 день                : '+ convert(varchar(12), dateadd(d, 1, @dt), 103);
print '+1 неделя              : '+ convert(varchar(12), dateadd(wk, 1, @dt), 103);
print '+1 месяц               : '+ convert(varchar(12), dateadd(m, 1, @dt), 103);
print '+1 квартал             : '+ convert(varchar(12), dateadd(q, 1, @dt), 103);
print '+1 год                 : '+ convert(varchar(12), dateadd(y, 1, @dt), 103);
print 'Разница в днях        : '+ convert(varchar(12), datediff(d, @dt,@ed));
print 'Разница в неделях      : '+ convert(varchar(12), datediff(wk, @dt,@ed));
print 'Разница в месяцах      : '+ convert(varchar(12), datediff(m, @dt,@ed));
print 'Разница в годах        : '+ convert(varchar(12), datediff(yy, @dt,@ed));
```

ФУНКЦИИ ДАТЫ И ВРЕМЕНИ

- DATEADD (datepart, number, date)
 - datepart:
 - year - yy, yyyy; quarter - qq, q; month - mm, m;
 - dayofyear - dy, y; day - dd, d; week - wk, ww;
 - weekday - dw, w; hour – hh; minute - mi, n;
 - second - ss, s
-
- DATEPART (datepart, date)
- DATEDIFF (datepart , startdate , enddate)

ФУНКЦИИ ДАТЫ И ВРЕМЕНИ

Текущая дата	:	05/03/2014
Текущая дата+200 дней	:	21/09/2014
День	:	5
День недели	:	4
Неделя	:	10
Месяц	:	3
Квартал	:	1
Год	:	2014
+1 день	:	06/03/2014
+1 неделя	:	12/03/2014
+1 месяц	:	05/04/2014
+1 квартал	:	05/06/2014
+1 год	:	06/03/2014
Разница в днях	:	200
Разница в неделях	:	29
Разница в месяцах	:	6
Разница в годах	:	0

ФУНКЦИИ ДАТЫ И ВРЕМЕНИ

```
declare @t time(7) = sysdatetime(), -- текущее время
        @et time(7) = dateadd(mi, 12*60, getdate()),
        @dt datetime2 = sysdatetime(); -- текущая дата и время
print 'Текущее время          : '+ convert(varchar(12), @t);
print 'Текущие дата и время   : '+ convert(varchar(20), @dt, 120);
print 'Текущие время + 12час : '+ convert(varchar(20), @et);
print 'Часы                  : '+ convert(varchar(20), datepart(hh, @t));
print 'Минуты                : '+ convert(varchar(20), datepart(mi, @t));
print 'Секунды               : '+ convert(varchar(20), datepart(ss, @t));
print 'Милисекунды          : '+ convert(varchar(20), datepart(ms, @t));
print 'Текущее время + 1час  : '+ convert(varchar(20), dateadd(hh, 1, @t));
print 'Текущее время + 1мин  : '+ convert(varchar(20), dateadd(mi, 1, @t));
print 'Текущее время + 1сек  : '+ convert(varchar(20), dateadd(ss, 1, @t));
print 'Разница в днях         : '+ convert(varchar(20), datediff(d, @t,@et));
print 'Разница в час           : '+ convert(varchar(20), datediff(hh, @t,@et));
print 'Разница в мс            : '+ convert(varchar(20), datediff(ms, @t,@et));
```

```
Текущее время          : 11:11:38.248
Текущие дата и время   : 2014-03-05 11:11:38
Текущие время + 12час : 23:11:38.2470000
Часы                  : 11
Минуты                : 11
Секунды               : 38
Милисекунды          : 248
Текущее время + 1час  : 12:11:38.2489297
Текущее время + 1мин  : 11:12:38.2489297
Текущее время + 1сек  : 11:11:39.2489297
Разница в днях         : 0
Разница в час           : 12
Разница в мс            : 43199999
```

Системные функции

```
select @@ROWCOUNT      'кол. обработанных записей',
       @@ERROR        'код ошибки',
       @@SERVERNAME    'имя сервера',
       @@IDENTITY      'последнее значение IDENTITY',
       @@SPID          'сист. идентификатор',
       @@VERSION       'версия SQL Server'

select @@TRANCOUNT     'уровень вложенности транзакции',
       @@FETCH_STATUS   'статус команды выборки',
       @@NESTLEVEL      'уровень вложенности'
go
```

кол. обработанных записей	код ошибки	имя сервера	последнее эн...	сист. иент...	версия SQL Server
1	0	SMW-IIS-SQL	NULL	52	Microsoft SQL Server 2008 R2 (RTM) - 10.50.1600...

уровень вложенности транзакции	статус команды выборки	уровень вложенности
0	0	0

Оператор RETURN

```
print 'Метка 1';
return;           -- выход из пакета
print 'Метка 2'; -- не выполняется
go
print 'Метка 3';
```

```

print 'пакет1: начало-----';
declare @x0 int = 0;
print 'пакет1:@x0 = '+ cast(@x0 as char(1));
begin
    declare @x1 int = 1;
    print 'блок1: начало -----';
    print 'блок1:@x0 = '+ cast(@x0 as char(1));
    print 'блок1:@x1 = '+ cast(@x1 as char(1));
    begin
        declare @x2 int = 2;
        print '    блок2: начало -----';
        print '    блок2:@x0 = '+ cast(@x0 as char(1));
        print '    блок2:@x1 = '+ cast(@x1 as char(1));
        print '    блок2:@x2 = '+ cast(@x2 as char(1));
        print '    блок2: конец -----';
    end;
    begin
        print '    блок3: начало -----';
        print '    блок3:@x0 = '+ cast(@x0 as char(1));
        print '    блок3:@x1 = '+ cast(@x1 as char(1));
        print '    блок3:@x2 = '+ cast(@x2 as char(1));
        return; -- выход из пакета
        print '    блок3: конец -----';
        print ''
    end;
    print 'блок1: конец -----';
end;
print 'пакет1: конец-----';
go
print 'пакет2: начало-----';
print 'пакет2: конец-----';

```

BEGIN END

```

пакет1: начало-----
пакет1:@x0 = 0
блок1: начало -----
блок1:@x0 = 0
блок1:@x1 = 1
    блок2: начало -----
    блок2:@x0 = 0
    блок2:@x1 = 1
    блок2:@x2 = 2
    блок2: конец -----
    блок3: начало -----
    блок3:@x0 = 0
    блок3:@x1 = 1
    блок3:@x2 = 2
пакет2: начало-----
пакет2: конец-----

```

BEGIN END

```
declare @x1 int = 0

set @x1 = (select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM)

if @x1 > 200
begin
    print 'общее количество мест больше 200'
    select CAST(COUNT(*) as numeric(5,3))/
           CAST((select COUNT(*) from AUDITORIUM) as numeric(5,3)) *
           100   '%'
    from AUDITORIUM
    where AUDITORIUM_CAPACITY > (select AVG(AUDITORIUM_CAPACITY) from AUDITORIUM);
end
else if @x1 > 100
    print 'общее количество мест от 100 до 200'
else if @x1 > 50
    print 'общее количество мест от 50 до 100'
else
    print 'общее количество мест меньше 50'
```

%

58.333333300

IF-ELSE

```
if exists(select * from PULPIT)
    print 'в таблице PULPIT есть строки'
else
    print 'в таблице PULPIT нет строк'
go
```

в таблице PULPIT есть строки

```
if (select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM) > 200
    print 'общее количество мест больше 200'
else
    print 'общее количество мест меньше 200'
go
```

общее количество мест больше 200

IF-ELSE

```
declare @x1 int = 0

set @x1 = (select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM)

if @x1 > 200
    print 'общее количество мест больше 200'
else if @x1 > 100
    print 'общее количество мест от 100 до 200'
else if @x1 > 50
    print 'общее количество мест от 50 до 100'
else
    print 'общее количество мест меньше 50'

go
```

CASE

```
declare @z1 numeric (8,3) = (select CAST(SUM(AUDITORIUM_CAPACITY) as numeric(8,3)) from AUDITO
print 'общее количество мест ' +
(
    case
        when @z1 > 200 then 'больше 200'
        when @z1 > 100 then 'от 100 до 200'
        else 'меньше 50'
    end
)
```

общее количество мест больше 200

CASE

```
select case (select COUNT(*) from TEACHER where TEACHER_NAME LIKE '%Владимир%')
    when 0 then 'нет'
    when 1 then 'один'
    when 2 then 'два'
    else 'тыма'
end 'Владимир'
```

Владимир

тыма

```
select * from TEACHER where TEACHER_NAME LIKE '%Владимир%'
```

TEACHER	TEACHER_NAME	PULPIT
БЗБРДВ	Безбородов Владимир Степанович	ОХ
МШКВСК	Машковский Владимир Петрович	ЛЧ
СМЛВ	Смелов Владимир Владиславович	ИСиТ

WHILE

```
select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM
```

(Отсутствует имя столбца)

1564

```
while (select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM) < 2000
begin
    update AUDITORIUM set AUDITORIUM_CAPACITY = AUDITORIUM_CAPACITY*1.05
end
```

```
select SUM(AUDITORIUM_CAPACITY) from AUDITORIUM
```

(Отсутствует имя столбца)

2015

WHILE

```
declare @i int = 0;
while @i <5           -- логическое выражение
    set @i+=1;         -- тело цикла
print '@i = '+ cast(@i as varchar(3));
```

```
@i = 5
```

```
declare @i int = 0;
while @i <5           -- логическое выражение
begin                  -- тело цикла
    set @i+=1;
    print '@i = '+ cast(@i as varchar(3));
end;
```

```
@i = 1
@i = 2
@i = 3
@i = 4
@i = 5
```

```
set nocount on; -- не выводить: row(s) affected
declare @ac int = (select sum(AUDITORIUM_CAPACITY)
                   from AUDITORIUM
                   where AUDITORIUM_TYPE in ('ЛК','ЛБ-К'));
print 'Общая вместимость аудиторий (ЛК,ЛБ-К) : ' + cast(@ac as varchar(6));
while (@ac < 1500)
begin
    update AUDITORIUM set AUDITORIUM_CAPACITY *= 1.1 where AUDITORIUM_TYPE = 'ЛК';
    update AUDITORIUM set AUDITORIUM_CAPACITY *= 1.05 where AUDITORIUM_TYPE = 'ЛБ-К';
    set @ac = (select sum(AUDITORIUM_CAPACITY)
               from AUDITORIUM
               where AUDITORIUM_TYPE in ('ЛК','ЛБ-К'));
    print space(38) + ':' + cast(@ac as varchar(6));
end;
print 'Общая вместимость аудиторий (ЛК,ЛБ-К) : ' + cast(@ac as varchar(6));
```

Общая вместимость аудиторий (ЛК,ЛБ-К) : 1245
: 1355
: 1465
: 1586

Общая вместимость аудиторий (ЛК,ЛБ-К) : 1586

WHILE

```
declare @i int = 0, @x int;
while (@i < 5)
begin
    set @x = @i;
    set @i+=1;
    print 'A' + cast(@x as varchar(2));
    if @x = 2 continue;      -- в начало
    if @x = 3 break;        -- выйти
    print 'B' + cast(@x as varchar(2));
end;
```

A0

B0

A1

B1

A2

A3

TRY/ CATCH

```
declare @x int = 3, @y int = 0, @z int;
begin try
    print 'begin try:A'
    set @z = @x/@y;          -- деление на 0
    print 'begin try:B'      -- если ошибка - не выполняется
end try
begin catch                  -- обработка ошибки
    print 'begin catch:C'
end catch
```

begin try:A
begin catch:C

TRY/ CATCH

```
begin try

update TEACHER set TEACHER = 'УРБ' where TEACHER = 'СМЛВ'

end try
begin catch
    print 'catch'
    print ERROR NUMBER()
    print ERROR MESSAGE()
    print ERROR LINE()
    print ERROR PROCEDURE()
    print ERROR SEVERITY()
    print ERROR STATE()
end catch

catch
2627
Нарушение "PK_TEACHER" ограничения PRIMARY KEY. Невозможно вставить повторяющийся ключ в объект "dbo.TEACHER".
3

14
1
```

RAISERROR

```
declare @x int = 3, @y int = 0, @z int;
begin try
    if @y = 0
        raiserror ('Деление на нуль', 10, 1);
    else set @z = @x/@y;          -- деление на 0
    if exists(select * from PULPIT where PULPIT = 'ИСиТ')
        raiserror ('Нарушение РК', 11, 2);
    else insert PULPIT(PULPIT) values('ИСиТ');
    print 'begin try:3'
end try
begin catch           -- обработка ошибки
    print 'номер ошибки : ' + cast(error_number()      as varchar(6));
    print 'сообщение   : ' + error_message();
    print 'уровень     : ' + cast(error_severity()    as varchar(6));
    print 'метка       : ' + cast(error_state()       as varchar(8));
    print 'номер строки : ' + cast(error_line()       as varchar(8));
    if error_procedure() is not null
        print 'имя процедуры : ' + error_procedure();
end catch
```

```
Деление на нуль
номер ошибки : 50000
сообщение   : Нарушение РК
уровень     : 11
метка       : 2
номер строки : 7
```

WAITFOR

```
declare @ct char(8)
set @ct=
    select CAST(DATEPART(hh,DATEADD(mi,1,SYSDATETIME())) as CHAR(2))
        +' : '+
        CAST(DATEPART(mi,DATEADD(mi,1,SYSDATETIME())) as CHAR(2))
)
select SYSDATETIME(), @ct
waitfor time @ct
select SYSDATETIME()
```

(Отсутствует имя столбца)	(Отсутствует имя столбца)
2011-03-11 02:16:19.2777872	2:17

(Отсутствует имя столбца)
2011-03-11 02:17:00.0016880

WAITFOR

```
select SYSDATETIME(), @ct
waitfor time @ct
select SYSDATETIME()

declare @i int = 0
select SYSDATETIME()
while @i < 5
begin
    waitfor delay '00:00:05'
    select SYSDATETIME()
    set @i = @i+1
end
```

```
2011-03-11 02:26:11.0584448
```

```
(Отсутствует имя столбца)
```

```
2011-03-11 02:26:16.0604000
```

```
(Отсутствует имя столбца)
```

```
2011-03-11 02:26:21.0613520
```

```
(Отсутствует имя столбца)
```

```
2011-03-11 02:26:26.0623040
```

```
(Отсутствует имя столбца)
```

```
2011-03-11 02:26:31.0632560
```

```
(Отсутствует имя столбца)
```

```
2011-03-11 02:26:36.0652112
```



Понятие курсора

- Курсор – программная конструкция, которая служит для хранения результата запроса и для обработки строк результирующего набора запись за записью

Понятие курсора

1. Курсор объявляется в операторе DECLARE.
2. Курсор открывается с помощью оператора OPEN.
3. С помощью оператора FETCH считывается одна или несколько строк результирующего набора, связанного с курсором SELECT-оператора, и обрабатывается нужным образом. Результат каждого считывания проверяется с помощью системной функции @@FETCH_STATUS.
4. Курсор закрывается оператором CLOSE.
5. Если курсор глобальный, то он должен быть освобожден с использованием оператора DEALLOCATE.

Понятие курсора

```
declare @tid char(10), @tnm char(40), @tgn char(1);
declare c_teacher cursor global -- 1
                                for select TEACHER, TEACHER_NAME, GENDER -- 1
                                from TEACHER where PULPIT = 'ПОИСОИ'; -- 1
```

Понятие курсора

```
open c_teacher;                                -- 2
fetch  c_teacher into @tid, @tnm, @tgn;        -- 3
while @@fetch_status = 0                         -- 3
begin
    print @tid + ' ' + @tnm + ' ' + @tgn;
    fetch  c_teacher into @tid, @tnm, @tgn;      -- 3
end;
close   c_teacher;                            -- 4
deallocate c_teacher;                          -- 5
```

БРТШВЧ

Барташевич Святослав Александрович

и

ЮДНКВ

Юденков Виктор Степанович

и

**

**

**

Программные объекты

- Хранимые процедуры
- Скалярные и табличные функции
- Триггеры

Хранимые процедуры

```
use BSTU
go
create procedure GET_PULPITS
as begin                      -- начало кода
    select * from PULPIT;      -- код процедуры
end;                            -- конец кода
```

```
use BSTU
го
-- допускается сокращение ключевого
-- слова EXECUTE
exec GET_PULPITS; -- вызов процедуры
```

PULPIT	PULPIT_NAME	FACULTY
БФ	Белорусской филологии	ИДиП
ИСиТ	Информационных систем и технологий	ИДиП
ЛВ	Лесоводства	NULL
ЛЗиДВ	Лесозащиты и древесиноведения	NULL

ФУНКЦИИ

```
create function COUNT_STUDENTS(@faculty varchar(20)) returns int
as begin
    declare @rc int = 0;
    set @rc = (
        select count(idstudent)
        from FACULTY f join GROUPS g on f.FACULTY = g.FACULTY
                           join STUDENT s on s.IDGROUP = g.IDGROUP
        where f.FACULTY = @faculty
    );
    return @rc;
end;
```

ФУНКЦИИ

```
declare @c int = dbo.COUNT_STUDENTS('ИДиП');  
print 'количество студентов на факультете ИДиП: '+  
      cast(@c as varchar(4));
```

количество студентов на факультете ИДиП: 64

Триггеры

```
create trigger AUD_AFTER_UPDA on Товары after UPDATE  
as print 'AUD_AFTER_UPDATE_A';  
return;
```

Вопросы?

БАЗЫ ДАННЫХ

Лекция 10 Курсы

Курсоры

- **Курсор** – механизм, позволяющий обрабатывать отдельные строки, полученные в результате select-запроса.

Курсоры

- **Курсор** – область памяти сервера, предназначенная для хранения и обработки результата select-запроса.

Курсы

1. Курсор объявляется в операторе DECLARE.
2. Курсор открывается с помощью оператора OPEN.
3. С помощью оператора FETCH считывается одна или несколько строк результирующего набора, связанного с курсором SELECT-оператора, и обрабатывается нужным образом. Результат каждого считывания проверяется с помощью системной функции @@FETCH_STATUS.
4. Курсор закрывается оператором CLOSE.
5. Если курсор глобальный, то он должен быть освобожден с использованием оператора DEALLOCATE.

Курсоры

```
declare @tid char(10), @tnm char(40), @tgn char(1);
declare c_teacher cursor global                                -- 1
                                         for select TEACHER, TEACHER_NAME, GENDER   -- 1
                                         from TEACHER where PULPIT = 'ПОИСОИ'; -- 1

open c_teacher;                                              -- 2

fetch c_teacher into @tid, @tnm, @tgn;                      -- 3

while @@fetch_status = 0                                     -- 3
begin
    print @tid + ' ' + @tnm + ' ' + @tgn;
    fetch c_teacher into @tid, @tnm, @tgn;                  -- 3
end;

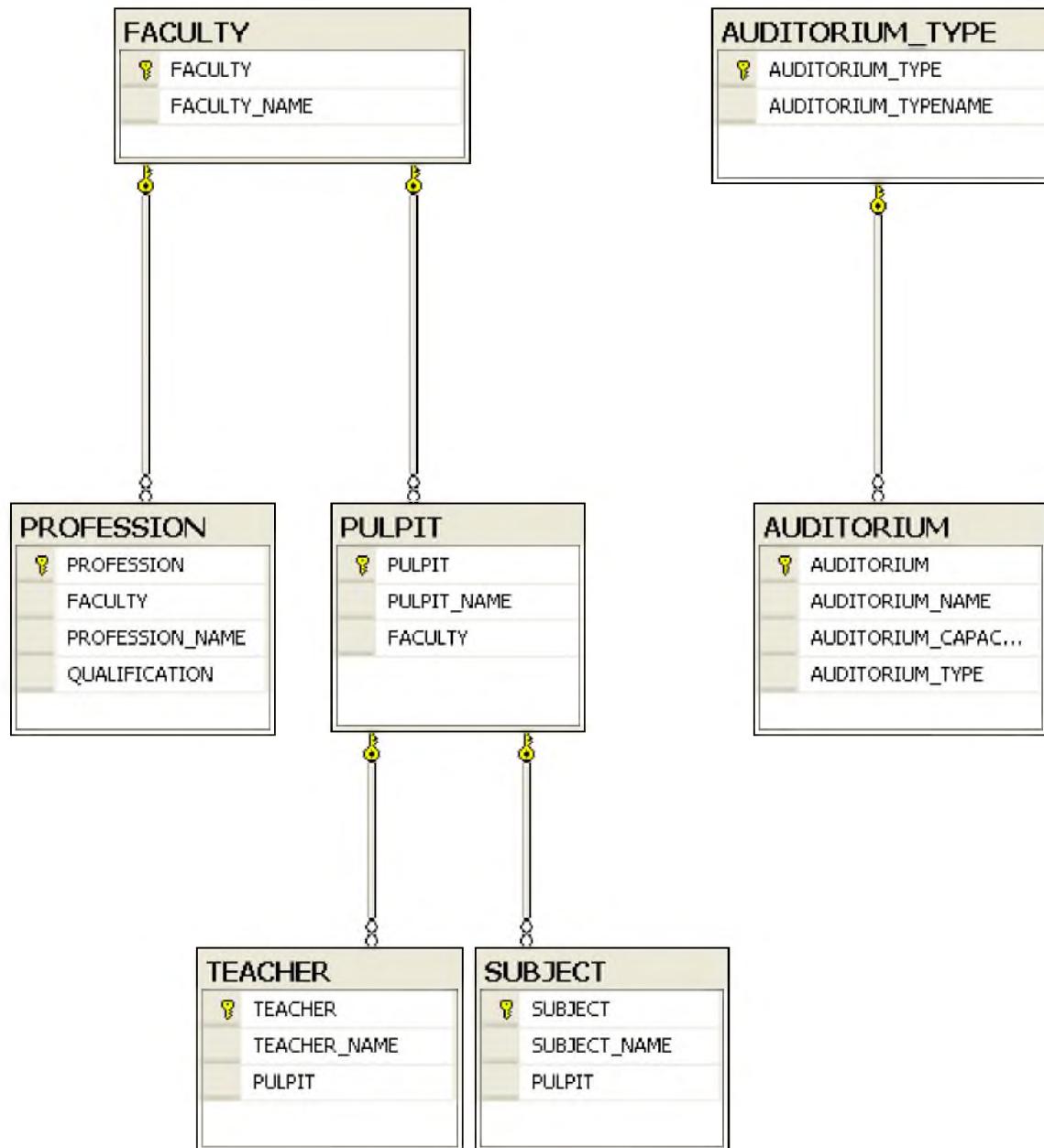
close c_teacher;                                            -- 4

deallocate c_teacher;                                       -- 5
```

Курсы

```
-- DROP TABLE SUBJECT
CREATE TABLE SUBJECT
(
    SUBJECT      CHAR(10)      NOT NULL,
    SUBJECT_NAME VARCHAR(50)   NOT NULL,
    PULPIT       CHAR(10)      NOT NULL,
    CONSTRAINT PK_SUBJECT PRIMARY KEY(SUBJECT),
    CONSTRAINT FK_SUBJECT_PULPIT FOREIGN KEY(PULPIT) REFERENCES PULPIT(PULPIT)
)
```

SUBJECT	SUBJECT_NAME	PULPIT
ИНФ	Информационные технологии	ИСиТ
КГ	Компьютерная геометрия	ИСиТ
КМС	Компьютерные мультимедийные системы	ИСиТ
БД	Базы данных	ИСиТ
МСОИ	Моделирование систем обработки информации	ИСиТ
ОАиП	Основы алгоритмизации и программирования	ИСиТ
ПЗ	Представление знаний в компьютерных системах	ИСиТ
ПИС	Проектирование информационных систем	ИСиТ
СУБД	Системы управления базами данных	ИСиТ
ПСП	Программирование сетевых приложений	ИСиТ
ЛВ	Лесоводство	ЛЗиДВ
ТиОЛ	Технология и оборудование лесозаготовок	ЛМиЛЗ
ОСПиЛ...	Основы садовопаркового и лесопаркового хозя...	ЛПиС...
ИГ	Инженерная геодезия	ЛУ
ЭП	Экономика природопользования	МиЭП
БЛЗиП...	Биология лесных зверей и птиц с осн. охотов.	ОВ
ОХ	Органическая химия	ОХ
ПМАПЛ	Полиграфические машины, автоматы и поточны...	ПОиС...
ОПП	Организация полиграфического производства	ПОиС...
ВТЛ	Водный транспорт леса	ТЛ
ТОПИ	Технология обогащения полезных ископаемых	ТНВи...
ТРИ	Технология резиновых изделий	ТНХС...
ПЭХ	Прикладная электрохимия	ХТЭП...
ЭТ	Экономическая теория	ЭТиМ



Курсы – пример

```
] declare ccc cursor
      for select subject, pulpit from subject
      -      read_only
declare @s char(10), @p char(10)
      -
open ccc
fetch ccc into @s, @p
print @s+' '+@p
close ccc
      -
go
```

Курсоры

- Глобальные курсоры
- Локальные курсоры

Курсы

```
declare ccc cursor
    for select subject, pulpit from subject
        read_only
declare @s char(10), @p char(10)

open ccc
fetch ccc into @s, @p
print @s+' '+@p
close ccc

go
```

Сообщение 16915, уровень 16, состояние 1, строка 2
Курсор с именем "ccc" уже существует.

Курсы

```
| declare ccc cursor
|         for select subject, pulpit from subject
|         read_only
declare @s char(10), @p char(10)

open ccc
fetch ccc into @s, @p
print @s+' '+@p
close ccc
deallocate ccc
go |
```

Курсы

```
declare ccc cursor local
                  for select subject, pulpit from subject
                  read_only
declare @s char(10), @p char(10)

open ccc
fetch ccc into @s, @p
print @s+' '+@p
close ccc
-- deallocate ccc
go |
```

Курсы

```
declare ccc cursor global
    for select subject, pulpit from subject
        read_only
declare @s char(10), @p char(10)

open ccc
fetch ccc into @s, @p
print @s+' '+@p
close ccc |
deallocate ccc
go
```

Типы курсоров

- Динамические
- Статические

Курсоры

- **Динамический курсор** – изменения данных отображаются в динамике

Курсоры

- **Статический курсор** – данные выбраны один раз и произошедшие изменения не видны

Курсы

- Асинхронное заполнение статических курсоров оптимизирует производительность
- Статические курсоры используют рабочие таблицы базы данных **tempdb** для хранения строк, составляющих курсор
- Если в соответствии с прогнозом оптимизатора запросов SQL Server ожидаемое число строк, возвращаемых в курсоре, превысит значение параметра **sp_configure cursor threshold**, сервер запускает отдельный поток для заполнения рабочей таблицы

Курсы

- Функция `@@CURSOR_ROWS` сообщает число строк в курсоре.
- `@@CURSOR_ROWS` для курсора, рабочая таблица которого продолжает заполняться, возвращается отрицательное число. Абсолютное значение возвращенного числа дает число строк в рабочей таблице, заполненных на данный момент времени.
- Например, если функция `@@CURSOR_ROWS` выбрана, пока идет заполнение набора ключей или курсора, управляемого набором ключей, но в наборе ключей уже имеется 143 ключа, функция возвращает значение -143.

Курсы

- @@CURSOR_ROWS
 - -n – количество записей при асинхронной выборке,
 - n – количество записей при синхронной выборке,
 - 0 – курсор не открыт

Курсы

```
| declare ccc cursor local
|     for select subject, pulpit from subject
|     read_only
|
| declare @s char(10), @p char(10)
| print ' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
| open ccc
| print ' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
| fetch ccc into @s, @p
| print @s+' '+@p+' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
| fetch ccc into @s, @p
| print @s+' '+@p+' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
| close ccc
|
| @@CURSOR_ROWS = 0
| @@CURSOR_ROWS = -1
БД           ИСиТ      @@CURSOR_ROWS = -1
БЛЭкПс00    ОВ        @@CURSOR_ROWS = -1
```

Курсы

```
declare ccc cursor local dynamic
      for select subject, pulpit from subject
      read_only
declare @s char(10), @p char(10)
print '@@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
open ccc
print '@@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
fetch ccc into @s, @p
print @s+' '+@p+' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
fetch ccc into @s, @p
print @s+' '+@p+' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
close ccc
|
@@CURSOR_ROWS = 0
@@CURSOR_ROWS = -1
ЗД           ИСиТ           @@CURSOR_ROWS = -1
БЛЭкПс00    ОВ           @@CURSOR_ROWS = -1
```

Курсы

```
declare ccc cursor local static
    for select subject, pulpit from subject
    read_only
declare @s char(10), @p char(10)
print '@@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
open ccc
print '@@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
fetch ccc into @s, @p
print @s+' '+@p+' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
fetch ccc into @s, @p
print @s+' '+@p+' @@CURSOR_ROWS = ' + CAST(@@CURSOR_ROWS as varchar(10))
close ccc
```

@@CURSOR_ROWS = 0

@@CURSOR_ROWS = 24

БД МСиТ

БЛЗиПс00 ОВ

@@CURSOR_ROWS = 24

@@CURSOR_ROWS = 24

Курсы

- @@FETCH_STATUS – возвращает состояние последней инструкции FETCH, вызванной в любом курсоре, открытом в данном соединении

Курсы

- @@FETCH_STATUS
 - 0 – успешная выборка,
 - -1 – вышли за диапазон таблицы,
 - -2 – запись удалена после открытия курсора

Курсы

```
declare ccc cursor local
    for select subject, pulpit from subject
        read_only
declare @s char(10), @p char(10)
print ' @@FETCH_STATUS = '+ CAST(@@FETCH_STATUS as varchar(10))
open ccc
print ' @@FETCH_STATUS = '+ CAST(@@FETCH_STATUS as varchar(10))
fetch ccc into @s, @p
while @@FETCH_STATUS = 0
begin
    print @s+' '+@p+' @@FETCH_STATUS = '+ CAST(@@FETCH_STATUS as varchar(10))
    fetch ccc into @s, @p
end
print ' @@FETCH_STATUS = '+ CAST(@@FETCH_STATUS as varchar(10))
close ccc
print ' @@FETCH_STATUS = '+ CAST(@@FETCH_STATUS as varchar(10))
go|
```

Курсы

<u>@@FETCH_STATUS = -1</u>		
<u>@@FETCH_STATUS = -1</u>		
БД	ИСит	<u>@@FETCH_STATUS = 0</u>
БЛЭиПс00	ОВ	<u>@@FETCH_STATUS = 0</u>
ВТЛ	ТЛ	<u>@@FETCH_STATUS = 0</u>
ИГ	ЛУ	<u>@@FETCH_STATUS = 0</u>
ИН*	ИСит	<u>@@FETCH_STATUS = 0</u>
КГ	ИСит	<u>@@FETCH_STATUS = 0</u>
КМС	ИСит	<u>@@FETCH_STATUS = 0</u>
ЛВ	ЛЭиДВ	<u>@@FETCH_STATUS = 0</u>
МСОИ	ИСит	<u>@@FETCH_STATUS = 0</u>
ОАиП	ИСит	<u>@@FETCH_STATUS = 0</u>
ОПП	ПОиСОИ	<u>@@FETCH_STATUS = 0</u>
ОСПиШХ	ЛШиСПС	<u>@@FETCH_STATUS = 0</u>
ОХ	ОХ	<u>@@FETCH_STATUS = 0</u>
ПЗ	ИСит	<u>@@FETCH_STATUS = 0</u>
ПИС	ИСит	<u>@@FETCH_STATUS = 0</u>
ПМАПП	ПОиСОИ	<u>@@FETCH_STATUS = 0</u>
ПСП	ИСит	<u>@@FETCH_STATUS = 0</u>
ПЭХ	ХТЭПиМЭ	<u>@@FETCH_STATUS = 0</u>
СУБД	ИСит	<u>@@FETCH_STATUS = 0</u>
ТиОП	ЛМиЛЭ	<u>@@FETCH_STATUS = 0</u>
ТОПИ	ТНВиОХТ	<u>@@FETCH_STATUS = 0</u>
ТРИ	ТНХСиППМ	<u>@@FETCH_STATUS = 0</u>
ЭП	МиЭП	<u>@@FETCH_STATUS = 0</u>
ЭТ	ЭТиМ	<u>@@FETCH_STATUS = 0</u>
<u>@@FETCH_STATUS = -1</u>		
<u>@@FETCH_STATUS = -1</u>		

Курсы

- CURSOR_STATUS – скалярная функция, позволяющая при вызове хранимой процедуры определить, вернула ли она курсор и результирующий набор для данного параметра

Курсы

```
DECLARE @a1 INT, @a2 INT, @a3 INT, @a4 INT, @a5 INT;

DECLARE CURS_SUBJECTS CURSOR FOR
SELECT SUBJECT FROM SUBJECT WHERE PULPIT='ИСиТ' ;
    SET @a1 = CURSOR_STATUS('GLOBAL', 'CURS_SUBJECTS');
DECLARE  @S1 NCHAR(5), @S NCHAR(300) = '';

OPEN CURS_SUBJECTS;
    SET @a2 = CURSOR_STATUS('GLOBAL', 'CURS_SUBJECTS');
FETCH CURS_SUBJECTS INTO @S1;
    SET @a3 = CURSOR_STATUS('GLOBAL', 'CURS_SUBJECTS');
PRINT 'Список кратких наименований предметов на кафедре ИСиТ: ';
WHILE (@@FETCH_STATUS=0)
BEGIN
    SET @S=RTRIM(@S1)+', '+@S;
    FETCH CURS_SUBJECTS INTO @S1;
END;
PRINT LEFT(@S, LEN(@S)-1);
    SET @a4 = CURSOR_STATUS('GLOBAL', 'CURS_SUBJECTS');
CLOSE CURS_SUBJECTS;
    SET @a5 = CURSOR_STATUS('GLOBAL', 'CURS_SUBJECTS');

SELECT @a1 AS 'AFTER DECLARE',
    @a2 AS 'AFTER OPEN',
    @a3 AS 'AFTER 1 FETCH',
    @a4 AS 'AFTER ALL FETCH',
    @a5 AS 'AFTER CLOSE';
```

	AFTER declare	AFTER OPEN	AFTER 1 FETCH	AFTER ALL FETCH	AFTER CLOSE
	1	-1	1	1	1

Курсы – SCROLL

```
declare ccc cursor local scroll
    for select subject, pulpit from subject order by pulpit
        --read_only -- не совместен со scroll
declare @s char(10), @p char(10)
open ccc
fetch ccc into @s, @p
while @@FETCH_STATUS = 0
begin
    print @s+' '+@p
    fetch ccc into @s, @p
end
close ccc
go
```

Курсоры

ИНФ	ИСиТ
КГ	ИСиТ
КМС	ИСиТ
ЗД	ИСиТ
ЧСОИ	ИСиТ
ЭАиП	ИСиТ
ЛЗ	ИСиТ
Лис	ИСиТ
СУБД	ИСиТ
ЛСП	ИСиТ
ОВ	ЛЗиДВ
ГиОЛ	ЛМиЛЗ
ЭСПиЛПХ	ЛМиСПС
ИГ	ЛУ
ЭП	МиЭП
БЛЗиПсОО	ОВ
ЭХ	ОХ
ЛМАПЛ	ПОиСОИ
ЭПП	ПОиСОИ
ЗТЛ	ТЛ
ГОПИ	ТНВиОХТ
ГРИ	ТНХСиППМ
ЛЭХ	ХТЭПиМЭЕ
ЭТ	ЭТИМ



```

declare ccc cursor local scroll
    for select subject, pulpit from subject order by pulpit
declare @s char(10), @p char(10)
open ccc

fetch last from ccc into @s, @p
print @s+' '+@p

fetch first from ccc into @s, @p
print @s+' '+@p

fetch absolute 10 from ccc into @s, @p
print @s+' '+@p

fetch relative 5 from ccc into @s, @p
print @s+' '+@p

fetch relative -5 from ccc into @s, @p
print @s+' '+@p

fetch absolute -10 from ccc into @s, @p
print @s+' '+@p

fetch next from ccc into @s, @p
print @s+' '+@p

fetch prior from ccc into @s, @p
print @s+' '+@p

close ccc
go

```

<u>fetch last</u> from ccc into @s, @p <u>fetch first</u> from ccc into @s, @p <u>fetch absolute 10</u> from ccc into @s, @p <u>fetch relative 5</u> from ccc into @s, @p <u>fetch relative -5</u> from ccc into @s, @p <u>fetch absolute -10</u> from ccc into @s, @p <u>fetch next</u> from ccc into @s, @p <u>fetch prior</u> from ccc into @s, @p	ЭТ ИИФ ПСП ЭП ПСП ЭП БЛЗиПсОО ЭП	ЭТИМ ИСиТ ИСиТ МиЭП ИСиТ МиЭП ОВ МиЭП
--	---	--

Курсы

```
declare @s char(10), @ps char(10),  @p char(10)
declare ccc cursor local dynamic scroll
    for select subject, pulpit from subject where pulpit = @ps
set @ps = 'ИСиТ'
open ccc
fetch ccc into @s, @p
while @@FETCH_STATUS = 0
begin
    print @s+' '+@p
    fetch ccc into @s, @p
end
close ccc
go
|
Выполнение команд успешно завершено.
```

Курсы

```
| declare @s char(10), @ps char(10) = 'ИСиТ', @p char(10)
| declare ccc cursor local dynamic scroll
-         for select subject, pulpit from subject where pulpit = @ps
-- set @ps = 'ИСиТ'


---


open ccc
fetch ccc into @s, @p
while @@FETCH_STATUS = 0
begin
    print @s+' '+@p
    fetch ccc into @s, @p
end |
close ccc
go
```

БД	ИСиТ
ИНФ	ИСиТ
КГ	ИСиТ
КМС	ИСиТ
МСОИ	ИСиТ
ОАиП	ИСиТ
ПЗ	ИСиТ
ПМС	ИСиТ
ПСП	ИСиТ
СУБД	ИСиТ

Курсы

```
declare @s char(10), @ps char(10), @p char(10)
set @ps = 'ИСиТ'

declare ccc cursor local dynamic scroll
    for select subject, pulpit from subject where pulpit = @ps
-- set @ps = 'ИСиТ'

open ccc
fetch ccc into @s, @p
while @@FETCH_STATUS = 0
begin
    print @s+' '+@p
    fetch ccc into @s, @p
end
close ccc
go
```

БД	ИСиТ
ИИФ	ИСиТ
КГ	ИСиТ
КМС	ИСиТ
МСОИ	ИСиТ
ОАиП	ИСиТ
ПЗ	ИСиТ
ПМС	ИСиТ
ПСП	ИСиТ
СУБД	ИСиТ

Курсы – UPDATE CURRENT OF

```
declare @s char(10), @ps char(10), @p char(10), @n varchar(200)
declare ccc cursor local dynamic scroll
    for select subject, pulpit, subject_name from subject
open ccc
fetch ccc into @s, @p, @n
while @@FETCH_STATUS = 0
begin
    if @s = 'БД' update subject set subject_name = 'Самая важная дисциплина!!!!'
        where current of ccc
    fetch ccc into @s, @p, @n
end
close ccc
go
```

Курсы – DELETE CURRENT OF

```
| declare @s char(10), @ps char(10), @p char(10), @n varchar(200)
| declare ccc cursor local dynamic scroll
|         for select subject, pulpit, subject_name from subject
open ccc
fetch ccc into @s, @p, @n
while @@FETCH_STATUS = 0
begin
    if @s = 'БД' delete subject where current of ccc
    fetch ccc into @s, @p, @n
end
close ccc
go|
```

Вложенные курсоры

```
SET NOCOUNT ON;

DECLARE @vendor_id int, @vendor_name nvarchar(50),
        @message varchar(80), @product nvarchar(50);

PRINT '----- Vendor Products Report -----';

DECLARE vendor_cursor CURSOR FOR
SELECT BusinessEntityID, Name
FROM Purchasing.Vendor
WHERE PreferredVendorStatus = 1
ORDER BY BusinessEntityID;

OPEN vendor_cursor

FETCH NEXT FROM vendor_cursor
INTO @vendor_id, @vendor_name

WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT ''
    SELECT @message = '----- Products From Vendor: ' +
                     @vendor_name

    PRINT @message
```

```
-- Declare an inner cursor based
-- on vendor_id from the outer cursor.
DECLARE product_cursor CURSOR FOR
SELECT v.Name
FROM Purchasing.ProductVendor pv, Production.Product v
WHERE pv.ProductID = v.ProductID AND
pv.BusinessEntityID = @vendor_id -- Variable value from the outer cursor

OPEN product_cursor
FETCH NEXT FROM product_cursor INTO @product
IF @@FETCH_STATUS <> 0
    PRINT '          <<None>>'

WHILE @@FETCH_STATUS = 0
BEGIN
    SELECT @message = '          ' + @product
    PRINT @message
    FETCH NEXT FROM product_cursor INTO @product
END
CLOSE product_cursor
DEALLOCATE product_cursor
```

-- Get the next vendor.

```
        FETCH NEXT FROM vendor_cursor
        INTO @vendor_id, @vendor_name
END
CLOSE vendor_cursor;
DEALLOCATE vendor_cursor;
```

Вложенные курсоры

----- Vendor Products Report -----

----- Products From Vendor: Australia Bike Retailer

- Thin-Jam Lock Nut 9
- Thin-Jam Lock Nut 10
- Thin-Jam Lock Nut 1
- Thin-Jam Lock Nut 2
- Thin-Jam Lock Nut 15
- Thin-Jam Lock Nut 16
- Thin-Jam Lock Nut 5
- Thin-Jam Lock Nut 6
- Thin-Jam Lock Nut 3
- Thin-Jam Lock Nut 4
- Thin-Jam Lock Nut 13
- Thin-Jam Lock Nut 14
- Thin-Jam Lock Nut 7
- Thin-Jam Lock Nut 8
- Thin-Jam Lock Nut 12
- Thin-Jam Lock Nut 11

----- Products From Vendor: Allenson Cycles

- Seat Post

Вложенные курсоры

----- Products From Vendor: Morgan Bike Accessories
HL Grip Tape

----- Products From Vendor: Cycling Master
<<None>>

----- Products From Vendor: Chicago Rent-All
Reflector

----- Products From Vendor: Greenwood Athletic Company
LL Mountain Pedal
ML Mountain Pedal

----- Products From Vendor: Compete Enterprises, Inc
Guide Pulley
Tension Pulley
HL Road Pedal

Процедуры, поддерживающие курсоры

- sp_cursor_list
- sp_describe_cursor
- sp_describe_cursor_columns
- sp_describe_cursor_tables

sp_cursor_list

```
DECLARE @REPORT CURSOR;
EXEC SP_CURSOR_LIST @CURSOR_RETURN = @REPORT OUTPUT, @CURSOR_SCOPE =3;
OPEN @REPORT;
FETCH NEXT FROM @REPORT;
WHILE (@FETCH_STATUS=0)
BEGIN
FETCH NEXT FROM @REPORT;
END;
CLOSE @REPORT;
```

	reference_name	cursor_name	cursor_scope	status	model	concurrency	scrollable
1	CURS SUBJECTS	CURS SUBJECTS	2	-1	3	3	0

	reference_name	cursor_name	cursor_scope	status	model	concurrency	scrollable	open_
	open_status	cursor_rows	fetch_status	column_count	row_count	last_operation	cursor_handle	

	status	cursor_rows	fetch_status	column_count	row_count	last_operation	cursor_handle
	0	0	-1	1	0	6	180150009

sp_describe_cursor

```
DECLARE @REPORT CURSOR;
EXEC SP_describe_CURSOR @CURSOR_return = @REPORT OUTPUT,
@CURSOR_Source ='global',
@CURSOR_Identity ='CURS SUBJECTS';

OPEN @REPORT;
FETCH NEXT FROM @REPORT;
WHILE (@@FETCH_STATUS=0)
BEGIN
FETCH NEXT FROM @REPORT;
END;
CLOSE @REPORT;
DEALLOCATE @REPORT;
```

	reference_name	cursor_name	cursor_scope	status	model	concurrency	scrollable
1	CURS SUBJECTS	CURS SUBJECTS	2	-1	3	3	0

	reference_name	cursor_name	cursor_scope	status	model	concurrency	scrollable	open

	open_status	cursor_rows	fetch_status	column_count	row_count	last_operation	cursor_handle
	0	0	-1	1	0	6	180150009

	status	cursor_rows	fetch_status	column_count	row_count	last_operation	cursor_handle

sp_describe_cursor_columns

```
-- SP_DESCRIBE_CURSOR_COLUMNS
-----
DECLARE @REPORT CURSOR;
EXEC SP_DESCRIBE_CURSOR_COLUMNS @CURSOR_return = @REPORT OUTPUT,
                                @CURSOR_Source ='global',
                                @CURSOR_Identity ='CURS_SUBJECTS';
OPEN @REPORT;
FETCH NEXT FROM @REPORT;
WHILE (@@FETCH_STATUS=0)
BEGIN
    FETCH NEXT FROM @REPORT;
END;
CLOSE @REPORT;
DEALLOCATE @REPORT;
```

Results | Messages |

column_name	ordinal_position	column_characteristics_flags	column_size	data_type_sql	column_precision
SUBJECT	0	18	10	175	0

column_name ordinal_position column_characteristics_flags column_size data_type_sql column_precision

column_scale	order_position	order_direction	hidden_column	columnid	objectid	dbid	dbname
0	0	NULL	0	1	565577053	11	B_BSTU

column_scale order_position order_direction hidden_column columnid objectid dbid dbname

sp_describe_cursor_tables

```
-- SP_DESCRIBE_CURSOR_TABLES
-----
DECLARE @REPORT CURSOR;
EXEC SP_DESCRIBE_CURSOR_TABLES @CURSOR_return = @REPORT OUTPUT,
                                @CURSOR_Source ='global',
                                @CURSOR_Identity ='CURS_SUBJECTS';
OPEN @REPORT;
FETCH NEXT FROM @REPORT;
WHILE (@@FETCH_STATUS=0)
BEGIN
FETCH NEXT FROM @REPORT;
END;
CLOSE @REPORT;
DEALLOCATE @REPORT;
```

The screenshot shows the SQL Server Management Studio interface with two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and displays a table with the following data:

	table_owner	table_name	optimizer_hint	lock_type	server_name	objectid	dbid	dbname
1	dbo	SUBJECT	0	0	1-VAIO	565577053	11	B_BSTU

Below the first table, there is another table structure with empty rows, indicating the continuation of the result set.

Вопросы?

БАЗЫ ДАННЫХ

Лекция 11 Процедуры и функции

Процедуры и функции

- Пакет (batch)
- Подпрограмма (routine)
 - Процедура
 - Функция
- Системные
- Пользовательские

Пакет

- Пакет — это последовательность инструкций, которые отправляются системе базы данных для совместного их выполнения
- Преимущество – одновременное исполнение всех инструкций позволяет получить улучшение производительности

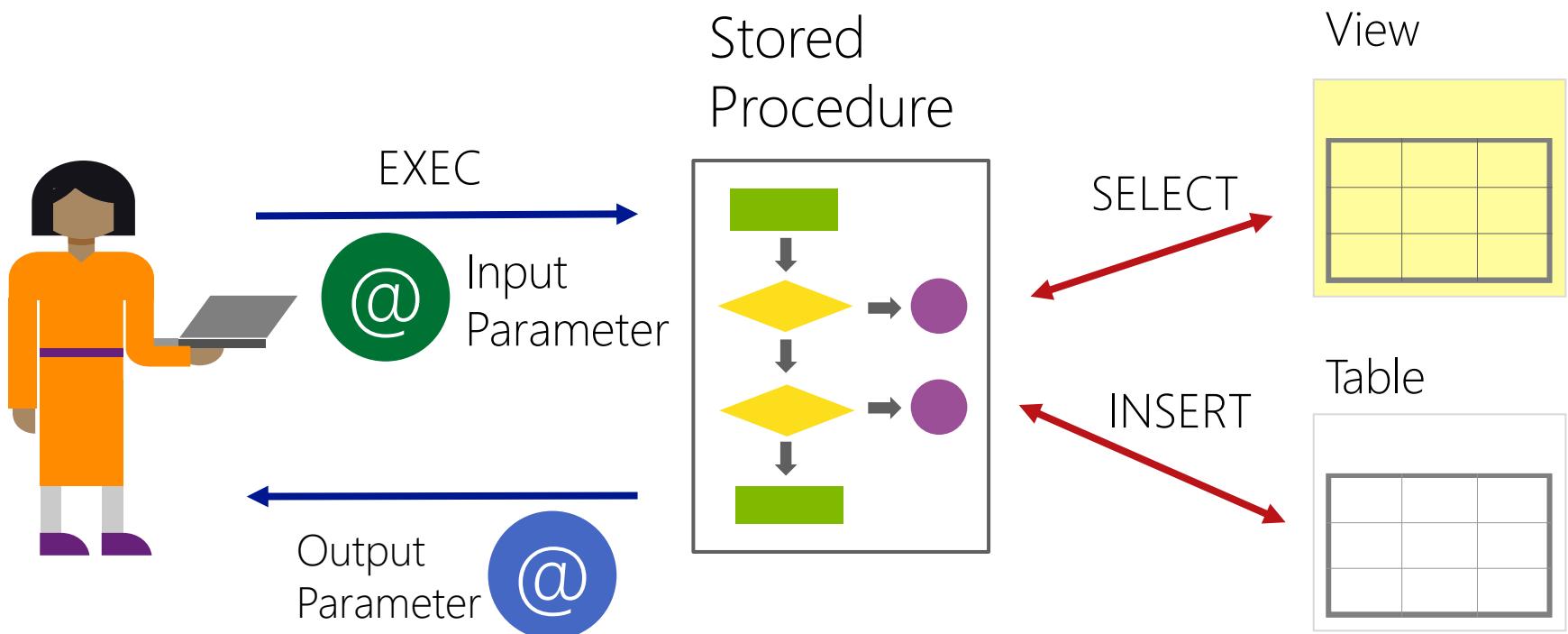
Пакет

- Ограничение на единственную инструкцию:
- CREATE VIEW
- CREATE PROCEDURE
- CREATE TRIGGER

Хранимые процедуры

- Хранимая процедура – объект базы данных
- Хранимая процедура – поименованный блок (BEGIN...END) операторов, хранящийся в базе данных в откомпилированном виде

Хранимые процедуры



Хранимые процедуры

- Принимает входные параметры
- Принимает и формирует выходные параметры

Хранимые процедуры

- Целочисленное значение, возвращаемое к точке вызова с помощью оператора RETURN
- Один или более результирующих наборов, сформированных операторами SELECT
- Содержимое стандартного выходного потока, полученного при выполнении операторов PRINT

Хранимые процедуры

Результирующий набор хранимой процедуры может быть использован в качестве источника строк для INSERT

Хранимые процедуры

- Допускается применение :
 - Основных DDL, DML и TCL-операторов
 - Конструкций TRY/CATCH
 - Курсоров
 - Временных таблиц

Хранимые процедуры

- Не допускается применение :
 - CREATE or ALTER FUNCTION
 - CREATE or ALTER TRIGGER
 - CREATE or ALTER PROCEDURE
 - CREATE or ALTER VIEW
 - USE databasename

Хранимые процедуры

```
CREATE TABLE TLESSON  
(  
    TLESSON      CHAR(10) NOT NULL,  
    TLESSON_NAME VARCHAR(30),  
    CONSTRAINT PK_TLESSON PRIMARY KEY(TLESSON)  
) ;
```

TLESSON	TLESSON_NAME
ДП	Дипломное проектирование
ЗП	Практическое занятие
ЗЧ	Зачет
КП	Курсовое проектирование
КЧ	Классный час
ЛК	Лекция
ЛР	Лабораторная работа
ПД	Преддипломная практика
ПИ	Инженерная практика
ПП	Программистская практика
ПР	Производственная практи...
СМ	Семинар
ФК	Занятие физкультурой
ЭК	Экзамен

Хранимые процедуры

```
drop procedure PrintTLESSON  
  
create procedure PrintTLESSON  
as  
    declare @lt char(10), @ln varchar(30)  
    declare ctl cursor local read_only for select * from TLESSON order by tlesson  
begin
```

```
    print '----- Типы занятий (TLESSON) -----'
```

```
    open ctl  
    fetch ctl into @lt, @ln  
    while @@FETCH_STATUS = 0  
        begin  
            print @lt+' '+@ln  
            fetch ctl into @lt, @ln  
        end  
    close ctl
```

```
end
```

```
exec PrintTLESSON
```

----- Типы занятий (TLESSON) -----	
ДП	Дипломное проектирование
ЗП	Практическое занятие
ЗЧ	Зачет
КП	Курсовое проектирование
КЧ	Классный час
ЛК	Лекция
ЛР	Лабораторная работа
ПД	Преддипломная практика
ПИ	Инженерная практика
ПП	Программистская практика
ПР	Производственная практика
СМ	Семинар
ФК	Занятие физкультурой
ЭК	Экзамен

Хранимые процедуры

```
drop procedure SelectTLESSON  
.  
  
create procedure SelectTLESSON  
as  
begin  
    select * from TLESSON  
end
```

```
exec SelectTLESSON
```

TLESSON	TLESSON_NAME
ДП	Дипломное проектирование
ЗП	Практическое занятие
ЗЧ	Зачет
КП	Курсовое проектирование
КЧ	Классный час
ЛК	Лекция
ЛР	Лабораторная работа
ПД	Преддипломная практика
ПИ	Инженерная практика
ПП	Программистская практика
ПР	Производственная практика
СМ	Семинар
ФК	Занятие физкультурой
ЭК	Экзамен

Хранимые процедуры

```
declare @x int, @y int
print '1) @@ROWCOUNT = '+CAST(@@ROWCOUNT as varchar(10))
select * from TLESSON
print '2) @@ROWCOUNT = '+CAST(@@ROWCOUNT as varchar(10))
print '3) @@ROWCOUNT = '+CAST(@@ROWCOUNT as varchar(10))
set @x = 1
print '4) @@ROWCOUNT = '+CAST(@@ROWCOUNT as varchar(10))
select @x = 2, @y = 3
print '5) @@ROWCOUNT = '+CAST(@@ROWCOUNT as varchar(10))
```

TLESSON	TLESSON_NAME
ДП	Дипломное проектирование
ЗП	Практическое занятие
ЗЧ	Зачет
КП	Курсовое проектирование
КЧ	Классный час
ЛК	Лекция
ЛР	Лабораторная работа
ПД	Преддипломная практика
ПИ	Инженерная практика
ПП	Программистская практика
ПР	Производственная практика
СМ	Семинар
ФК	Занятие физкультурой
ЭК	Экзамен

1) @@ROWCOUNT = 0

(строк обработано: 14)

2) @@ROWCOUNT = 14

3) @@ROWCOUNT = 0

4) @@ROWCOUNT = 1

5) @@ROWCOUNT = 1

|

Хранимые процедуры

```
create procedure SelectTLESSON
as
    declare @rc int = 0
begin
    select * from TLESSON
    if @@ROWCOUNT = 0 set @rc = -1
    return @rc
end

declare @code int = 77
exec @code = SelectTLESSON
print 'код возврата = '+ CAST(@code as varchar(10))

(строк обработано: 14)
код возврата = 0
```

Возможен
только
числовой код
возврата

Хранимые процедуры

```
create procedure GetExamResults -- создать процедуру с именем GetExamResults
as
begin -- формирует результирующий набор
    select st.NAME,pr.PDATE,sb.SUBJECT_NAME, pr.NOTE
        from STUDENT st join PROGRESS pr on st.IDSTUDENT = pr.IDSTUDENT
                           join [SUBJECT] sb on pr.[SUBJECT] = sb.[SUBJECT];
end;
go;
execute GetExamResults; -- вызов процедуры
```

NAME	PDATE	SUBJECT_NAME	NOTE
Манакова Анастасия Владимировна	2013-01-10	Основы алгоритмизации и программирования	6
Хартанович Екатерина Александровна	2013-01-10	Основы алгоритмизации и программирования	8
Горбач Елизавета Юрьевна	2013-01-10	Основы алгоритмизации и программирования	7
Зыкова Кристина Дмитриевна	2013-01-10	Основы алгоритмизации и программирования	5

```
use BSTU
go
insert [#Результаты экзамена] execute GetExamResults;
```

ROW_NUMBER() OVER

```
select ROW_NUMBER() over (order by TEACHER), TEACHER, TEACHER_NAME from TEACHER
```

(Отсутствует имя столбца)	TEACHER	TEACHER_NAME
1	?	Неизвестный
2	АКНВЧ	Акунович Станислав Иванович
3	БЗБРДВ	Безбородов Владимир Степанович
4	БРКВЧ	Бракович Андрей Игорьевич
5	БРНВСК	Барановский Станислав Иванович
6	БРТШВЧ	Барташевич Святослав Александрович
7	ГРМН	Герман Олег Витольдович
8	ГРН	Гурин Николай Иванович
9	ДДК	Дедко Александр Аркадьевич
10	ДМДК	Демидко Марина Николаевна
11	ЕЩНК	Ещенко Людмила Семеновна
12	ЖЛК	Жиляк Надежда Александровна
13	ЖРСК	Жарский Иван Михайлович
14	ЗВГЦВ	Звягинцев Вячеслав Борисович
15	КБЛ	Кабайло Александр Серафимович
16	КЛСНВ	Колесников Леонид Валерьевич
17	ЛБХ	Лабоха Константин Валентинович
18	ЛЩНК	Лащенко Анатолий Павлович
19	МХВ	Мохов Сергей Петрович
20	МШКВСК	Машковский Владимир Петрович
21	НВРВ	Неверов Александр Васильевич
22	НСКВЦ	Насковец Михаил Трофимович
23	ПРКПЧК	Прокопчук Николай Романович
24	ПСТВЛВ	Пустовалова Наталия Николаевна
25	РРКИ	Рыжиков Евгений Евгеньевич

ROW_NUMBER() OVER

```
select ROW_NUMBER() over (partition by PULPIT order by TEACHER),  
       PULPIT, TEACHER, TEACHER_NAME from TEACHER
```

1	ИСиТ	?	Неизвестный
2	ИСиТ	АКНВЧ	Акунович Станислав Иванович
3	ИСиТ	БРКВЧ	Бракович Андрей Игорьевич
4	ИСиТ	ГРМН	Герман Олег Витольдович
5	ИСиТ	ГРН	Гурин Николай Иванович
6	ИСиТ	ДДК	Дедко Александр Аркадьевич
7	ИСиТ	ЖЛК	Жилик Надежда Александровна
8	ИСиТ	КБЛ	Кабайло Александр Серафимович
9	ИСиТ	КЛСНВ	Колесников Леонид Валерьевич
10	ИСиТ	ЛЩНК	Лащенко Анатолий Павлович
11	ИСиТ	ПСТВЛВ	Пустовалова Наталия Николаевна
12	ИСиТ	РМНК	Романенко Дмитрий Михайлович
13	ИСиТ	СМЛВ	Смелов Владимир Владиславович
14	ИСиТ	УРБ	Урбанович Павел Павлович
1	ЛВ	ЛБХ	Лабоха Константин Валентинович
1	ЛЗидВ	ЗВГЦВ	Звягинцев Вячеслав Борисович
1	ЛМиЛЗ	МХВ	Мохов Сергей Петрович
1	ЛПиСПС	ДМДК	Демидко Марина Николаевна
1	ЛУ	МШКВСК	Машковский Владимир Петрович
1	МиЭП	НВРВ	Неверов Александр Васильевич
1	ОВ	РВКЧ	Ровкач Андрей Иванович
1	ОХ	БЗБРДВ	Безбородов Владимир Степанович
1	ПОиСОИ	БРТШВЧ	Барташевич Святослав Александрович
2	ПОиСОИ	ЮДНКВ	Юденков Виктор Степанович
1	ТЛ	НСКВЦ	Насковец Михаил Трофимович

WITH

```
with RRR as
```

```
(  
select ROW_NUMBER() over (order by TEACHER) rn, TEACHER, TEACHER_NAME from TEACHER  
)
```

```
select * from RRR where rn < 10
```

n	TEACHER	TEACHER_NAME
1	?	Неизвестный
2	АКНВЧ	Акунович Станислав Иванович
3	БЗБРДВ	Безбородов Владимир Степанович
4	БРКВЧ	Бракович Андрей Игорьевич
5	БРНВСК	Барановский Станислав Иванович
6	БРТШВЧ	Барташевич Святослав Александрович
7	ГРМН	Герман Олег Витольдович
8	ГРН	Гурин Николай Иванович
9	ДДК	Дедко Александр Аркадьевич

Передача параметров

```
create procedure SelectTEACHER
    @p char(10),
    @n int = 5,
    @c int output
as
    declare @rc int = 0
begin
    begin try
        select * from (
            select ROW_NUMBER() over (order by TEACHER) rn, TEACHER, TEACHER_NAME
            from TEACHER
            where PULPIT = @p
            )rrr
            where rrr.rn < @n
        set @c = (select COUNT(*) from TEACHER)
    end try
    begin catch
        set @rc = -1
    end catch
    return @rc
end
```

Передача параметров

```
declare @code int, @ttt int  
exec @code = SelectTEACHER 'ИСиТ', default, @ttt output  
print 'код возврата = ' + CAST(@code as varchar(10))  
print 'output = ' + CAST(@ttt as varchar(10))
```

n	TEACHER	TEACHER_NAME
1	?	Неизвестный
2	АКНВЧ	Акунович Станислав Иванович
3	БРКВЧ	Бракович Андрей Игорьевич
4	ГРМН	Герман Олег Витольдович

(строк обработано: 4)

код возврата = 0

output = 29

Передача параметров

```
declare @code int, @ttt int
exec @code = SelectTEACHER 'ИСиТ', 2, @ttt output
print 'код возврата = ' + CAST(@code as varchar(10))
print 'output = ' + CAST(@ttt as varchar(10))
```

m	TEACHER	TEACHER_NAME
1	?	Неизвестный

(строк обработано: 1)

код возврата = 0

output = 29

Хранимые процедуры

Диаграммы баз данных

Таблицы

Представления

Синонимы

Программирование

Хранимые процедуры

Системные хранимые процедуры

dbo.PrintTLESSON

dbo.SelectTEACHER

Параметры

- @r (char(10), Ввод, Нет значения по умолчанию)
- @n (int, Ввод, По умолчанию)
- @c (int, Ввод-вывод, Нет значения по умолчанию)
- Возвращает целое значение

dbo.SelectTLESSON

Функции

Хранимые процедуры

- ALTER PROCEDURE
- DROP PROCEDURE
- sp_rename

Хранимые процедуры

```
use BSTU
go
drop procedure GET_PULPITS; -- удаление процедуры
go
create procedure GET_PULPITS -- создание процедуры
as begin
    declare @rc int = (select count(*) from PULPIT) ;
    select * from PULPIT;
    return @rc; -- код возврат
end;
```

Хранимые процедуры

```
use BSTU
go
alter procedure GET_PULPITS -- модификация процедуры
    @f varchar(20),
    @c int output
as begin
    declare @rc int = (select count(*) from PULPIT) ;
    print 'параметры: @f = '+@f +', @c = '+ cast(@c as varchar(3));
    select * from PULPIT where FACULTY = @f;
    set @c = @@rowcount;
    return @rc; -- код возврат
end;
```

Динамический SQL

- sp_executesql

```
DECLARE @sqlcode AS NVARCHAR(256) =
    N'<code_to_run>';
EXEC sys.sp_executesql @statement = @sqlcode;
```

```
DECLARE @sqlcode AS NVARCHAR(256) =
    N'SELECT GETDATE() AS dt';
EXEC sys.sp_executesql @statement = @sqlcode;
```

ФУНКЦИИ

- Встроенные
 - Математические
 - Строковые
 - Работа с датами
- Пользовательские

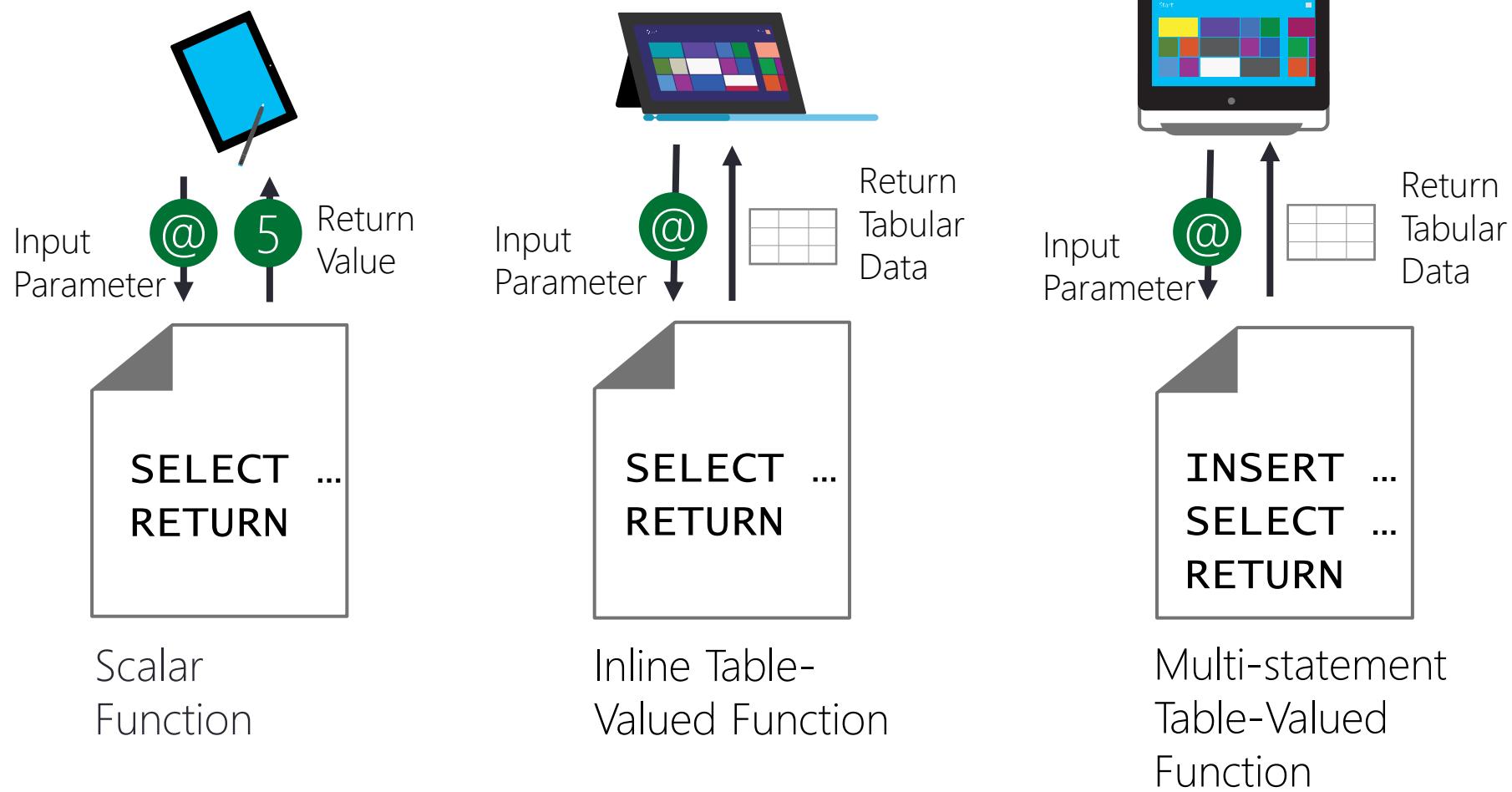
ФУНКЦИИ

- Не допускается применение:
 - DDL-операторов
 - DML-операторов изменяющих данные (INSERT, DELETE, UPDATE)
 - конструкций TRY/CATCH
 - транзакций

ФУНКЦИИ

- Скалярные
- Inline
- Multi-Statement

ФУНКЦИИ



ФУНКЦИИ

- Скалярные
 - возвращает одно значение
 - нельзя timestamp, cursor, text, ntext, image, table

Хранимые процедуры

```
drop function countTEACHER  
  
create function countTEACHER(@p char(10)) returns int  
begin  
    declare @rc int = (select COUNT(*) from TEACHER where PULPIT = @p)  
    return @rc  
end  
  
declare @x int = dbo.countTEACHER('ИСиТ')  
print 'преподавателей ИСиТ = ' + cast(@x as varchar(10))  
select dbo.countTEACHER('ИСиТ')
```

(Отсутствует имя столбца)

ФУНКЦИИ

- Inline
 - одна команда SELECT
 - возврат table

```
create function dbo.fnPROFESSION() returns table  
as return select PROFESSION, FACULTY, QUALIFICATION  
from PROFESSION;
```

```
select * from fnPROFESSION()
```

PROFESSION	FACULTY	QUALIFICATION
1-25 01 07	ИЭФ	экономистменеджер
1-25 01 09	ИЭФ	экономист

ФУНКЦИИ

```
alter function dbo.fnPROFESSION(@f varchar(20) = null, @like_q varchar(30) = null)
returns table
as return select PROFESSION, FACULTY, QUALIFICATION
from PROFESSION
where FACULTY = isnull(@f,FACULTY) and
      1 = case
            when @like_q is null          then 1
            when  QUALIFICATION like @like_q then 1
            else 0
      end;
```

ФУНКЦИИ

```
select * from fnPROFESSION(default, default);
```

PROFESSION	FACULTY	QUALIFICATION
------------	---------	---------------

1-25 01 07	ИЭФ	экономист-менеджер
------------	-----	--------------------

```
select * from fnPROFESSION('ИЭФ', default);
```

PROFESSION	FACULTY	QUALIFICATION
------------	---------	---------------

1-25 01 07	ИЭФ	экономист-менеджер
------------	-----	--------------------

1-25 01 08	ИЭФ	экономист
------------	-----	-----------

ФУНКЦИИ

```
select * from fnPROFESSION(default, 'экономист');
```

PROFESSION	FACULTY	QUALIFICATION
1-25 01 07	ИЭФ	экономист-менеджер
1-25 01 08	ИЭФ	экономист

```
select * from fnPROFESSION('ХИТ', 'механик');
```

PROFESSION	FACULTY	QUALIFICATION
1-36 01 08	ХИТ	инженер-механик
1-36 07 01	ХИТ	инженер-механик

ФУНКЦИИ

```
select count(*) 'кол. механиков на ХТИТ'
from GROUPS g join fnPROFESSION('ХТИТ', 'механик') p
    on g.PROFESSION = p.PROFESSION
join STUDENT s
    on s.IDGROUP = g.IDGROUP
```

кол. механиков на ХТИТ

21

```
select f.FACULTY, p.QUALIFICATION
    from FACULTY f left outer join fnPROFESSION(default, 'механик') p
        on f.FACULTY = p.FACULTY
```

FACULTY	QUALIFICATION
ИДиП	инженер-электромеханик
ИЗФ	NULL
ЛХФ	NULL
ТОВ	NULL
ТТЛП	инженер-механик

ФУНКЦИИ

- Multi-Statement
- возвращает table
- несколько команд

ФУНКЦИИ

```
create function FACULTY_REPORT(@c int)
returns @fr table
(
    [Факультет] varchar(50),
    [Количество кафедр] int,
    [Количество групп] int,
    [Количество студентов] int,
    [Количество специальностей] int
)
```

```
as begin
    declare cc cursor static for
        select FACULTY from FACULTY
        where dbo.COUNT_STUDENTS(FACULTY, default)> @c;
    declare @f varchar(30);
    open cc;
    fetch cc into @f;
    while @@fetch_status = 0
    begin
        insert @fr values(
            @f,
            (select count(PULPIT) from PULPIT where FACULTY = @f),
            (select count(IDGROUP) from GROUPS where FACULTY = @f),
            dbo.COUNT_STUDENTS(@f, default),
            (select count(PROFESSION) from PROFESSION where FACULTY = @f)
        );
        fetch cc into @f;
    end;
    return;
end;
```

ФУНКЦИИ

```
select * from FACULTY_REPORT(28)
```

Факультет	Количество кафедр	Количество групп	Количество студентов	Количество специальностей
ИДиП	5	11	64	3
ИЗФ	3	5	29	2

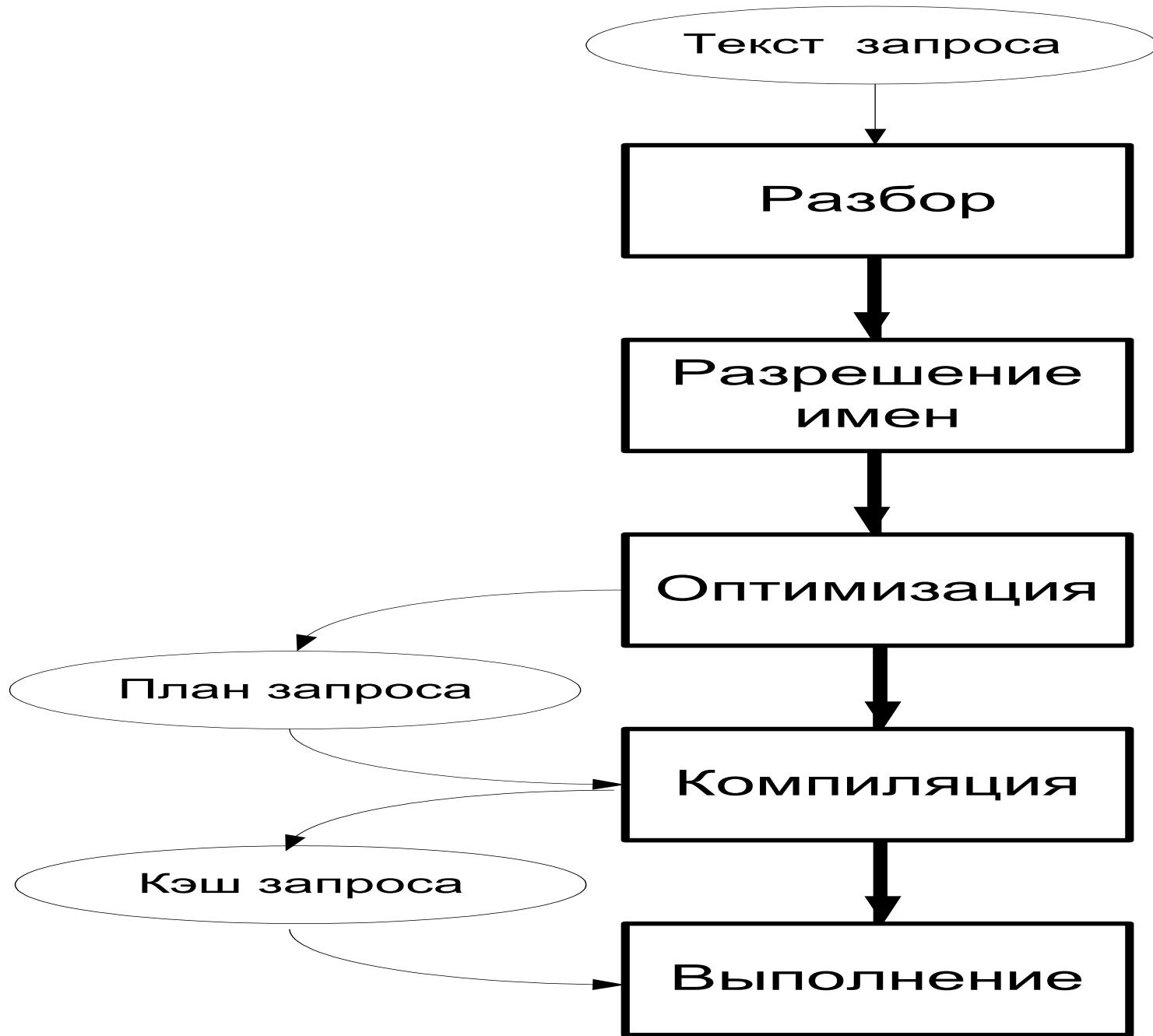
Хранимые процедуры

- DROP FUNCTION
- ALTER FUNCTION

Вопросы?

БАЗЫ ДАННЫХ

Лекция 12 Индексы



Обработка SQL-запроса

- Разбор
- Разрешение имен
- Оптимизация
- Компиляция
- Выполнение

Синтаксический разбор

- Синтаксический разбор текста запроса для проверки его на соответствие правилам языка.

Разрешение имен

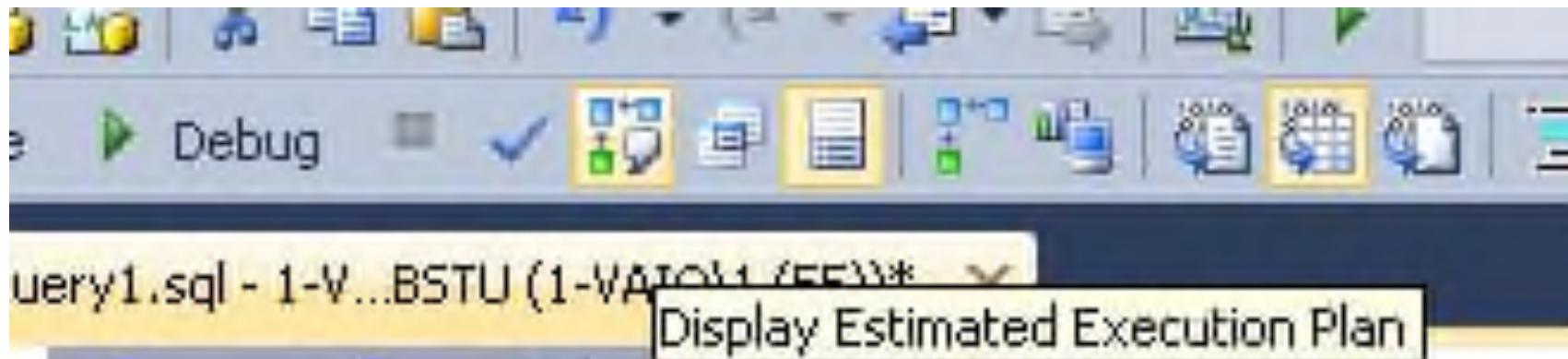
- Проверка наличия используемых в запросе объектов Бд:
 - Таблиц
 - Представлений
 - Столбцов
 - Пользовательских и встроенных функций и пр.

Оптимизация

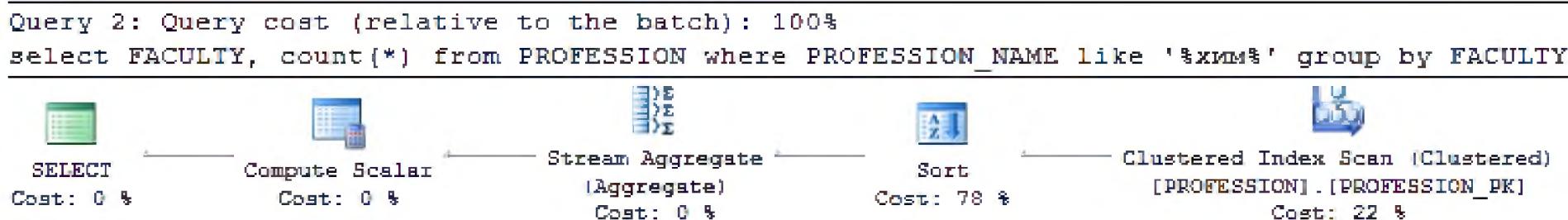
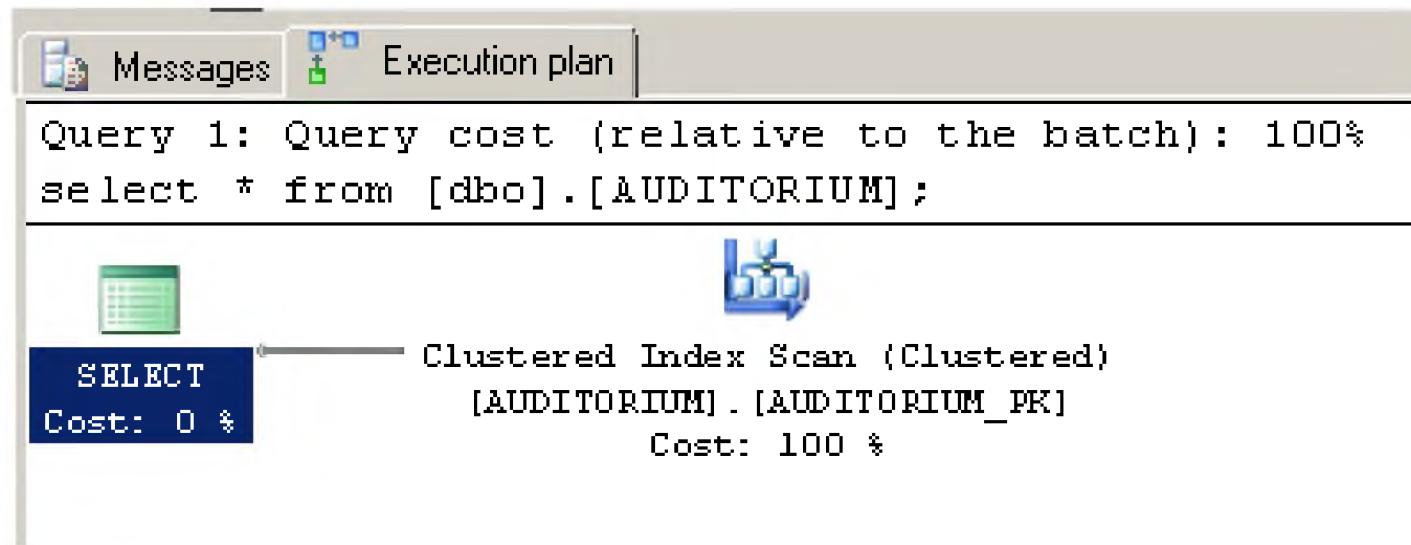
- Специальная компоненте сервера – оптимизатор
- Основная задача оптимизатора – построение плана запроса
- План запроса представляет собой алгоритм выполнения SQL-запроса

План запроса

- **Display Estimated Execution Plan** - Показать предполагаемый план выполнения



План запроса



План запроса

- Для каждого шага вычисляется стоимость – величина, пропорциональная продолжительности выполнения шага
- Суммарная стоимость шагов плана составляет стоимость всего запроса
- Задача – минимизация общей стоимости запроса

Компиляция

- Откомпилированный план запроса помещается в специальную область памяти, называемую библиотечным кэшем
- Кэш используется для ускорения выполнения будущих аналогичных запросов

Выполнение

- Откомпилированный план выполняется сервером СУБД

Индексы

- Индекс представляет собой отдельную физическую структуру данных, которая позволяет получать быстрый доступ к одной или нескольким строкам данных

Индексы

- Индексы сохраняются в страницах индексов
- Для каждой индексируемой строки имеется элемент индекса, который сохраняется на странице индексов
- Каждый элемент индекса состоит из ключа индекса и указателя

Индексы

- Индексы создаются по сбалансированному дереву B+
- B+-дерево имеет древовидную структуру, в которой все листья находятся на расстоянии одинакового количества уровней от вершины дерева
- Это свойство поддерживается при добавлении или удалении данных в индексированном столбце

Индексы

- CREATE
- ALTER
- DROP

Индексы

- Индекс всегда связан с таблицей или с подмножеством столбцов таблицы

Индексы

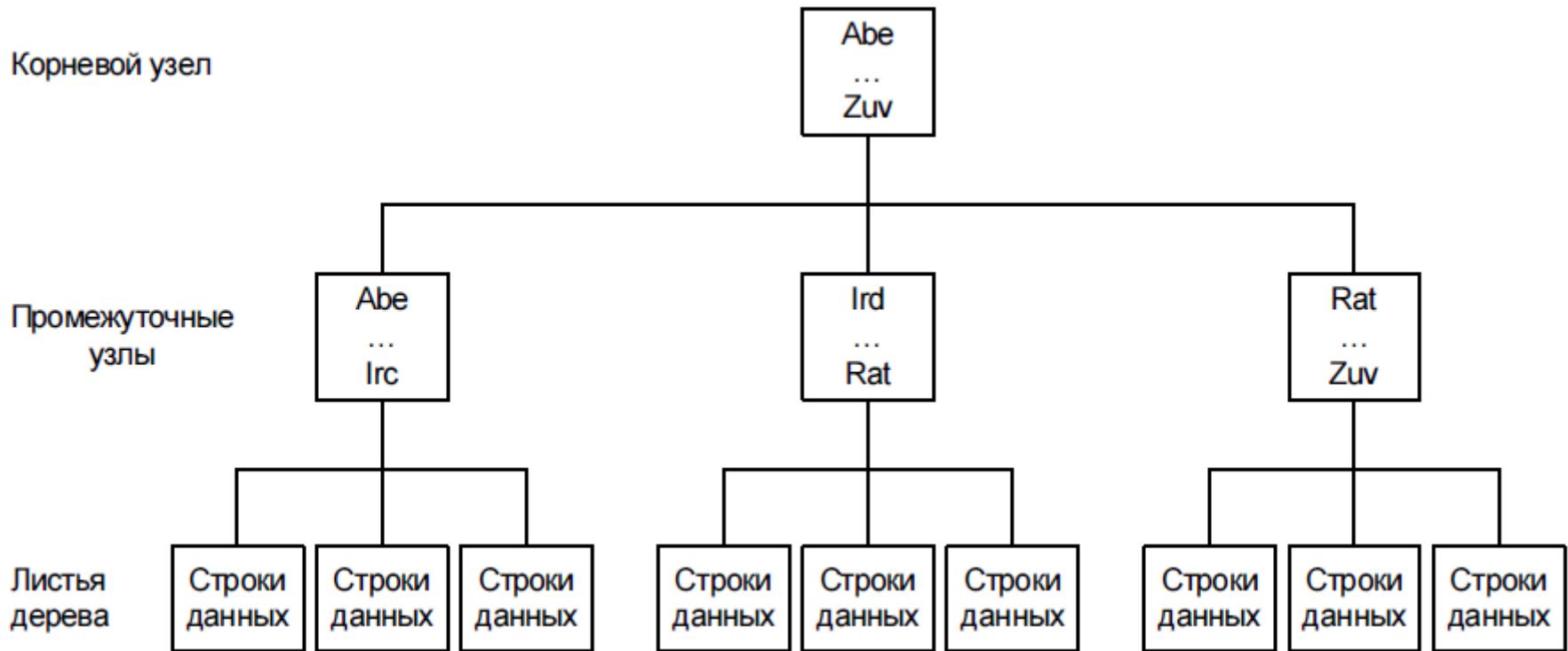
- Кластеризованные индексы
- Некластеризованные индексы

Кластеризованный индекс

- Определяет физический порядок данных в таблице
- Может только один для одной таблицы
- Таблица перестраивается в порядке индекса
- Листья дерева индекса содержат страницы данных

Кластеризованный индекс

Корневой узел



Кластеризованный индекс

- Создается по умолчанию для каждой таблицы, для которой определен первичный ключ
- Уникальный – в столбце, для которого определен кластеризованный индекс, каждое значение данных может встречаться только один раз
- Если кластеризованный индекс создается для столбца, содержащего повторяющиеся значения, СУБД принудительно добавляет четырехбайтовый идентификатор к строкам, содержащим дубликаты значений

Индекс

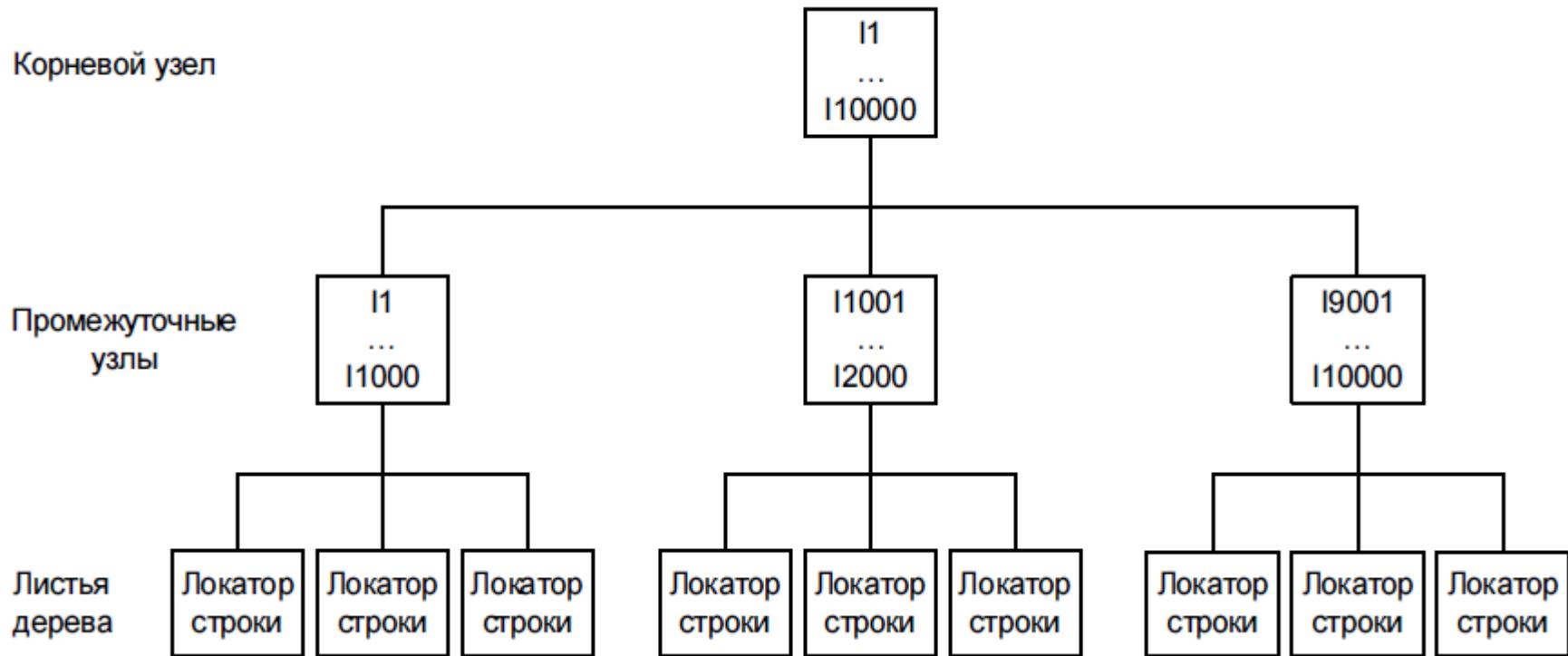
- Кластеризованная таблица – таблица с кластеризованным индексом
- Куча (heap) – таблица без кластеризованного индекса

Некластеризованный индекс

- физически находится отдельно от таблицы
- страницы листьев состоят из ключей индекса и закладок
- может быть несколько для одной таблицы
- не изменяет физическое упорядочивание строк таблицы

Некластеризованный индекс

Корневой узел



Некластеризованный индекс

- Если есть кластеризованный индекс, то закладка некластеризованного индекса показывает B+-дерево кластеризованного индекса таблицы
- Если нет кластеризованного индекса, закладка идентична RID — Row Identifier, состоящего из:
 - Адреса файла, в котором хранится таблица,
 - Адреса физического блока (страницы), в котором хранится строка,
 - Смещения строки в странице.

Некластеризованный индекс

- Поиск данных с использованием некластеризованного индекса в зависимости от типа таблицы:
- Куча — прохождение при поиске по структуре некластеризованного индекса, после чего строка извлекается, используя идентификатор строки.
- Кластеризованная таблица — прохождение при поиске по структуре некластеризованного индекса, после чего следует прохождение по соответствующему кластеризованному индексу.

Индексы

- **SP_HELPINDEX** - получить перечень индексов, связанных с заданной таблицей

```
exec SP_HELPINDEX 'TEACHER'
```

index_name	index_description	index_keys
TEACHER_PK	clustered, unique, primary key located on FG1	TEACHER

Свойства индексов

- Индекс может иметь максимум 900 байтов и не более 16 столбцов
- Разрешено максимум 249 некластеризованных индексов для таблицы
- В UNIQUE составном индексе - однозначная комбинация значений всех столбцов каждой строки
- Если UNIQUE не указывается, то повторяющиеся значения разрешаются
- Параметр NONCLUSTERED по умолчанию

Индексы

- Таблицы
 - + Системные таблицы
 - + dbo.AUDITORIUM
 - + dbo.AUDITORIUM_TYPE
 - + dbo.CURRICULUM
 - + dbo.DT
 - + dbo.FACULTY
 - + dbo.PROFESSION
 - + dbo.PULPIT
 - + dbo.SUBJECT
 - dbo.SUBSTGROUP
 - Столбцы
 - PK_SUBSTGROUP (PK, char(1), Не NULL)
 - STGROUP (PK, char(2), Не NULL)
 - FACULTY (PK, FK, char(10), Не NULL)
 - YYYY (PK, char(4), Не NULL)
 - PROFESSION (FK, char(20), Не NULL)
 - + Ключи
 - Ограничения
 - Триггеры
 - Индексы
 - (簇) PK_SUBSTGROUP (Кластеризованный)
 - + Статистика

Индексы - пример

```
create table #EXPLORE
(
    TKEY    int,
    CC      int identity(1,1),
    TFIELD varchar(100)
);
go
-- добавление в таблицу 10000 строк
set nocount on;      -- не выводить сообщения о вводе строк
declare @i int = 0;
while  @i < 10000
begin
    insert #EXPLORE(TKEY, TFIELD) values(floor(30000*RAND())), replicate('строка ', 10));
    if (@i%100 = 0) print @i;  -- вывести сообщение
    set @i = @i+1;
end;
go
select count(*) [количество строк] from #EXPLORE;
```

количество строк

10000

Индексы - пример

```
checkpoint;          -- фиксация БД  
dbcc dropcleanbuffers; -- очистить буферный кэш  
go  
select * from #EXPLORE where tkey between 15000 and 25000 order by tkey;
```



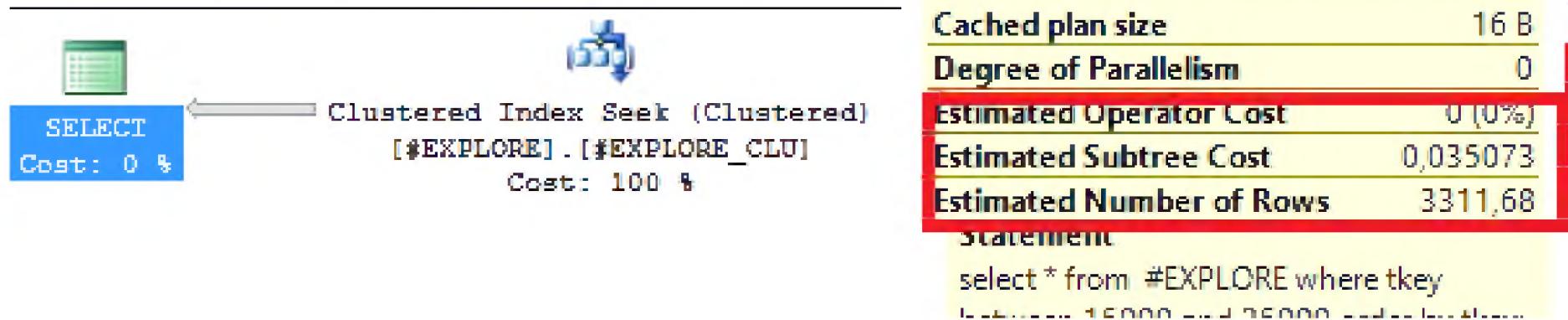
Cached plan size	16 B
Degree of Parallelism	0
Memory Grant	2672
Estimated Operator Cost	0 (0 %)
Estimated Subtree Cost	0,17378
Estimated Number of Rows	3311,68

Statement

```
select * from #EXPLORE where tkey  
between 15000 and 25000 order by tkey;
```

Индексы - пример

```
checkpoint;          -- фиксация ВД  
dbcc dropcleanbuffers; -- очистить буферный кэш  
go  
create clustered index #EXPLORE_CLU on #EXPLORE(tkey asc) -- создать индекс  
go  
select * from #EXPLORE where tkey between 15000 and 25000 order by tkey;
```



Индексы

- Кластеризованные и некластеризованные
- Уникальные и неуникальные
- Простые и составные
- XML-индексы
- Пространственные индексы

Индексы

```
CREATE TABLE SUBSTGROUP -- студенческая подгруппа
(
    SUBGROUP  CHAR(1)  NOT NULL,  -- подгруппа
    STGROUP   CHAR(2)  NOT NULL,  -- группа
    FACULTY   CHAR(10) NOT NULL, -- факультет
    YYYY      CHAR(4)  NOT NULL, -- год поступления
    PROFESSION CHAR(20) NOT NULL, -- специальность
    CONSTRAINT FK_SUBSTGROUP_FACULTY FOREIGN KEY(FACULTY) REFERENCES FACULTY(FACULTY),
    CONSTRAINT FK_SUBSTGROUP_PROFESSION FOREIGN KEY(PROFESSION) REFERENCES PROFESSION(PROFESSION),
    CONSTRAINT PK_SUBSTGROUP PRIMARY KEY NONCLUSTERED(SUBGROUP, STGROUP, FACULTY, YYY)
);
```

The screenshot shows the object browser in SQL Server Management Studio. A tree view displays the structure of the `dbo.SUBSTGROUP` table. The table has five columns: `SUBGROUP`, `STGROUP`, `FACULTY`, `YYYY`, and `PROFESSION`. The primary key constraint is shown as a non-clustered index covering the columns `(SUBGROUP, STGROUP, FACULTY, YYYY)`. Other visible objects include triggers and statistics.

- Создание таблицы `dbo.SUBSTGROUP`.
- Создание индекса `PK_SUBSTGROUP` (уникальный, некластеризованный).
- Создание ограничения `FK_SUBSTGROUP_FACULTY`.
- Создание ограничения `FK_SUBSTGROUP_PROFESSION`.
- Создание ограничения `PK_SUBSTGROUP`.

Индексы

```
create clustered index YYYY on SUBSTGROUP (YYYY) ;
```

The screenshot shows the Object Explorer pane of SQL Server Management Studio. A tree view displays the structure of the 'SUBSTGROUP' table:

- dbo.SUBSTGROUP**:
 - Столбцы**:
 - SUBGROUP (PK, char(1), Не NULL)
 - STGROUP (PK, char(2), Не NULL)
 - FACULTY (PK, FK, char(10), Не NULL)
 - YYYY (PK, char(4), Не NULL)
 - PROFESSION (FK, char(20), Не NULL)
 - Ключи**
 - Ограничения**
 - Триггеры**
 - Индексы**:
 - PK_SUBSTGROUP (уникальный, некластеризованный)
 - YYYY (Кластеризованный)**
 - Статистики**

Индексы

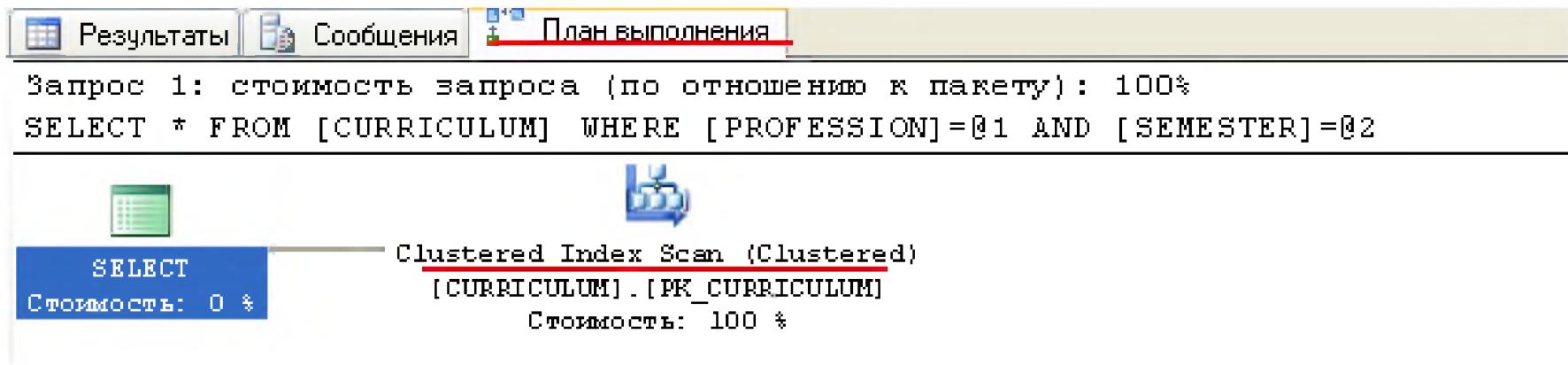
```
create index PROFESSION_SEMESTER on CURRICULUM(PROFESSION, SEMESTER);
```

The screenshot shows a tree view of a database schema. At the top level, there is a folder icon labeled 'Таблицы' (Tables). Under it, there is a plus sign icon followed by a folder icon labeled 'Системные таблицы' (System tables). Below that are two table icons: 'dbo.AUDITORIUM' and 'dbo.AUDITORIUM_TYPE'. Under the 'Таблицы' node, there is a minus sign icon followed by a folder icon labeled 'dbo.CURRICULUM'. This table is currently selected, as indicated by a dark blue selection bar at the bottom of its node. Under 'dbo.CURRICULUM', there is a minus sign icon followed by a folder icon labeled 'Столбцы' (Columns). Inside this folder, there are seven entries, each preceded by a key icon: 'APPROVE (PK, date, Не NULL)', 'PROFESSION (PK, FK, char(20), Не NULL)', 'SUBJECT (PK, FK, char(10), Не NULL)', 'SEMESTER (PK, tinyint, Не NULL)', 'TLESSON (PK, FK, char(10), Не NULL)', 'HOURS (smallint, NULL)', and 'HALT (date, NULL)'. Below the 'Столбцы' folder, there are four more folders with plus signs: 'Ключи' (Keys), 'Ограничения' (Constraints), 'Триггеры' (Triggers), and 'Индексы' (Indexes). The 'Индексы' folder is expanded, showing two index entries: 'PK_CURRICULUM (Кластеризованный)' and 'PROFESSION_SEMESTER (не уникальный, некластеризованный)'. At the very bottom of the tree, there is a plus sign icon followed by a folder icon labeled 'Статистика' (Statistics).

- Таблицы
 - + Системные таблицы
 - + dbo.AUDITORIUM
 - + dbo.AUDITORIUM_TYPE
 - dbo.CURRICULUM
 - Столбцы
 - APPROVE (PK, date, Не NULL)
 - PROFESSION (PK, FK, char(20), Не NULL)
 - SUBJECT (PK, FK, char(10), Не NULL)
 - SEMESTER (PK, tinyint, Не NULL)
 - TLESSON (PK, FK, char(10), Не NULL)
 - HOURS (smallint, NULL)
 - HALT (date, NULL)
 - + Ключи
 - + Ограничения
 - + Триггеры
 - Индексы
 - (PK_CURRICULUM (Кластеризованный)
 - (PROFESSION_SEMESTER (не уникальный, некластеризованный))
 - + Статистика

Индексы

```
select * from CURRICULUM where PROFESSION = '1-40 01 02' and SEMESTER = 3
```



Индексы

Clustered Index Scan (Clustered)

Просмотр всего кластеризованного индекса или его части.

Физическая операция	Clustered Index Scan
Логическая операция	Clustered Index Scan
Фактическое количество строк	7
Предполагаемая стоимость операций ввода-вывода	0,003125
Предполагаемая стоимость процессного ресурса	0,0002153
Количество выполнений	1
Предполагаемое количество выполнений	1
Предполагаемая стоимость оператора	0,0033403 (100%)
Предполагаемая стоимость поддерева	0,0033403
Предполагаемое количество строк	7
Предполагаемый размер строки	56 Б
Фактическое число повторных привязок	0
Фактическое число сбросов на начало	0
Отсортировано	False
Идентификатор узла	0

Предикат

[Smelow].[dbo].[CURRICULUM].[SEMESTER]=(3) AND [Smelow].[dbo].[CURRICULUM].[PROFESSION]='1-40 01 02'

Объект

[Smelow].[dbo].[CURRICULUM].[PK_CURRICULUM]

Список столбцов

[Smelow].[dbo].[CURRICULUM].APPROVE; [Smelow].[dbo].[CURRICULUM].PROFESSION; [Smelow].[dbo].[CURRICULUM].SUBJECT; [Smelow].[dbo].[CURRICULUM].SEMESTER; [Smelow].[dbo].[CURRICULUM].TLESSON; [Smelow].[dbo].[CURRICULUM].HOURS; [Smelow].[dbo].[CURRICULUM].HALT

Индексы

```
checkpoint;          -- фиксация БД  
dbcc dropcleanbuffers; -- очистить буферный кэш  
go  
select * from #EXPLORE where tkey between 15000 and 25000 order by tkey;
```



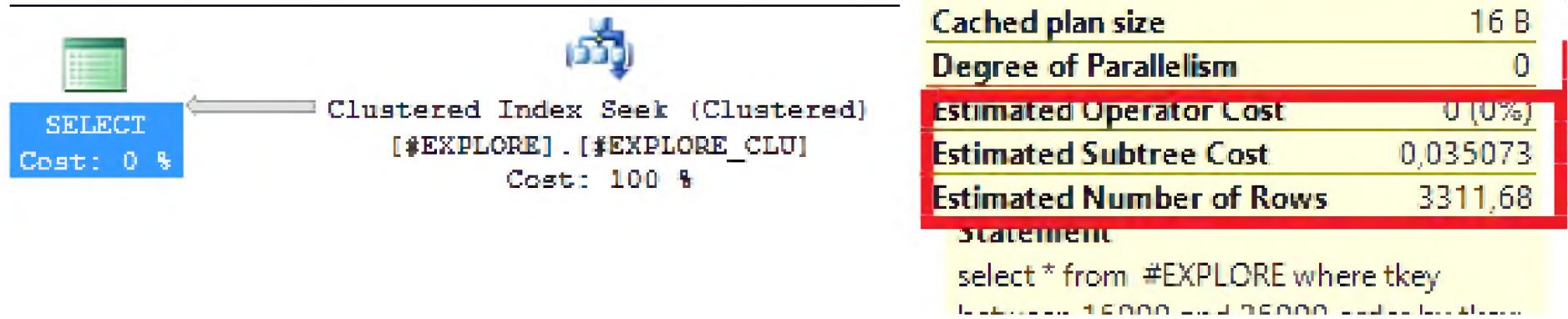
Cached plan size	16 B
Degree of Parallelism	0
Memory Grant	2672
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,17378
Estimated Number of Rows	3311,68

Statement

```
select * from #EXPLORE where tkey  
between 15000 and 25000 order by tkey;
```

Индексы

```
checkpoint; -- фиксация БД  
dbcc dropcleanbuffers; -- очистить буферный кэш  
go  
create clustered index #EXPLORE_CLU on #EXPLORE(tkey asc) -- создать индекс  
go  
select * from #EXPLORE where tkey between 15000 and 25000 order by tkey;
```



Неуникальные индексы

```
select t.cc [количество значений], count(*) [количество случаев]
  from (
    select TKEY, count(*) cc
      from #EXPLORE group by TKEY having count(*) > 1
    ) t
 group by t.cc
 order by [количество случаев] desc
```

количество значений	количество случаев
2	1226
3	142
4	7

Неуникальные индексы

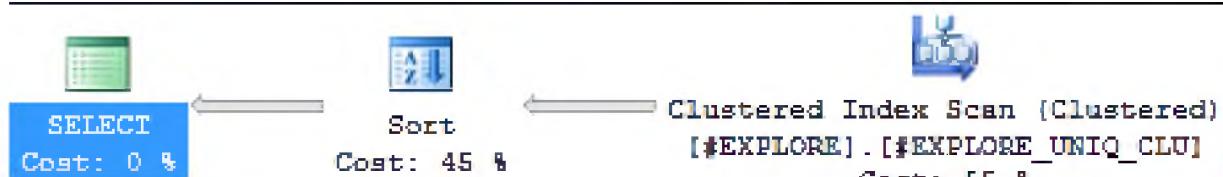
```
drop index #EXPLORE_CLU on #EXPLORE
go
---- создание уникального кластеризованного индекса
create unique clustered index #EXPLORE_UNIQ_CLU on #EXPLORE(tkey asc) -- создать индекс
```

```
Msg 1505, Level 16, State 1, Line 2
The CREATE UNIQUE INDEX statement terminated because a duplicate
key was found for the object name 'dbo.#EXPLORE_00000000001E' and the index name '#EXPLORE_UNIQ_CLU'.
The duplicate key value is (4).
The statement has been terminated.
```

Неуникальные индексы

---- создание уникального конструированного индекса

```
create unique clustered index #EXPLORE_UNIQ_CLU on #EXPLORE(CC asc) -- создать индекс  
go  
select * from #EXPLORE where TKEY between 15000 and 25000 order by TKEY;
```



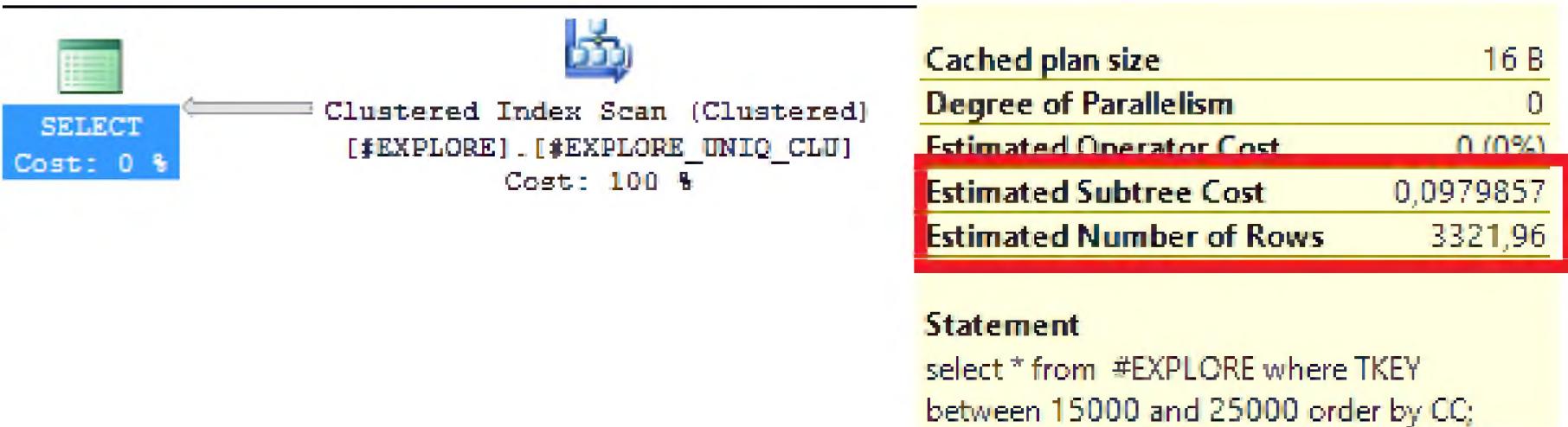
Cached plan size	16 B
Degree of Parallelism	0
Memory Grant	2672
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,178768
Estimated Number of Rows	3321,96

Statement

```
select * from #EXPLORE where TKEY  
between 15000 and 25000 order by TKEY;
```

Неуникальные индексы

```
select * from #EXPLORE where TKEY between 15000 and 25000 order by CC;
```



Неуникальные индексы

```
select * from #EXPLORE where CC between 4500 and 6500 order by CC;
```

	Clustered Index Seek (Clustered)	
	[#EXPLORE].[#EXPLORE_UNIQ_CLU]	
SELECT Cost: 0 %	Cost: 100 %	
		Cached plan size 16 B
		Degree of Parallelism 0
		Estimated Operator Cost 0 (0%)
		Estimated Subtree Cost 0,0217795
		Estimated Number of Rows 2001,07

Statement

```
select * from #EXPLORE where CC between  
4500 and 6500 order by CC;
```

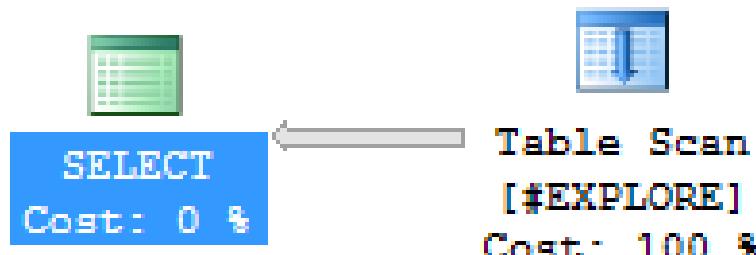
Составной индекс

```
create index #EXPLORE_NONCLU on #EXPLORE (TKEY, CC)
go
exec SP_HELPINDEX '#EXPLORE'
```

index_name	index_description	index_keys
#EXPLORE_NONCLU	nonclustered located on PRIMARY	TKEY, CC

Некластеризованный составной индекс

```
select * from #EXPLORE where TKEY > 1500 and CC < 4500;
```

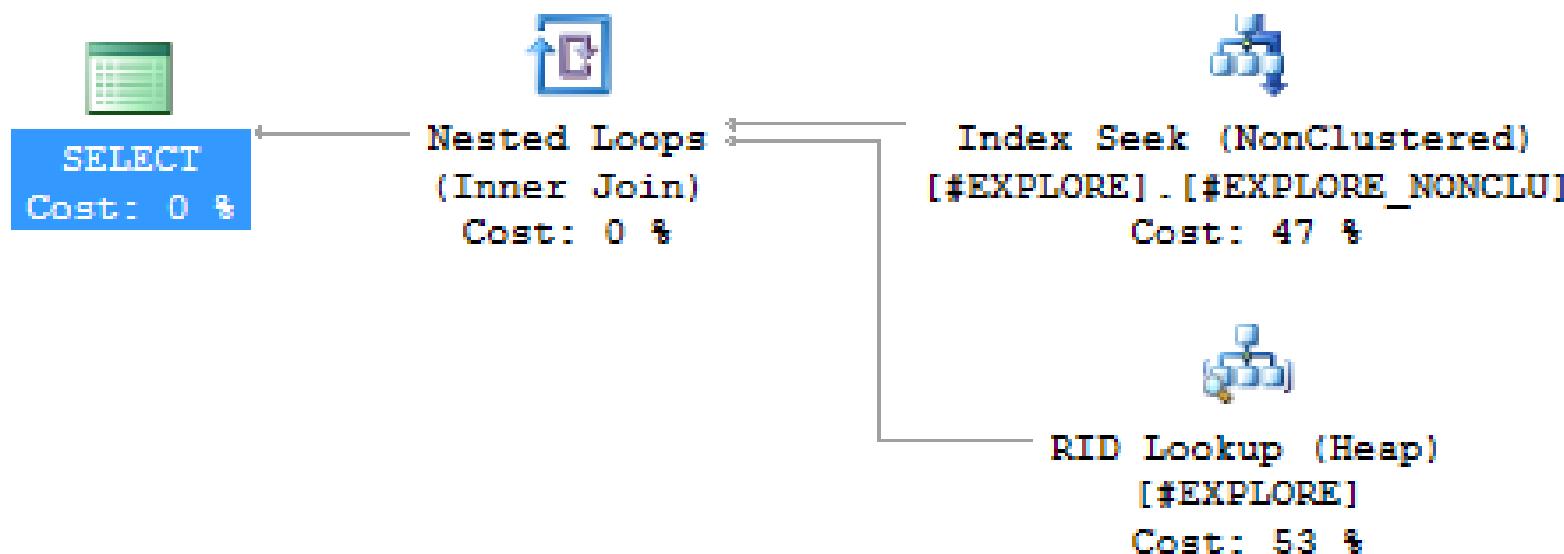


```
select * from #EXPLORE order by TKEY, CC;
```



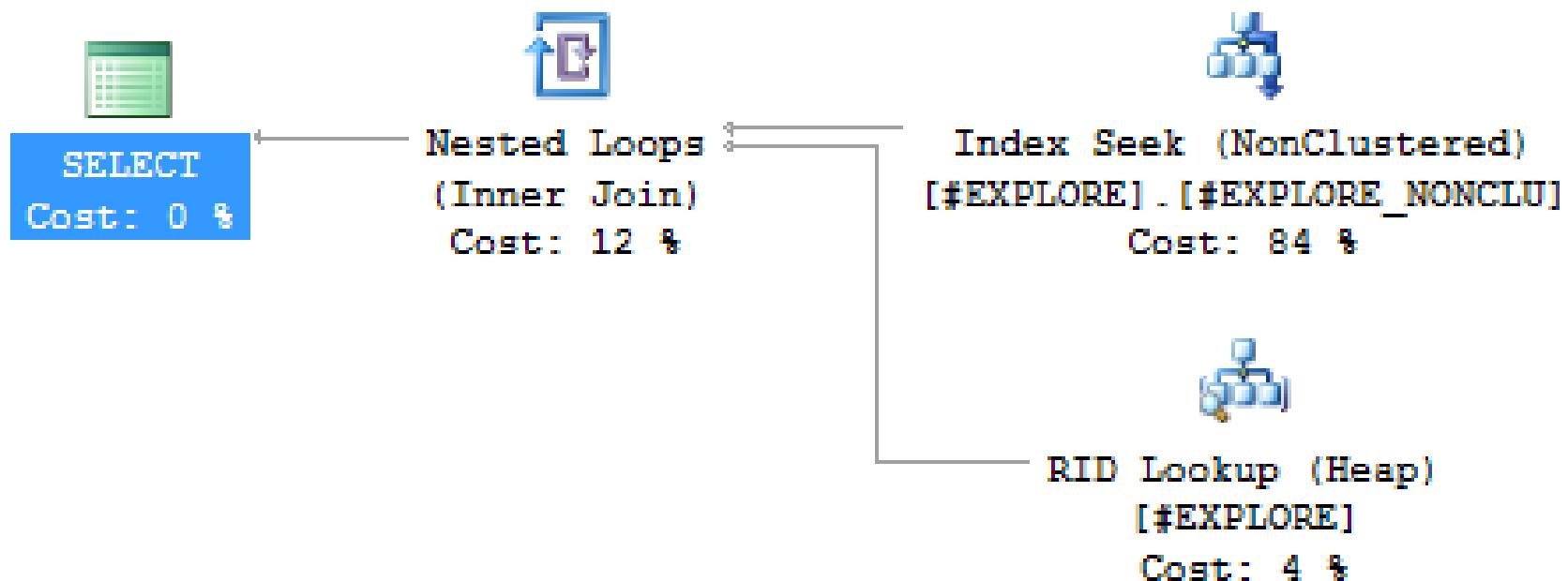
Некластеризованный составной индекс

```
select * from #EXPLORE where TKEY = 556 and CC > 3
```



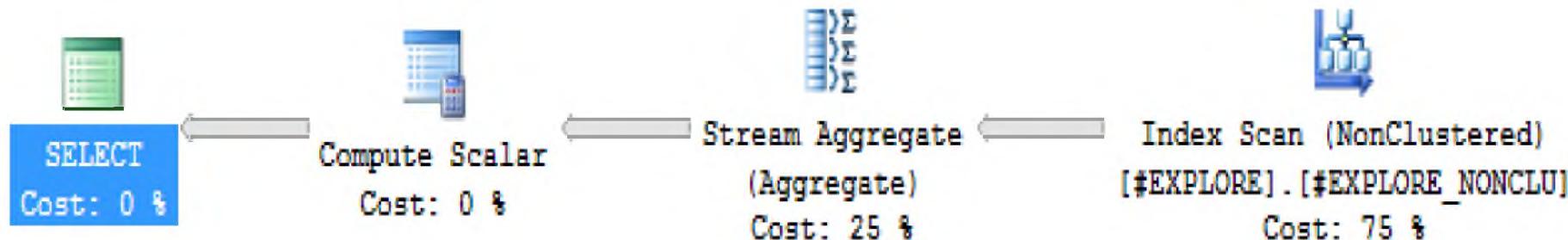
Некластеризованный составной индекс

```
select * from #EXPLORE where TKEY > 1000 and CC = 2222
```



Составной индекс

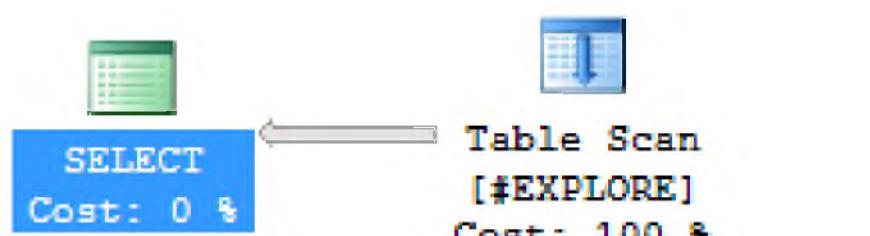
```
select max(TKEY), max(CC), count(*) from #EXPLORE group by TKEY, CC;
```



Индекс

```
drop index #EXPLORE_NONCLU on #EXPLORE; -- удаление индекса
go
create index #EXPLORE_TKEY on #EXPLORE (TKEY);
go
create index #EXPLORE_CC on #EXPLORE (CC);
go

select * from #EXPLORE where TKEY > 1500 and CC < 4500;
select * from #EXPLORE where TKEY > 1500;
select * from #EXPLORE where CC < 4500;
```



Индекс

```
select TKEY from #EXPLORE where TKEY > 1500;
```



SELECT
Cost: 0 %



Index Seek (NonClustered)
[#EXPLORE].[#EXPLORE_TKEY]
Cost: 100 %

```
select CC          from #EXPLORE where TKEY > 1500;  
select TFIELD     from #EXPLORE where TKEY > 1500;
```



SELECT
Cost: 0 %



Table Scan
[#EXPLORE]
Cost: 100 %

Индекс покрытия

- Включение всех столбцов запроса в индекс может значительно повысить производительность запроса
- Фильтруемые столбцы должны быть первыми ключевыми столбцами в индексе

```
create index #EXPLORE_WHERE_I on #EXPLORE (TKEY) include (CC)  
WHERE (TKEY >= 15000 and CC < 45000);
```

INCLUDE

- INCLUDE позволяет указать столбцы, которые добавляются к страницам листьев некластеризованного индекса
- Имена столбцов в списке INCLUDE не должны повторяться
- Если все столбцы запроса включены в индекс, то оптимизатор запросов может определить местонахождение всех значений столбцов по страницам индекса, не обращаясь к данным в таблице

Фильтрующие индексы

- Фильтрующий индекс создается только для строк таблицы, которые удовлетворяют логическому условию
- Если множество строк, выбранное WHERE-выражением SELECT-запроса, является подмножеством множества индексируемых фильтрующим индексом строк, оптимизатор будет его применять в плане запроса
- Можно объединять с индексами покрытия

Фильтрующие индексы

```
use TEMPDB  
go  
create index #EXPLORE_WHERE on #EXPLORE(TKEY) WHERE (TKEY >= 15000 and TKEY <= 20000)  
go  
exec SP_HELPINDEX '#EXPLORE';
```

index_name	index_description	index_keys
#EXPLORE_TKEY	nonclustered located on PRIMARY	TKEY
#EXPLORE_WHERE	nonclustered located on PRIMARY	TKEY

```
create index #EXPLORE_WHERE_I on #EXPLORE(TKEY)  
include(CC) WHERE (TKEY >= 15000 and CC < 45000);
```

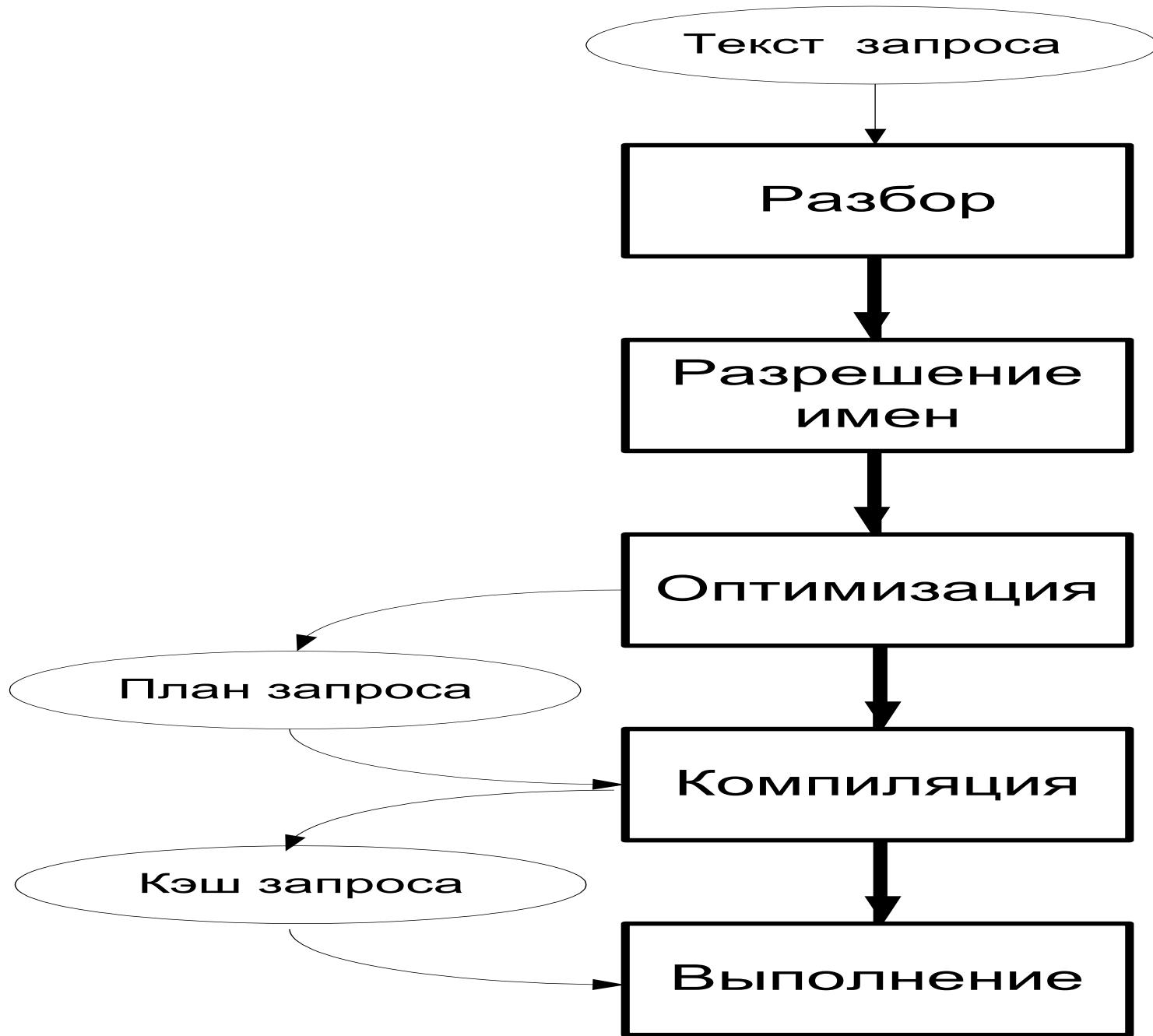
Применение индексов

- Индекс занимает определенный объем дискового пространства
- Индекс используется для выборки данных
- Для вставки и удаления данных необходимо обслуживания индекса
- Чем больше индексов имеет таблица, тем больше объем работы по их реорганизации
- **Правило:**
- Выбирать индексы для частых запросов
- Затем оценивать их использование
- Не индексировать столбцы, по которым нет поиска

Вопросы?

БАЗЫ ДАННЫХ

Лекция 9
Индексы



Индексы

- Индекс представляет собой физическую структуру данных, которая позволяет получать быстрый доступ к одной или нескольким строкам данных таблицы

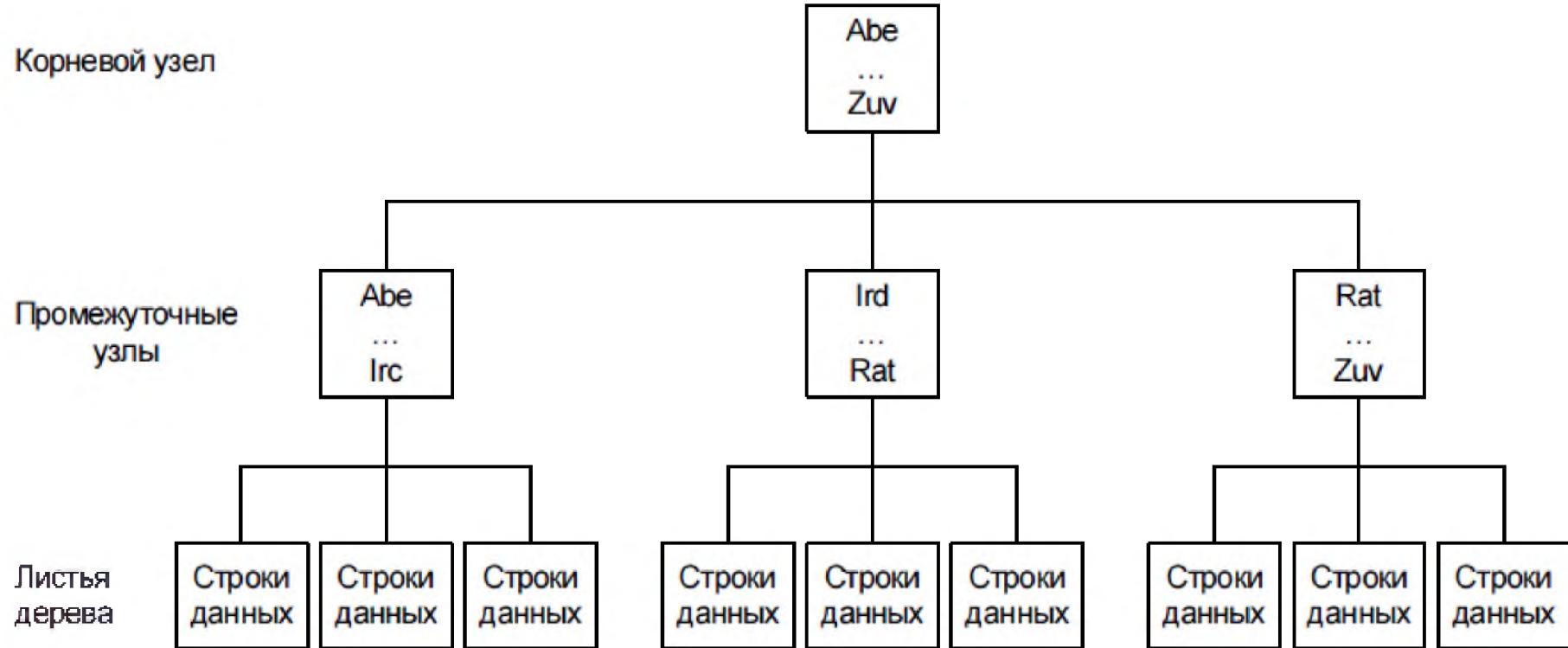
Индексы

- Индексы сохраняются в страницах индексов
- Для каждой индексируемой строки имеется элемент индекса, который сохраняется на странице индексов
- Каждый элемент индекса состоит из ключа индекса и указателя
- SP_HELPINDEX
- sys.indexes
- sys.index_columns

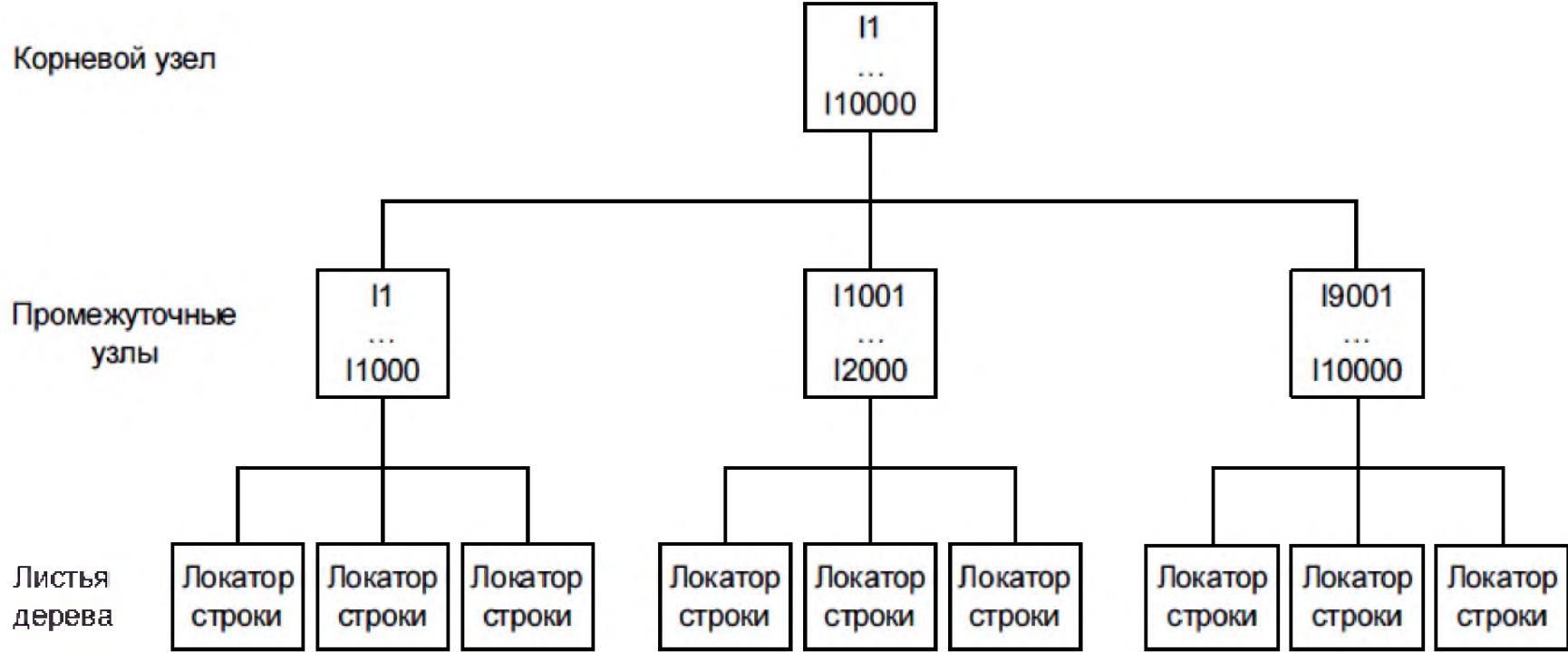
Индексы

- Индексы создаются по сбалансированному дереву B+
- Это свойство поддерживается при добавлении или удалении данных в индексированном столбце

Корневой узел



Корневой узел



Кластеризованный индекс

- Определяет физический порядок данных в таблице
- Может только один для одной таблицы
- Таблица перестраивается в порядке индекса
- Листья дерева индекса содержат страницы данных
- Создается по умолчанию для каждой таблицы, для которой определен РК или UNIQUE

Индекс

- Кластеризованная таблица – таблица с кластеризованным индексом
- Куча (heap) – таблица без кластеризованного индекса

Некластеризованный индекс

- физически находится отдельно от таблицы
- страницы листьев состоят из ключей индекса и закладок
- может быть несколько для одной таблицы
- не изменяет физическое упорядочивание строк таблицы
- В листе находится RID — Row Identifier:
 - Адрес файла таблицы
 - Адрес страницы для строки
 - Смещение строки в странице

Поиск в некластеризованном индексе

- Куча — прохождение по структуре индекса, после чего строка извлекается по идентификатору строки
- Кластеризованная таблица — прохождение при поиске по структуре индекса, после чего следует прохождение по соответствующему кластеризованному индексу

Применение индексов

- Индекс занимает определенный объем дискового пространства
- Для вставки и удаления данных необходимо обслуживание индекса
- Чем больше индексов имеет таблица, тем больше объем работы по их реорганизации
- **Правила:**
 - Выбирать индексы для частых запросов
 - Затем оценивать их использование
 - Не индексировать столбцы, по которым нет поиска

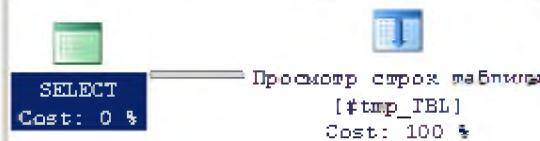
Применение индексов

```
CREATE TABLE #tmp_TBL
(
    tb_1 int IDENTITY(1,1),
    tb_2 int,
    tb_3 varchar(100))

SET NOCOUNT ON;
DECLARE @I INT = 0;
WHILE (@I<20000)
BEGIN
    INSERT #tmp_TBL (tb_2, tb_3) VALUES (FLOOR(30000*rand()), REPLICATE('STR',10));
    SET @I=@I+1;
END;

SELECT * FROM #tmp_TBL
WHERE tb_1 BETWEEN 100 AND 900 AND tb_2<4500;
```

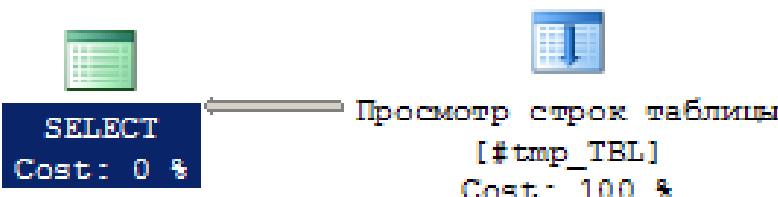
```
SELECT * FROM #tmp_TBL WHERE tb_1 BETWEEN 100 AND 900 AND tb_2<4500
```



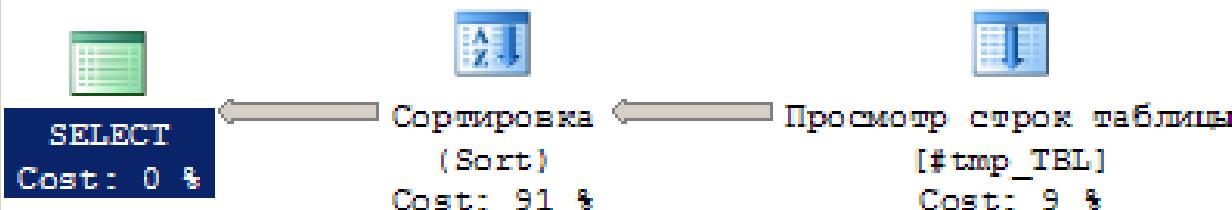
Применение составного индекса - нет

```
CREATE INDEX #tmp_TBL_NOCL ON #tmp_TBL(tb_1, tb_2);
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM #tmp_TBL WHERE tb_1 BETWEEN 100 AND 900 AND tb_2<4500
Missing Index (Impact 63.5242): CREATE NONCLUSTERED INDEX [<Name of Missing
```



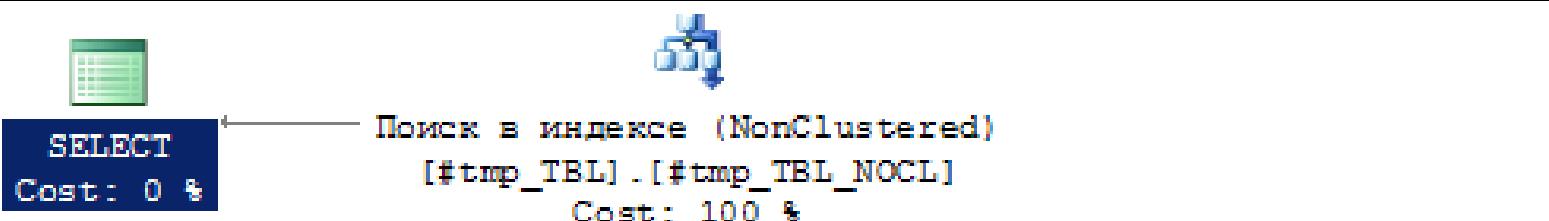
```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM #tmp_TBL ORDER BY tb_1
```



Применение составного индекса - есть

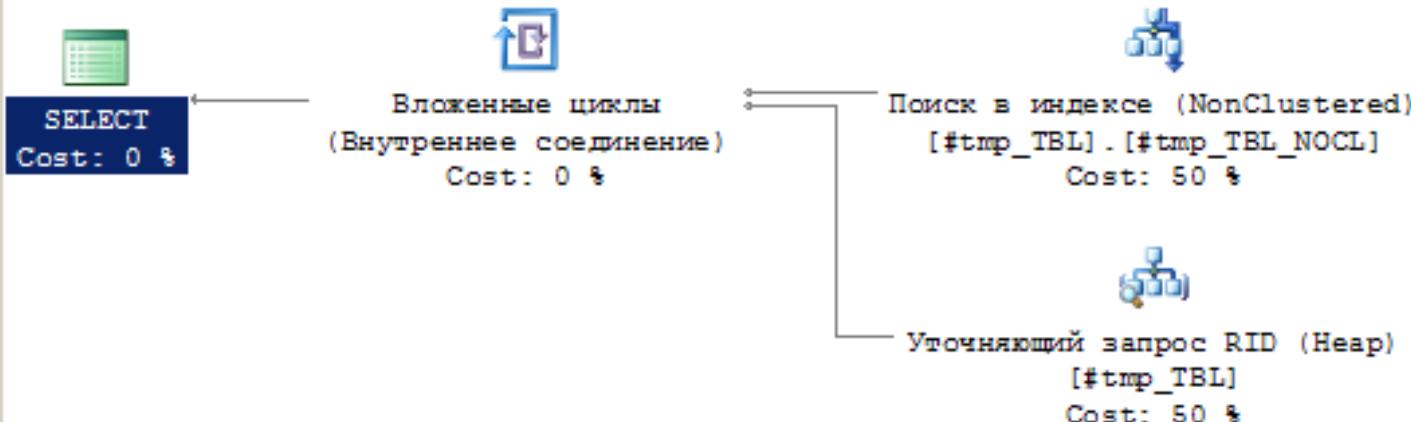
Query 1: Query cost (relative to the batch): 100%

```
SELECT tb_1, tb_2 FROM #tmp_TBL WHERE tb_1=666 AND tb_2=4500
```



Query 1: Query cost (relative to the batch): 100%

```
SELECT * FROM #tmp_TBL WHERE tb_1=666 AND tb_2=4500
```



Фильтрующие индексы

- Фильтрующий индекс создается только для строк таблицы, которые удовлетворяют логическому условию
- Если множество строк, выбранное WHERE-выражением SELECT-запроса, является подмножеством множества индексируемых фильтрующим индексом строк, оптимизатор будет его применять в плане запроса
- Можно объединять с индексами покрытия

Индекс покрытия

- Включение всех столбцов запроса в индекс может значительно повысить производительность запроса
- Фильтруемые столбцы должны быть первыми ключевыми столбцами в индексе
- Пересоздаем таблицу

INCLUDE

- INCLUDE позволяет указать столбцы, которые добавляются к страницам листьев некластеризованного индекса
- Имена столбцов в списке INCLUDE не должны повторяться
- Если все столбцы запроса включены в индекс, то оптимизатор запросов может определить местонахождение всех значений столбцов по страницам индекса, не обращаясь к данным в таблице

Query 1: Query cost (relative to the batch): 33%

```
SELECT tb_2 FROM #tmp_TBL where tb_2 BETWEEN 5000 AND 19999
```



SELECT
Cost: 0 %

Просмотр строк таблицы
[#tmp_TBL]
Cost: 100 %

Query 2: Query cost (relative to the batch): 33%

```
; SELECT tb_2 FROM #tmp_TBL where tb_2 > 15000 AND tb_2 < 19999
```



SELECT
Cost: 0 %

Просмотр строк таблицы
[#tmp_TBL]
Cost: 100 %

Query 3: Query cost (relative to the batch): 33%

```
; SELECT tb_2 FROM #tmp_TBL where tb_2 = 17999
```

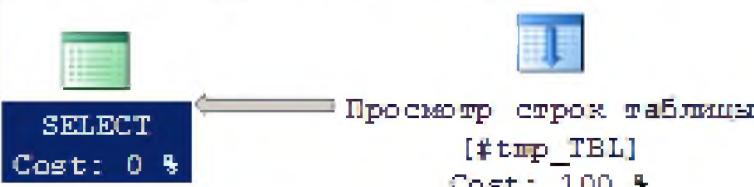


SELECT
Cost: 0 %

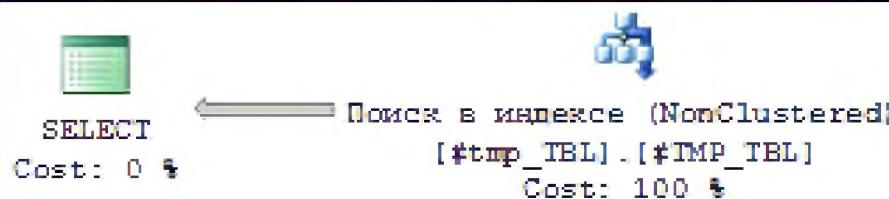
Просмотр строк таблицы
[#tmp_TBL]
Cost: 100 %

```
CREATE INDEX #TMP_TBL ON #tmp_TBL(tb_2)
WHERE tb_2>=15000 AND tb_2<20000;
```

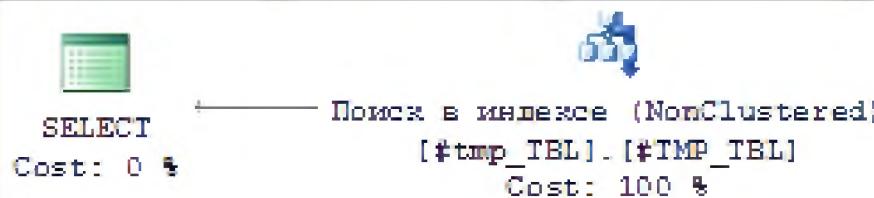
```
Query 1: Query cost (relative to the batch): 92%
SELECT tb_2 FROM #tmp_TBL where tb_2 BETWEEN 5000 AND 19999
```



```
Query 2: Query cost (relative to the batch): 6%
; SELECT tb_2 FROM #tmp_TBL where tb_2 > 15000 AND tb_2 < 19999
```

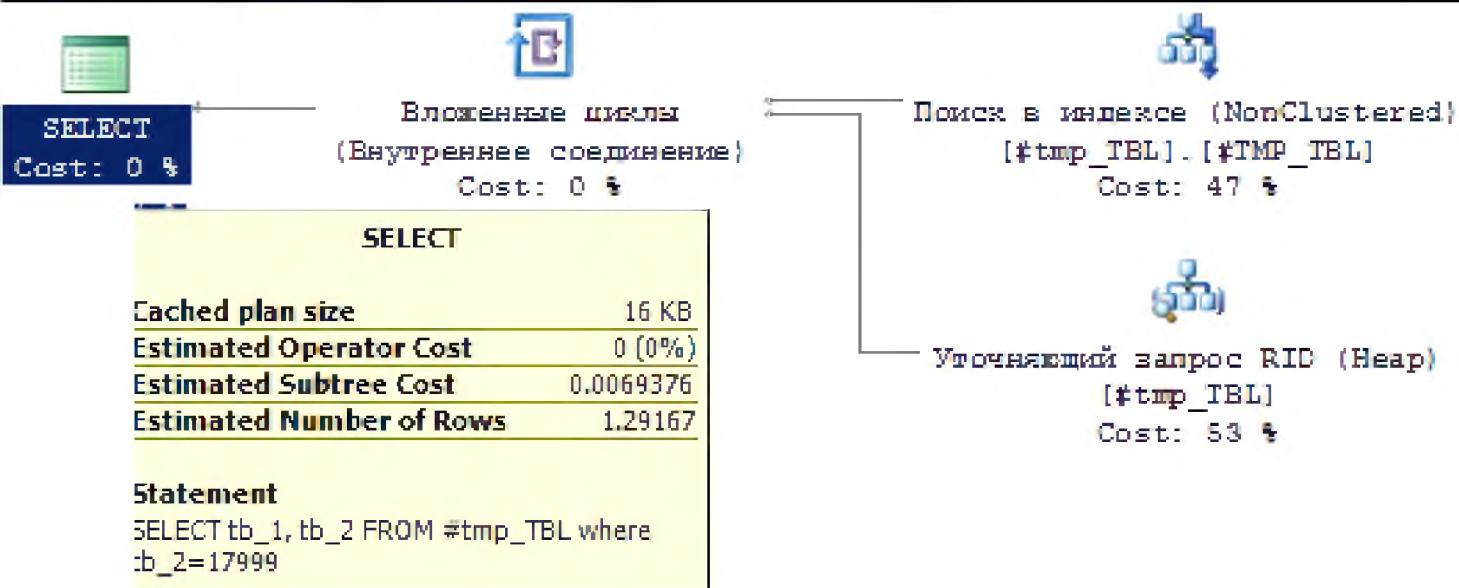


```
Query 3: Query cost (relative to the batch): 2%
; SELECT tb_2 FROM #tmp_TBL where tb_2 = 17999
```



Query 1: Query cost (relative to the batch): 100%

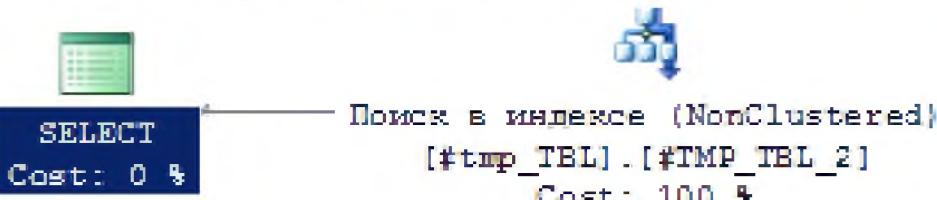
```
SELECT tb_1, tb_2 FROM #tmp_TBL where tb_2=17999
```



```
CREATE INDEX #TMP_TBL_2 ON #tmp_TBL(tb_2)
include (tb_1) WHERE tb_2>=15000 AND tb_2<21000;
```

Query 1: Query cost (relative to the batch): 100%

```
SELECT tb_1, tb_2 FROM #tmp_TBL where tb_2=17999
```



Cached plan size	16 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0032834
Estimated Number of Rows	1.29167

Statement:

```
SELECT tb_1, tb_2 FROM #tmp_TBL where tb_2=17999
```

Оптимизация запроса

- анализ запроса
- выбор индекса
- выбор порядка выполнения операций соединения
- выбор метода выполнения операций соединения

Селективность и плотность

- Селективность запроса – соотношение количества строк, удовлетворяющих условию, к общему количеству строк в таблице
 - Индекс успешно работает при $\leq 5\%$.
 - Не нужен индекс при 80% или более
- Плотность запроса – количество возвращаемых строк запроса

Анализ запроса

- Наличие аргументов поиска
- Использование оператора OR
- Существование критериев соединения

Анализ запроса

- Аргумент поиска — это часть запроса, которая ограничивает промежуточный результирующий набор запроса:
- name = 'Иванов С.П.'
- capacity >= 100
- name = 'Иванов С.П.' AND idgroup = 512

Анализ запроса

- Нельзя использовать в качестве аргументов поиска:
 - Выражение с оператором отрицания NOT
 - <>
 - Выражение
- NOT IN('d1', 'd2')
- aud_no <> 9031
- capacity * 0.6 > 100

Создание индексов

- SELECT ... WHERE column_name – для этого столбца следует создать индекс
- SELECT ... WHERE column_name1 AND column_name2 – создать составной индекс по всем столбцам

Индексы при соединении

- Для каждого соединяемого столбца
- Некластеризованный индекс для столбца внешнего ключа

Выбор индексов

- Оптимизатор проверяет селективность выражения с индексированным столбцом, используя статистические данные
- AUTO_CREATE_STATISTICS ON(OFF)

Фрагментация индекса

- Внутренняя фрагментация - объем данных, хранящихся в каждой странице
- Внешняя фрагментация - нарушение логического порядка страниц

```
CREATE TABLE test (p int);

SET NOCOUNT ON;
DECLARE @I INT = 0;
WHILE (@I<200000)
    BEGIN
        INSERT test (p) VALUES (FLOOR(30000*rand()));
        SET @I=@I+1;
    END;
```

```
CREATE NONCLUSTERED INDEX TMP_p ON test (p)
```

Index Properties - TMP_p

Ready

Select a page

- General
- Options
- Storage
- Filter
- Fragmentation
- Extended Properties

Script Help

Page fullness 68.92 %

Total fragmentation 98.88 %

General

Average row size	23
Depth	3
Forwarded records	0
Ghost rows	0
Index type	NONCLUSTERED INDEX
Leaf-level rows	400000
Maximum row size	30
Minimum row size	16
Pages	1792
Partition ID	1
Version ghost rows	0

Average row size
The average leaf-level row size.

OK Отмена Справка

```

SELECT a.index_id,
       name,
       avg_fragmentation_in_percent,
       fragment_count,
       avg_fragment_size_in_pages
FROM sys.dm_db_index_physical_stats (DB_ID('Test1'),
                                     OBJECT_ID('TMP_p'), NULL, NULL, NULL) AS a
JOIN sys.indexes AS b
ON a.object_id = b.object_id AND a.index_id = b.index_id;

```

	index_id	name	avg_fragmentation_in_percent	fragment_count	avg_fragment_size_in_pages
1	1	PK_PRODUCTS_59EE6463D5BD6110	0	1	1
2	1	PK_OFFICES_6A36566EE58513E5	0	1	1
3	1	PK_SALESREP_1D72C150D274422D	0	1	1
4	1	PK_CUSTOMER_6A01161A279E1D84	0	1	1
5	1	PK_ORDERS_6C473C958C5BCF61	0	1	1
6	0	NULL	0	0	0
7	0	NULL	19.6629213483146	38	36.5473684210526
8	2	TMP_p	98.8839285714286	1792	1

```
ALTER INDEX TMP_p ON test REORGANIZE;
```

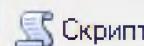
Page fullness		99.45 %
Total fragmentation		1.13 %
General		
index_id	name	avg_fragmentation_in_percent
1	PK_PRODUCTS_59EE6463D5BD6110	0
2	PK_OFFICES_6A36566EE58513E5	0
3	PK_SALESREP_1D72C150D274422D	0
4	PK_CUSTOMER_6A01161A279E1D84	0
5	PK_ORDERS_6C473C958C5BCF61	0
6	NULL	0
7	NULL	19.6629213483146
8	TMP_p	1.12721417069243

```
ALTER INDEX TMP_p ON test REBUILD WITH (ONLINE=OFF);
```

Page fullness		99.81 %
Total fragmentation		0.11 %
General		
index_id	name	avg_fragmentation_in_percent
1	PK_PRODUCTS_59EE6463D5BD6110	0
2	PK_OFFICES_6A36566EE58513E5	0
3	PK_SALESREP_1D72C150D274422D	0
4	PK_CUSTOMER_6A01161A279E1D84	0
5	PK_ORDERS_6C473C958C5BCF61	0
6	NULL	0
7	NULL	19.6629213483146
8	TMP_p	0.112233445566779

Выбор страницы

-  Общие
-  Память
-  Процессоры
-  Безопасность
-  Соединения
-  Параметры базы данных
-  Дополнительно
-  Разрешения



Скрипт



Справка

Коэффициент заполнения индекса, используемый по умолчанию:

Резервное копирование и восстановление

Укажите для SQL Server срок ожидания новой ленты.

Без ограничений

```
CREATE TABLE test (p int);

SET NOCOUNT ON;
DECLARE @I INT = 0;
WHILE (@I<20000)
BEGIN
    INSERT test (p) VALUES (FLOOR(30000*rand()));
    SET @I=@I+1;
END;

CREATE NONCLUSTERED INDEX ind_for_rebuild ON test(p)

SELECT * FROM SYS.dm_db_index_physical_stats(DB_id('Test'), OBJECT_ID('Test'), NULL,NULL,NULL)

ALTER INDEX ind_for_rebuild ON test REORGANIZE

ALTER INDEX ind_for_rebuild ON test REBUILD WITH (FILLFACTOR = 80);
```

Фрагментация

- Заполненность страниц 65,77 %
- Общая фрагментация 67,75 %

Фрагментация

- Заполненность страниц 79,97 %
- Общая фрагментация 0,00 %

Изменение индекса

- FillFactor указывает, сколько листовых страниц индекса заполняется
- При высоком FillFactor больше строк помещается в одну страницу данных, но может стать больше разбиений страниц
- При низком FillFactor на странице данных меньше записей, что снизит число разбиений страниц, но потребуется больше операций чтения, так как данные будут распределены по большему числу страниц

Изменение индекса

- ALTER INDEX
- ALLOW_ROW_LOCKS, ALLOW_PAGE_LOCKS, IGNORE_DUP_KEY
- REBUILD - пересоздание индекса
- REORGANIZE - реорганизация страниц листьев индекса
- DISABLE - отключение индекса

DISABLE

- DISABLE отключает указанный индекс
- Отключенный индекс недоступен, пока он не будет снова включен
- REBUILD
- При отключенном кластеризованном индексе таблицы данные этой таблицы будут недоступны, так как все страницы данных таблицы с кластеризованным индексом хранятся в его листьях дерева

БАЗЫ ДАННЫХ

Лекция 13 Оптимизация запросов

FILLFACTOR

- FILLFACTOR = n задает заполнение в процентах каждой страницы индекса
- При n = 100 нет свободного места для вставки новых строк - только для статических таблиц.
- При n = 0 страницы листьев индекса заполняются полностью, а каждая из промежуточных страниц содержит свободное место для одной записи

PAD_INDEX

- PAD_INDEX указывает, что значение параметра FILLFACTOR применяется как к страницам индекса, так и к страницам данных в индексе

Фрагментация индекса

- Внутренняя фрагментация - объем данных, хранящихся в каждой странице
- Внешняя фрагментация - нарушение логического порядка страниц

sys.dm_db_index_physical_stat

```
DECLARE @db_id INT;
DECLARE @tab_id INT;
DECLARE @ind_id INT;

SET @db_id = DB_ID('TMP1_BSTU');
SET @tab_id = OBJECT_ID('AUDITORIUM');

SELECT avg_fragmentation_in_percent, avg_page_space_used_in_percent
FROM sys.dm_db_index_physical_stats
(@db_id, @tab_id, NULL, NULL, NULL)
```

The screenshot shows a SQL Server Management Studio (SSMS) window. At the top, there is a code editor pane containing the T-SQL script. Below it is a toolbar with a percentage icon and a refresh button. The results pane is visible, showing tabs for 'Results' and 'Messages'. The 'Results' tab is selected, displaying a table with two columns: 'avg_fragmentation_in_percent' and 'avg_page_space_used_in_percent'. A single row is present in the table, with values '0' and 'NULL' respectively.

avg_fragmentation_in_percent	avg_page_space_used_in_percent
0	NULL

Представления индексов

- sys.indexes
- sys.index_columns
- sp_helpindex
- sys.dm_db_index_usage_stats
- sys.dm_db_missing_index_details

Изменение индекса

- ALTER INDEX
- ALLOW_ROW_LOCKS, ALLOW_PAGE_LOCKS, IGNORE_DUP_KEY
- REBUILD - пересоздание индекса
- REORGANIZE - реорганизация страниц листьев индекса
- DISABLE - отключение индекса

REBUILD

- Параметр REBUILD применяется для пересоздания индексов
- ALL - все индексы таблицы

REBUILD

```
alter index #EXPLORE_TKEY on #EXPLORE rebuild with (online = off);
go
```

----- данные о фрагментации -----

```
select name [Индекс], avg_fragmentation_in_percent [Фрагментация (%)]
from sys.dm_db_index_physical_stats(DB_ID(N'TEMPDB'), OBJECT_ID(N'#EXPLORE'), NULL, NULL, NULL) ss
join sys.indexes ii on ss.object_id = ii.object_id and ss.index_id = ii.index_id
where name is not null;
go
```

Индекс	Фрагментация (%)
#EXPLORE_TKEY	0

REORGANIZE

- REORGANIZE задает реорганизацию страниц листьев индекса, чтобы физический порядок страниц совпадал с их логическим порядком — слева направо

REORGINEZE

```
alter index #EXPLORE_TKEY on #EXPLORE reorganize;  
go
```

----- данные о фрагментации -----

```
select name [Индекс], avg_fragmentation_in_percent [Фрагментация (%)]  
from sys.dm_db_index_physical_stats(DB_ID(N'TEMPDB'), OBJECT_ID(N'#EXPLORE'), NULL, NULL, NULL) ss  
join sys.indexes ii on ss.object_id = ii.object_id and ss.index_id = ii.index_id  
where name is not null;
```

Индекс	Фрагментация (%)
#EXPLORE_TKEY	3,94736842105263

DISABLE

- DISABLE отключает указанный индекс
- Отключенный индекс недоступен, пока он не будет снова включен
- REBUILD
- При отключенном кластеризованном индексе таблицы данные этой таблицы будут недоступны, так как все страницы данных таблицы с кластеризованным индексом хранятся в его листьях дерева

DROP INDEX

- DROP INDEX
- Для кластеризованного индекса - потребуется пересоздать все некластеризованные индексы
- MOVE TO - куда переместить строки данных, находящиеся в страницах листьев кластеризованного индекса - файловая группа по умолчанию или именованная файловая группа
- Нельзя для PRIMARY KEY и UNIQUE

Вычисляемые столбцы

- Вычисляемым называется столбец таблицы, в котором сохраняются результаты вычислений данных таблицы.
 - Виртуальный
 - Постоянный

Вычисляемые столбцы

```
CREATE TABLE Orders
(orderid INT NOT NULL,
price MONEY NOT NULL,
quantity INT NOT NULL,
orderdate DATETIME NOT NULL,
total AS price * quantity,
shippeddate AS DATEADD (DAY, 7, orderdate));

CREATE CLUSTERED INDEX i1 ON orders (total);
```

Индексированные представления

- Создается представление CREATE VIEW с предложением SCHEMABINDING
- Создается кластеризованный индекс для этого представления – в инструкции CREATE INDEX вместо имени таблицы указывается имя представления

Индексированные представления

- все используемые в представлении функции должны быть детерминированными
- представление должно ссылаться только на базовые таблицы
- представление и базовые таблицы должны иметь одного владельца и принадлежать к одной и той же базе данных
- инструкция SELECT в представлении не должна содержать DISTINCT, UNION, TOP, ORDER BY, MIN, MAX, COUNT, OUTER, SUM (для выражений, допускающих значения NULL), подзапросы или производные таблицы

Индексированные представления

```
create view Aud(AUDITORIUM_NAME, AUDITORIUM_CAPACITY)
WITH SCHEMABINDING
as
select AUDITORIUM_NAME, AUDITORIUM_CAPACITY from dbo.AUDITORIUM;
go

SELECT objectproperty(object_id('dbo.AUDITORIUMS'), 'IsIndexable');
SELECT objectproperty(object_id('dbo.Aud'), 'IsIndexable');
```

The screenshot shows the SQL Server Management Studio interface with two result panes. The top pane is titled 'Results' and displays the output of the first two SELECT statements. The bottom pane is also titled 'Results' and displays the output of the last two SELECT statements.

Results	Results
(No column name)	(No column name)
0	1

Индексированные представления

- запросы, которые обрабатывают большое количество строк и содержат операции соединения или агрегатные функции
- операции соединения и агрегатные функции, которые часто выполняются в одном или нескольких запросах

Колоночные индексы

- Строки таблиц сохраняются в страницах - построчное хранение - row store
- Данные группируются и сохраняются по одному столбцу - постолбцовое хранение - column store
- реализуется посредством использования колоночного индекса - columnstore index

Колоночные индексы

- Система извлекает только требуемые столбцы
- Оптимальное сжатие значений.
- Значительное ускорение времени выполнения запросов с особыми характеристиками

```
- CREATE NONCLUSTERED COLUMNSTORE INDEX cs_index1  
| ON AUDITORIUM (AUDITORIUM_NAME, AUDITORIUM_CAPACITY, AUDITORIUM_TYPE);
```

Колоночные индексы

- Таблица с колоночным индексом доступна только для чтения
- Колоночные индексы поддерживают только типы данных CHAR, VARCHAR, INT, DECIMAL и FLOAT
- Ограничений на кластеризованные и некластеризованные колоночные индексы – не больше одного некластеризованного колоночного индекса, а кластеризованные колоночные индексы не поддерживаются вообще

Оптимизация запроса

- анализ запроса
- выбор индекса
- выбор порядка выполнения операций соединения
- выбор метода выполнения операций соединения

Селективность и плотность

- Селективность запроса – соотношение количества строк, удовлетворяющих условию, к общему количеству строк в таблице
- Плотность запроса – количество возвращаемых строк запроса

Селективность

- Индекс успешно работает при $\leq 5\%$.
- Не нужен индекс при 80% или более

Анализ запроса

- Наличие аргументов поиска
- Использование оператора OR
- Существование критериев соединения

Анализ запроса

- Аргумент поиска — это часть запроса, которая ограничивает промежуточный результирующий набор запроса:
- name = 'Иванов С.П.'
- capacity >= 100
- name = 'Иванов С.П.' AND idgroup = 512

Анализ запроса

- Нельзя использовать в качестве аргументов поиска:
 - Выражение с оператором отрицания NOT
 - <>
 - Выражение
- NOT IN('d1', 'd2')
- aud_no <> 9031
- capacity * 0.6 > 100

Создание индексов

- SELECT ... WHERE column_name – для этого столбца следует создать индекс
- SELECT ... WHERE column_name1 AND column_name2 – создать составной индекс по всем столбцам

Индексы при соединении

- Для каждого соединяемого столбца
- Некластеризованный индекс для столбца внешнего ключа

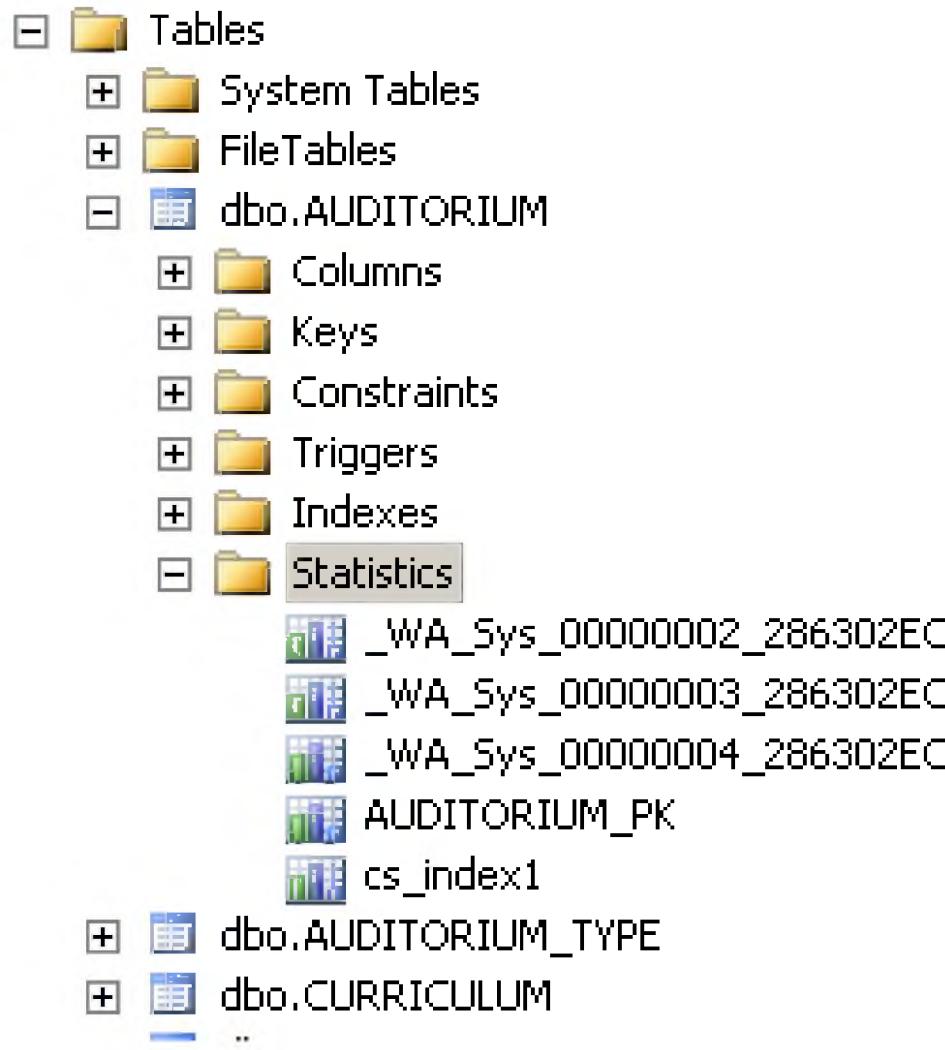
Выбор индексов

- Оптимизатор проверяет селективность выражения с индексированным столбцом, используя статистические данные
- AUTO_CREATE_STATISTICS ON(OFF)

Обход индекса

- Для кучи сначала выполняется обход некластеризованного индекса, а затем извлекается строка, используя идентификатор строки
- Для кластеризованной таблицы, после обхода структуры некластеризованного индекса следует обход структуры кластеризованного индекса таблицы

Сбор статистики



Сбор статистики

New Statistics on Table dbo.AUDITORIUM

Select a page Script Help

General Filter

Table Name: dbo.AUDITORIUM

Statistics Name:

Statistics Columns:

Name	Data Type	Size	Identity	Allow
AUDITORIUM	char	20	<input type="checkbox"/>	<input type="checkbox"/>
AUDITORIUM_TYPE	char	10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AUDITORIUM_CAPACI...	int	4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AUDITORIUM_NAME	varchar	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add... Remove Move Up Move Down

Connection

Server: 1\VAIO

Update Statistics: _____

Выбор порядка соединения

- Вложенный цикл
- Соединение слиянием
- Соединение хешированием

Вложенный цикл

- Для каждой строки внешней таблицы извлекается и сравнивается каждая строка внутренней таблицы
- Сканирование внешней таблицы и n раз – сканирование внутренней

Соединение слиянием

- Строки соединяемых таблиц должны быть физически упорядочены с использованием значений столбца соединения
- Выполняется сканирование обеих таблиц в порядке столбцов соединения, сопоставляя строки с одинаковыми значениями для столбцов соединения

Соединение хешированием

- Вычисляется хеш для значения соединяемого столбца из меньшей таблицы и сохраняется в определенном сегменте
- Вычисляется хеш для значения соединяемого столбца из большей таблицы и сравнивается с хешами на предыдущем этапе
- Совпадающие строки попадают в результирующий набор

Кэширование планов

- При первом выполнении запроса его скомпилированная версия сохраняется в кэше планов — `plan cache`
- При повторном выполнении запроса проверяется, нет ли для него плана в кэше планов

Просмотр планов кэша

- sys.dm_exec_cached_plans
- sys.dm_exec_query_stats
- sys.dm_exec_sql_text

Просмотр планов

- SET
- среда Management Studio

Инструкция SET

```
SET SHOWPLAN_TEXT ON;
GO
```

```
SELECT * FROM FACULTY JOIN PULPIT
ON FACULTY.FACULTY = PULPIT.FACULTY
AND FACULTY.FACULTY = 'ИДиП';
GO
```

```
SET SHOWPLAN_TEXT OFF;
GO
```

The screenshot shows the SQL Server Management Studio interface with the following details:

- Toolbar:** Includes a percentage icon (%), a dropdown arrow, and a refresh button.
- Tab Bar:** Shows "Results" and "Messages" tabs.
- Results Grid:** Contains the query text:

```
SELECT * FROM FACULTY JOIN PULPIT ON FACULTY...
```
- Execution Plan Grid:** Contains the execution plan steps:
 - Statement Text: I-Nested Loops(Inner Join)
 - Step 1: I-Clustered Index Seek(OBJECT:[TMP1_BSTU].[dbo].[FACULTY].[FACULTY_PK]), SEEK:[TMP1_BSTU].[dbo].[FACULTY].[FACULTY]='ИДиП') ORDERED FORWARD
 - Step 2: I-Clustered Index Scan(OBJECT:[TMP1_BSTU].[dbo].[PULPIT].[PULPIT_PK]), WHERE:[TMP1_BSTU].[dbo].[PULPIT].[FACULTY]='ИДиП')

Инструкция SET

- |--Nested Loops(Inner Join)
- |--Clustered Index Seek (OBJECT: ([TMP1_BSTU].[dbo].[FACULTY].[FACULTY_PK]), SEEK:([TMP1_BSTU].[dbo].[FACULTY].[FACULTY]='ИдиП') ORDERED FORWARD)
- |--Clustered Index Scan(OBJECT: ([TMP1_BSTU].[dbo].[PULPIT].[PULPIT_PK]), WHERE:([TMP1_BSTU].[dbo].[PULPIT].[FACULTY]='ИдиП'))

Просмотр плана выполнения

- Оператор с самым большим отступом выполняется первым
- Если два или более операторов имеют одинаковый отступ, они выполняются в порядке сверху вниз
- Compute Scalar вычисляет выражение, выдавая в результате скалярное значение
- Clustered Index Seek выполняет поиск строк по кластеризованным индексам
- Index Scan - обрабатываются все листья страницы дерева индексов

Подсказки оптимизации

- Подсказки оптимизации являются частью инструкции SELECT, которые указывают оптимизатору запросов, что нужно выполнять данную инструкцию определенным образом

Подсказки оптимизации hints

- табличные подсказки
- подсказки соединения
- подсказки запросов
- структуры планов

Табличные подсказки

- INDEX
- NOEXPAND
- FORCESEEK

```
SELECT * FROM AUDITORIUM WITH (INDEX(cs_index1))
WHERE AUDITORIUM = '461-2';
```

Подсказки соединения

- FORCE ORDER
- LOOP
- HASH
- MERGE

```
SELECT * FROM FACULTY JOIN PULPIT  
ON FACULTY.FACULTY = PULPIT.FACULTY  
AND FACULTY.FACULTY = 'ИДиП'  
OPTION (FORCE ORDER);
```

Подсказки запросов

- FAST n
- OPTIMIZE FOR
- OPTIMIZE FOR UNKNOWN
- USE PLAN

```
SELECT * FROM TEACHER OPTION (FAST 10);
```

|

Структуры планов

- sp_create_plan_guide
- sys.plan_guides

```
sp_create_plan_guide @name = N'Example_hint_15',
@stmt = N'SELECT * FROM FACULTY JOIN PULPIT
ON FACULTY.FACULTY = PULPIT.FACULTY',
@type = N'SQL',
@Module_or_batch = NULL,
@params = NULL,
@hints = N'OPTION (HASH JOIN)';

select * from sys.plan_guides;
```

The screenshot shows a SQL query window with two parts. The first part is a T-SQL script to create a plan guide named 'Example_hint_15' for the query 'SELECT * FROM FACULTY JOIN PULPIT ON FACULTY.FACULTY = PULPIT.FACULTY' with a hint of 'OPTION (HASH JOIN)'. The second part is a 'select * from sys.plan_guides;' command to verify the creation. Below the window, the results pane shows a table with one row, matching the created plan guide.

plan_guide_id	name	is_disabled	query_text
65537	Example_hint_15	0	SELECT * FROM FACULTY JOIN PULPIT ON FACULTY.FACULTY = PULPIT.FACULTY

Настройка производительности

- Факторы, влияющие на производительность
- Мониторинг производительности
- Средства для настройки производительности

Факторы производительности

- прикладные программы баз данных
- система баз данных
- системные ресурсы

Приложения базы данных и производительность

- эффективность кода приложения
- проектирование на физическом уровне

Приложения базы данных и производительность

- использовать кластеризованные индексы
- исключить использование предиката NOT IN

Проектирование на физическом уровне

- Денормализация таблиц означает соединение вместе двух или более нормализованных таблиц в одну с некоторой избыточностью данных

Денормализация

- позволяет избежать использования операции соединения
- для денормализованных данных требуется меньшее число таблиц, чем для нормализованных
- для хранения денормализованной таблицы требуется больший объем дискового пространства
- модификация данных усложняется вследствие избыточности данных

СУБД и производительность

- оптимизатор запросов
- блокировки

Оптимизатор запросов

- Оптимизатор формулирует несколько планов выполнения запроса для выборки

Блокировки

- Используются для управления одновременным доступом к данным и для предотвращения потенциальных ошибок в случае одновременного доступа к одним и тем же данным
- Оказывают влияние на производительность системы вследствие гранулярности
- Блокировка на уровне строк обеспечивает наилучшую производительность
- Уровни изоляции влияют на длительность блокировки для инструкций SELECT

Системные ресурсы и производительность

- Центральный процессор
- Оперативная память
- Дисковые операции ввода/вывода
- Сетевое окружение

Процессор

- выполняет пользовательские процессы и взаимодействует с другими ресурсами системы
- Проблема:
 - операционная система и пользовательские программы обращаются со слишком большим количеством запросов

Оперативная память

- Динамическое освобождение памяти
- Проблемы:
 - Недостаточно памяти для выполнения требуемой работы.

Дисковые операции ввода/вывода

- Скорость передачи данных определяется объемом данных, который можно записать на диск в единицу времени
- При одновременном использовании системы базы данных большим количеством пользователей, несколько дисков лучше, чем один диск

Сетевая инфраструктура

- если сервер баз данных отправляет приложению какие-либо строки, отправлять следует только лишь те строки, которые действительно требуются приложению
- если продолжительное пользовательское приложение выполняется только на стороне клиента, то его следует переместить на сторону сервера

Зависимость ресурсов

- При увеличении количества процессоров нагрузка на них распределяется равномерно, что может решить проблему узкого места с дисковыми операциями
- Большой объем оперативной памяти повышает шансы, что страница будет найдена в памяти
- Чтение данных с диска вместо получения их из кэша тормозит систему при большом количестве конкурентных процессов

Дисковые операции ввода/вывода

- Выполняется большой объем дисковых операций
- Операции чтения с диска и записи на него являются двумя наиболее затратными операциями
- системы баз данных
- Данные хранятся в страницах размером в 8 Кбайт
- Кэш оперативной памяти разделен на страницы размером по 8 Кбайт
- Система читает данные по страницам
- Операции чтения выполняются для операций выборки и модификации

Дисковые операции ввода/вывода

- Если требуемая страница отсутствует в кэше, то она считывается с диска и помещается в буферный кэш - физический ввод/вывод или физическое чтение
- Буферный кэш является памятью совместного использования - к одной и той же странице могут обращаться несколько пользователей
- При модификации данных в буферном кэше выполняется операция логической записи
- Операция физической записи происходит только при сохранении данных из кэша на диск
 - упреждающее чтение
 - контрольные точки

Упреждающее чтение

- Выполнение чтения данных только из памяти и никогда не ожидать завершения операции чтения с диска
- Знать, какие следующие несколько страниц потребуются пользователю, и считывать эти страницы до того, как пользовательский процесс запросит их

Упреждающее чтение

- Read Ahead Manager
- Единица памяти для упреждающих операций чтения
64 Кбайт
- Выполнение объемных сканирований таблиц и сканирований диапазонов индексов
- Read Ahead Manager считывает до 2 Мбайт данных за один раз
- Каждый экстент считывается с помощью одной операции

Упреждающее чтение

- Множественные одновременные операции упреждающего чтения для каждого файла
- Используется информация из промежуточного уровня индексных страниц, находящегося сразу же над уровнем листьев
- В процессе распознаются смежные страницы и считаются один раз
- Может иметь отрицательное воздействие на производительность, если приходится читать слишком много страниц, излишне заполняя кэш
- Создавать индексы, которые в действительности необходимы

Вопросы?

БАЗЫ ДАННЫХ

Лекция 14 Транзакции

Модели конкурентного доступа

- пессимистический одновременный конкурентный доступ
- оптимистический одновременный конкурентный доступ

Пессимистический доступ

- Предполагается, что между процессами в любое время может возникнуть конфликт и ресурс блокируется

Оптимистический доступ

- Предполагается, что одновременное изменение данных маловероятно

Транзакция

- Одна или несколько команд SQL, которые либо успешно выполняются как единое целое, либо отменяются как единое целое

Транзакция

- Логическая единица работы, обеспечивающая переход базы данных из одного согласованного состояния в другое согласованное состояние

Транзакции

- Неявные
- Явные

Неявные транзакции

- Неявная транзакция — задает любую отдельную инструкцию INSERT, UPDATE или DELETE как единицу транзакции

Явные транзакции

- Явная транзакция — группа инструкций, начало и конец которой обозначаются инструкциями:
 - BEGIN TRANSACTION
 - COMMIT
 - ROLLBACK

ACID

- Atomicity - Атомарность
- Consistency - Согласованность
- Isolation - Изолированность
- Durability - Долговечность

Atomicity - Атомарность

- Выполняются или все изменения данных в транзакции или ни одна

Consistency - Согласованность

- Выполняемые транзакцией трансформации данных переводят базу данных из одного согласованного состояния в другое

Isolation - Изолированность

- Все параллельные транзакции отделяются друг от друга.
- Активная транзакция не может видеть модификации данных в параллельной или незавершенной транзакции

Durability - Долговечность

- Транзакцию после фиксации нельзя отменить, кроме как другой транзакцией

Инструкции

- BEGIN TRANSACTION
- BEGIN DISTRIBUTED TRANSACTION
- COMMIT [WORK]
- ROLLBACK [WORK]
- SAVE TRANSACTION
- SET IMPLICIT_TRANSACTIONS

BEGIN TRANSACTION

```
 BEGIN TRANSACTION;

SELECT COUNT(*) FROM AUDITORIUM;

INSERT INTO AUDITORIUM VALUES ('123-1', 'ЛК', 60, '123-1');

SELECT COUNT(*) FROM AUDITORIUM;

COMMIT;

SELECT COUNT(*) FROM AUDITORIUM;
```

The screenshot shows a window titled 'Результаты' (Results) with three rows of data. The first row has two columns: 'Сообщения' (Messages) and '(Отсутствует имя столбца)' (No column name specified). The value '14' is displayed in the second column. The second row has the same structure and displays the value '15'. The third row also has the same structure and displays the value '15'.

Сообщения	(Отсутствует имя столбца)
	14
	15
	15

ROLLBACK

```
BEGIN TRANSACTION;

SELECT COUNT(*) FROM AUDITORIUM;

INSERT INTO AUDITORIUM VALUES ('128-1', 'ЛК-К', 60, '128-1');

SELECT COUNT(*) FROM AUDITORIUM;

ROLLBACK;

SELECT COUNT(*) FROM AUDITORIUM;
```

The screenshot shows a SQL query window with two tabs: 'Результаты' (Results) and 'Сообщения' (Messages). The Results tab displays three rows of data from the AUDITORIUM table. The first row shows 15 rows before the insert. The second row shows 16 rows after the insert. The third row shows 15 rows again after the rollback.

id	name	capacity	auditorium_id
15			
16			
15			

SAVE TRANSACTION

```
BEGIN TRANSACTION;

INSERT INTO AUDITORIUM VALUES ('127-1', 'ЛК', 60, '127-1');
SAVE TRANSACTION A;

INSERT INTO AUDITORIUM VALUES ('128-1', 'ЛК-К', 60, '128-1');
SAVE TRANSACTION B;

INSERT INTO AUDITORIUM VALUES ('129-1', 'ЛК', 60, '129-1');
ROLLBACK TRANSACTION B;

INSERT INTO AUDITORIUM VALUES ('130-1', 'ЛК-К', 60, '130-1');
ROLLBACK TRANSACTION A;

COMMIT TRANSACTION;
```

|

```
SELECT * FROM AUDITORIUM
WHERE AUDITORIUM IN('127-1', '128-1', '129-1', '130-1');
```

%

Результаты Сообщения

AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
127-1	ЛК	60	127-1

SAVE TRANSACTION

- Точка сохранения определяет точку в транзакции, такую что все последующие изменения данных могут быть отменены без отмены всей транзакции
- **SAVE TRANSACTION** создает метку для последующей инструкции **ROLLBACK**, имеющей такую же метку, как и данная инструкция **SAVE TRANSACTION**

BEGIN DISTRIBUTED TRANSACTION

- Запускается распределенная транзакция
- Управляется Microsoft Distributed Transaction Coordinator
- Распределенная транзакция — это транзакция, которая используется на нескольких базах данных и на нескольких серверах
- Координатор - сервер, запустивший инструкцию BEGIN DISTRIBUTED TRANSACTION

SET IMPLICIT_TRANSACTIONS ON

- Режим неявных транзакций
- Если транзакцию явно не зафиксировать, то все изменения, выполненные в ней, откатываются при отключении пользователя
- Любая из следующих инструкций запускает транзакцию:
- CREATE (ALTER, DROP, TRUNCATE) TABLE
- OPEN FETCH
- GRANT REVOKE
- INSERT DELETE UPDATE
- SELECT

SET IMPLICIT_TRANSACTIONS OFF

- BEGIN TRANSACTION
- COMMIT или ROLLBACK
- Явные транзакции можно вкладывать друг в друга
- Вложенные транзакции используются в хранимых процедурах, которые сами содержат транзакции и вызываются внутри другой транзакции
- @@TRANCOUNT содержит число активных транзакций для текущего пользователя

@@TRANCOUNT

```
----- @@TRANCOUNT
```

```
SELECT COUNT(*) '1', @@TRANCOUNT 'TRANCOUNT' FROM AUDITORIUM;  
BEGIN TRANSACTION A;
```

```
INSERT INTO AUDITORIUM VALUES ('128-1', 'JK', 60, '128-1');  
SELECT COUNT(*) '2', @@TRANCOUNT 'TRANCOUNT' FROM AUDITORIUM;
```

```
BEGIN TRANSACTION B;
```

```
DELETE AUDITORIUM where Auditorium = '128-1';
```

```
SELECT COUNT(*) '3', @@TRANCOUNT 'TRANCOUNT' FROM AUDITORIUM;
```

```
COMMIT TRANSACTION B;
```

```
SELECT COUNT(*) '4', @@TRANCOUNT 'TRANCOUNT' FROM AUDITORIUM;
```

```
INSERT INTO AUDITORIUM VALUES ('128-1', 'JK', 60, '128-1');
```

```
SELECT COUNT(*) '5', @@TRANCOUNT 'TRANCOUNT' FROM AUDITORIUM;
```

```
COMMIT TRANSACTION A;
```

```
SELECT COUNT(*) '6', @@TRANCOUNT 'TRANCOUNT' FROM AUDITORIUM;
```

1	TRANCOUNT
16	0

2	TRANCOUNT
17	1

3	TRANCOUNT
16	2

4	TRANCOUNT
16	1

5	TRANCOUNT
17	1

6	TRANCOUNT
17	0

Журнал транзакций

- Журнал транзакций применяется для отката или восстановления транзакции
- Для каждой базы данных собственный журнал транзакций
- Database Engine сохраняет значения до и после транзакции в журналах транзакций (transaction log)
 - Исходные образы записей (before image)
 - Преобразованные образы записей (after image)
 - LSN – порядковый номер для каждой записи
- Процессы:
 - Процесс отмены записей (undo)
 - Процесс повторного выполнения действий (redo)

Блокировки

- Блокировки – механизм обеспечения согласованности данных в случае одновременного обращения к данным нескольких пользователей
- Свойства:
 - Длительность блокировки
 - Режим блокировки
 - Гранулярность блокировки

Длительность блокировки

- Длительность блокировки — это период времени, в течение которого ресурс удерживает определенную блокировку

Режим блокировки

- Разделяемая (shared lock)
 - Монопольная (exclusive lock)
 - Обновления (update lock)
-
- СУБД автоматически выбирает соответствующий режим блокировки, в зависимости от типа операции (чтение или запись)

Разделяемая блокировка

- Разделяемая блокировка резервирует ресурс только для чтения
- Другие процессы не могут изменять заблокированный ресурс
- Может быть несколько разделяемых блокировок

Монопольная блокировка

- Монопольная блокировка резервирует страницу или строку для монопольного использования одной транзакции
- Применяется при INSERT, UPDATE и DELETE
- Монопольную блокировку нельзя установить, если на ресурс уже установлена какая-либо блокировка

Блокировка обновления

- Можно устанавливать на объекты с разделяемой блокировкой, накладывается еще одна разделяемая блокировка
- Нельзя устанавливать при наличии на нем другой блокировки обновления или монопольной блокировки
- При COMMIT транзакции обновления, блокировка обновления преобразовывается в монопольную блокировку
- У объекта может быть только одна блокировка обновления

Гранулярность блокировки

- Гранулярность блокировки определяет, какой объект блокируется:
 - строки
 - страницы
 - индексный ключ или диапазон индексных ключей
 - таблицы
 - экстент
 - база данных
- СУБД выбирает гранулярность блокировки автоматически

Гранулярность блокировки

- Процесс преобразования большого числа блокировок уровня строки, страницы или индекса в одну блокировку уровня таблицы называется эскалацией блокировок (lock escalation)
- ALTER TABLE
- SET (LOCK_ESCALATION = {**TABLE** | AUTO | DISABLE})
- Подсказки блокировок (locking hints)
- SET LOCK_TIMEOUT - период в миллисекундах, в течение которого транзакция будет ожидать снятия блокировки с объекта (-1 по умолчанию, не установлен)

БЛОКИРОВКИ

- sys.dm_tran_locks

```
SELECT resource_type,
DB_NAME(resource_database_id) as database_name,
request_session_id, request_mode,
request_status
FROM sys.dm_tran_locks;
```

The screenshot shows the results of the SQL query in a results grid. The columns are labeled: resource_type, database_name, request_session_id, request_mode, and request_status. There are five rows, each representing a lock on the 'BSTU' database. All locks are held by session ID 55 and have mode 'S' (Share) and status 'GRANT'.

resource_type	database_name	request_session_id	request_mode	request_status
DATABASE	BSTU	55	S	GRANT
DATABASE	BSTU	51	S	GRANT
DATABASE	BSTU	54	S	GRANT
DATABASE	BSTU	53	S	GRANT
DATABASE	BSTU	52	S	GRANT

Взаимоблокировки

- Взаимоблокировка (deadlock) — это особая проблема одновременного конкурентного доступа, в которой две транзакции блокируют друг друга

Взаимоблокировки

```
-----deadlock1
BEGIN TRANSACTION
---
UPDATE AUDITORIUM
    SET AUDITORIUM_TYPE = 'ЛБ-К'
    WHERE AUDITORIUM_NAME = '128-1';
WAITFOR DELAY '00:00:10';
UPDATE AUDITORIUM
    SET AUDITORIUM_CAPACITY = 70
    WHERE AUDITORIUM_NAME = '127-1';
COMMIT;
```

```
-----deadlock2
BEGIN TRANSACTION
UPDATE AUDITORIUM
    SET AUDITORIUM_TYPE = 'ЛК'
    WHERE AUDITORIUM_NAME = '127-1';
WAITFOR DELAY '00:00:10';
UPDATE AUDITORIUM
    SET AUDITORIUM_CAPACITY = 66
    WHERE AUDITORIUM_NAME = '128-1';
COMMIT;
```

Messages

```
Msg 1205, Level 13, State 51, Line 4
Transaction (Process ID 57) was deadlocked on lock resources
```

with another process and has been chosen as the deadlock victim. Rerun the transaction.

Уровни изоляции

- Уровень изоляции задает степень защищенности данных в транзакции от возможности изменения другими транзакциями

Проблемы

- Потеря обновлений
- Грязное чтение
- Неповторяющееся чтение
- Фантомное чтение

Потеря обновлений

- Несколько транзакций одновременно могут считывать и обновлять одни и те же данные
- Теряются все обновления данных, за исключением обновлений, выполненных последней транзакцией

Грязное чтение

- Происходит чтение несуществующих данных или потеря модифицированных данных

Неповторяющее чтение

- Один процесс считывает данные несколько раз, а другой процесс изменяет эти данные между двумя операциями чтения первого процесса
- Значения двух чтений будут разными

Фантомное чтение

- Последовательные операции чтения могут возвратить разные значения
- Считывание разного числа строк при каждом чтении
- Возникают дополнительные фантомные строки, которые вставляются другими транзакциями

УРОВНИ ИЗОЛЯЦИИ

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE
- SNAPSHOT

Уровни изоляции

- Пессимистическая модель:
 - READ UNCOMMITTED
 - REPEATABLE READ
 - SERIALIZABLE
- Оптимистическая модель:
 - SNAPSHOT
- Обе модели:
 - READ COMMITTED

READ UNCOMMITTED

- Не изолирует операции чтения других транзакций
- Транзакция не задает и не признает блокировок
- Допускает проблемы:
 - Грязное чтение
 - Неповторяющее чтение
 - Фантомное чтение

READ UNCOMMITTED

```
-- 1  
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED  
BEGIN TRAN  
SELECT COUNT(*) FROM AUDITORIUM -- запускаем транзакцию, Результат: 17
```

Results	
(No column name)	
	17

```
-- 2  
BEGIN TRAN -- открываем параллельную транзакцию  
DELETE FROM AUDITORIUM WHERE AUDITORIUM='128-1' -- удаляем строку из таблицы  
-- 3  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 16, налицо неподтвержденное чтение
```

Results	
(No column name)	
	16

READ UNCOMMITTED

```
-- 4  
ROLLBACK TRAN -- откатываем транзакцию
```

```
-- 5  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 17, после отката транзакции В  
COMMIT TRAN
```

Results	Messages
(No column name)	
17	

READ COMMITTED

- Транзакция выполняет проверку только на наличие монопольной блокировки для данной строки
- Является уровнем изоляции по умолчанию
- Проблемы:
 - Неповторяющее чтение
 - Фантомное чтение

READ COMMITTED

----- Покажем, что уровень изолированности READ COMMITTED не допускает неподтвержденное чтение

-- 6

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```

```
BEGIN TRAN
```

```
SELECT COUNT(*) FROM AUDITORIUM -- запускаем транзакцию, Результат: 17
```

Results	Messages
(No column name)	
17	

-- 7

```
BEGIN TRAN -- открываем параллельную транзакцию
```

```
DELETE FROM AUDITORIUM WHERE AUDITORIUM='128-1' -- удаляем строку из таблицы
```

-- 8

```
SELECT COUNT(*) FROM AUDITORIUM -- Результат: ожидание, неподтвержденного чтения нет
```



-- 9

```
ROLLBACK TRAN -- откатываем транзакцию
```

-- 10

```
SELECT COUNT(*) FROM AUDITORIUM -- сразу после отката транзакции В Результат: 17,
```

```
COMMIT TRAN
```

Results	Messages
(No column name)	
17	

READ COMMITTED

----- Покажем, что уровень изолированности READ COMMITTED допускает неповторяющееся чтение

```
-- 11  
SET TRANSACTION ISOLATION LEVEL READ COMMITTED  
BEGIN TRAN  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 17
```

Results	
(No column name)	
1	17

```
-- 12  
BEGIN TRAN -- открываем параллельную транзакцию  
DELETE FROM AUDITORIUM WHERE AUDITORIUM = '128-1' -- удаляем строку из таблицы  
COMMIT TRAN
```

```
-- 13  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 16  
-- пока вторая транзакция удаляла запись, данные дважды прочитались по-разному.  
COMMIT TRAN
```

Results	
(No column name)	
16	

REPEATABLE READ

- Устанавливает разделяемые блокировки на все считывающие данные и удерживает эти блокировки до тех пор, пока транзакция не будет подтверждена или отменена
- Не препятствует другим инструкциям вставлять новые строки
- Проблема:
 - Фантомное чтение

REPEATABLE READ

----- Покажем, что уровень изолированности REPEATABLE READ не допускает неповторяющееся чтение

```
INSERT INTO AUDITORIUM VALUES ('128-1', 'ЛК', 60, '128-1'); -- вернем запись
```

-- 14

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
```

```
BEGIN TRAN
```

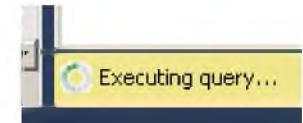
```
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 17
```

Results	Messages
(No column name)	
17	

-- 15

```
BEGIN TRAN -- открываем параллельную транзакцию
```

```
DELETE FROM AUDITORIUM WHERE AUDITORIUM = '128-1' -- удаляем строку из таблицы, результат - ожидание
```



-- 16

```
COMMIT TRAN -- сразу после фиксации транзакции A в окне B
```

-- Строк обработано:1 - прошло выполнение оператора удаления

-- 17

```
COMMIT TRAN -- завершаем транзакцию
```

REPEATABLE READ

----- Покажем, что уровень изолированности REPEATABLE READ допускает проблему фантомных записей

```
INSERT AUDITORIUM VALUES ('128-1', 'ЛК', 60, '128-1'); -- вернем запись  
-- 18  
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ  
BEGIN TRAN  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 17
```

Results	
	(No column name)
1	17

```
-- 19  
BEGIN TRAN  
INSERT INTO AUDITORIUM VALUES ('437-1', 'ЛК', 20, '437-1') -- Строк обработано:1  
COMMIT TRAN -- завершаем транзакцию
```

```
-- 20  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 18  
--- в рамках одной транзакции А два результата  
COMMIT TRAN
```

Results	
	(No column name)
	18

SERIALIZABLE

- Устанавливает блокировку на всю область данных, считываемых соответствующей транзакцией
- Предотвращает вставку новых строк другой транзакцией до тех пор, пока первая транзакция не будет подтверждена или отменена

SERIALIZABLE

- Реализуется с использованием метода блокировки диапазона ключа
- Блокировка диапазона ключа блокирует элементы индексов

SERIALIZABLE

----- Покажем, что уровень изолированности SERIALIZABLE не допускает проблему фантомных записей

-- 21

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

```
BEGIN TRAN
```

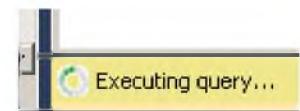
```
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 18
```

Results	
(No column name)	
1	18

--- 22

```
BEGIN TRAN
```

```
INSERT INTO AUDITORIUM VALUES ('446-1', 'ЛК', 20, '446-1') -- ожидание
```



-- 23

COMMIT TRAN -- после выполнения этой команды в сценарии В - Строк обработано:1

Установка уровня изоляции

- SET TRANSACTION ISOLATION LEVEL:
 - READ UNCOMMITTED
 - READ COMMITTED
 - REPEATABLE READ
 - SERIALIZABLE

Уровень изоляции

DBCC USEROPTIONS

Set Option	Value
textsize	2147483647
language	us_english
dateformat	mdy
datefirst	7
lock_timeout	-1
quoted_identifier	SET
arithabort	SET
ansi_null_dflt_on	SET
ansi_warnings	SET
ansi_padding	SET
ansi_nulls	SET
concat_null_yields_null	SET
isolation level	read committed



Управление версиями строк

- Механизм управления оптимистическим одновременным конкурентным доступом основан на управлении версиями строк
- Для всех изменений данных создаются и поддерживаются логические копии
- При каждом изменении строки СУБД сохраняет в tempdb исходный вид записи

Управление версиями строк

- Каждая версия строки помечается порядковым номером транзакции (XSN — transaction sequence number)
- Самая последняя версия строки сохраняется в базе данных и соединяется в связанном списке с версией, сохраненной в tempdb

Управление версиями строк

- Поддержка уровней изоляции READ COMMITTED
SNAPSHOT и SNAPSHOT
- Создание в триггерах таблиц inserted и deleted

READ COMMITTED SNAPSHOT

- Любая другая транзакция будет читать значения зафиксированные на момент начала этой транзакции
- ALTER DATABASE
- SET ISOLATION LEVEL
- READ COMMITTED SNAPSHOT

SNAPSHOT

- Уровень изоляции на уровне транзакций
- Любая другая транзакция будет читать подтвержденные значения в том виде, в каком они существовали непосредственно перед началом выполнения этой транзакции

SNAPSHOT

- На уровне базы данных включается
- ALLOW_SNAPSHOT_ISOLATION
- SET TRANSACTION ISOLATION LEVEL SNAPSHOT

SNAPSHOT

```
-- Установим ALLOW_SNAPSHOT_ISOLATION
USE master
GO
ALTER DATABASE B_BSTU SET ALLOW_SNAPSHOT_ISOLATION ON
GO
USE B_BSTU
GO
-- 24
SET TRANSACTION ISOLATION LEVEL SNAPSHOT
BEGIN TRAN
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 19
```

--- 25

```
BEGIN TRAN
INSERT INTO AUDITORIUM VALUES ('447-1', 'ЛК', 20, '447-1') -- выполняем вставку - Строк обработано:1
```

-- 26

```
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 19 - результат прежний
```

Results	Messages
(No column name)	
19	

Results	Messages
(No column name)	
19	

SNAPSHOT

```
--27  
COMMIT -- накатываем транзакцию В  
  
-- 28  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 19 - а результат все равно прежний  
  
-- 29  
COMMIT -- накатываем транзакцию А  
SELECT COUNT(*) FROM AUDITORIUM -- Результат: 20 - изменился
```

Results	Messages
(No column name)	
20	

Results	Messages
(No column name)	
19	

Вопросы?

БАЗЫ ДАННЫХ

Лекция 15 Триггеры

Триггер

- Триггер - специальный вид хранимых процедур, выполняющихся при событиях базы данных

Триггер

- Имя
- Действие
- Исполнение

Имя триггера - максимум 128 символов

Триггер

- DML-триггеры
- DDL-триггеры

DML-триггеры

- Создаются для таблицы или представления
- Реагируют на события INSERT, DELETE, UPDATE

Триггер

- CREATE TRIGGER
- [schema_name.] trigger_name
- ON { table_name | view_name }
- [WITH dml_trigger_option [...]]
- { FOR | AFTER | INSTEAD OF }
- { [INSERT] [,] [UPDATE] [,] [DELETE] }
- [WITH APPEND]
- { AS sql_statement |
- EXTERNAL NAME method_name }

DML-триггеры

```
create trigger AUD_AFTER_INSERT
on AUDITORIUM after INSERT
as
    print 'AUD_AFTER_INSERT';
    return;
go

create trigger AUD_AFTER_DELETE
on AUDITORIUM after DELETE
as
    print 'AUD_AFTER_DELETE';
    return;
go

create trigger AUD_AFTER_UPDATE
on AUDITORIUM after UPDATE
as
    print 'AUD_AFTER_UPDATE';
    return;
go
```



Таблицы `deleted` и `inserted`

Две специальные виртуальные таблицы

- `deleted` — содержит копии строк, удаленных из таблицы
- `inserted` — содержит копии строк, вставленных в таблицу
- Структура этих таблиц эквивалентна структуре таблицы, для которой определен триггер

Таблицы deleted и inserted

- Таблица deleted – в инструкции CREATE TRIGGER указывается DELETE или UPDATE
- Таблица inserted – в инструкции CREATE TRIGGER указывается INSERT или UPDATE

DML-триггеры

```
use BSTU
go
delete AUDITORIUM where AUDITORIUM in ('301-4', '401-4');

go
create trigger AUD_AFTER
on AUDITORIUM after INSERT, DELETE, UPDATE
as
declare @ins int = (select count(*) from INSERTED),
        @del int = (select count(*) from DELETED);

if      @ins > 0 and @del = 0  print 'Событие: INSERT';
else if @ins = 0 and @del > 0  print 'Событие: DELETE';
else if @ins > 0 and @del > 0  print 'Событие: UPDATE';
return;
go
```

DML-триггеры

```
alter trigger AUD_AFTER_INSERT
on AUDITORIUM after INSERT
as
    select 'insert:INSERTED', * from INSERTED
union
    select 'insert:DELETED', * from DELETED;
return;
go
```

DML-триггеры

```
alter trigger AUD_AFTER_UPDATE
on AUDITORIUM after UPDATE
as
select 'update:INSERTED', * from INSERTED
union
select 'update:DELETED', * from DELETED;
return;
go
```

DML-триггеры

```
alter trigger AUD_AFTER_DELETE
on AUDITORIUM after DELETE
as
select 'delete:INSERTED', * from INSERTED
union
select 'delete:DELETED', * from DELETED;
return;
go
```

```

insert AUDITORIUM values ('301-4', 'ЛК', 75, '301-4'),
                           ('401-4', 'ЛК', 80, '301-4');

update AUDITORIUM set AUDITORIUM_CAPACITY += 5
where AUDITORIUM in ('301-4', '401-4');

delete AUDITORIUM where AUDITORIUM in ('301-4', '401-4');

```

(No column name)	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
insert:INSERTED	401-4	ЛК	80	301-4
insert:INSERTED	301-4	ЛК	75	301-4

(No column name)	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
update:INSERTED	401-4	ЛК	85	301-4
update:INSERTED	301-4	ЛК	80	301-4
update:DELETED	401-4	ЛК	80	301-4
update:DELETED	301-4	ЛК	75	301-4

(No column name)	AUDITORIUM	AUDITORIUM_TYPE	AUDITORIUM_CAPACITY	AUDITORIUM_NAME
delete:DELETED	401-4	ЛК	85	301-4
delete:DELETED	301-4	ЛК	80	301-4

DML-триггеры

- AFTER-триггеры
- INSTEAD OF-триггеры

AFTER-триггеры

- Триггеры AFTER можно создавать только для базовых таблиц
- Можно использовать для:
 - создания журнала аудита действий в таблицах базы данных
 - реализации бизнес-логики
 - принудительного обеспечения ссылочной целостности

```
CREATE TRIGGER On_View_STUDENT
ON |dbo.all_| Не удалось создать триггер "On_View_STUDENT" для dbo.all_stdents". Для представлений допустимы только триггеры INSTEAD OF.
AS DECLAR
IF (@C >100)
BEGIN
```

AFTER-триггеры

- AFTER-триггеры - триггеры уровня оператора
- Выполняются по одному разу для каждого оператора
- Выполняются после наступления события

AFTER-триггеры

- AFTER-триггер вызывается после выполнения активизирующего его оператора
- Если оператор нарушает ограничение целостности, то возникшая ошибка не допускает выполнения этого оператора и соответствующих триггеров

AFTER-триггеры

```
alter table AUDITORIUM  
add constraint CH_AUDITORIUM check(AUDITORIUM_CAPACITY >=15)  
go  
update AUDITORIUM set AUDITORIUM_CAPACITY = 10 where AUDITORIUM = '206-1';
```

Msg 547, Level 16, State 0, Line 1

The UPDATE statement conflicted with the CHECK constraint "CH_AUDITORIUM".

The conflict occurred in database "BSTU",

table "dbo.AUDITORIUM", column 'AUDITORIUM_CAPACITY'.

The statement has been terminated.

AFTER-триггеры - транзакция

```
alter trigger AUD_AFTER
on AUDITORIUM after DELETE, UPDATE
as
declare @c int = (select sum(AUDITORIUM_CAPACITY) from AUDITORIUM);
if (ISNULL(@c, 0) < 1900)
begin
raiserror('Общая вместимость аудиторий не может быть < 1900',10,1);
rollback;
end;
return;
```

AFTER-триггеры - транзакция

```
use BSTU  
go  
delete AUDITORIUM where AUDITORIUM_TYPE = 'ЛБ-К';
```

Общая вместимость аудиторий не может быть < 1900

Msg 3609, Level 16, State 1, Line 1

The transaction ended in the trigger. The batch has been aborted.

INSTEAD OF-триггеры

- Триггеры уровня оператора
- Выполняются по одному разу для каждого оператора
- Выполняются вместо операции - сама операция не выполняется

INSTEAD OF-триггеры

- Всегда использует таблицы `inserted` и `deleted`
- Выполняется после создания таблиц `inserted` и `deleted`
- Выполняется перед выполнением проверки ограничений целостности или каких-либо других действий
- INSTEAD OF можно создавать для таблиц и для представлений - выполняется вместо выполнения любых действий с любой таблицей

INSTEAD OF-триггеры

```
use BSTU
go
create trigger AUDTYPE_INSTED
on AUDITORIUM_TYPE instead of INSERT, DELETE, UPDATE
as
    raiserror (N'изменение данных запрещено!!!!', 10, 1);
return;
```

INSTEAD OF-триггеры

```
use BSTU
go
insert AUDITORIUM_TYPE values('ЛК', 'XXX');
update AUDITORIUM_TYPE set AUDITORIUM_TYPENAME = 'YYY'
    where AUDITORIUM_TYPE = 'ЛК';
delete AUDITORIUM_TYPE where AUDITORIUM_TYPE = 'ЛК';
```

изменение данных запрещено!!!

изменение данных запрещено!!!

изменение данных запрещено!!!

INSTEAD OF-триггеры

- Не могут вызываться рекурсивно (если в триггере сработает операция, снова вызвавшая работу триггера)
- Если образуется рекурсия вызовов триггеров, то будет сделана попытка выполнить оператор

Право на создание триггера

- Владелец базы данных
- Администратор DDL
- Владелец таблицы, для которой определяется триггер
- Это разрешение не может передаваться

Порядок DML-триггеров

- Можно указать порядок выполнения для нескольких триггеров
- `sp_settriggerorder ()` имеет параметр `@order`:
 - `first` — указывает, что триггер является первым триггером AFTER, выполняющимся для действия
 - `last` — указывает, что триггер является последним триггером AFTER, выполняющимся для действия
 - `none` — отсутствует порядок выполнения (чтобы выполнить сброс)
- Остальные триггеры выполняются в неопределенном порядке

Порядок DML-триггеров

Представления каталога

```
use BSTU
go
select t.name, e.type_desc
from sys.triggers t join sys.trigger_events e
                      on t.object_id = e.object_id
where OBJECT_NAME(t.parent_id)='AUDITORIUM' and
      e.type_desc = 'UPDATE' ;
```

name	type_desc
AUD_AFTER_UPDATE	UPDATE
AUD_AFTER	UPDATE
AUD_AFTER_UPDATEA	UPDATE
AUD_AFTER_UPDATEB	UPDATE
AUD_AFTER_UPDATEC	UPDATE

sys.triggers

```
select * from sys.triggers;
```

name	object_id	parent_class	parent_class_desc	parent_id	type	type_desc	is_instead_of_trigger	is_disabled
TR_TEACHER_INS	370100359	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0
TR_TEACHER_DEL	386100416	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0
TR_TEACHER_UPD	402100473	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0
TR_TEACHER	418100530	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0
TR_TEACHER_DEL1	434100587	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0
TR_TEACHER_DEL2	450100644	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0
TR_TEACHER_DEL3	466100701	1	OBJECT_OR_COLUMN	501576825	TR	SQL_TRIGGER	0	0

sys.trigger_events

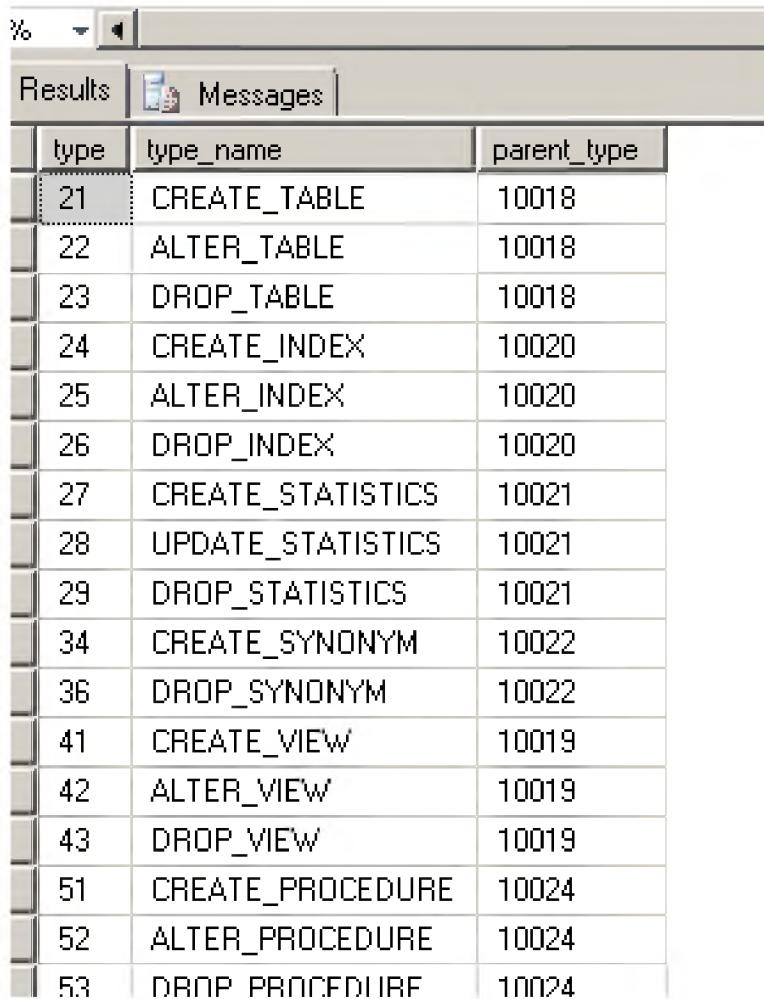
```
select * from sys.trigger_events;
```

The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top-left corner, there is a dropdown menu with the percentage symbol (%) and a back arrow icon. Below the menu, there are two tabs: "Results" (which is selected) and "Messages". The main area displays the results of the query "select * from sys.trigger_events;". The results are presented in a grid table with the following columns:

object_id	type	type_desc	is_first	is_last	event_group_type	event_group_type_desc	is_trigger_event
370100359	1	INSERT	0	0	NULL	NULL	1
386100416	3	DELETE	0	0	NULL	NULL	1
402100473	2	UPDATE	0	0	NULL	NULL	1
418100530	1	INSERT	0	0	NULL	NULL	1
418100530	2	UPDATE	0	0	NULL	NULL	1
418100530	3	DELETE	0	0	NULL	NULL	1
434100587	3	DELETE	0	0	NULL	NULL	1
450100644	3	DELETE	0	1	NULL	NULL	1
466100701	3	DELETE	1	0	NULL	NULL	1

sys.trigger_event_types

```
| select * from sys.trigger_event_types;
```



type	type_name	parent_type
21	CREATE_TABLE	10018
22	ALTER_TABLE	10018
23	DROP_TABLE	10018
24	CREATE_INDEX	10020
25	ALTER_INDEX	10020
26	DROP_INDEX	10020
27	CREATE_STATISTICS	10021
28	UPDATE_STATISTICS	10021
29	DROP_STATISTICS	10021
34	CREATE_SYNONYM	10022
36	DROP_SYNONYM	10022
41	CREATE_VIEW	10019
42	ALTER_VIEW	10019
43	DROP_VIEW	10019
51	CREATE_PROCEDURE	10024
52	ALTER_PROCEDURE	10024
53	DROP PROCEDURE	10024

Порядок DML-триггеров

- Если для таблицы или представления созданы INSTEAD OF и AFTER-триггеры, реагирующие на одно и то же событие, то выполняется только INSTEAD OF- триггер

Порядок DML-триггеров

- sp_helptrigger

```
exec sp_helptrigger @tablename = 'TEACHER';
```

The screenshot shows a SQL Server Management Studio window with the 'Results' tab selected. The query `exec sp_helptrigger @tablename = 'TEACHER';` has been run, and the results are displayed in a table.

trigger_name	trigger_owner	isupdate	isdelete	isinsert	isafter	isinsteadof	trigger_schema
TR_TEACHER_INS	dbo	0	0	1	1	0	dbo
TR_TEACHER_DEL	dbo	0	1	0	1	0	dbo
TR_TEACHER_UPD	dbo	1	0	0	1	0	dbo
TR_TEACHER	dbo	1	1	1	1	0	dbo
TR_TEACHER_DEL1	dbo	0	1	0	1	0	dbo
TR_TEACHER_DEL2	dbo	0	1	0	1	0	dbo
TR_TEACHER_DEL3	dbo	0	1	0	1	0	dbo

DDL-триггеры

- CREATE TRIGGER
- [schema_name.] trigger_name
- ON { ALL SERVER | DATABASE }
- [WITH { ENCRYPTION | EXECUTE AS clause_name }]
- { FOR | AFTER }
- { event_group | event_type | LOGON }
- AS { batch | EXTERNAL NAME method_name }

DDL-триггеры

- триггеры уровня сервера (ALLSERVER)
- триггеры уровня базы данных (DATABASE)

Триггеры уровня сервера

- Триггеры уровня сервера обрабатывают события сервера СУБД:
 - Создание объектов сервера
 - Изменение объектов сервера
 - Удаление объектов сервера
 - Подключение к серверу

DDL-триггеры

```
use MASTER
go
create trigger DROP_DB
on all server
for DROP_DATABASE
as
    print 'удаление базы данных';
    insert SERVER_EVENT_LOG values(EVENTDATA ());
go
```

Eventdata

```
<EVENT_INSTANCE>
  <EventType>DROP_DATABASE</EventType>
  <PostTime>2014-06-09T15:05:54.027</PostTime>
  <SPID>53</SPID>
  <ServerName>WIN-FPSVQQ7K4B3</ServerName>
  <LoginName>sa</LoginName>
  <DatabaseName>DB1</DatabaseName>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON"
               ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON"
               ENCRYPTED="FALSE" />
    <CommandText>drop database [DB1]; </CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Logon

```
use MASTER
go
create trigger LOGON_SERVER
on all server
with execute as 'sa'
for LOGON
as
    insert SERVER_EVENT_LOG values(EVENTDATA());
go
```

Logon

```
<EVENT_INSTANCE>
  <EventType>LOGON</EventType>
  <PostTime>2014-06-09T20:53:19.510</PostTime>
  <SPID>56</SPID>
  <ServerName>WIN-FPSVQQ7K4B3</ServerName>
  <LoginName>ss</LoginName>
  <LoginType>SQL Login</LoginType>
  <SID>XXBAqoGtbUOpWPCPifiLXQ==</SID>
  <ClientHost>192.168.0.200</ClientHost>
  <IsPooled>0</IsPooled>
</EVENT_INSTANCE>
```

Триггеры уровня базы данных

- Триггеры уровня базы данных – обработка событий, происходящих в рамках базы данных

Триггеры уровня базы данных

```
use BSTU
go
create trigger DDL_DB
on database
for DDL_DATABASE_LEVEL_EVENTS
as
    insert MASTER.DBO.SERVER_EVENT_LOG values(EVENTDATA());
go
```

Триггеры уровня базы данных

```
<EVENT_INSTANCE>
  <EventType>CREATE_TABLE</EventType>
  <PostTime>2014-06-09T22:01:30.650</PostTime>
  <SPID>51</SPID>
  <ServerName>WIN-FPSVQQ7K4B3</ServerName>
  <LoginName>sa</LoginName>
  <UserName>dbo</UserName>
  <DatabaseName>BSTRU</DatabaseName>
  <SchemaName>dbo</SchemaName>
  <ObjectName>t1</ObjectName>
  <ObjectType>TABLE</ObjectType>
  <TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON"
               ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON"
               ENCRYPTED="FALSE" />
    <CommandText>create table t1 ( x int); </CommandText>
  </TSQLCommand>
</EVENT_INSTANCE>
```

Триггеры уровня базы данных

```
create trigger DDL_DB_DROP_TABLE
on database
for DROP_TABLE
as
declare @t varchar(50)=
EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]', 'varchar(50)');
if @t = 'TEACHER'
begin
raiserror( N'удаление таблицы TEACHER запрещено', 16, 1);
rollback;
end;
```

Уровень 1	Уровень 2	Уровень 3
DDL_EVENTS	DDL_DATABASE_LEVEL_EVENTS	<p>ALTER_INSTANCE</p> <p>DDL_DATABASE_SECURITY_EVENTS</p> <p>DDL_DEFAULT_EVENTS</p> <p>DDL_FUNCTION_EVENTS</p> <p>DDL_PROCEDURE_EVENTS</p> <p>DDL_SYNONYM_EVENTS</p> <p>DDL_TABLE_VIEW_EVENTS</p> <p>DDL_TYPE_EVENTS</p> <p>RENAME</p>
	DDL_SERVER_LEVEL_EVENTS	<p>DLL_DATABASE_EVENTS</p> <p>DLL_ENDPOINTS_LEVEL</p> <p>DLL_EVENT_SESSION_EVENTS</p> <p>DLL_MESSAGE_EVENTS</p>

Вопросы?

БАЗЫ ДАННЫХ

Лекция 16 XML

XML

- XML – Extensible Markup Language
- Является подмножеством языка SGML – Standard Generalized Markup Language – метаязыка для определения языков разметки

W3C – стандартизация

- Консорциум Всемирной паутины – World Wide Web Consortium – организация, разрабатывающая и внедряющая технологические стандарты для web
- Глава – Тимоти Джон Бернерс-Ли
- Ок. 15 стандартов утверждены для XML:
 - XML Schema
 - XPath
 - XSLT
 - XQuery

XML-документ

- Текстовый файл
- Древовидная структура
- Теги
- Атрибуты
- Данные

XML

- Элементы
- Символьные данные
- Древовидная структура документа
- Корневой элемент, дочерние элементы и листья
- Атрибуты

```
<?xml version="1.0" encoding="UTF-8"?>
<PersonList Type="Employee">
    <Title> Value="Employee List"</Title>
    <Contents>
        <Employee>
            <Name>Ann Heathers</Name>
            <No>12202</No>
            <Deptno>d3</Deptno>
            <Address>
                <City>Dallas</City>
                <Street>Main St</Street>
            </Address>
        </Employee>
        <Employee>
            <Name>John Dow</Name>
            <No>12216</No>
            <Deptno>d1</Deptno>
            <Address>
                <City>Seattle</City>
                <Street>Abbey Rd</Street>
            </Address>
        </Employee>
    </Contents>
</PersonList>
```

XML-документ

- Начинается специальным тегом с именем `<xml>` (объявление XML)
- Каждый элемент начинается открывающим тегом и завершается закрывающим
- Могут быть вложенные теги

Пространства имен XML

- Тег XML - namespace:name
- URI - Uniform Resource Identifier — унифицированный идентификатор ресурса

```
<Faculty xmlns="http://www.belstu.by/isit"
          xmlns:lib="http://www.belstu.by/library">
    <Name>Book</Name>
    <Feature>
        <lib:Title>Introduction to XML in Databases</lib:Title>
        <lib:Author>A. S. Filkenstein</lib:Author>
    </Feature>
</Faculty>
```

XML

- Правильно построенный документ – well-formed – соответствует синтаксическим правилам XML
- Валидный документ – valid – соответствует правилам описания типа документа (Document Type Definition, DTD)

XML

- наличие корневого элемента
- каждый открывающий тег имеет соответствующий закрывающий тег
- правильное вложение элементов документа
- атрибут должен иметь значение, которое берется в кавычки

```
<?xml version="1.0" encoding="UTF-8"?>
<PersonList Type="Employee">
    <Title> Value="Employee List"</Title>
    <Contents>
        <Employee>
            <Name>Ann Heathers</Name>
            <No>12202</No>
            <Deptno>d3</Deptno>
            <Address>
                <City>Dallas</City>
                <Street>Main St</Street>
            </Address>
        </Employee>
        <Employee>
            <Name>John Dow</Name>
            <No>12216</No>
            <Deptno>d1</Deptno>
            <Address>
                <City>Seattle</City>
                <Street>Abbey Rd</Street>
            </Address>
        </Employee>
    </Contents>
</PersonList>
```

Языки схем

- язык DTD - Document Type Definition
 - Набор правил для структурирования XML
 - Внутренний DTD – часть XML-документа
 - Внешний DTD – адрес URL
- язык XML Schema

DTD – Document Type Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PersonList SYSTEM "D:\DTDS\EMP4.dtd">
<!ELEMENT EmployeeList (Title, Contents)>
<!ELEMENT Title EMPTY>
<!ELEMENT Contents (Employee*)>
<!ELEMENT Employee (Name, No, Deptno, Address)>
<!ELEMENT Name (Fname, Lname)>
<!ELEMENT Fname (#PCDATA)>
<!ELEMENT Lname (#PCDATA)>
<!ELEMENT No (#PCDATA)>
<!ELEMENT Deptno (#PCDATA)>
<!ELEMENT Address (City, Street) >
<!ELEMENT City (#PCDATA)>
<!ELEMENT Street (#PCDATA)>
<!ATTLIST EmployeeList Type CDATA #IMPLIED
Date CDATA #IMPLIED>
<!ATTLIST Title Value CDATA #REQUIRED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<PersonList Type="Employee">
    <Title> Value="Employee List"</Title>
    <Contents>
        <Employee>
            <Name>Ann Heathers</Name>
            <No>12202</No>
            <Deptno>d3</Deptno>
            <Address>
                <City>Dallas</City>
                <Street>Main St</Street>
            </Address>
        </Employee>
        <Employee>
            <Name>John Dow</Name>
            <No>12216</No>
            <Deptno>d1</Deptno>
            <Address>
                <City>Seattle</City>
                <Street>Abbey Rd</Street>
            </Address>
        </Employee>
    </Contents>
</PersonList>
```

DTD – Document Type Definition

- Имя DTD должно соответствовать имени тега корневого элемента XML-документа
- XML-документ нужно связать с соответствующим файлом DTD
- Объявления типов элементов должны начинаться с инструкции ELEMENT
- Порядок элементов XML-документа
- Элементы без подчиненных - #PCDATA
- * наличие элементов (от нуля и больше)
- ? наличие не более одного элемента
- + наличие по крайней мере одного элемента
- Объявление атрибута <!ATTLIST имя атрибута и тип данных>
- #IMPLIED атрибут необязательный, #REQUIRED обязательный

DTD – Document Type Definition

- Атрибут типа ID - определение уникального значения
- Атрибут типа IDREF должен ссылаться на действительный идентификатор, объявленный в этом же документе
- Атрибут типа IDREFS задает список разделенных пробелами строк, на которые ссылаются значения атрибута типа ID

XML Schema

- XML Schema — язык описания структуры XML-документа – предназначен для определения правил, которым должен подчиняться документ
- Создается модель данных документа:
 - словарь (названия элементов и атрибутов)
 - модель содержания (отношения между элементами и атрибутами и их структура)
 - типы данных

Задачи

- преобразование XML в строки реляционных таблиц
- преобразование данных в таблицах в XML

Декомпозиция XML

- sp_xml_prepare_document
- sp_xml_remove_document

```
DECLARE @hdoc INT  
DECLARE @doc VARCHAR(1000)  
SET @doc = '<ROOT>  
    <Employee>  
        <Name>Ann Heathers</Name>  
        <No>12202</No>  
        <Deptno>d3</Deptno>  
        <Address>  
            <City>Dallas</City>  
            <Street>Main St</Street>  
        </Address>  
    </Employee>  
    <Employee>  
        <Name>John Dow</Name>  
        <No>12216</No>  
        <Deptno>d1</Deptno>  
        <Address>  
            <City>Seattle</City>  
            <Street>Abbey Rd</Street>  
        </Address>  
    </Employee>  
</ROOT>'  
EXEC sp_xml_preparedocument @hdoc OUTPUT, @doc'
```

Декомпозиция XML

```
SELECT * FROM OPENXML (@hdoc, '/ROOT/Employee', 1)
WITH (name VARCHAR(20) 'Name',
no INT 'No',
deptno VARCHAR(6) 'Deptno',
address VARCHAR(50) 'Address');
```

The screenshot shows a SQL query being run in a query editor window. The query uses the OPENXML function to decompose an XML document into a table. The results are displayed in a grid titled 'Results'.

	name	no	deptno	address
	Ann Heathers	12202	d3	Dallas Main St
	John Dow	12216	d1	Seattle Abbey Rd

Тип данных XML

- столбцы таблицы
- переменные
- входные или выходные параметры в хранимых процедурах или функциях

Нельзя использовать UNIQUE, PRIMARY KEY или FOREIGN KEY

```
CREATE TABLE xmltab(id INTEGER NOT NULL PRIMARY KEY,  
                     xml_column XML);
```

Представление данных в XML

- RAW – каждая строка РН в строку XML
- AUTO – каждая строка РН в XML-элемент с подчиненными
- PATH – сочетание атрибутной и элементной форм
- EXPLICIT – расширенная форма РН

RAW

```
]SELECT  
    [Title],  
    [FirstName],  
    [MiddleName],  
    [LastName],  
    [AdditionalContactInfo]  
FROM [Person].[Person]  
FOR XML RAW;
```



The screenshot shows a SQL Server Management Studio interface. The top part contains a T-SQL query for selecting XML data from the Person table. The bottom part shows the execution results in a tabular format. The 'Results' tab is selected, displaying the XML output. The XML structure includes a root element 'row' with attributes FirstName='Ken', MiddleName='J', and LastName='Sá...'. There is also a 'Messages' tab visible.

row
<pre><row FirstName='Ken' MiddleName='J' LastName='Sá...'></pre>

RAW

```
<row FirstName="Ken" MiddleName="J" LastName="Sánchez" />
<row FirstName="Terri" MiddleName="Lee" LastName="Duffy" />
<row FirstName="Roberto" LastName="Tamburello" />
<row FirstName="Rob" LastName="Walters" />
<row Title="Ms." FirstName="Gail" MiddleName="A" LastName="Erickson" />
<row Title="Mr." FirstName="Jossef" MiddleName="H" LastName="Goldberg" />
<row FirstName="Dylan" MiddleName="A" LastName="Miller" />
<row FirstName="Diane" MiddleName="L" LastName="Margheim" />
<row FirstName="Gigi" MiddleName="N" LastName="Matthew" />
<row FirstName="Michael" LastName="Raheem" />
<row FirstName="Ovidiu" MiddleName="V" LastName="Cracium" />
<row FirstName="Thierry" MiddleName="B" LastName="D'Hers" />
<row Title="Ms." FirstName="Janice" MiddleName="M" LastName="Galvin" />
<row FirstName="Michael" MiddleName="I" LastName="Sullivan" />
<row FirstName="Sharon" MiddleName="B" LastName="Salavaria" />
<row FirstName="David" MiddleName="M" LastName="Bradley" />
<row FirstName="Kevin" MiddleName="F" LastName="Brown" />
<row FirstName="John" MiddleName="L" LastName="Wood" />
<row FirstName="Mary" MiddleName="A" LastName="Dempsey" />
<row FirstName="Wanida" MiddleName="M" LastName="Benshoof" />
<row FirstName="Terry" MiddleName="J" LastName="Eminhizer" />
<row FirstName="Sariya" MiddleName="E" LastName="Harnpadoungsataya" />
<row FirstName="Mary" MiddleName="E" LastName="Gibson" />
<row Title="Ms." FirstName="Jill" MiddleName="A" LastName="Williams" />
<row FirstName="James" MiddleName="R" LastName="Hamilton" />
<row FirstName="Peter" MiddleName="J" LastName="Krebs" />
<row FirstName="Jo" MiddleName="A" LastName="Brown" />
<row FirstName="Guiv" MiddleName="R" LastName="Gilbert" />
```

AUTO

```
SELECT
    Title,
    FirstName,
    MiddleName,
    LastName,
    AdditionalContactInfo,
    DepartmentID,
    ShiftID,
    EndDate
FROM Person.Person inner join HumanResources.EmployeeDepartmentHistory
on Person.Person.BusinessEntityID= HumanResources.EmployeeDepartmentHistory.BusinessEntityID
FOR XML AUTO;
```

%



Results

Messages

XML_F52E2B61-18A1-11d1-B105-00805F49916B

<Person.Person FirstName="Ken" MiddleName="J" La...

AUTO

```
<HumanResources.EmployeeDepartmentHistory DepartmentID="16" ShiftID="1" />
</Person.Person>
<Person.Person FirstName="Terri" MiddleName="Lee" LastName="Duffy">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="1" ShiftID="1" />
</Person.Person>
<Person.Person FirstName="Roberto" LastName="Tamburello">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="1" ShiftID="1" />
</Person.Person>
<Person.Person FirstName="Rob" LastName="Walters">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="1" ShiftID="1" EndDate="2004-06-01" />
</Person.Person>
<Person.Person FirstName="Rob" LastName="Walters">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="2" ShiftID="1" />
</Person.Person>
<Person.Person Title="Ms." FirstName="Gail" MiddleName="A" LastName="Erickson">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="1" ShiftID="1" />
</Person.Person>
<Person.Person Title="Mr." FirstName="Jossef" MiddleName="H" LastName="Goldberg">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="1" ShiftID="1" />
</Person.Person>
<Person.Person FirstName="Dylan" MiddleName="A" LastName="Miller">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="6" ShiftID="1" />
</Person.Person>
<Person.Person FirstName="Diane" MiddleName="L" LastName="Margheim">
    <HumanResources.EmployeeDepartmentHistory DepartmentID="6" ShiftID="1" />
</Person.Person>
```

PATH

```
SELECT DepartmentID "@Department",
       Title "EmpName/Title",
       FirstName "EmpName/FirstName",
       MiddleName "EmpName/MiddleName",
       LastName "EmpName/LastName"

  FROM Person.Person inner join HumanResources.EmployeeDepartmentHistory
    on Person.Person.BusinessEntityID= HumanResources.EmployeeDepartmentHistory.BusinessEntityID
   FOR XML path;
```

Шаблон	Описание
eN	XML-элемент eN
@aN	XML-атрибут aN
eN1/eN2	XML-элемент eN1 и вложенный элемент eN2
eN/@aN	XML-элемент eN и атрибут aN

PATH

```
<row Department="1">
  <EmpName>
    <FirstName>Roberto</FirstName>
    <LastName>Tamburello</LastName>
  </EmpName>
</row>
<row Department="1">
  <EmpName>
    <FirstName>Rob</FirstName>
    <LastName>Walters</LastName>
  </EmpName>
</row>
<row Department="2">
  <EmpName>
    <FirstName>Rob</FirstName>
    <LastName>Walters</LastName>
  </EmpName>
</row>
<row Department="1">
  <EmpName>
    <Title>Ms.</Title>
    <FirstName>Gail</FirstName>
    <MiddleName>A</MiddleName>
    <LastName>Erickson</LastName>
  </EmpName>
</row>
```



Директивы

- TYPE
- ROOT
- ELEMENTS

Директива TYPE

- сохранять результат реляционного запроса как XML-документ или фрагмент типа данных XML

```
DECLARE @x xml;
SET @x = (SELECT FirstName FROM Person.Person
FOR XML AUTO, TYPE);
SELECT @x;
```

The screenshot shows the SQL Server Management Studio interface. The top bar has tabs for 'Results' and 'Messages'. The 'Results' tab is selected. Below it, there's a row for '(No column name)'. Underneath, the results of the query are displayed as XML fragments:

```
<Person.Person FirstName="Syed" /><Person.Person...
```

```
<Person.Person FirstName="Kim" />
<Person.Person FirstName="Kim" />
<Person.Person FirstName="Kim" />
<Person.Person FirstName="Hazem" />
<Person.Person FirstName="Sam" />
<Person.Person FirstName="Humberto" />
<Person.Person FirstName="Gustavo" />
<Person.Person FirstName="Pilar" />
<Person.Person FirstName="Pilar" />
<Person.Person FirstName="Aaron" />
<Person.Person FirstName="Adam" />
<Person.Person FirstName="Alex" />
<Person.Person FirstName="Alexandra" />
<Person.Person FirstName="Allison" />
<Person.Person FirstName="Amanda" />
<Person.Person FirstName="Amber" />
<Person.Person FirstName="Andrea" />
<Person.Person FirstName="Angel" />
<Person.Person FirstName="Bailey" />
<Person.Person FirstName="Ben" />
<Person.Person FirstName="Blake" />
<Person.Person FirstName="Carla" />
<Person.Person FirstName="Carlos" />
<Person.Person FirstName="Charles" />
<Person.Person FirstName="Chloe" />
<Person.Person FirstName="Connor" />
<Person.Person FirstName="Courtney" />
<Person.Person FirstName="Dalton" />
<Person.Person FirstName="Devin" />
.
```

Директива ROOT

- Добавление к результирующему набору XML одного элемента верхнего уровня

```
DECLARE @x xml;
SET @x = (SELECT FirstName FROM Person.Person
FOR XML AUTO, ROOT ('AllPersons'));
SELECT @x;
```

Директива ELEMENTS

```
DECLARE @x xml;
SET @x = (SELECT FirstName FROM Person.Person
FOR XML AUTO, elements);
SELECT @x;
```

```
<Person>
  <FirstName>Syed</FirstName>
</Person>
<Person>
  <FirstName>Catherine</FirstName>
</Person>
<Person>
  <FirstName>Kim</FirstName>
</Person>
<Person>
  <FirstName>Kim</FirstName>
</Person>
<Person>
  <FirstName>Hazem</FirstName>
</Person>
<Person>
  <FirstName>Sam</FirstName>
</Person>
<Person>
  <FirstName>Humberto</FirstName>
</Person>
- - -
```

XML Schema

- встроенный механизм, позволяющий проверять на корректность XML-документы
- **XML SCHEMA COLLECTION**
- может содержать один или более XML-SCHEMA-документов
- для XML-столбца можно указать имя коллекции схем

XML Schema

```
CREATE XML SCHEMA COLLECTION EmployeeSchema AS  
N'<?xml version="1.0" encoding="UTF-16"?>  
  <xsd:schema elementFormDefault="unqualified"  
    attributeFormDefault="unqualified"  
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" >  
    <xsd:element name="employees">  
      <xsd:complexType mixed="false">  
        <xsd:sequence>  
          <xsd:element name="fname" type="xsd:string"/>  
          <xsd:element name="lname" type="xsd:string"/>  
          <xsd:element name="department" type="xsd:string"/>  
          <xsd:element name="salary" type="xsd:integer"/>  
          <xsd:element name="comments" type="xsd:string"/>  
        </xsd:sequence>  
      </xsd:complexType>  
    </xsd:element>  
  </xsd:schema>';
```

Индексирование XML

- Первичный XML-индекс:
 - Индексируются все теги, значения и пути
 - Используется для возвращения скалярных значений или поддеревьев

```
- CREATE PRIMARY XML INDEX index_xml_column ON xmltab(xml_column);|
```

Индексирование XML

- Три типа вторичных типа XML-индексов:
 - FOR PATH — по структуре
 - FOR VALUE — по значениям элементов и атрибутов
 - FOR PROPERTY — по свойствам

```
CREATE XML INDEX i_xmlcolumn_path ON xmltab(xml_column)
USING XML INDEX index_xml_column FOR PATH;
```

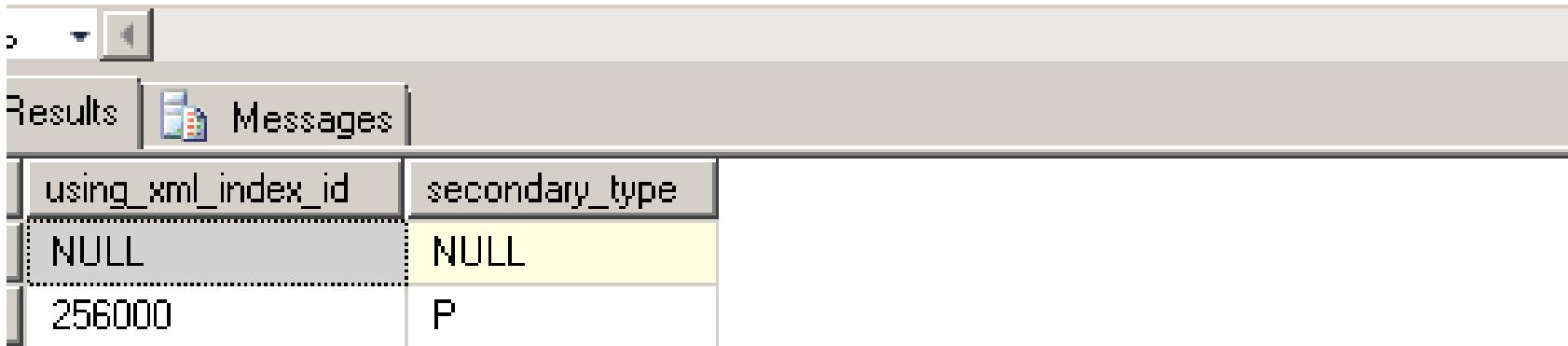
Индексирование XML

- Не могут быть составными
- Не могут быть кластеризованными

Индексирование XML

- sys.xml_indexes

```
SELECT USING_XML_INDEX_ID, SECONDARY_TYPE FROM SYS.XML_INDEXES;
```



The screenshot shows a SQL query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and displays the following data:

using_xml_index_id	secondary_type
NULL	NULL
256000	P

Представления каталога

```
SELECT * FROM SYS.XML_SCHEMA_ATTRIBUTES;
```

```
SELECT * FROM SYS.XML_SCHEMA_ELEMENTS;
```

```
SELECT * FROM SYS.XML_SCHEMA_COMPONENTS;
```

The screenshot shows a SQL Server Management Studio window with the 'Results' tab selected. The query results are displayed in a grid table.

xml_component_id	xml_collection_id	xml_namespace_id	is_qualified	name	symbol
65554	65548	1	1	employees	E
65557	65548	1	0	fname	E
65558	65548	1	0	lname	E
65559	65548	1	0	department	E
65560	65548	1	0	salary	E
65561	65548	1	0	comments	E

Запрос данных из XML

- язык запросов Xpath
 - XML Path Language — язык запросов к элементам XML-документа
- язык запросов Xquery
 - XQuery — язык запросов, разработанный для обработки данных в формате XML

Запрос данных XPath

- для доступа к элементам и атрибутам XML-документа
 - Дочерние элементы узла `/customer/*`
 - Все атрибуты узла `/customer/!?*`
 - Чтобы вернуть только покупателей из региона Dallas
`/customer[@region = "Dallas"]`

Оси XPath

Имя	Описание
ancestor	Содержит родительский узел и все вышестоящие родительские узлы вплоть до корневого узла
ancestor-or-self	Содержит узлы-предки вместе с самим контекстным узлом, вплоть до корневого узла
attribute	Содержит атрибуты контекстного узла, если контекстный узел — узел элемента
child	Содержит дочерние узлы
descendant	Содержит дочерние и дальнейшие узлы
descendant-or-self	Содержит сам контекстный узел и все его дочерние и дальнейшие узлы
following	Содержит все узлы того же документа, к которому принадлежит контекстный узел, которые находятся после текущего контекстного узла в порядке документа, но не включает никаких потомков, пространства имен или узлов атрибутов
following-sibling	То же, что и following, но содержит все узлы, которые имеют того же родителя, что и контекстный узел
namespace	Содержит пространство имен контекстного узла, до тех пор, пока контекстный узел является элементом
parent	Содержит родительский узел контекстного узла Корневой элемент не имеет родителя

Запрос данных

- `query()` – выполнение запроса
- `exist()` – проверка существования
- `value()` – возвращает значение атрибута
- `nodes()` – возвращает набор узлов
- `modify()` – изменение документа
 - `insert`
 - `delete`
 - `replace value of`

Запрос данных

```
SELECT xml_column.query('/PersonList/Title')
FROM xmltab
FOR XML AUTO, TYPE;
```

The screenshot shows a database interface with a query window and a results pane. The query window contains the T-SQL code above. The results pane has tabs for 'Results' and 'Messages'. The 'Results' tab is selected, showing the output of the query. The output is an XML document with two root nodes, each containing a single 'Title' element with the value 'Employee List'.

Results	Messages
(No column name)	
<xmltab><Title> Value="Employee List"></Title>	
<xmltab><Title> Value="Employee List"></Title>	

```
Ǝ<xmltab>
└─<Title> Value="Employee List"&gt;</Title>
└─<xmltab>
└─<Title> Value="Employee List"&gt;</Title>
└─<xmltab>
```

Запрос данных

```
SELECT xml_column.exist('/PersonList/Title/@Value=EmployeeList') AS a  
FROM xmltab  
FOR XML AUTO, TYPE;
```

The screenshot shows the SQL Server Management Studio interface with the following details:

- Query Editor:** The top pane contains the XML query shown above.
- Results Pane:** The bottom pane displays the results of the query. It has tabs for "Results" and "Messages".
- Results Data:** The "Results" tab shows one row of data:
 - Column 1: (No column name)
 - Column 2: <xmltab a="1" /><xmltab a="1" />

Изменение данных

```
DECLARE @myXMLDoc xml;
SET @myXMLDoc = '<Root>
    <ProductDescription ProductID="1" ProductName="Road Bike">
        <Features>
            </Features>
    </ProductDescription>
    <ProductDescription ProductID="2" ProductName="Mountain Bike">
        <Features>
            </Features>
    </ProductDescription>
</Root>' ;
SELECT @myXMLDoc;

-- insert
SET @myXMLDoc.modify(
insert <Maintenance>3 year parts and labor extended maintenance is available</Maintenance>
into (/Root/ProductDescription/Features)[1]) ;
SELECT @myXMLDoc ;

-- update
SET @myXMLDoc.modify(
 replace value of (/Root/ProductDescription/@ProductName)[1]
 with      "The new bike"  );
SELECT @myXMLDoc ;

-- delete
SET @myXMLDoc.modify(
    delete /Root/ProductDescription/@ProductName
')
SELECT @myXMLDoc;
```

Вопросы?