**Вопросы к экзамену по дисциплине «Базы данных»**для студентов специальности

1-40 01 01 «Программное обеспечение информационных технологий»

1. Основные понятия теории баз данных: база данных, система управления базами данных, основные требования к информации в БД.

**База данных** – это совокупность взаимосвязанных данных

* Хранилище динамически обновляемой информации
* Информация отражает состояние некоторой предметной области (объекта) и должна быть полезной, точной, актуальной и непротиворечивой
* Информация представлена в вид:
  + 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

**СУБД** – это программная реализация технологии хранения, извлечения, обновления и обработки данных в базе 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

1. Модели данных, основная терминология реляционных баз данных.

**Модели:**

* Иерархическая



* Сетевая



* Реляционная
* Основана на теории множеств
* 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:**

* 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

**Аспекты (?):**

* 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)

1. Нормализация таблиц базы данных. Нормальные формы таблиц.

**Нормализация данных** – процесс преобразования таблиц базы данных к нормальной форме

**Шесть** нормальных форм – 1NF, 2NF, ... 6NF.

Широкое практическое применение имеют формы 1NF, 2NF, 3NF

**1НФ:**

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

**Алгоритм:**

* Устранить повторяющиеся группы в отдельных таблицах
* Создать отдельную таблицу для каждого набора связанных данных

Идентифицировать каждый набор связанных данных с помощью первичного ключа

**2НФ:**

* 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

**Алгоритм:**

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

**3НФ:**

* Таблица находится во второй нормальной форме
* Отсутствуют транзитивные зависимости

Пример:

Студент – Группа – Факультет – Университет

#### Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.  
  
Например, есть таблица «Автомобили»:

|  |  |
| --- | --- |
| Фирма | Модели |
| BMW | M5, X5M, M1 |
| Nissan | GT-R |

Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным. Преобразуем таблицу к 1НФ:

|  |  |
| --- | --- |
| Фирма | Модели |
| BMW | M5 |
| BMW | X5M |
| BMW | M1 |
| Nissan | GT-R |

#### Вторая нормальная форма

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК).  
  
Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.  
  
Например, дана таблица:

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Фирма | Цена | Скидка |
| M5 | BMW | 5500000 | 5% |
| X5M | BMW | 6000000 | 5% |
| M1 | BMW | 2500000 | 5% |
| GT-R | Nissan | 5000000 | 10% |

Таблица находится в первой нормальной форме, но не во второй. Цена машины зависит от модели и фирмы. Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

|  |  |  |
| --- | --- | --- |
| Модель | Фирма | Цена |
| M5 | BMW | 5500000 |
| X5M | BMW | 6000000 |
| M1 | BMW | 2500000 |
| GT-R | Nissan | 5000000 |

|  |  |
| --- | --- |
| Фирма | Скидка |
| BMW | 5% |
| Nissan | 10% |

#### Третья нормальная форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.  
  
Рассмотрим таблицу:

|  |  |  |
| --- | --- | --- |
| Модель | Магазин | Телефон |
| BMW | Риал-авто | 87-33-98 |
| Audi | Риал-авто | 87-33-98 |
| Nissan | Некст-Авто | 94-54-12 |

Таблица находится во 2НФ, но не в 3НФ.  
В отношении атрибут «Модель» является первичным ключом. Личных телефонов у автомобилей нет, и телефон зависит исключительно от магазина.  
Таким образом, в отношении существуют следующие функциональные зависимости: Модель → Магазин, Магазин → Телефон, Модель → Телефон.  
Зависимость Модель → Телефон является транзитивной, следовательно, отношение не находится в 3НФ.  
В результате разделения исходного отношения получаются два отношения, находящиеся в 3НФ:

|  |  |
| --- | --- |
| Магазин | Телефон |
| Риал-авто | 87-33-98 |
| Некст-Авто | 94-54-12 |

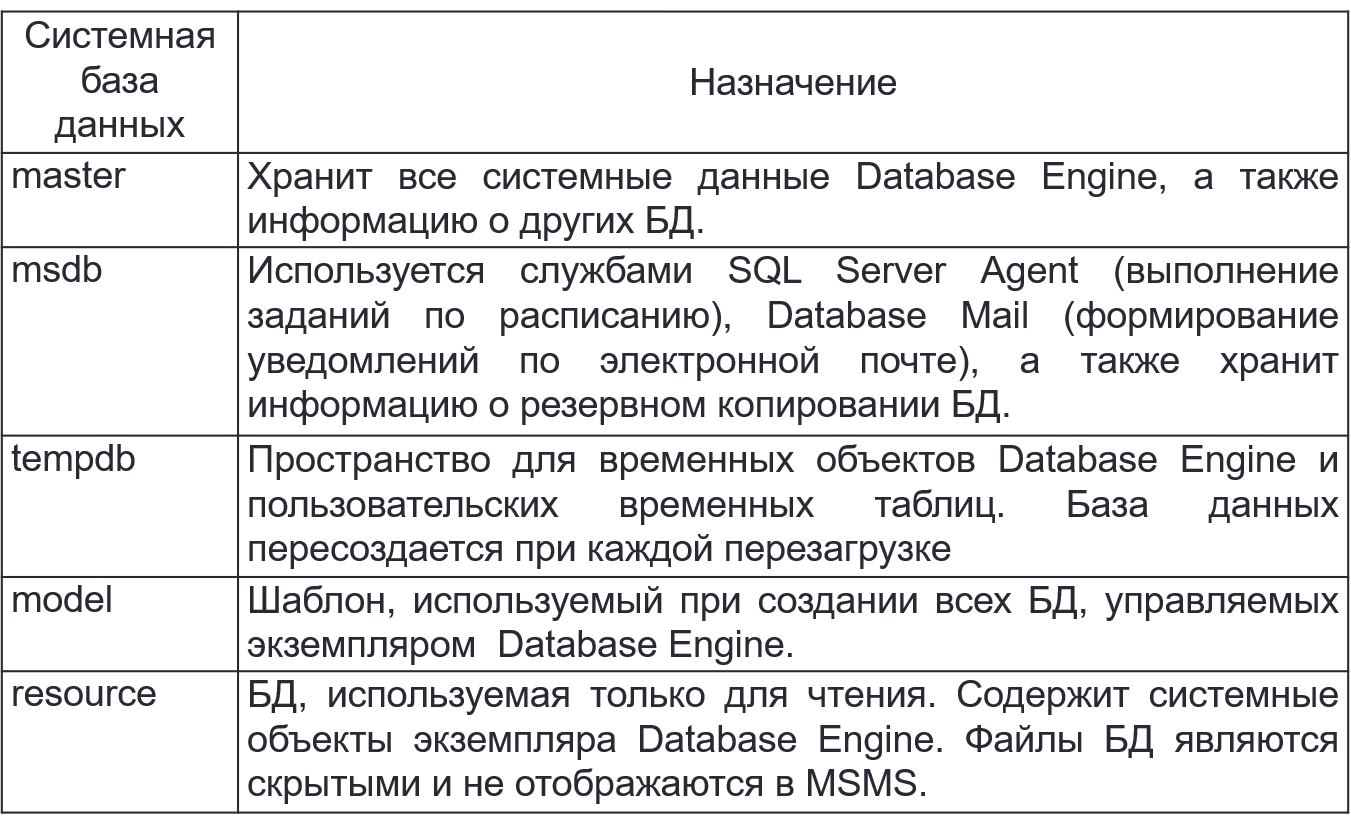
|  |  |
| --- | --- |
| Модель | Магазин |
| BMW | Риал-авто |
| Audi | Риал-авто |
| Nissan | Некст-Авто |

1. Язык SQL. Группы операторов SQL.

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

* **DDL** - Data Definition Language - язык определения данных (**создания**, **удаления** и **изменения** объектов БД или сервера СУБД, create, alter, drop)
* **DML** - Data Manipulation Language - язык манипулирования данными (для **работы** **со строками таблиц,** select, insert, update, delete)
* **TCL** - Transaction Control Language - язык управления транзакциями (для **управления транзакциями,** begin tran, save tran, commit, rollback)
* **DCL** - Data Control Language - язык управления данными (для **управления процессом авторизации,** grant, revoke, deny)

1. Системные базы данных: master, msdb, model, tempdb.



1. Структура файлов базы данных.

**Основные файлы БД:**

* страницы
* экстенты
* файлы
* файловые группы
* Основной единицей хранилища данных является страница
* Размер страницы постоянен и составляет 8 Кб
* Каждая страница имеет заголовок размером в 96 байтов для системной информации
* Строки данных размещаются на странице сразу же после заголовка
* Виды страниц:
  + страницы данных
  + страницы индексов
* 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
* **Экстент** – физическая единица дискового пространства, используемая для выделения памяти для таблиц и индексов
* Размер экстента составляет 8 последовательно расположенных страниц или иначе 64 Кбайт
* Все страницы данных имеют фиксированный размер (8 Кб) и состоят из следующих частей:
  + заголовка страницы (96 байтов)
  + пространства для данных
  + таблицы смещений строк

1. Создание файлов базы данных. Файловые группы. Создание таблиц в файловой группе. Дисковое хранение файлов базы данных. Страницы, экстенты.

* 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 БД**

* 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

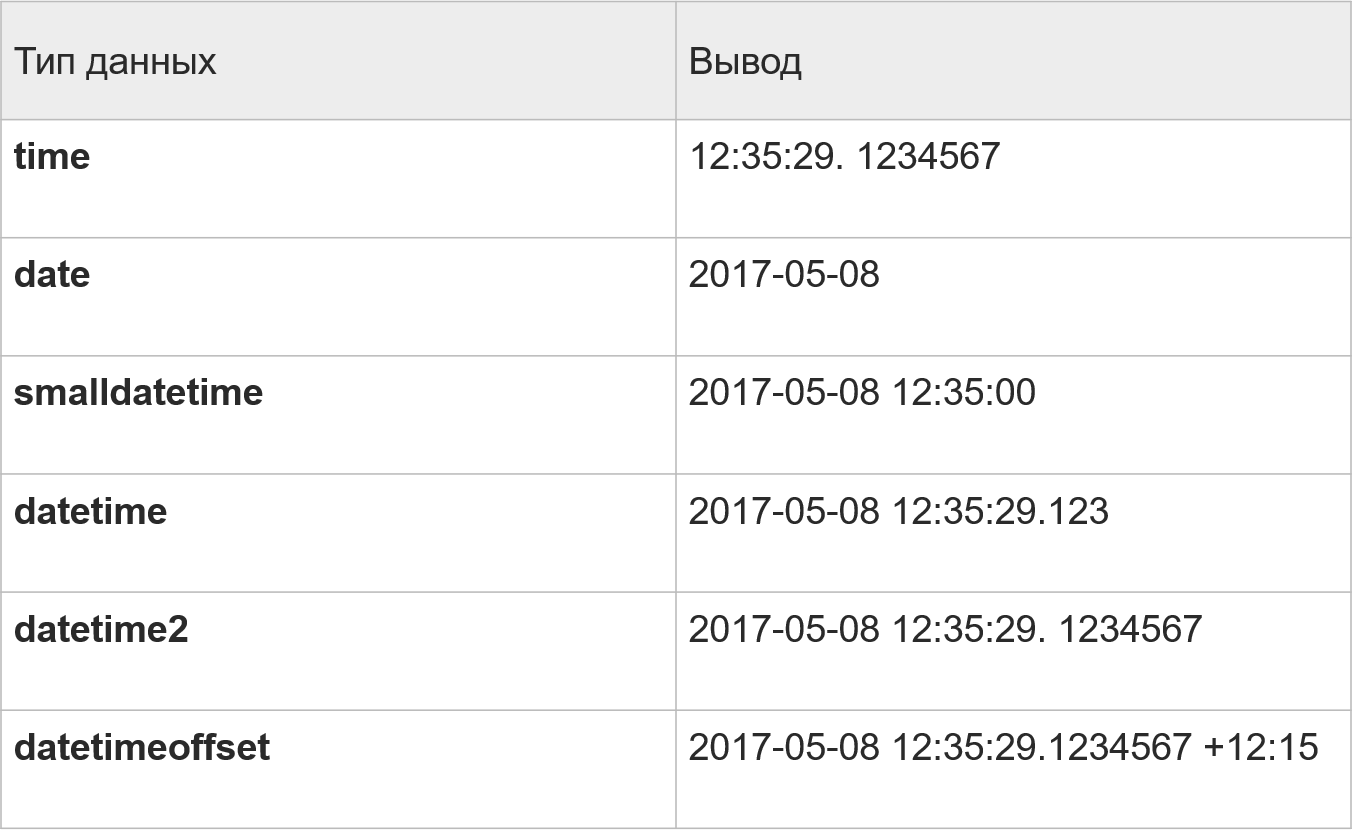
**Файловые группы:**

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

1. Типы данных Microsoft SQL Server.

**Типы:**

* Числовые
  + Точные ((tiny/small/big)int, bit, decimal, numeric)
  + Приближенные (float, real (синоним float(24)))
* Денежные (money (8), smallmoney(4))
* Символьные ((n(юникод))(var(переменная длина))char)
* Дата и время (date, time, smalldatetime, datetime (точность до тысячных секунды), datetime2 (точность до сотен наносекунд), datetimeoffset (хрен знает, в чём отличие от datetime2))



**Двоичные (**binary, varbinary**):**

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

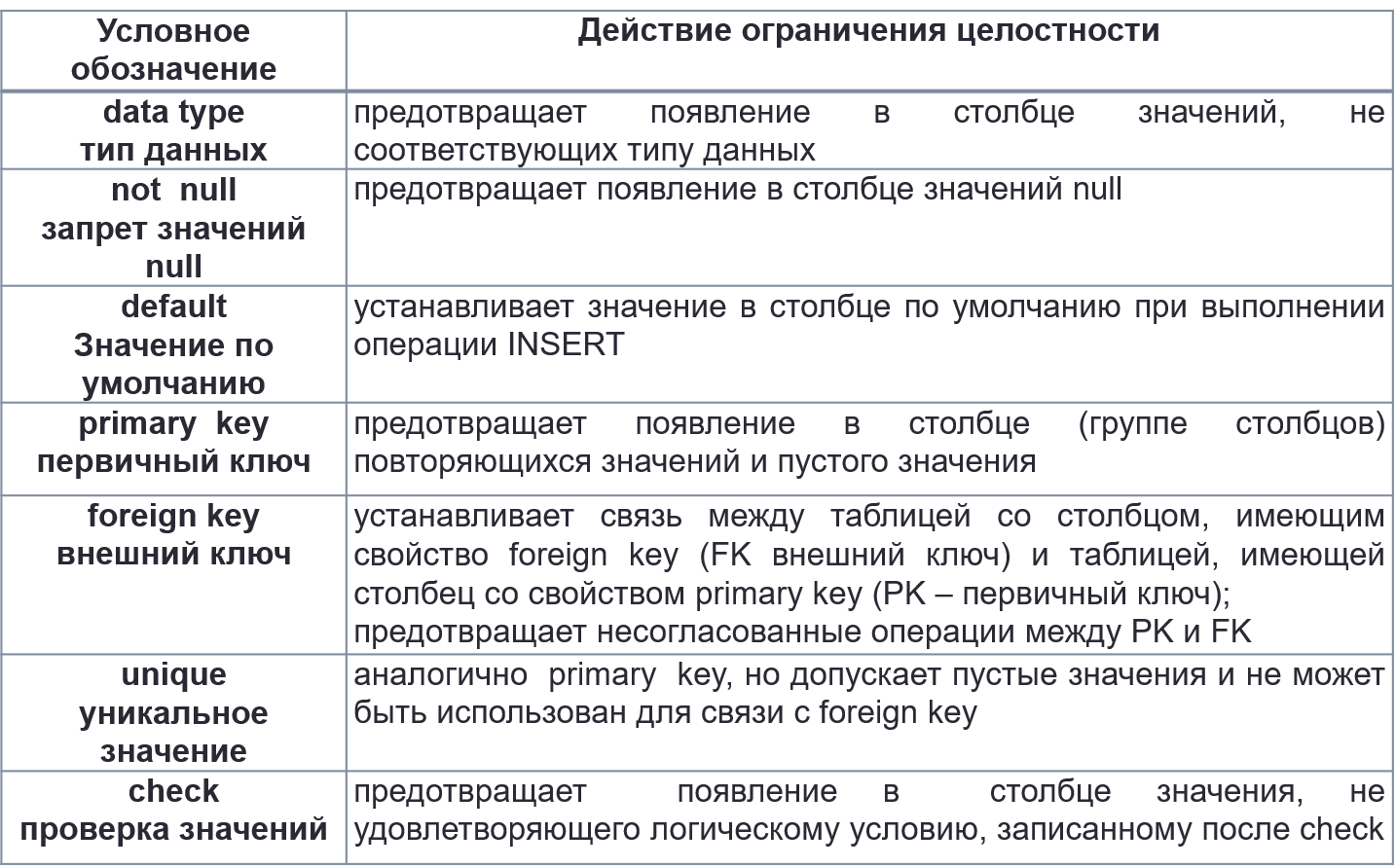
**Прочие:**

* TIMESTAMP
* UNIQUEIDENTIFIER
* xml
* hierarchyid
* Пространственные (geography, geometry)
* FILESTREAM
* sqlvariant
* text, ntext, image

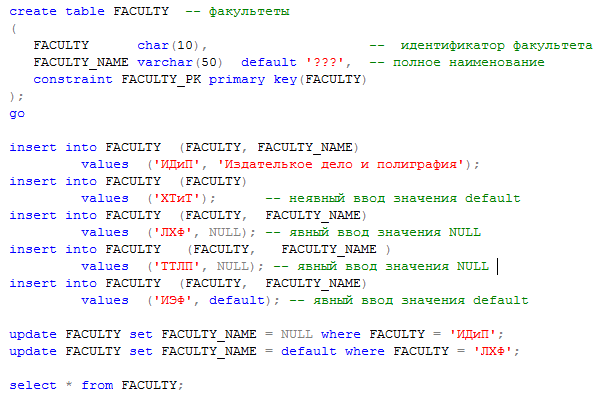
1. Таблицы. Создание, изменение и удаление таблиц.
2. Таблицы. Локальные и глобальные временные таблицы.

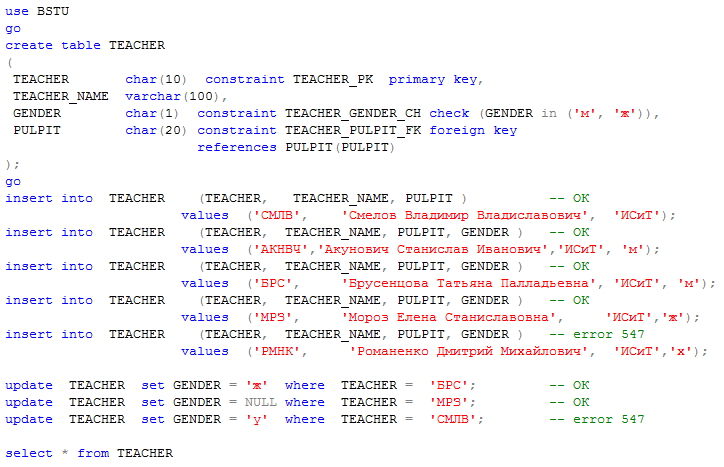
* Временные таблицы создаются для временного хранения результатов SELECT-запросов
* Локальные временные таблицы
  + имена, начинаются с символа #
  + доступны только пользователю, ее создавшему
  + автоматически удаляется при отключении пользователя
* Глобальные временные таблицы
  + имена, начинаются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

1. Ограничения целостности. Создание, изменение и удаление ограничений целостности.



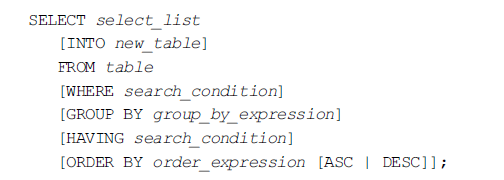
* Для ограничений целостности
  + PRIMARY KEY
  + FOREIGN KEY
  + UNIQUE
  + CHECK
* может быть задано **имя**
* Если это имя не задано, при создании таблицы сервер назначает ограничениям собственные имена
* Not null, default, check (



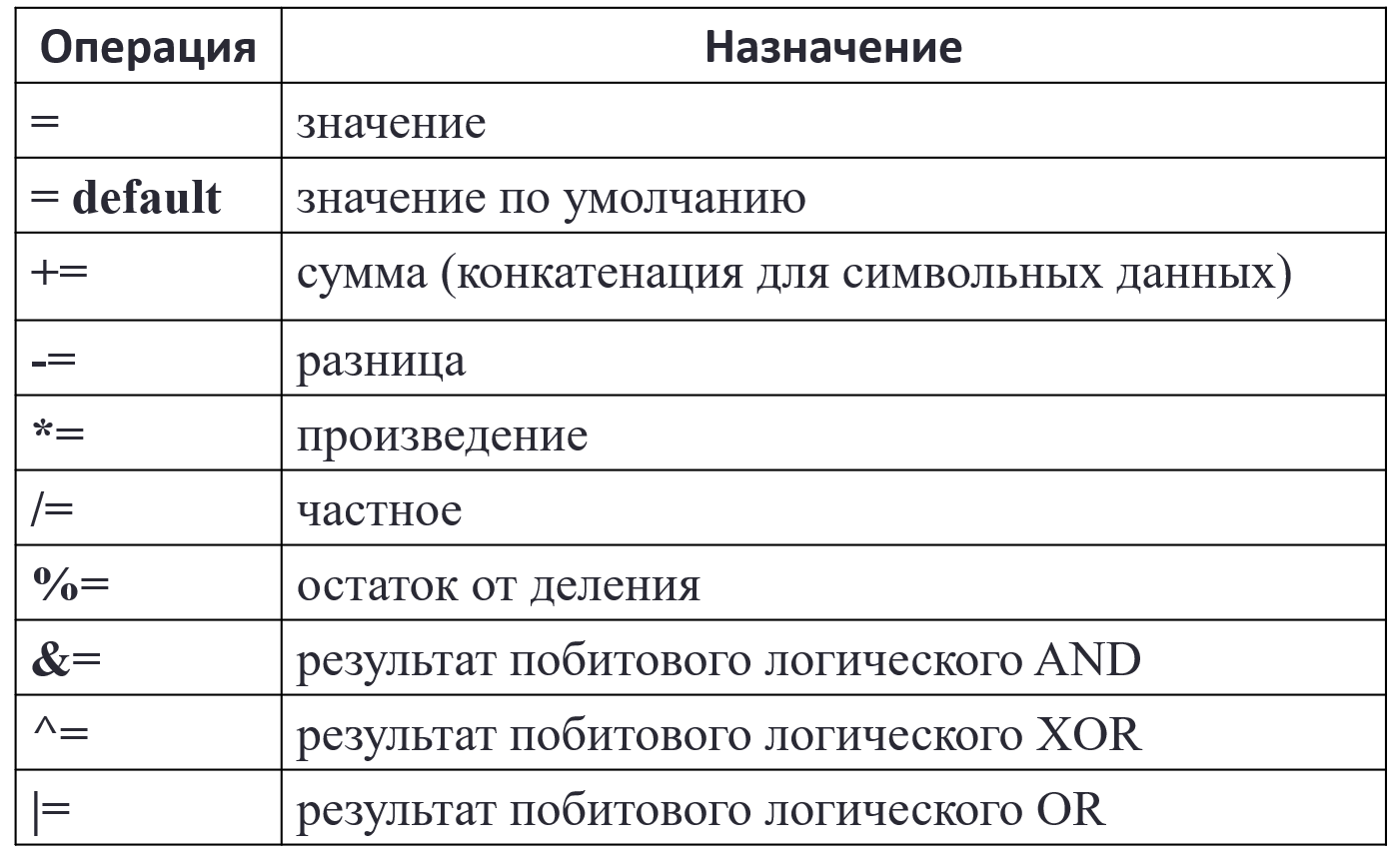


)

1. Операторы DML: SELECT, INSERT, UPDATE, DELETE.

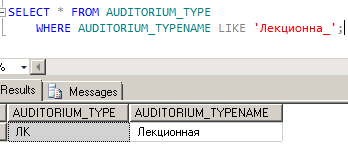


**Update**:



1. Операторы DDL: CREATE, ALTER, DROP.
2. Оператор SELECT. Секции FROM, WHERE, GROUP BY, HAVING, ORDER BY, DISTINCT, TOP.

**Where:**

* Сравнение строк (CHAR, VARCHAR, NCHAR и NVARCAR) 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
* Приоритет выполнения:
  + оператор NOT
  + оператор AND
  + оператор OR
* (not) in/ between
* is null
* like (% для любого количества символов, \_ для одного символа ())

**Group by:**

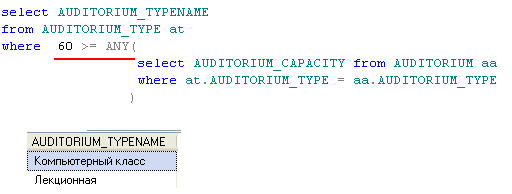
* Каждый столбец в списке выборки запроса также должен присутствовать в предложении GROUP BY
* Не распространяется на константы и столбцы, являющиеся частью агрегатной функции
  + MIN
  + MAX
  + SUM
  + AVG
  + COUNT
* Последовательность имен столбцов в GROUP BY не обязательно должна быть такой же,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

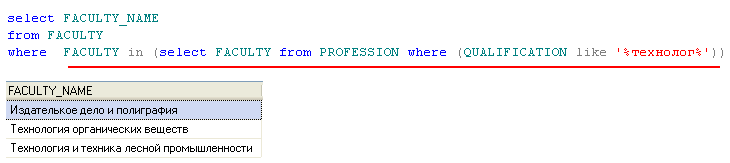
1. Подзапросы. Конструкции IN, EXISTS, ALL, ANY, NOT.

Подзапрос – конструкция SELECT, расположенная в другой конструкции SELECT

Внешняя конструкция называется внешним запросом, внутренняя – вложенным. Сначала выполняется внутренний запрос

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





SELECT DISTINCT s.Name

FROM Sales.Store AS s

WHERE **EXISTS**

(SELECT \*

FROM Purchasing.Vendor AS v

WHERE s.Name = v.Name) ;

GO

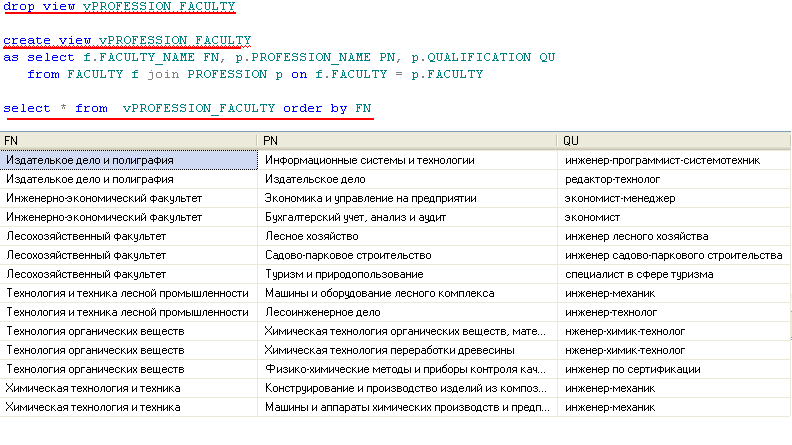
1. Оператор SELECT. Группировка данных.
2. Оператор SELECT. Использование агрегатных функций.
3. Оператор SELECT. Группировка данных с использованием CUBE, ROLLUP

**Виды группировок**:

* GROUP BY
* GROUP BY HAVING
* GROUP BY WITH CUBE
* GROUP BY WITH ROLLUP
* Оператор ROLLUP добавляет суммирующую строку в результирующий набор
* CUBE похож на ROLLUP за тем исключением, что CUBE добавляет суммирующие строки для каждой комбинации групп

1. Представления. Создание, изменение и удаление представлений.

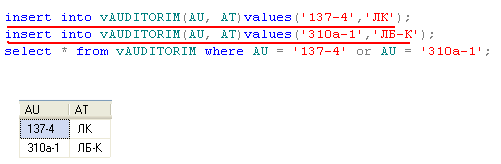
**Представление** – поименованный SELECT-запрос

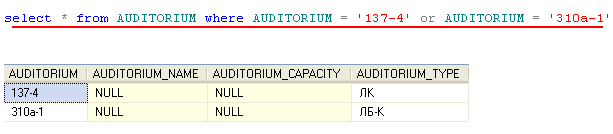


* Упрощение
* Механизм безопасности
* Обеспечение обратной совместимости

Использование операций DML-операций

1. Представления. Операции DML над представлениями. Использование представлений с указанием WITH CHECK OPTION.





* Любая операция выполняется над базовой таблицей
* Все существующие **ограничения** **целостности** базовой таблицы наследуются представлением
* Любые изменения, в том числе инструкции UPDATE, INSERT и DELETE, должны ссылаться на столбцы только одной базовой таблицы
* Изменяемые в представлении столбцы должны непосредственно ссылаться на данные столбцов базовой таблицы. Столбцы нельзя сформировать каким-либо другим образом
* WITH CHECK OPTION является параметром CREATE VIEW. Параметр WITH CHECK OPTION назначается для того, чтобы гарантировать, что все запросы UPDATE и INSERT удовлетворяют условиям в определении представления. Если они не удовлетворяют условию, запрос UPDATE или INSERT возвращает ошибку

1. Представления. Использование представлений с указанием 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
* Предложение SCHEMABINDING привязывает представление к схеме таблицы, по которой оно создается. Когда это предложение указывается, имена объектов баз данных в инструкции SELECT должны состоять из двух частей, т.е. в виде schema.db\_object, где schema - владелец, а db\_object может быть таблицей, представлением или определяемой пользователем функцией.
* Любая попытка модифицировать структуру представлений или таблиц, на которые ссылается созданное таким образом представление, будет неудачной. Чтобы такие таблицы или представления можно было модифицировать (инструкцией ALTER) или удалять (инструкцией DROP), нужно удалить это представление или убрать из него предложение SCHEMABINDING.

1. Функции преобразования типов данных CAST и CONVERT.

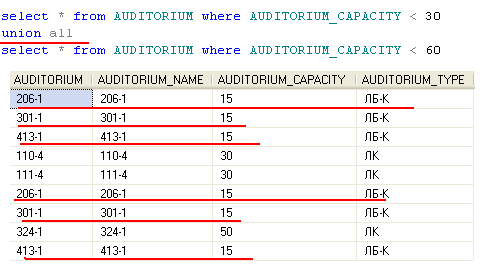
Cast(выражение as тип\_данных), convert(тип\_данных, выражение, [стиль]). Стиль позволяет по-разному преобразовывать выражения, например, некоторые значения для форматирования данных типа money в строку:

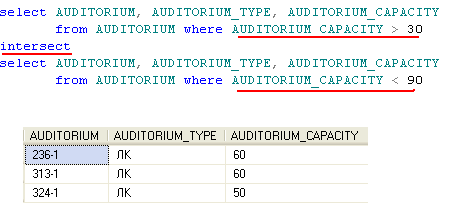
0 - в дробной части числа остаются только две цифры (по умолчанию)

1 - в дробной части числа остаются только две цифры, а для разделения разрядов применяется запятая

2 - в дробной части числа остаются только четыре цифры

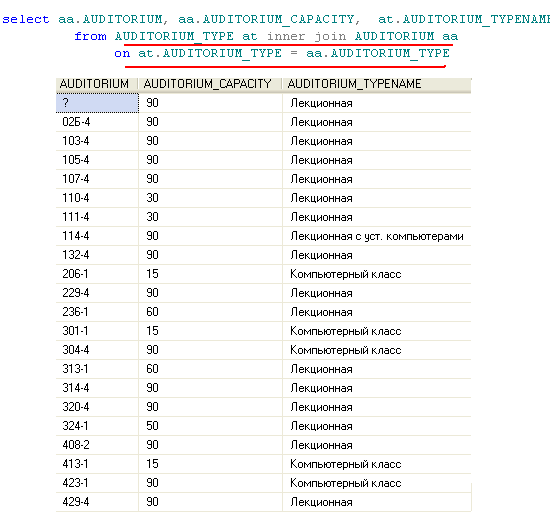
1. Операторы работы с множествами UNION (ALL), INTERSECT, EXCEPT.





1. Соединение таблиц. Внутреннее соединение INNER JOIN.

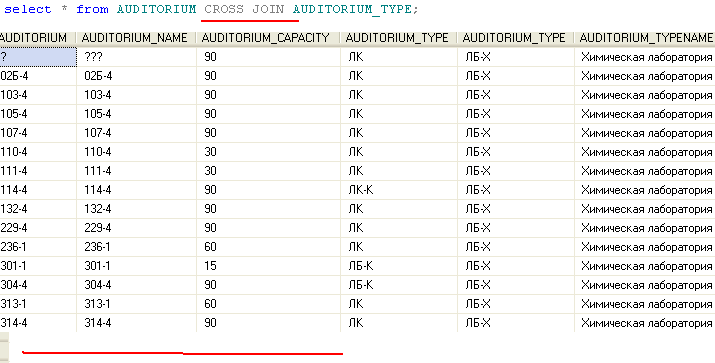
* **Естественное соединение** (Внутреннее соединение содержит только те строки одной таблицы, для которых имеются соответствующие строки в другой таблице)



* **Тета-соединение** (соединение, в котором используется общее условие сравнения столбцов соединения), **самосоединение** (таблица соединяется сама с собой, столбец таблицы сравнивается сам с собой) и **полусоединение** (возвращает набор всех строк из одной таблицы, для которой в другой таблице есть одно или несколько совпадений)

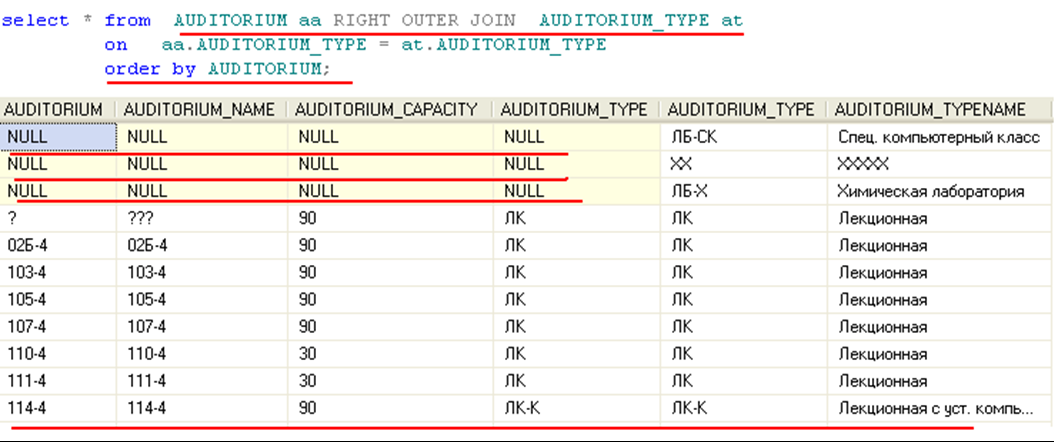
1. Ортогональное соединение CROSS JOIN.

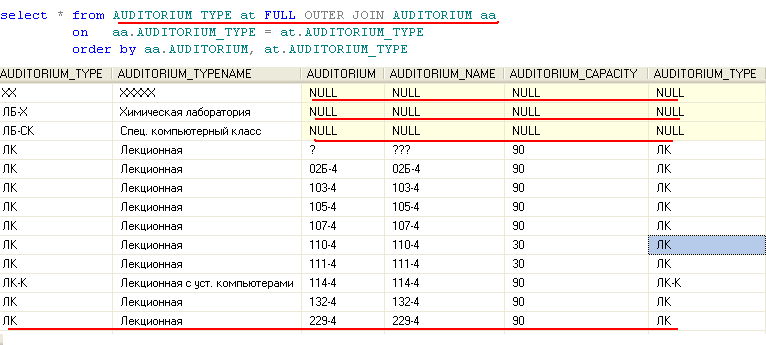
* Декартово произведение (перекрестное соединение) (Ортогональное соединение, перекрестное соединение. CROSS JOIN. Декартово произведение соединяет каждую строку первой таблицы с каждой строкой второй таблицы. Количество строк n × m)



1. Внешние соединения: LEFT(RIGHT) OUTER JOIN, FULL OUTER JOIN.

* Внешнее соединение (результирующий набор содержит все строки одной таблицы и те из второй таблицы, для которых имеются соответствующие строки в первой таблице)





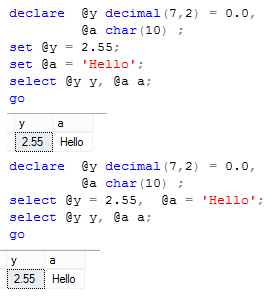
1. Язык T-SQL. Пакеты. Объявление переменных.

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

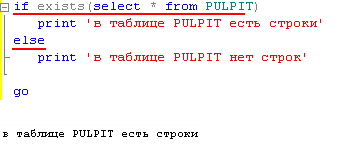
* DECLARE - объявление используемых переменных
* Для каждой переменной указывается имя и тип
* Областью видимости переменной являются все инструкции между ее объявлением и концом пакета или хранимой процедуры, где она объявлена.

1. Язык T-SQL. Операторы присвоения.

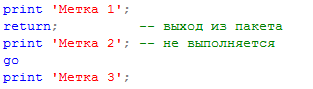
* инициализировать в DECLARE
* присвоить значение SET
* присвоить значение SELECT

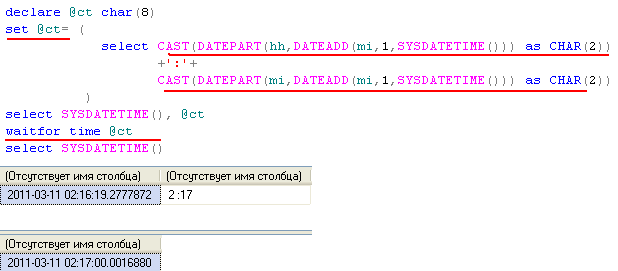


1. Язык T-SQL. Операторы print, if-else, case.



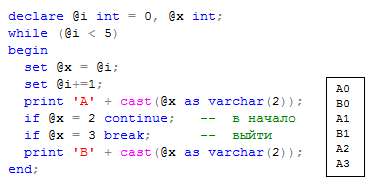
1. Язык T-SQL. Операторы begin-end, waitfor и return.



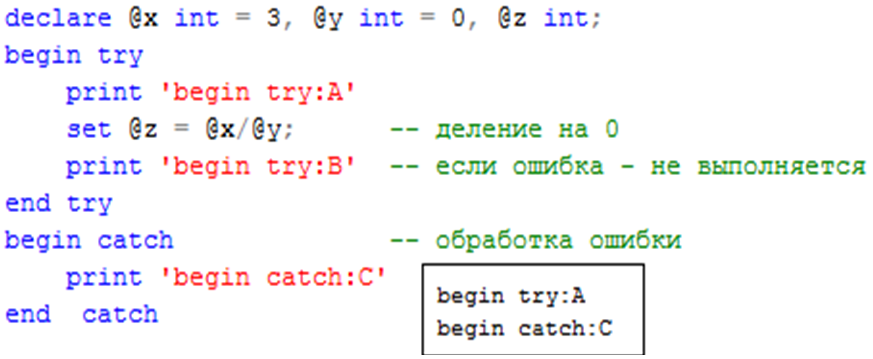


Waitfor блокирует выполнение пакета, хранимой процедуры или транзакции, пока не наступит указанное время, не истечет заданный интервал времени либо заданная инструкция не изменит или не возвратит по крайней мере одну строку.

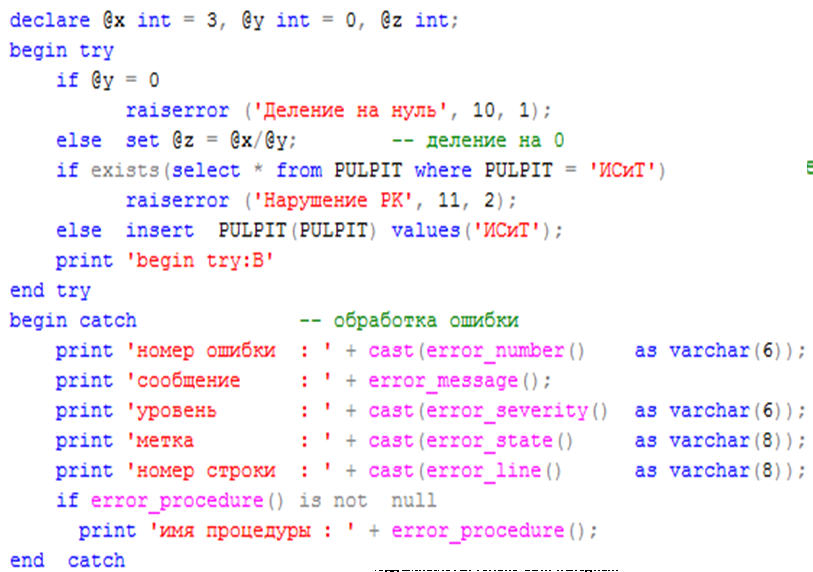
1. Язык T-SQL. Оператор цикла while.



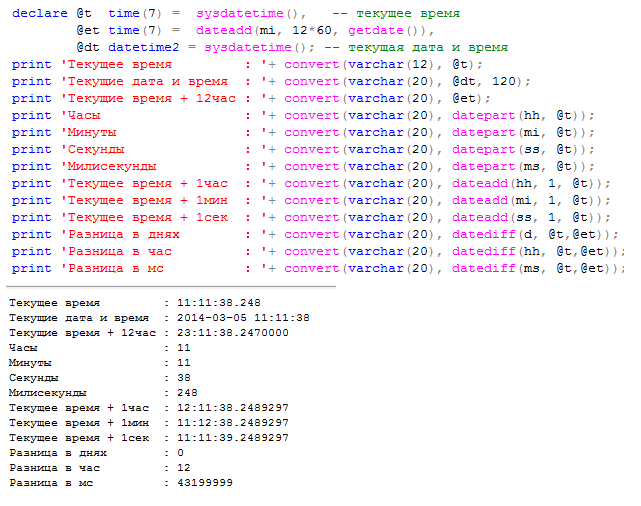
1. Язык T-SQL. Обработка ошибок в конструкциях try-catch. Функция RAISERROR.



**RAISEERROR**:

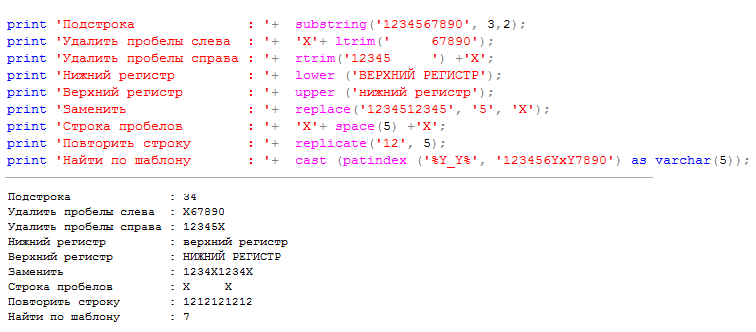


1. Язык T-SQL. Встроенные функции работы с датами.

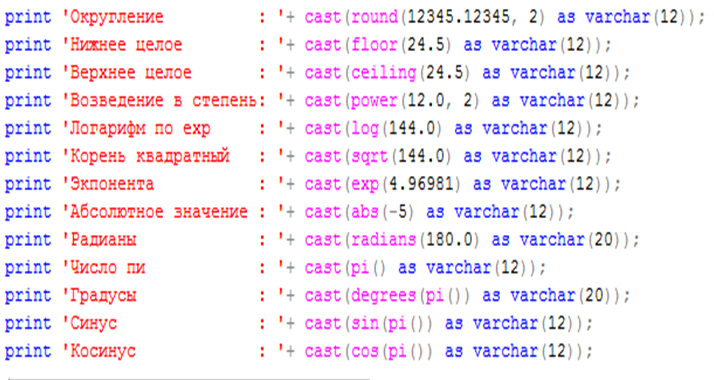


* DATEADD (datepart, number, date)
  + datepart:
  + year - yy, yyyy; quarter - qq, q; month - mm, m;
  + dayofyear - dy, y; day - dd, d; week - wk, ww;
  + weekday - dw, w; hour – hh; minute - mi, n;
  + second - ss, s
* DATEPART (datepart, date)
* DATEDIFF ( datepart , startdate , enddate )

1. Язык T-SQL. Встроенные функции работы со строками.



1. Язык T-SQL. Встроенные функции работы с числовыми данными.

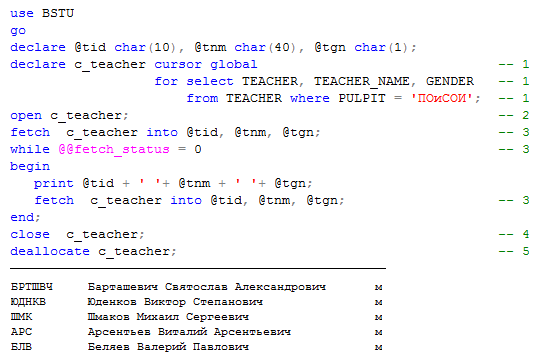


1. Курсоры. Объявление курсора. Общая схема работы с курсором: declare, open, fetch, close, deallocate.

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

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

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.



1. Курсоры. Типы курсоров: global/local, static/dynamic.

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

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

LOCAL

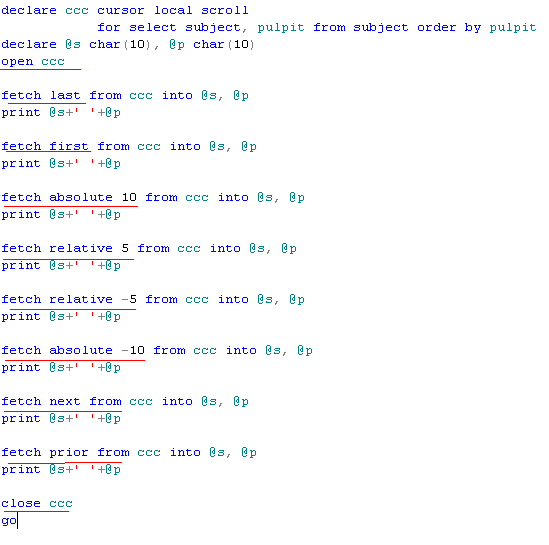
Указывает, что курсор является локальным по отношению к пакету, хранимой процедуре или триггеру, в котором он был создан. Имя курсора допустимо только внутри этой области. На курсор могут ссылаться локальные переменные пакета, хранимые процедуры, триггеры или выходной параметр OUTPUT хранимой процедуры. Параметр OUTPUT используется для передачи локального курсора вызывающему его пакету, хранимой процедуре или триггеру, который затем может присвоить параметр переменной курсора с целью последующего обращения к курсору после завершения хранимой процедуры. Курсор неявно освобождается после завершения выполнения пакета, хранимой процедуры или триггера, за исключением случая, когда курсор был передан параметру OUTPUT. Если курсор был передан параметру OUTPUT, то курсор освобождается при освобождении всех ссылающихся на него переменных или при выходе из области видимости.

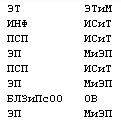
GLOBAL

Указывает, что курсор является глобальным по отношению к соединению. Имя курсора может использоваться любой хранимой процедурой или пакетом, которые выполняются в соединении. Курсор неявно освобождается только в случае разрыва соединения

1. Курсоры. Опция scroll. Способы позиционирования в курсоре: relative/absolute, next/prior, first/last.

Scroll указывает, что доступны все параметры выборки (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE). Если в инструкции DECLARE CURSOR стандарта ISO не указан параметр SCROLL, то поддерживается только параметр выборки NEXT. Если указан аргумент FAST\_FORWARD, задать SCROLL невозможно. Если параметр SCROLL не указан, для выборки доступен только механизм NEXT, и курсор становится FORWARD\_ONLY.



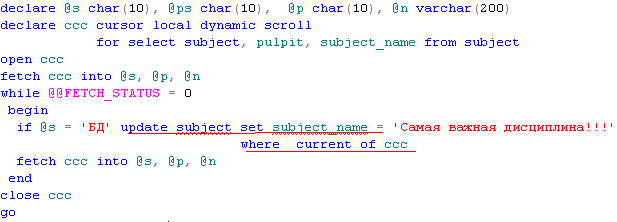


1. Курсоры. Функция fetch\_status.

@@FETCH\_STATUS – возвращает состояние последней инструкции FETCH, вызванной в любом курсоре, открытом в данном соединении

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

1. Курсоры. Применение секции where current of в операторах update, delete.



1. Хранимые процедуры. Создание, изменение и удаление хранимых процедур. Вызов хранимых процедур. Передача параметров.

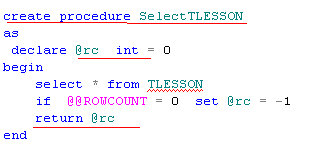
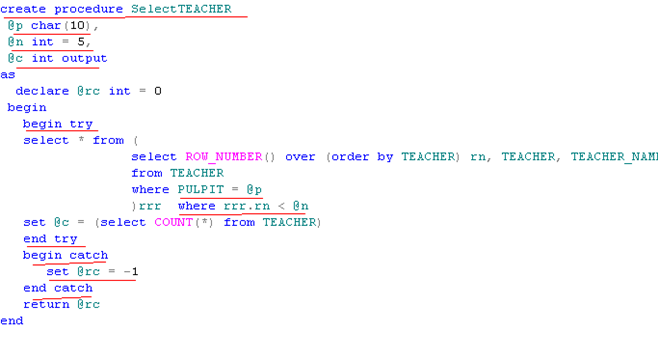
* Хранимая процедура – объект базы данных
* Хранимая процедура – поименованный блок (BEGIN…END) операторов, хранящийся в базе данных в откомпилированном виде
* Принимает входные параметры
* Принимаете и формирует выходные параметры
* Целочисленное значение, возвращаемое к точке вызова с помощью оператора RETURN
* Один или более результирующих наборов, сформированных операторами SELECT
* Содержимое стандартного выходного потока, полученного при выполнении операторов PRINT
* Результирующий набор хранимой процедуры может быть использован в качестве источника строк для INSERT

**Допускается применение**:

* Основных DDL, DML и TCL-операторов
* Конструкций TRY/CATCH
* Курсоров
* Временных таблиц

**Не допускается применение**:

* CREATE or ALTER FUNCTION
* CREATE or ALTER TRIGGER
* CREATE or ALTER PROCEDURE
* CREATE or ALTER VIEW
* USE databasename

1. Хранимые процедуры. DML в процедурах. Входные и выходные параметры.

Входные и выходные параметры были в прошлом билете

1. Функции, определенные пользователем. Виды функций. Обращение к функции. Передача параметров.

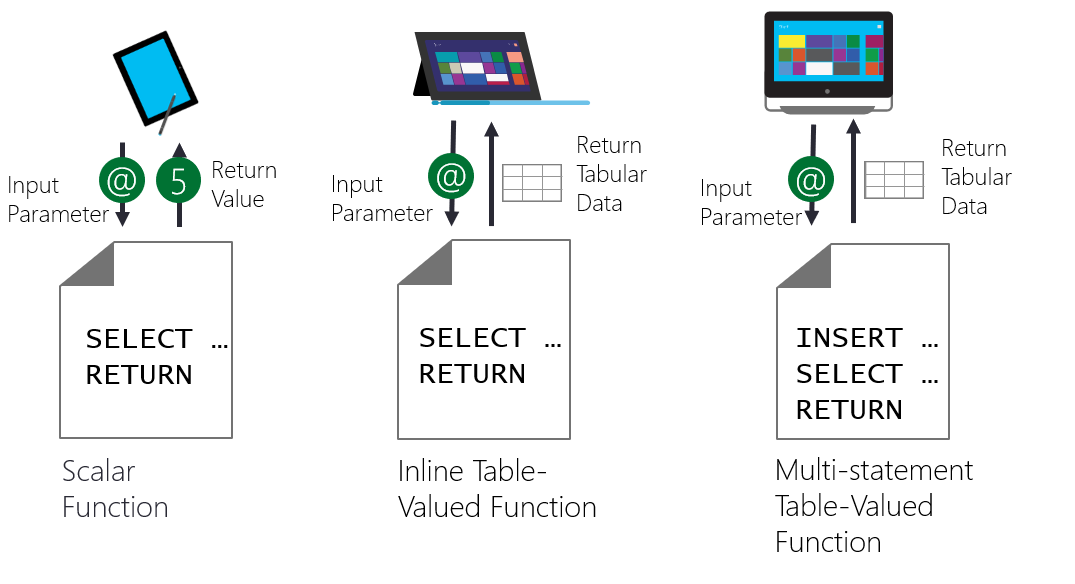
* Встроенные
  + Математические
  + Строковые
  + Работа с датами
* Пользовательские

Не допускается применение:

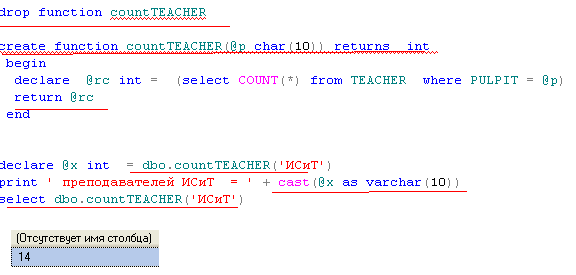
* DDL-операторов
* DML-операторов, изменяющих данные (INSERT, DELETE, UPDATE)
* конструкций TRY/CATCH
* транзакций

Функции:

* Скалярные (возвращает одно значение, нельзя timestamp, cursor, text, ntext, image, table)
* Inline (один SELECT, возвращает таблицу)
* Multi-Statement (несколько команд, возвращает таблицу)



1. Функции, определенные пользователем. Создание, изменение и удаление функций. DML в функциях.



1. План запроса. Этапы обработки SELECT запроса. Понятие стоимости запроса. Понятия селективности и плотности. Оптимизация запросов.

**Этапы обработки**:

1. Разбор
2. Разрешение имен
3. Оптимизация
4. Компиляция
5. Выполнение

* Для каждого шага вычисляется **стоимость** – величина, пропорциональная продолжительности выполнения шага
* Суммарная стоимость шагов плана составляет стоимость всего запроса
* Задача – минимизация общей стоимости запроса
* **Селективность запроса** – соотношение количества строк, удовлетворяющих условию, к общему количеству строк в таблице
  + Индекс успешно работает при <= 5%.
  + Не нужен индекс при 80% или более
* **Плотность запроса** – количество возвращаемых строк запроса

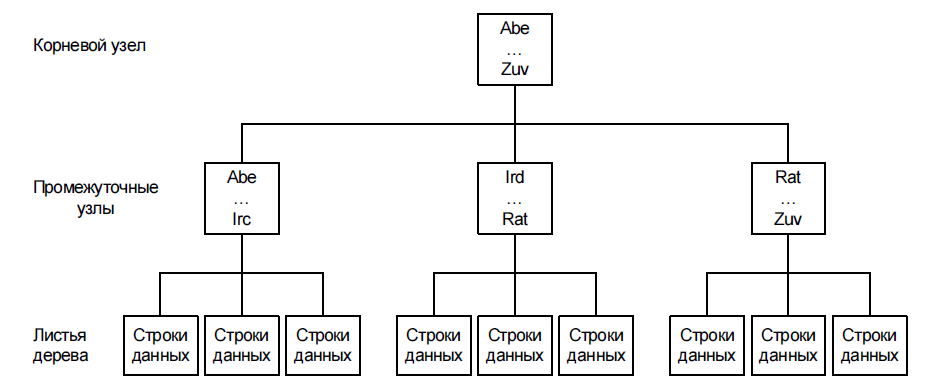
**Оптимизация**:

* анализ запроса
* выбор индекса
* выбор порядка выполнения операций соединения
* выбор метода выполнения операций соединения

1. Индексы. Назначение и применение индексов.

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

* 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 создаются по сбалансированному дереву B+
* Это свойство поддерживается при добавлении или удалении данных в индексированном столбце



1. Индексы. Виды индексов. Применение различных видов индексов.

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 PK или UNIQUE

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

**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**:

* 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

1. Индексы. Реорганизация, перестроение, включение и отключение индексов.

Внутренняя фрагментация - объем данных, хранящихся в каждой странице

Внешняя фрагментация - нарушение логического порядка страниц

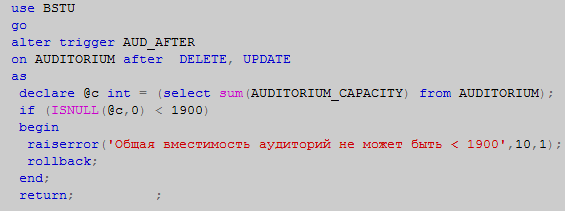
Реорганизация влияет на листья, перестройка ~~рушит ссср~~ влияет на весь индекс

* 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

1. Триггеры. Типы триггеров. Создание и назначение after-триггера.

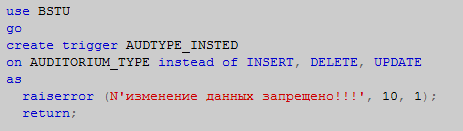
**Триггер** - специальный вид хранимых процедур, выполняющихся при событиях базы данных

* DML-триггеры (Создаются для таблицы или представления. Реагируют на события INSERT, DELETE, UPDATE)
* DDL-триггеры
  + триггеры уровня сервера (ALLSERVER)
  + триггеры уровня базы данных (DATABASE)
* 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
* AFTER-триггеры - триггеры уровня оператора
* Выполняются по одному разу для каждого оператора
* Выполняются после наступления события
* AFTER-триггер вызывается после выполнения активизирующего его оператора
* Если оператор нарушает ограничение целостности, то возникшая ошибка не допускает выполнения этого оператора и соответствующих триггеров



1. Триггеры. Создание и назначение instead of-триггеров.

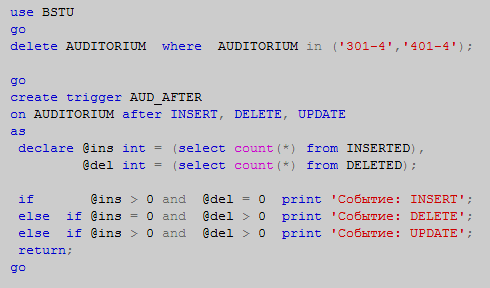
* Триггеры уровня оператора
* Выполняются по одному разу для каждого оператора
* Выполняются вместо операции - сама операция не выполняется
* Всегда использует таблицы inserted и deleted
* Выполняется после создания таблиц inserted и deleted
* Выполняется перед выполнением проверки ограничений целостности или каких-либо других действий
* INSTEAD OF можно создавать для таблиц и для представлений - выполняется вместо выполнения любых действий с любой таблицей



1. Триггеры. Использование таблиц inserted, deleted.

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
* Таблица deleted – в инструкции CREATE TRIGGER указывается DELETE или UPDATE
* Таблица inserted – в инструкции CREATE TRIGGER указывается INSERT или UPDATE



1. Транзакции. Явные и неявные транзакции.

Транзакция – одна или несколько команд SQL, которые либо успешно выполняются как единое целое, либо отменяются как единое целое; Логическая единица работы, обеспечивающая переход базы данных из одного согласованного состояния в другое согласованное состояние

Может быть явной и неявной (insert, update, delete являются единицами транзакции)

* Явная транзакция — группа инструкций, начало и конец которой обозначаются инструкциями:
  + BEGIN TRANSACTION
  + COMMIT
  + ROLLBACK

1. Транзакции. Свойства ACID.

* 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)

1. Транзакции. Уровни изолированности транзакций.

**Уровень изоляции** задает степень защищенности данных в транзакции от возможности изменения другими транзакциями

Проблемы:

* 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)
* Неповторяемое 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)
* READ UNCOMMITTED (ваще пахую)
* READ COMMITTED (выполняет проверку только на наличие монопольной блокировки для данной строки. Проблемы: неповторяемое и фантомное чтения)
* REPEATABLE READ (Устанавливает разделяемые блокировки на все считываемые данные и удерживает эти блокировки до тех пор, пока транзакция не будет подтверждена или отменена. Проблема – фантомное чтение)
* SERIALIZABLE (Устанавливает блокировку на всю область данных, считываемых соответствующей транзакцией. Проблем нет. Уровень изоляции сына маминой подруги)
* SNAPSHOT (Реализуется с использованием метода блокировки диапазона ключа. Блокировка диапазона ключа блокирует элементы индексов)

1. Транзакции. Вложенные транзакции. Функция TRANCOUNT.

TRANCOUNT возвращает число открытых транзакций

Вложенные транзакции используются в хранимых процедурах, которые сами содержат транзакции и вызываются внутри другой транзакции

1. Транзакции. Блокировки. Эскалация блокировок. Взаимные блокировки.

* Блокировки – механизм обеспечения согласованности данных в случае одновременного обращения к данным нескольких пользователей

Свойства:

* Длительность блокировки (период времени, в течение которого ресурс удерживает определенную блокировку)
* Режим блокировки (СУБД автоматически выбирает соответствующий режим блокировки, в зависимости от типа операции (чтение или запись))
  + Разделяемая (shared lock)
  + Монопольная (exclusive lock)
  + Обновления (update lock)
* Гранулярность блокировки определяет, какой объект блокируется:
  + строки
  + страницы
  + индексный ключ или диапазон индексных ключей
  + таблицы
  + экстент
  + база данных

Гранулярность выбирается СУБД автоматически

* Взаимоблокировка (deadlock) — это особая проблема одновременного конкурентного доступа, в которой две транзакции блокируют друг друга

**Эскалация**:

* 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)

1. XML в SQL Server. Секция for XML в SELECT. Директивы PATH, AUTO, RAW. Директивы TYPE, ELEMENTS, ROOT.
2. XML в SQL Server. Применение процедур sp\_xml\_preparedocument, sp\_xml\_removedocument. Применение OPENXML.
3. XML в SQL Server. Схема XML-документа. Коллекция XML SCHEMA.
4. XML в SQL Server. Индексирование XML. Инструкции xPath. Методы типа данных xml.

Ассистент О.А. Нистюк