Введение в SQL

Краткая историческая справка

В начале 70-х годов доктор Э.Ф.Кодд, работавший в компании IBM, разработал теорию реляционных баз данных. Для воплощения идей реляционной модели он разработал язык реляционных баз данных и назвал его Alpha. IBM предпочла передать дальнейшую разработку группе программистов, неподконтрольной доктору Кодду. Нарушив некоторые принципы реляционной модели, они реализовали её в экспериментальной реляционной СУБД IBM System R, для которой затем был создан специальный язык SEQUEL (Structured English QUEry Language — «структурированный английский язык запросов»), позволявший относительно просто управлять данными в этой СУБД. Поскольку SEQUEL было уже зарегистрированной торговой маркой, название сократили до SQL, и таким оно осталось по сей день. Язык ориентирован **главным образом** на удобную и понятную пользователям формулировку запросов к реляционным БД. В действительности SQL обеспечивает и другие возможности.

В настоящее время язык SQL реализован во всех коммерческих реляционных СУБД. Все компании-производители провозглашают соответствие своей реализации стандарту SQL, но на самом деле реализуют диалекты SQL.

Стандартизация SQL

Формальное название стандарта SQL - это ISO/IEC 9075 "Database Language SQL" (Язык баз данных SQL). Время от времени выпускается пересмотренная версия этого стандарта; наиболее свежее обновление было выпущено в 2011 году. Версия 2011 известна как ISO/IEC 9075:2011 или просто как SQL:2011. Версиями до неё были: SQL-86 (SQL1), SQL-92 (SQL2), SQL:1999 (SQL3), SQL:2003, SQL:2006 и SQL:2008. Каждая версия замещает предыдущую, так что требования соответствия стандарту более ранних версий не имеют официальной силы. Стандарт SQL:2011 не является свободно доступным. Полный стандарт можно приобрести у организации ISO. Общая организация стандарта SQL:2011 имеет вид:

- 9075-1, SQL/Framework;
- 9075-2, SQL/Foundation;
- 9075-3, SQL/CLI; (спецификация интерфейса уровня вызова)
- 9075-4, SQL/PSM; (спецификация хранимых процедур)
- 9075-9, SQL/MED; (управление внешними данными)
- 9075-10, SQL/OLB; (связывание с объектно-ориентированными языками программирования)
- 9075-11, SQL/Schemata;
- 9075-13, SQL/JRT; (использование подпрограмм и типов SQL в языке программирования Java)
- 9075-14, SQL/XML. (спецификации языковых средств, позволяющих работать с XML-документами в среде SQL)

Несмотря на наличие международного стандарта, многие производители СУБД вносят изменения в язык SQL, тем самым отступая от стандарта. В результате у разных производителей СУБД в ходу разные диалекты SQL, в общем случае между собой несовместимые. В настоящее время проблема совместимости решается так: описание языка имеет модульную структуру, основная часть стандарта вынесена в раздел «SQL/Foundation», все остальные выведены в отдельные модули, остался только один уровень совместимости – «Core», что означает поддержку этой основной части. Поддержка остальных возможностей оставлена на усмотрение производителей СУБД.

При всех своих изменениях, SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В тоже время, современные СУБД предоставляют пользователю развитые средства визуального построения запросов. Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов, он стал настолько сложным, что превратился в инструмент профессионального программиста.

Transact-SQL (T-SQL)

Transact-SQL (T-SQL) – расширение языка SQL, созданное компанией Microsoft (для Microsoft SQL Server) и Sybase (для Sybase ASE). **Замечание.** Последняя версия Sybase – «SAP ADAPTIVE SERVER ENTERPRISE 16».

Классификация инструкций T-SQL

Инструкции T-SQL принято делить на следующие категории:

- **инструкции языка описания данных** (Data Definition Language, DDL):
    - CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.) [54]
    - ALTER изменяет объект [49]
    - DROP удаляет объект [52]
    - ENABLE TRIGGER включает триггер DML, DDL или logon
    - DISABLE TRIGGER отключает триггер

    - TRUNCATE TABLE удаляет все строки в таблице, не записывая в журнал удаление отдельных строк
    - UPDATE STATISTICS обновляет статистику оптимизации запросов для таблицы или индексированного представления
- **инструкции языка обработки данных** (Data Manipulation Language, DML):
    - SELECT считывает данные, удовлетворяющие заданным условиям
    - INSERT добавляет новые данные
    - UPDATE изменяет существующие данные
    - DELETE удаляет данные
    - MERGE выполняет операции вставки, обновления или удаления для целевой таблицы на основе результатов соединения с исходной таблицей
    - BULK INSERT выполняет импорт файла данных в таблицу или представление базы данных в формате, указанном пользователем
    - READTEXT считывает значения text, ntext или image из столбцов типа text, ntext или image начиная с указанной позиции
    - WRITETEXT обновляет и заменяет все поле text, ntext или image
    - UPDATETEXT обновляет часть поля text, ntext или image
- **инструкции безопасности** (ранее инструкции языка доступа к данным - Data Control Language, DCL):
    - GRANT предоставляет пользователю разрешения на определенные операции с объектом
    - REVOKE отзывает ранее выданные разрешения
    - DENY задает запрет, имеющий приоритет над разрешением
    - ADD SIGNATURE добавляет цифровую подпись для хранимой процедуры, функции, сборки или триггера
    - OPEN MASTER KEY открывает главный ключ в текущей базе данных
    - CLOSE MASTER KEY закрывает главный ключ в текущей базе данных
    - OPEN SYMMETRIC KEY расшифровывает симметричный ключ и делает его доступным для использования
    - CLOSE SYMMETRIC KEY закрывает симметричный ключ или все симметричные ключи, открытые в текущем сеансе
    - EXECUTE AS контекст выполнения сеанса переключается на заданное имя входа и имя пользователя
    - REVERT переключает контекст выполнения в контекст участника, вызывавшего последнюю инструкцию EXECUTE AS
    - SETUSER позволяет члену предопределенной роли сервера sysadmin или члену предопределенной роли базы данных db_owner олицетворять другого пользователя
- **инструкции управления транзакциями** (Transaction Control Language, TCL):
    - BEGIN DISTRIBUTED TRANSACTION запускает распределенную транзакцию, управляемую координатором распределенных транзакций
    - BEGIN TRANSACTION отмечает начальную точку явной локальной транзакции
    - COMMIT TRANSACTION отмечает успешное завершение явной или неявной транзакции
    - COMMIT WORK действует так же, как и инструкция COMMIT TRANSACTION
    - ROLLBACK TRANSACTION откатывает явные или неявные транзакции до начала или до точки сохранения транзакции
    - ROLLBACK WORK действует так же, как и инструкция ROLLBACK TRANSACTION
    - SAVE TRANSACTION устанавливает точку сохранения внутри транзакции
- **инструкции управления потоком** (Control-of-Flow Language, CFL):
    - BEGIN...END
    - BREAK
    - CONTINUE
    - GOTO
    - IF...ELSE
    - RETURN
    - THROW
    - TRY...CATCH
    - WAITFOR
    - WHILE
    - PRINT
    - EXECUTE
    - RAISERROR
- **инструкции курсоров**:
    - DECLARE CURSOR определяет атрибуты серверного курсора
    - OPEN открывает серверный курсор и заполняет его
    - FETCH получает определенную строку из серверного курсора
    - CLOSE закрывает открытый курсор, высвобождая текущий результирующий набор и снимая блокировки курсоров для строк, на которых установлен курсор
    - DEALLOCATE удаляет ссылку курсора и освобождает структуры данных, составляющие курсор
- **инструкции BACKUP и RESTORE** - позволяют создавать резервные копии баз данных и восстанавливать базы данных

- **команды управления**:
  - CHECKPOINT создает ручную контрольную точку в базе данных SQL Server, с которой в данный момент установлено соединение
  - DBCC выступают в качестве консольных команд базы данных для SQL Server
  - KILL прерывает пользовательский процесс, определяемый идентификатором сеанса или единицей работы
  - KILL QUERY NOTIFICATION SUBSCRIPTION удаляет подписки на уведомления о запросах из экземпляра SQL Server
  - KILL STATS JOB останавливает асинхронное задание обновление статистики
  - RECONFIGURE изменяет значение параметра конфигурации с помощью хранимой системной процедуры sp_configure
  - SHUTDOWN немедленно останавливает SQL Server
- **инструкции SET** - изменяют текущий сеанс, управляя специфическими данными. Инструкции SET группируются в следующие категории:
  - **Инструкции даты и времени**
    - SET DATEFIRST
    - SET DATEFORMAT
  - **Инструкции блокировки**
    - SET DEADLOCK_PRIORITY
    - SET LOCK_TIMEOUT
  - **Прочие инструкции**
    - SET CONCAT_NULL_YIELDS_NULL
    - SET CURSOR_CLOSE_ON_COMMIT
    - SET FIPS_FLAGGER
    - SET IDENTITY_INSERT
    - SET LANGUAGE
    - SET OFFSETS
    - SET QUOTED_IDENTIFIER
  - **Инструкции выполнения запросов**
    - SET ARITHABORT
    - SET ARITHIGNORE
    - SET FMTONLY
    - SET NOCOUNT
    - SET NOEXEC
    - SET NUMERIC_ROUNDABORT
    - SET PARSEONLY
    - SET QUERY_GOVERNOR_COST_LIMIT
    - SET ROWCOUNT
    - SET TEXTSIZE
  - **Инструкции настроек ISO**
    - SET ANSI_DEFAULTS
    - SET ANSI_NULL_DFLT_OFF
    - SET ANSI_NULL_DFLT_ON
    - SET ANSI_NULLS
    - SET ANSI_PADDING
    - SET ANSI_WARNINGS
  - **Статистические инструкции**
    - SET FORCEPLAN
    - SET SHOWPLAN_ALL
    - SET SHOWPLAN_TEXT
    - SET SHOWPLAN_XML
    - SET STATISTICS IO
    - SET STATISTICS XML
    - SET STATISTICS PROFILE
    - SET STATISTICS TIME
  - **Инструкции управления транзакциями**
    - SET IMPLICIT_TRANSACTIONS
    - SET REMOTE_PROC_TRANSACTIONS
    - SET TRANSACTION ISOLATION LEVEL
    - SET XACT_ABORT

Полный справочник по инструкциям языка T-SQL находится по адресу
http://msdn.microsoft.com/ru-ru/library/bb510741.aspx

Системы типов данных языка SQL

A. Типы данных SQL:2011

Все допустимые в SQL типы данных, которые можно использовать при определении столбцов, разбиваются на следующие категории:

- *точные числовые типы* (exact numerics);
- *приближенные числовые типы* (approximate numerics);
- *типы символьных строк* (character strings);
- *типы битовых строк* (bit strings);
- *типы даты и времени* (datetimes);
- *типы временных интервалов* (intervals);
- *булевский тип* (Booleans);
- *типы коллекций* (collection types);
- *анонимные строчные типы* (anonymous row types);
- *типы, определяемые пользователем* (user-defined types);
- *ссылочные типы* (reference types).

Комментарии к некоторым типам данных (при первом знакомстве пропустить)

Булевский тип

При определении столбца булевского типа указывается просто спецификация BOOLEAN. Булевский тип состоит из трех значений: true, false и unknown (соответствующие литералы обозначаются TRUE, FALSE и UNKNOWN). Поддерживается возможность построения булевских выражений, которые вычисляются в трехзначной логике.

Типы коллекций

Под термином коллекция обычно понимается одно из следующих образований: *массив*, *список*, *множество* и *мультимножество*. В варианте SQL:2011 специфицированы только *массивы* и *мультимножества*.

Анонимные строчные типы

Анонимный строчный тип – это конструктор типов ROW, позволяющий производить безымянные типы строк (кортежей). При определении столбца, значения которого должны принадлежать некоторому строчному типу, используется конструкция ROW (fld_1, fld_2, …, fld_n), где каждый элемент fld_i, определяющий поле строчного типа, задается в виде тройки *fldname*, *fldtype*, *fldoptions*. В качестве типа данных поля строчного типа можно использовать любой допустимый в SQL тип данных, включая типы коллекций, определяемые пользователями типы и другие строчные типы. Необязательный элемент fldoptions может задаваться для указания применяемого по умолчанию порядка сортировки, если соответствующий подэлемент fldtype указывает на тип символьных строк, а также должен задаваться, если fldtype указывает на ссылочный тип. Степенью строчного типа называется число его полей.

Типы, определяемые пользователем

Эта категория типов данных связана с объектными расширениями языка SQL. Различают:

- Структурные типы (Structured Types). Можно определить долговременный хранимый, именованный тип данных, включающий один или более атрибутов любого из допустимых в SQL типа данных, в том числе другие структурные типы, типы коллекций, строчные типы и т. д. Стандарт SQL не накладывает ограничений на сложность получаемой в результате структуры данных, однако не запрещает устанавливать такие ограничения в реализации. Дополнительные механизмы определяемых пользователями методов, функций и процедур позволяют определить поведенческие аспекты структурного типа.
- Индивидуальные типы (Distinct Types). Можно определить долговременно хранимый, именованный тип данных, опираясь на единственный предопределенный тип. Например, можно определить индивидуальный тип данных PRICE, опираясь на тип DECIMAL (5, 2). Тогда значения типа PRICE представляются точно так же, как значения типа DECIMAL (5, 2). В существующем стандарте индивидуальный тип не наследует от своего опорного типа набор операций над значениями. Например, чтобы сложить два значения типа PRICE требуется явно сообщить системе, что с этими значениями нужно обращаться как со значениями типа DECIMAL (5, 2). Другая возможность состоит в явном определении методов, функций и процедур, связанных с данным индивидуальным типом. Похоже, что в будущих версиях стандарта появятся и другие, более удобные возможности.

Ссылочные типы

Эта категория типов данных связана с объектными расширениями языка SQL. Обеспечивается механизм конструирования типов (ссылочных типов), которые могут использоваться в качестве типов столбцов некоторого вида таблиц (типизированных таблиц). Фактически значениями ссылочного типа являются строки соответствующей типизированной таблицы. Более точно, каждой строке типизированной таблицы приписывается уникальное значение (нечто вроде первичного ключа, назначаемого системой или приложением), которое может использоваться в методах, определенных для табличного типа, для уникальной идентификации строк соответствующей таблицы. Эти уникальные значения называются ссылочными значениями, а их тип – ссылочным типом. Ссылочный тип может содержать только те значения, которые действительно ссылаются на экземпляры указанного типа (т. е. на строки соответствующей типизированной таблицы).

B. Типы данных SQL Server

- Точные числа: bigint int smallint tinyint bit decimal numeric money smallmoney
- Приблизительные числа: float real
- Дата и время: date, datetime2, datetime, datetimeoffset, smalldatetime, time
- Символьные строки: char varchar text
- Символьные строки в Юникоде: nchar nvarchar ntext
- Двоичные данные: binary varbinary image
- Прочие типы данных: cursor, hierarchyid, sql_variant, table, timestamp, uniqueidentifier, xml, пространственные типы (geography и geometry)

В зависимости от параметров хранения, некоторые типы данных в SQL Server относятся к следующим группам:

- типы данных больших значений: varchar(max), nvarchar(max) и varbinary(max);
- типы данных больших объектов: text, ntext, image, varchar(max), nvarchar(max), varbinary(max) и xml.

Идентификаторы T-SQL

Идентификаторы в SQL Server могут присваиваться любым сущностям. Для большинства объектов идентификаторы необходимы, а для некоторых, например ограничений, необязательны. Существует два класса идентификаторов:

a) обычные и
b) с разделителями.

Правила для обычных идентификаторов

1. Первым символом должен быть один из следующих:
    a) латинская буква,
    b) подчеркивание '_',
    c) коммерческое at '@',
    d) решетка '#'.
2. Определенные символы в начале идентификатора имеют особое значение:
    a) идентификатор, начинающийся символом @, означает локальную переменную или параметр,
    b) идентификатор, начинающийся символом #, означает локальный временный объект,
    c) идентификатор, начинающийся двойным символом ##, означает глобальный временный объект,
    d) некоторые функции языка T-SQL имеют имена, начинающиеся двойным символом @@, и во избежание путаницы с этими функциями не следует использовать имена, начинающиеся символами @@.
3. Последующие символы могут включать:
    a) латинские буквы,
    b) десятичные цифры,
    c) символы @, $, # и _.

Замечание. Правила записи обычных идентификаторов зависят от уровня совместимости базы данных, который можно установить с помощью инструкции ALTER DATABASE.

Идентификаторы с разделителями могут содержать то же количество символов, что и обычные идентификаторы. Это может быть от 1 до 128 символов, не включая символы-разделители. Идентификаторы локальных временных таблиц могут быть максимум 116 символов.

Правила для идентификаторов с разделителями

Идентификаторы, не соответствующие правилам, могут содержать любую комбинацию символов в текущей кодовой странице и должны заключаться в разделители:

a) двойные кавычки и
b) квадратные скобки.

В этом случае SET QUOTED_IDENTIFIER должен быть установлен в состояние ON.

Примеры: "Blanks in Table Name" или [Blanks in Table Name].

Использование идентификаторов в качестве имен объектов

Полное имя объекта состоит из четырех идентификаторов: имени сервера, имени базы данных, имени схемы и имени объекта, которые отображаются в следующем формате:

имя_сервера.[*имя_базы_данных*].[*имя_схемы*].*имя_объекта*

Замечания.

1) Большинство ссылок на объекты используют трехкомпонентные имена.
2) По умолчанию в качестве *имени_сервера* используется локальный сервер.
3) По умолчанию в качестве *имени_базы_данных* используется текущая база данных соединения.
4) По умолчанию в качестве *имени_схемы* обычно используется **dbo**, если в явном виде не были заданы иные настройки.
5) Четырехсоставные имена обычно используются в распределенных запросах и удаленных вызовах хранимых процедур.
6) Для ссылки на столбцы используется дополнительная точечная нотация.
7) Для обращения к свойствам столбцов UDT используется дополнительная точечная нотация.

Зарезервированные ключевые слова T-SQL

- список зарезервированных слов SQL Server (185 штук)
- список зарезервированных ключевых слов ODBC (235 штук)
- список зарезервированных ключевых слов на будущее (273 штуки)

Константы T-SQL

Константа, также называемая литералом или скалярным значением, зависит от типа данных.

Символьные строки

Заключаются в одинарные кавычки, например, 'O''Brien'. Если для соединения значение параметра QUOTED_IDENTIFIER было задано как OFF, то символьные константы также могут заключаться в двойные кавычки. Если символьная строка, заключенная в одинарные кавычки, содержит также внедренную одинарную кавычку, то эту кавычку необходимо заключить в дополнительные одинарные кавычки. Данное требование не распространяется на строки, заключенные в двойные кавычки.

Символьные строки в Юникоде

Начинается с идентификатора N, например, N'Русская Редакция'.

Двоичные строки

Начинаются с префикса 0x, за которым следует строка шестнадцатеричных чисел, которая не заключается в кавычки, например, 0x12Ef.

Константы типа **bit**

Содержат последовательности нулей и единиц и в кавычки не заключаются. Все числа, больше единицы, преобразуются в единицу. Примеры: 0101010101, 000000000.

Константы **datetime**

Задаются с помощью символьных значений даты специального формата, заключенных в одинарные кавычки.

Константы **integer**

Состоят из числовых строк, которые не заключаются в кавычки и не содержат десятичного разделителя.

Константы **decimal**

Состоят из числовых строк, которые не заключаются в кавычки и содержат десятичный разделитель.

Константы типа **float** и **real**

Представляются в экспоненциальной форме.

Константы **money**

Состоят из числовых строк, которые не заключаются в кавычки, могут содержать десятичный разделитель и префикс в виде знака валюты.

Константы **uniqueidentifier**

Состоят из строки, представляющей идентификатор GUID. Могут указываться с помощью символьного или двоичного символьного формата.

Константы **xml**

Скалярные выражения T-SQL

Сочетание операндов и операторов, используемое компонентом SQL Server для получения одиночного значения данных. В простейшем случае выражение может быть:

- литералом (константой);
- функцией;
- именем столбца;
- переменной;
- вложенным запросом;
- функцией CASE.

В выражениях символы и значения типа datetime необходимо заключать в одинарные кавычки. Символьные константы, используемые в качестве шаблона для предложения LIKE, должны быть заключены в одинарные кавычки.

expression ::=
{ *constant* |
scalar_function |
[*table_name*.] *column* |
variable |
(*expression*) |
(*scalar_subquery*) |
{ *unary_operator* } expression |
expression { *binary_operator* } *expression* |
ranking_windowed_function |
aggregate_windowed_function
}

Категории операторов T-SQL

Категория	Операторы
Арифметические операторы	+, -, *, /, %
Логические операторы	ALL, AND, ANY, BETWEEN, EXISTS, IN, LIKE, NOT, OR, SOME
Оператор присваивания	=
Оператор разрешения области	:: (обеспечивает доступ к статическим элементам составного типа данных)
Битовые операторы	&, |, ^
Операторы наборов	EXCEPT, INTERSECT, UNION
Операторы сравнения	=, >, <, >=, <=, <>, !=, !<, !>
Оператор объединения строк	+ (сцепление строк)
Составные операторы	+=, -=, *=, /=, %=, &=, ^=, |=
Унарные операторы	+, -, ~

Приоритет операторов T-SQL

Уровень	Операторы
1	~
2	*, /, %
3	+ (положительное), - (отрицательное), + (сложение), (+ сцепление), - (вычитание), & , ^, |
4	=, >, <, >=, <=, <>, !=, !>, !<
5	NOT
6	AND
7	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
8	= (присваивание)

Замечания.
1. Если два оператора в выражении имеют один и тот же уровень старшинства, они выполняются в порядке слева направо по мере их появления в выражении.
2. Чтобы изменить приоритет операторов в выражении, следует использовать скобки.

Приоритет типов данных T-SQL

Если оператор связывает два выражения различных типов данных, то по правилам приоритета типов данных определяется, какой тип данных имеет меньший приоритет и будет преобразован в тип данных с большим приоритетом. Если неявное преобразование не поддерживается, возвращается ошибка. Если оба операнда выражения имеют одинаковый тип данных, результат операции будет иметь тот же тип данных.

В SQL Server используется следующий приоритет типов данных:

1. UDT (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).

Приведение и преобразование типов данных

Следующие функции поддерживают приведение и преобразование типов данных:
- CAST и CONVERT
- PARSE
- TRY_CAST
- TRY_CONVERT
- TRY_PARSE

Использование функций CAST и CONVERT для преобразования выражений одного типа в другой

CAST (*выражение* AS *целевой_тип_данных* [(*длина*)])
CONVERT (*целевой_тип_данных* [(*длина*)] , *выражение* [, *стиль*])

Пример 1. Получить названия деталей и их цены для тех деталей, у которых первая цифра цены по прейскуранту – 2.

```
SELECT PName, Price
```

```
FROM P
WHERE CAST(Price AS int) LIKE '2%';
```

Можно и так

```
SELECT PName, Price
FROM P
WHERE CAST(CAST(Price AS int) AS varchar(10)) LIKE '2%';
```

Пример 2. Объединение несимвольных недвоичных выражений с помощью функции CAST.

```
SELECT ' Цена детали '+Pname+' по прейскуранту составляет '+CAST(Price AS varchar(12)) AS
[Цена]
FROM P
WHERE Price BETWEEN 10.00 AND 20.00;
```

Пример 3. Показать текущую дату и время, используя функцию `CAST` для изменения текущей даты и времени в символьный тип данных и затем использовать `CONVERT` для отображения даты и времени в формате `ISO 8901`.

```
SELECT
   GETDATE() AS [До преобразования],
   CAST(GETDATE() AS nvarchar(30)) AS [Используя Cast],
   CONVERT(nvarchar(30), GETDATE(), 126) AS [Используя Convert в ISO8601]  ;

2010-11-24 00:33:45.403 Nov 24 2010 12:33AM           2010-11-24T00:33:45.403
```

Пример 4. Частичная противоположность предыдущему примеру. Отобразить дату и время в виде символьных данных, использует функцию CAST для изменения символьных данных в тип данных datetime, а затем использует CONVERT для изменения символьных данных в тип данных datetime.

```
SELECT
   '2006-04-25T15:50:59.997' AS UnconvertedText,
   CAST('2006-04-25T15:50:59.997' AS datetime) AS UsingCast,
   CONVERT(datetime, '2006-04-25T15:50:59.997', 126) AS UsingConvertFrom_ISO8601 ;

2006-04-25T15:50:59.997 2006-04-25 15:50:59.997 2006-04-25 15:50:59.9970
```

Условие поиска T-SQL

Сочетание одного или нескольких предикатов, в котором используются логические операторы AND, OR и NOT. Параметр *search_condition* в инструкции DELETE, MERGE, SELECT или UPDATE указывает, сколько строк возвращается или обрабатывается инструкцией.

<*search_condition*> ::=
  { [NOT] <*predicate*> | (<*search_condition*>) } [{ AND | OR } [NOT] { <*predicate*> | (<*search_condition*>) }]
[,...n]

<*predicate*> ::=
{ *expression* { = | < > | ! = | > | > = | ! > | < | < = | ! < } *expression*
  | *match_expression* [NOT] LIKE *pattern* [ESCAPE '*escape_character*']
  | *expression* [NOT] BETWEEN *expression* AND *expression*
  | *expression* IS [NOT] NULL
  | CONTAINS ({ *column* | * } , '<*contains_search_condition*>')
  | FREETEXT ({ *column* | * } , '*freetext_string*')
  | *test_expression* [NOT] IN (*subquery* | *expression* [,...n])
  | *expression* { = | < > | ! = | > | > = | ! > | < | < = | ! < } { ALL | SOME | ANY} (*subquery*)
  | EXISTS (*subquery*)
}

Комментарий.

1) *expression* – скалярное выражение.
2) *match_expression* – любое допустимое выражение символьного типа данных.

3) *pattern* – конкретная строка символов для поиска в *match_expression*, которая может содержать следующие допустимые символы-шаблоны: %, _, [], [^]. Длина значения *pattern* не может превышать 8000 байт.
4) *escape_character* – символ, помещаемый перед символом-шаблоном, чтобы символ-шаблон рассматривался как обычный символ, а не как шаблон.
5) CONTAINS – осуществляет поиск столбцов, содержащих символьные данные с заданной точностью (*fuzzy*), соответствующие заданным отдельным словам и фразам на основе похожести словам и точному расстоянию между словами, взвешенному совпадению. Этот параметр может быть использован только в инструкции SELECT.
6) FREETEXT – предоставляет простую форму естественного языка ввода запросов на осуществление поиска столбцов, содержащих символьные данные, совпадающие с содержанием предиката не точно, а по смыслу. Этот параметр может быть использован только в инструкции SELECT Список значений необходимо заключать в скобки.
7) *subquery* – может рассматриваться как ограниченная инструкция SELECT и являющаяся подобной на *query_expresssion* в инструкции SELECT. Использование предложений ORDER BY, COMPUTE и ключевого слова INTO не допускается.

Архитектура SQL Server

SQL Server 2014 содержит следующие технологии управления и анализа данных:

- Database Engine - основная служба (реляционное ядро) для хранения, обработки и обеспечения безопасности данных.
- Data Quality Services - построение базы знаний и ее использование для выполнения разнообразных задач по обеспечению качества данных, включая исправление, дополнение, стандартизацию и устранение дубликатов данных.
- Analysis Services - многомерные данные и интеллектуальный анализ данных и совместная работа с PowerPivot, Excel и SharePoint. Для обнаружения в данных закономерностей и тенденций можно применять сочетание этих функций и средств, а затем использовать найденные закономерности и тенденции для принятия обоснованных решений в отношении сложных бизнес-задач.
- Integration Services - интеграцию и преобразование данных, например, экспорт-импорт данных.
- Master Data Services - управления основными данными. Решение, построенное на основе Master Data Services, позволяет обеспечить правильность информации, используемой для построения отчетов и выполнения анализа. С помощью Master Data Services можно создать центральный репозиторий основных данных и поддерживать запись этих данных по мере их изменения, защищенную и доступную для аудита.
- Replication - копирование и распространение данных и объектов баз данных между базами данных, а также синхронизации баз данных для поддержания согласованности.
- Reporting Services - создания отчетов с поддержкой веб-интерфейса

Ключевые компоненты Database Engine:

Query Optimizer (см. Query Optimizer Deep Dive - Part 1..Part 4)
http://sqlblog.com/blogs/paul_white/archive/2012/04/28/query-optimizer-deep-dive-part-1.aspx

Performance|Improving .NET Application Performance and Scalability (Chapter 1.. Chapter 17)
http://msdn.microsoft.com/en-us/library/ff647813.aspx

Логические операторы описывают операции реляционной алгебры, используемые для обработки инструкции.
Физические операторы реализуют действия, описанные логическими операторами. Каждый физический оператор является объектом или процедурой, выполняющей операцию.

Оптимизатор запросов использует операторы для построения плана запроса. План запроса — это дерево физических операторов. Можно просмотреть план запроса с помощью инструкций SET SHOWPLAN.

Логические операторы описывают операции реляционной алгебры, используемые для обработки инструкции.
Физические операторы реализуют действия, описанные логическими операторами. Каждый физический оператор является объектом или процедурой, выполняющей операцию.

Справочник по логическим и физическим операторам Showplan (104 штуки)
http://msdn.microsoft.com/ru-ru/library/ms191158.aspx