ЯЗЫК SQL.
Создание запросов в СУБД Microsoft Access средствами SQL

Оглавление

1. История развития языка SQL

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД IBM System R, для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language — «структурированный английский язык запросов». Позже по юридическим соображениям язык SEQUEL был переименован в SQL.

Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования. Собственно разработкой языка запросов занимались Дональд Чэмбэрлин (Donald D. Chamberlin) и Рэй Бойс (Ray Boyce).

SQL (англ. Structured Query Language — «язык структурированных запросов») — универсальный компьютерный информационно-логический язык, появившийся в результате разработки реляционной модели данных, применяемый для создания, модификации и управления данными в реляционных базах данных.

Изначально, SQL был основным способом работы пользователя с базой данных и представлял собой небольшую совокупность команд (операторов) допускающих создание таблиц, добавление в таблицы новых записей, извлечение записей из таблиц (в соответствии с заданным условием), удаление записей и изменение структур таблиц. В связи с усложнением язык SQL стал более прикладным языком программирования, а пользователи получили возможность использовать визуальные построители запросов.

SQL принципиально отличается от традиционных алгоритмических языков программирования прежде всего тем, что он относится к непроцедурным языкам. На языке типа Кобол или Си можно записать шаг за шагом все инструкции, необходимые для исполнения программы.

Язык SQL позволяет задать только то, “что нужно делать”, а само исполнение отдельных операций (“как делать”) возлагается непосредственно на СУБД. Такой подход в значительной мере определяется самой философией реляционных баз данных. СУБД в данном случае рассматривается как “черный ящик”, и что происходит внутри него, пользователя не должно касаться. Его должно интересовать только внесение в базу данных необходимых изменений и получение правильного ответа на запрос.

Все языки манипулирования данными, созданные для многих СУБД до появления реляционных баз данных, были ориентированы на операции с данными, представленными в виде логических записей файлов. Разумеется, это требовало от пользователя детального знания организации хранения данных и серьезных усилий для указания того, какие данные необходимы, где они размещаются и как их получить.

Благодаря работающим с файловыми серверами СУБД, множество пользователей получают доступ к одним и тем же базам данных. Упрощается разработка различных автоматизированных систем управления организациями. Однако при таком подходе вся обработка запросов из программ или с терминалов пользовательских ЭВМ на них и выполняется, поэтому для реализации даже простого запроса необходимо считывать с файлового сервера или записывать на него целые файлы, а это ведет к конфликтным ситуациям и перегрузке сети. Для исключения указанных недостатков была предложена технология клиент-сервер, но при этом понадобился единый язык общения с сервером – выбор пал на SQL.

Рассматриваемый язык SQL ориентирован на операции с данными, ленными в виде логически взаимосвязанных совокупностей таблиц-ий. Важнейшая особенность его структур – ориентация на конечный г обработки данных, а не на процедуру этой обработки. Язык SQL сам определяет, где находятся данные, индексы и даже какие наиболее эффективные последовательности операций следует использовать для получения результата, а потому указывать эти детали в запросе к базе данных не требуется.

SQL в настоящее время получил очень широкое распространение и фактически превратился в стандартный язык реляционных баз данных. Стандарт на язык SQL был выпущен Американским национальным институтом стандартов (ANSI) в 1986 г., а в 1987 г. Международная организация стандартов (ISO) приняла его в качестве международного. Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992 году нового расширенного стандарта (ANSI SQL-92 или просто SQL2). Следующим стандартом стал SQL:1999 (SQL3). В настоящее время действует стандарт, принятый в 2003 году (SQL:2003) с небольшими модификациями, внесёнными позже.

Разработка, в основном, шла в отделениях фирмы IBM (языки ISBL, SQL, QBE) и университетах США (PIQUE, QUEL). Последний создавался для СУБД INGRES (Interactive Graphics and Retrieval System), которая была разработана в начале 70-х годов в Университете шт. Калифорния и сегодня входит в пятерку лучших профессиональных СУБД. Сегодня из всех этих языков полностью сохранились и развиваются QBE (Query-By-Example - запрос по образцу) и SQL, а из остальных взяты в расширение внутренних языков СУБД только наиболее интересные конструкции.

В начале 70-х годов плодотворный труд исследователя из IBM доктора Кодда (E. F. Codd) привел к созданию продукта, связанного с реляционной моделью данных под названием SEQUEL (Structured English Query Language, структурированный английский язык для запросов), который в 1980 г. был переименован в SQL (Structured Query Language, струк турированный язык запросов).

Некоторые популярные диалекты SQL:

- PL/SQL. Используется в Oracle. PL/SQL – это сокращение от Procedural Language/SQL. Он во многом похож на язык Ada.
- Transact-SQL. Используется в Microsoft SQL Server и Sybase Adaptive Server. По мере того как Microsoft и Sybase все больше отходят от общей платформы, которую они использовали в начале 90-х годов, их реализации Transact-SQL также подвергаются дивергенции.

Некоторые популярные диалекты SQL:

- **PL/pgSQL.** Название диалекта и расширений SQL, реализованных в PostgreSQL. Является сокращением от Procedural Language/postgreSQL.
- **SQLPL.** Самый новый диалект от DB2 (SQLProcedural Language). Основан на стандартных операторах управления SQL. Большинство других диалектов предшествовало стандарту, и это означает, что вы найдете в них массу отличий от стандарта SQL.

2. Достоинства языка SQL:

1. Независимость от конкретных СУБД. Если при создании БД не использовались нестандартные возможности языка SQL предоставляемые некоторой СУБД, то такую БД можно без изменений перенести на СУБД другого производителя. К сожалению большинство БД используют особенности СУБД, на которой работают, что затрудняет их перенос на другую СУБД без изменений;
2. Реляционная основа. Реляционная модель имеет солидный теоретический фундамент. Язык SQL основан на реляционной модели и является единственным языком для реляционных БД;

Достоинства языка SQL:

3. SQL обладает высокоуровневой структурой, напоминающей английский язык.
4. SQL позволяет создавать различные представления данных для различных пользователей;
5. SQL является полноценным языком для работы с БД;
6. Стандарты языка SQL. Официальный стандарт языка SQL опубликован ANSI и ISO в 1989 году и значительно расширен в 1992 году.

Синтаксис языка SQL

Синтаксические конструкции SQL делятся на 4 основные категории

1. **Идентификаторы.** Представляют собой пользовательские или системные имена объектов баз данных, таких, как база данных, таблица, ограничение в таблице, столбцы таблицы, представления и т. п.
2. **Константы.** Представляют собой созданные пользователем или системой строки или значения, не являющиеся идентификаторами или ключевыми словами. Константы могут представлять собой строки, например «hello», числа, например «1234», даты, например «1 января 2002», или булевы значения, например TRUE.

Синтаксические конструкции SQL делятся на 4 основные категории

3. **Операторы.** Символы, показывающие, какое действие выполняется над одним или несколькими выражениями, чаще всего в инструкциях DELETE, INSERT, SELECT или UPDATE. Операторы также часто применяются для создания объектов базы данных.
4. **Зарезервированные и ключевые слова.** Имеют специальный смысл для обработчика кода SQL. Например, SELECT, GRANT, DELETE или CREATE. Зарезервированные слова (Reserved words), обычно команды и инструкции SQL, нельзя использовать в качестве идентификаторов на данной платформе. Ключевые слова (keywords) - это слова, которые могут стать зарезервированными в будущем.

Соглашения об именах

- *Выбирайте имя так, чтобы оно было осмысленным, наглядным и соответствовало назначению объекта.*
- *Используйте в именах один и тот же регистр по всей базе.*
- *Будьте последовательны в использовании сокращений.*
- *Для удобства восприятия используйте полные, наглядные и осмысленные имена с символами подчеркивания.*
- *Не помещайте название компании и продуктов в имена объектов баз данных.*
- *Не используйте слишком очевидные префиксы и суффикс.*
- *Не заполняйте все пространство, отведенное для имени объекта.*
- *Не используйте идентификаторы с разделителями*

Правила создания идентификаторов

Идентификаторы должны быть уникальны в пределах своей области действия. Таким образом, в иерархии объектов имена баз данных не должны повторяться в пределах данного экземпляра сервера базы, а имена таблиц, представлений, функций, триггеров и хранимых процедур - уникальны в пределах данной схемы.

Имена столбцов, ключей и индексов должны быть уникальны в пределах одной таблицы или представления и т. д.

Константы

В SQL константами считаются любые числовые значения, строки символов, значения, связанные с представлением времени (дата и время), и булевы значения, которые не являются идентификаторами или ключевыми словами. Базы данных на основе SQL разрешают использовать в коде SQL различные константы. Допустимы большинство числовых, символьных и булевых типов данных, а также даты. Например, к числовым типам данных SQL Server можно (среди прочих) отнести типы INTEGER, REAL и MONEY.

Булевы значения, строковые константы и даты выглядят примерно так.:

- TRUE
- 'Hello world!'
- 10CT-28-1966 22:14:30:00'

Таким образом, числовые константы могут выглядеть так.

- 30
- -17
- -853 3888
- -6.66
- $70000
- 2E5
- 7E-3

Операторы

Оператор - это символ, обозначающий действие, выполняемое над одним или несколькими выражениями. Операторы наиболее часто используются в инструкциях DELETE, INSERT, SELECT или UPDATE, а также часто применяются при создании объектов базы данных, таких, как хранимые процедуры, функции, триггеры и представления.

Категории операторов:

- *Арифметические операторы.* Поддерживаются всеми базами данных.
- *Операторы присваивания.* Поддерживаются всеми базами данных.
- *Побитовые операторы.* Поддерживаются Microsoft SQL Server.
- *Операторы сравнения.* Поддерживаются всеми базами данных.
- *Логические операторы.* Поддерживаются в DB2, Oracle, SQL Server и PostgreSQL. *Унарные операторы.* Поддерживаются в DB2, Oracle и SQL Server.

Арифметические операторы

Арифметические операторы выполняют математические действия над двумя значениями любого типа, относящегося к числовой категории.

Арифметический оператор	**Действие**
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток отделения (только SQL Server). Возвращает остаток от операции деления в виде целого числа (integer)

Оператор присваивания

За исключением Oracle, где для этой цели применяется оператор **:=** , оператор присваивания (=) присваивает значение переменной или псевдониму (alias) заголовка столбца. В SQL Server в качестве оператора для присваивания псевдонимов таблицам или заголовкам столбцов может служить ключевое слово **AS**.

Операторы сравнения

Операторы сравнения проверяют равенство или неравенство двух выражений. Результатом операции сравнения является булево значение: TRUE, FALSE или UNKNOWN. Также заметьте, что по стандарту ANSI сравнение выражений, когда одно или оба значения равны NULL, дает результат NULL.

Оператор сравнения	**Действие**
=	Равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
<>	Не равно
! =	Не равно (не соответствует стандарту ANSI)
!<	Не меньше (не соответствует стандарту ANSI)
!>	Не больше (не соответствует стандарту ANSI)

Логические операторы

Логический оператор	Действие
ALL	TRUE, если весь набор сравнений дает результат TRUE
AND	TRUE, если оба булевых выражения дают результат TRUE
ANY	TRUE, если хотя бы одно сравнение из набора дает результат TRUE
BETWEEN	TRUE, если операнд находится внутри диапазона
EXISTS	TRUE, если подзапрос возвращает хотя бы одну строку
IN	TRUE, если операнд равен одному выражению из списка или одной или нескольким строкам, возвращаемым подзапросом
LIKE	TRUE, если операнд совпадает с шаблоном
NOT	Обращает значение любого другого булевого оператора
OR	TRUE, если любое булево выражение равно TRUE
SOME	TRUE, если несколько сравнений из набора дают результат TRUE

Приоритет операторов

1. +, -, ~ (унарные операторы)
2. *, /, % (математические операторы)
3. +, - (арифметические операторы)
4. =, >, <, >=, <=, <>, !=, !>, !< (операторы сравнения)
5. ^ (побитовое исключающее ИЛИ), & (побитовое И), | (побитовое ИЛИ)
6. NOT, AND, ALL, ANY, BETWEEN IN LIKE, OR, SOME = (присваивание значение переменной)

Язык определения данных

- CREATE – создает объектов базы данных
- ALTER – изменяет объект
- DROP –удаляет объект
- Стандарт SQL-92 определяет команды для следующих объектов:
- ASSERTION – утверждения для проверки
- CHARACTER SET – набор символов
- COLLATION – правила сортировки для набора символов
- DOMAIN – домен (пользовательского типа данных столбца).
- SCHEMA – схема (именованной группы объектов)
- TABLE – таблица базы данных
- TRANSLATION – правила преобразования (трансляции) из одного набора символов в другой (используется в операторе TRANSLATE)
- VIEW – представления данных

Типы данных

1. **Символьные строки:**

   CHARACTER(*n*)или CHAR(*n*) — строка фиксированной длины в *n* символов, разделенная пробелами;

   CHARACTER VARYING(*n*)или VARCHAR(*n*) — строка переменной длины с максимальным количеством символов *n*;

   NATIONAL CHARACTER(*n*)или NCHAR(*n*) — строка фиксированной длины с поддержкой международных кодировок

   NATIONAL CHARACTER VARYING(*n*)или NVARCHAR(*n*) — строка переменной длины NCHAR.

Типы данных

2. Битовые данные:

- BIT(*n*) — массив из *n* битов
- BIT VARYING(*n*) — массив длинной до *n* битов

3. Числа:

- INTEGER и SMALLINT — целые числа;
- FLOAT, REAL и DOUBLE PRECISION — вещественные числа;
- NUMERIC(*precision*, *scale*)или DECIMAL(*precision*, *scale*)— вещественное число с указанием в скобках количество знаков до запятой и после запятой.

Типы данных

4. Дата и время:

- DATE — дата (2010-05-30);
- TIME — время (14:55:37);
- TIME WITH TIME ZONE или TIMESTAMP — тоже самое, что и TIME, только исключаются данные о часовом поясе;
- TIMESTAMP — это DATE и TIME соединенные вместе в одной переменной (2010-05-30 14:55:37).
- TIMESTAMP WITH TIME ZONE or TIMESTAMPTZ — тоже самое, что и TIMESTAMP, только исключаются данные о часовом поясе.

Создание запросов в СУБД Access средствами SQL

Запрос — объект базы данных, используемый для выборки или модификации хранимых данных.

В режиме конструктора можно открывать различные запросы: запрос на выборку, перекрестный запрос и запрос на изменение.

Запрос на выборку и перекрестный запрос также можно открыть в режиме таблицы для просмотра результатов.

Запросы на выборку и их использование

Запрос на выборку является наиболее часто используемым типом запроса. Запросы этого типа выбирает данные из одной или нескольких таблиц и отображают их в виде таблицы, записи в которой можно обновлять (с некоторыми ограничениями). Запросы на выборку можно также использовать для группировки записей и вычисления сумм, средних значений, подсчета записей и нахождения других типов итоговых значений.

Для подготовки запросов используются:

- QBE (Query By Example) — язык запросов по образцам,
- SQL (Structured Query Language) — язык структурированных запросов.

В основу языка структурированных запросов входят 4 основных оператора:

- SELECT – используется для выборки записей из таблиц;
- INSERT –используется для добавления записей в таблицу;
- UPDATE – используется для обновления записей таблицы;
- DELETE – используется для удаления записей из таблицы.

Оператор SELECT

Основой SQL является инструкция SELECT, используемая для создания запросов на выборку.

Синтаксис инструкции:

SELECT [ALL | DISTINCT | DISTINCTROW]
список_выбора
FROM имена таблиц
[WHERE критерий поиска]
[GROUP BY имя столбца, имя столбца,...]
[HAVING условие поиска]
[ORDER BY критерий столбца [ASC | DESC]];

SELECT — выбрать (директива) данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями

FROM — из (условие) перечисленных таблиц, в которых расположены эти столбцы

WHERE — где (условие) строки из указанных таблиц должны удовлетворять указанному перечню условий отбора строк

GROUP BY — группируя по (условие) указанному перечню столбцов с тем, чтобы получить для каждой группы единственное агрегированное значение, используя во фразе SELECT SQL – функции: SUM (сумма), COUNT (количество), MIN (минимум), MAX (максимум), AVG (среднее значение)

HAVING — имея в результате лишь те группы, которые удовлетворяют указанному перечню условий отбора групп (условие)

ORDER BY — спецификация сортировки (условие) определяет порядок сортировки: ASC – сортировка по возрастанию, DESC - сортировка по убыванию.

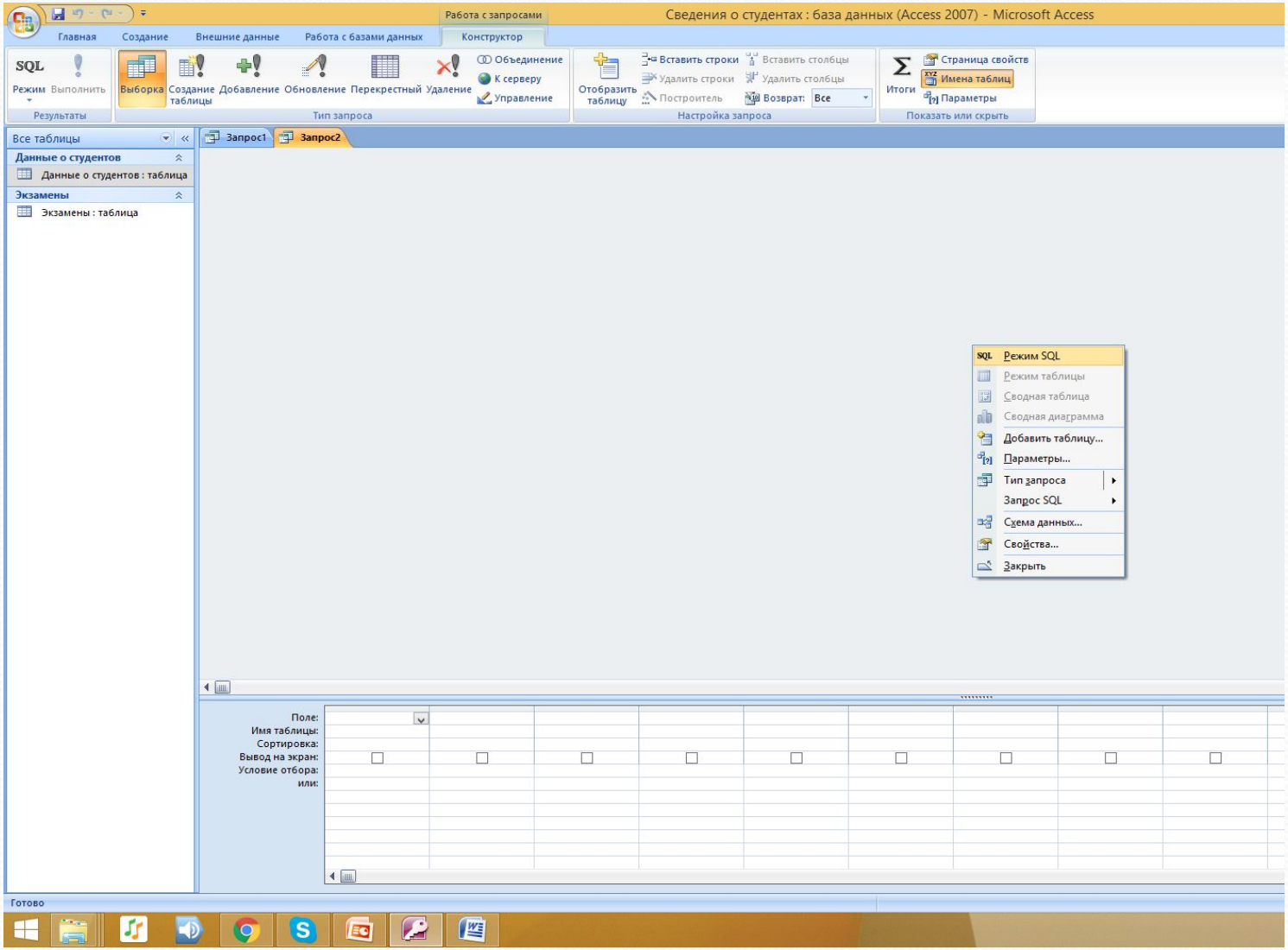
Запросы с использованием единственной таблицы

Все запросы на получение практически любого количества данных из одной или нескольких таблиц выполняются с помощью единственного предложения SELECT. В общем случае результатом реализации предложения SELECT является другая таблица. К этой новой (рабочей) таблице может быть снова применена операция SELECT и т.д., т.е. такие операции могут быть вложены друг в друга. Представляет исторический интерес тот факт, что именно возможность включения одного предложения SELECT внутрь другого послужила мотивировкой использования прилагательного "структуризированный" в названии языка SQL.

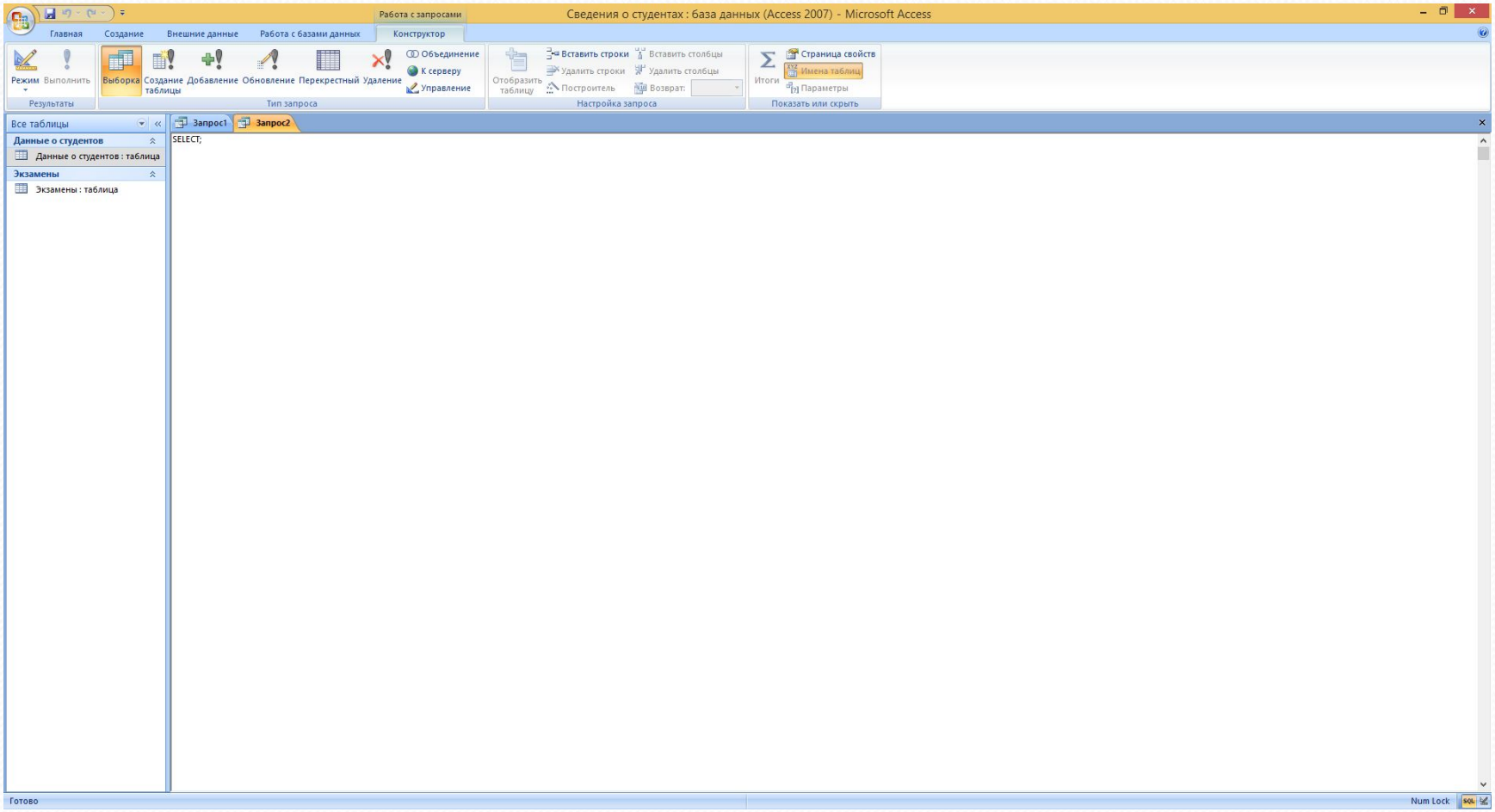
ПРЕДИКАТЫ

1. Сравнения =, <>, >=, <, <=
2. В интервале - “между” BETWEEN a1 and a2
3. Входит в множество IN (=[Предмет] IN (“История”,“Информатика”)
4. Подобие < имя > Like < образец >
   (что) (с чем сравнивать)

Режим SQL в MS Access



Окно SQL


Работа с запросами
Сведения о студентах : база данных (Access 2007) - Microsoft Access
Главная
Создание
Внешние данные
Работа с базами данных
Конструктор
Режим
Выполнить
Выборка
Создание таблицы
Добавление
Обновление
Перекрестный
Удаление
Объединение
К серверу
Управление
Отобразить таблицу
Вставить строки
Удалить строки
Построитель
Вставить столбцы
Удалить столбцы
Возврат:
Итоги
Страница свойств
Имена таблиц
Параметры
Результаты
Тип запроса
Настройка запроса
Показать или скрыть
Все таблицы
Данные о студентов
Данные о студентов : таблица
Экзамены
Экзамены : таблица
Запрос1
Запрос2
SELECT;
Готово
Num Lock

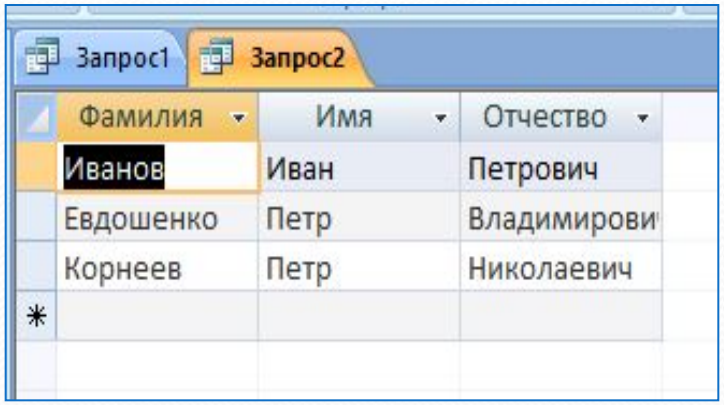
Рассмотрим синтаксис запросов на выборку:

1. Запрос на выборку фамилии, имени и даты рождения студента

   SELECT Фамилия, Имя, Отчество

   FROM Данные;

Результат:



При необходимости получения полной информации о Студенте, можно было бы дать запрос

SELECT Фамилия, Имя, Отчество, Город, Адрес, Телефон (и т.д.)

FROM Данные

или использовать его более короткую нотацию:

SELECT * (Звездочка (*) может применяться для вывода полного списка столбцов)

FROM Данные

Результат:

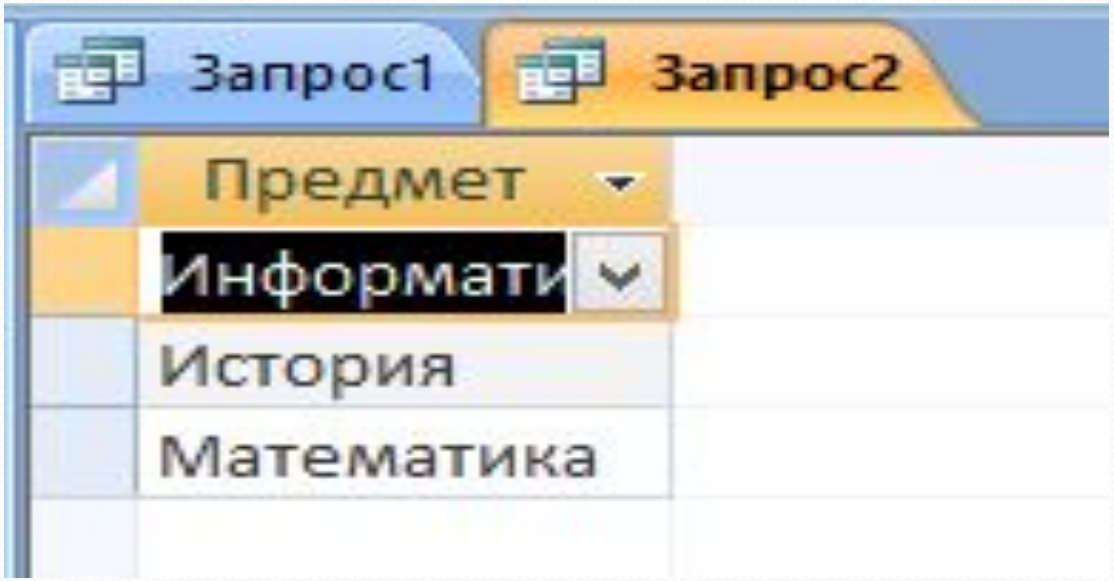
Запрос1

Номер заче	Фамилия	Имя	Отчество	Факультет	Курс	Группа	Дата_рожд	Стипендия	Город	Адрес	Телефон
2131	Иванов	Иван	Петрович	ФЭУ	2	14ЭУП	01.02.1997	2 500,00 р.	Оренбург	Загородное, 5	231254
2133	Евдошенко	Петр	Владимирови	ФЭУ	2	14ГиМУ	02.03.1998	1 980,00 р.	Гай	Полевая	215740
2405	Корнеев	Петр	Николаевич	ФЭФ	3	13ПИ	11.12.1996	4 500,00 р.	Оренбург	Победа	124441

Для исключения дубликатов и одновременного упорядочения перечня необходимо дополнить запрос ключевым словом DISTINCT (различный, различные), как показано в следующем примере:

```
SELECT DISTINCT Предмет;
FROM Экзамены;
```

Результат:

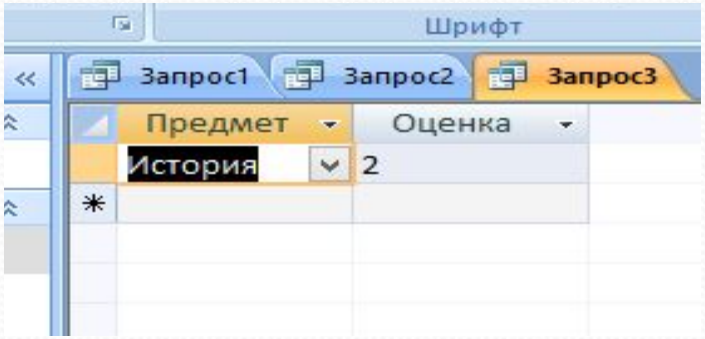


В синтаксисе фразы WHERE показано, что для отбора нужных строк таблицы можно использовать операторы сравнения = (равно), <> (не равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), которые могут предваряться оператором NOT, создавая, например, отношения "не меньше" и "не больше".

Так, для получения перечня предметов, по которым были получены 2, можно сформировать запрос

```
SELECT Экзамены.Предмет, Экзамены.Оценка
FROM Экзамены
WHERE (((Экзамены.Оценка)="2"));
```

Результат:



Создайте запросы:

1. На получения предметов, по которым были получены 5 или 4.
2. На получение списка студентов, проживающих в г.Оренбурге.
3. Список студентов, получающих стипендию более 1600.

Оператор INSERT

```
INSERT INTO <имя_таблицы> [(<имя столбца_1> [, <имя
  столбца_1> ...])] {VALUES (<значение_1> [,
  <значение_2> ...]) | <выражение SELECT>};
```

Так, например, чтобы ввести строку в таблицу Продавцов, вы можете использовать следующее условие:

1. `INSERT INTO Salespeople VALUES (1001, 'Peel', 'London', .12);`
2. `INSERT INTO Customers (city, cname, cnum) VALUES ('London', 'Honman', 2001);`

Оператор UPDATE

Теперь, вы должны узнать как изменять некоторые или все значения в существующей строке. Это выполняется командой UPDATE.

```
UPDATE TABLE <имя_таблицы>
  SET <имя столбца_1> = <значение_1> [, <имя
  столбца_2> = <значение_2> ...]
   [WHERE <условие>];
```

Например

1. `UPDATE Customers SET rating = 200;`
2. `UPDATE Customers  SET rating = 200 WHERE snum = 1001;`
3. `UPDATE Salespeople SET sname = 'Gibson',city = 'Boston',comm = .10 WHERE snum = 1004;`

Оператор DELETE

Вы можете удалять строки из таблицы командой модификации - DELETE. Она может удалять только введенные строки, а не индивидуальные значения полей.

```
DELETE FROM <имя_таблицы> [WHERE <условие>];
```

Например

1. `DELETE FROM Salespeople WHERE snum = 1003;`
2. `DELETE FROM Salespeople WHERE city = 'London';`

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия – режим доступа: ru./wiki/SQL
2. Вопросы практического програмирования – режим доступа: mstu.edu/education/materials/zelenkov/ch_4_7.html
3. Введение в структурированный язык запросов SQL – режим доступа: intuit/department/database/sql/1/
4. Всё про Sql – режим доступа: sql/
5. Введение в стандарты языка баз данных SQL – режим доступа: citforum/database/sqlbook/index.shtml

СПАСИБО ЗА ВНИМАНИЕ