Санкт-петербургский государственный университет
Кафедра информационно-аналитических систем

Помыткина Татьяна Борисовна

Базы данных и СУБД

Учебный практикум

Дисциплина [002191] «Базы данных и СУБД»
по направлению 02.03.03
Математическое обеспечение и администрирование информационных систем

Санкт-Петербург, 2022

Содержание

Введение

Учебный практикум предназначен для студентов 2 курса СПбГУ, получающих образование по направлению «Математическое обеспечение и администрирование информационных систем», освоивших программирование в объеме предыдущих курсов и владеющих базовыми навыками работы с компьютером.

Задачи практикума: изучение основных понятий, связанных с системами управления базами данных, получение практических навыков проектирования реляционных баз данных, изучение стандартного языка баз данных SQL, построение SQL запросов различной степени сложности.

По окончании курса студенты будут уметь:
1. Описывать структуры данных в терминах модели «Сущность-связь».
2. Создавать объекты базы данных.
3. Описывать ограничения целостности для данных.
4. Писать запросы на языке SQL.
5. Создавать представления, функции, процедуры и триггеры.

Базовые понятия

I. Базы данных и СУБД

<u>База данных</u> – это набор информации, относящейся к некоторой предметной области,структурированной,сохраненной в электронном виде, обрабатываемой специальными программами.

<u>Система управления базами данных</u> (СУБД, Data Base Management System) - это совокупность программных средств, предназначенных для создания и использования баз данных.

<u>Основные задачи СУБД</u>:
- предоставлять пользователям доступ к данным без знания их физической организации;
- обеспечивать безопасность хранения данных и доступа к ним;
- обеспечивать бесконфликтность одновременного доступа разных пользователей к одним данным;
- реализовывать транзакции.

<u>СУБД различаются</u>:
- по поддержке моделей данных (реляционные, иерархические, сетевые, объектные,…)
- по способу доступа к БД (файл-серверные, клиент-серверные, встраиваемые)
- по масштабам поддерживаемых баз
- по платформам реализации

<u>Реляционная модель</u> – это модель “сущность-связь”. Реляционную базу данных можно рассматривать как коллекцию таблиц, связанных между собой.

Примеры реляционных СУБД:

Oracle, DB2, MS SQL Server, PostgreSQL, FireBird, MySQL.

Преимущества реляционной модели:

- универсальность;
- относительная простота реализации;
- поддержка стандартным интерфейсом SQL.

Независимо от физической организации, таблицу можно рассматривать как набор записей и полей. Записи таблицы содержат сведения об однотипных объектах, имеющих стандартный набор параметров.

Данные в таблицах должны удовлетворять следующим принципам:

- Каждое значение на пересечении строки и столбца *атомарно*.
- Значения данных в одном и том же столбце принадлежат к одному и тому же типу данных.
- Каждая запись в таблице уникальна (т. е. в таблице не может быть двух и более полностью совпадающих строк).
- Каждый столбец имеет уникальное имя.
- Последовательность записей в таблице и последовательность полей в записи несущественна.

Связи между таблицами позволяют логически организовать разнородные данные.

Как проектировать реляционную структуру?

1. Выделить сущности предметной области и их взаимосвязи.
2. Нормализовать получившуюся структуру.

Нормализация – это разделение информации по разным таблицам таким образом, чтобы максимально устранить дублирование данных и при этом минимально усложнить поиск данных.

При нормализации таблицы последовательно приводятся в следующие нормальные формы:

- Первая нормальная форма (разделение исходных данных на логические составляющие – таблицы, и назначение первичных ключей)
- Вторая нормальная форма (помещение в отдельную таблицу данных, которые только частично зависят от первичного ключа)
- Третья нормальная форма (устранение из таблицы данных, не зависящих от ее первичного ключа)
- Четвертая нормальная форма
- Пятая нормальная форма

Таблица БД называется таблицей в 1NF, если она удовлетворяет следующим условиям:

- поле любой записи содержит ровно одно значение (атомарно)
- столбцы таблицы однозначно поименованы (в таблице отсутствуют повторяющиеся группы полей)

Таблица БД называется таблицей во второй нормальной форме, если:

- она в первой нормальной форме
- любое непервичное поле полностью функционально зависит от любого возможного ключа (любое неключевое поле должно однозначно идентифицироваться ключем).

Чтобы перейти от первой нормальной формы ко второй, нужно выполнить следующие шаги:

- Определить, на какие части можно разбить первичный ключ, так чтобы некоторые из неключевых полей зависели от одной из этих частей (эти части не обязаны состоять из одной колонки!).
- Создать новую таблицу для каждой такой части ключа и группы зависящих от нее полей и переместить их в эту таблицу. Часть бывшего первичного ключа станет при этом первичным ключом новой таблицы.
- Удалить из исходной таблицы поля, перемещенные в другие таблицы, кроме тех их них, которые станут внешними ключами.

Чтобы перейти от второй нормальной формы к третьей, нужно выполнить следующие шаги:
- Определить все поля (или группы полей), от которых зависят другие поля.
- Создать новую таблицу для каждого такого поля (или группы полей) и группы зависящих от него полей и переместить их в эту таблицу. Поле (или группа полей), от которого зависят все остальные перемещенные поля, станет при этом первичным ключом новой таблицы.
- Удалить перемещенные поля из исходной таблицы, оставив лишь те из них, которые станут внешними ключами.

Чем выгодна нормализация:
- Лучшая общая организация базы данных
- Сокращение избыточности информации
- Непротиворечивость информации внутри базы данных
- Более гибкий проект базы данных
- Большая безопасность данных

Контрольные вопросы

- Что такое база данных?
- Что такое СУБД?
- В чем главные преимущества реляционной модели?
- Для чего используются транзакции?
- Каковы главные критерии, по которым данные распределяются по таблицам?
- Что такое атомарность?
- Почему необходима уникальность записей в таблицах?
- Чем первичный ключ отличается от внешнего?
- Зачем нужна нормализация?
- Каковы основные принципы нормализации?

II. SQL

SQL (Structured Query Language) – язык структуризованных запросов.

Это универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

SQL основывается на реляционной алгебре.

Основные составляющие языка:

- операторы определения данных Data Definition Language (DDL)
- операторы манипуляции данными Data Manipulation Language (DML)
- операторы управления доступом к данным Data Control Language (DCL)
- операторы управления транзакциями Transaction Control Language (TCL)

Data Manipulation Language

Операторы DML:

Добавление записи - INSERT
Изменение записи - UPDATE
Удаление записи - DELETE
Выборка данных – SELECT

Синтаксис оператора SELECT:

```
SELECT [DISTINCT]
{{функция агрегирования.. | выражение для вычисления значения
[AS имя столбца] }.,..}
| { спецификатор,* }
| *
FROM { { имя таблицы [AS] [имя корреляции]
[(имя столбца.,..)]}
| { подзапрос [AS] имя корреляции
[ имя столбца.,..]}
| соединенная таблица
} .,..
[ WHERE предикат ]
GROUP BY { { [ имя таблицы |имя корреляции }.] имя столбца}.,..
[ HAVING предикат]
[ { UNION | INTERSECT | EXCEPT ) (ALL]
[ CORRESPONDING [ BY ( имя столбца.,..) ] ]
оператор select | {TABLE имя таблицы}
| конструктор значений таблицы]
[ ORDER BY { { столбец-результат [ ASC | DESC ]}.,..}
| { { положительное целое [ASC | DESC ] }.,..};
```

Функции агрегирования:

COUNT(*)	количество записей
COUNT(DISTINCT expression)	количество различных записей
AVG([DISTINCT] expression)	среднее значение [среди различных]

MAX(expression)	максимальное значение
MIN(expression)	минимальное значение
SUM([DISTINCT] expression)	сумма значений [различных]

Фильтры:

1. expression (=, >, <, AND, OR)
   expression [NOT] BETWEEN expression AND expression
2. field-name IS [NOT] NULL
3. field-name [NOT] LIKE ‘string’ [ESCAPE ‘character’]
4. expression [NOT] IN (value-list | SELECT-statement)

Группирование - это перекомпоновка таблицы по группам записей, с последующим применением фразы SELECT; в результате получается столько строк, сколько нашлось групп.

Каждое выражение между SELECT и FROM должно быть способным принимать единственное значение для группы:

- значение столбца, указанного в GROUP BY,
- арифметическое выражение, включающее это значение,
- константа,
- функция, которая выдает единственное значение.

Сложные по синтаксису запросы:

- Вложенные запросы классические
- Использование псевдонимов (в т.ч. во вложенных запросах)
- Вложенные запросы в выражениях
- Объединение запросов

Механизмы обработки вложенных запросов:

- Первым обрабатывается вложенный подзапрос самого нижнего уровня; множество значений, полученное в результате его выполнения, используется при реализации подзапроса более высокого уровня и т.д.
- Выбирается первая строка результата внешнего запроса, значения ее столбцов используются во внутреннем запросе, часть его реализуется, затем берется вторая строка и т.д. (т.н. коррелированные запросы).

Объединение запросов:

Результаты двух и более запросов м.б. объединены, если они выводят одинаковое число столбцов совместимых типов.

Синтаксис операторов редактирования данных:

```
INSERT  INTO <таблица>
                    [ (<столбец> [,<столбец>] ...) ]
          VALUES (<константа> | <переменная>}
                    [,{<константа> | <переменная>}] ...);
```

```
INSERT INTO <таблица> [ (<столбец> [,<столбец>] ...) ]
                                                                   <подзапрос>;

UPDATE <таблица>
        SET <столбец> = <значение>
              [,<столбец> = <значение>] ...
            [WHERE фраза]

UPDATE  <таблица> SET <столбец> = <значение>
                                 [, <столбец> = <значение>] ...
                 [WHERE  фраза с подзапросом];

DELETE  FROM <таблица> [WHERE  фраза];


DELETE  FROM <таблица>
              [WHERE  фраза с подзапросом];
```

<u>Data Definition Language</u>

Основные объекты базы данных:

1. Таблица - Table
2. Ограничение - Constraint
3. Представление - View
4. Индекс - Index
5. Секвенция - Sequence
6. Триггер - Trigger
7. Хранимая процедура - Stored Procedure

1. Таблица (Table).

Создание таблицы:

```
CREATE TABLE <имя_таблицы> (
      <имя_столбца> <тип_данных>,
      <имя_столбца> <тип_данных>,
  ...);
```

Модификация таблицы:

Добавление столбца

```
        ALTER TABLE <имя_таблицы>
        ADD COLUMN <имя_столбца> <тип_данных>
```

Удаление столбца

```
   ALTER TABLE <имя_таблицы>
         DROP COLUMN <имя_столбца>
```

Изменение столбца

```
   ALTER TABLE <имя_таблицы>
          ALTER COLUMN <имя_столбца> [options]
```

Добавление правил целостности:

```
  ALTER TABLE <имя_таблицы>
```

```
        ADD [CONSTRAINT <имя_правила>]
          PRIMARY KEY (<имя_поля>)

ALTER TABLE <имя_таблицы>
        ADD [CONSTRAINT <имя_правила>]
          FOREIGN KEY (<имя_поля>)
            REFERENCES student (<имя_поля>)
              [ON DELETE\UPDATE CASCADE]
```

Удаление правил целостности:

```
ALTER TABLE <имя_таблицы>
          DROP CONSTRAINT <имя_правила>
```

Удаление таблицы:

```
DROP TABLE <имя_таблицы>
```

2. Ограничения (Constraint)

   - Целостность по ссылкам
     PK
     FK
     CASCADE
   - Целостность по сущностям
     NOT NULL
     UNIQUE
     CHECK

3. Представление (VIEW) – это именованный запрос, описание которого хранится в базе данных (его можно интерпретировать как пустую таблицу с “правилом заполнения”).

   Для каких целей используются представления:
   - Обеспечение независимости пользовательских программ от изменения логической структуры базы данных.
   - Разграничение доступа к данным.

   Синтаксис:

   ```
   CREATE VIEW <имя представления> [(<список полей>)]
                                  AS <оператор SELECT>;

   DROP VIEW <имя представления>;
   ```

4. Индексы (Index) - это именованные, специальным образом организованные структуры (обычно b-деревья), хранящие уникальные значения указанных полей таблицы со ссылками на те ее записи, где встречаются эти значения (подобие предметного указателя книги). В операторах SQL DML нет упоминаний индексов, решение о использовании какого-либо индекса при обработке запроса принимает оптимизатор СУБД, учитывая при этом множество факторов.

   Синтаксис:

```
CREATE [UNIQUE] INDEX <имя_индекса>
      ON <имя_таблицы>
                       <имя_столбца>   [DESC],
                       <имя_столбца>   [DESC],
                       …;

DROP INDEX <имя_индекса> ON <имя_таблицы>;
```

5. Секвенция (Sequence) (Последовательности, Генераторы, …). Используется для генерации последовательности чисел. Секвенция не связана с таблицами и полями. Ее можно представлять себе как число с правилом его изменения. Используется посредством вызова специальных функций. Синтаксис создания и использования различается для разных СУБД.

6. Триггер (Trigger) - это программа обработчик событий.

   Основные события, которые могут произойти в базе:

   - Создание записи
   - Изменение записи
   - Удаление записи

   (в некоторых СУБД триггеры могут обрабатывать и другие события, например, выборку данных, слияние данных, создание, изменение и удаление объектов схемы).

   Синтаксис создания и использования различается для разных СУБД.

7. Хранимая процедура (Stored Procedure) - это процедур, хранящаяся в БД, написанная на языке СУБД.

   Хранимая процедура может быть вызвана из другой хранимой процедуры, из триггера, из любого программного кода. Вызов может осуществляться с передачей значений параметров. Синтаксис создания и использования различается для разных СУБД.

Контрольные вопросы

- Является ли SQL языком программирования? Почему?
- Какое отношение SQL имеет к реляционной алгебре?
- Чем отличаются друг от друга стандарты SQL?

- На какие большие группы разделяются все операторы SQL?
- Каково назначение операторов манипуляции данными?
- Что такое функции агрегирования?
- Зачем нужны фильтры в операторе SELECT?
- В чем состоит принцип группирования данных?
- Каково предназначение четырех типов сложных по синтаксису запросов?
- В чем состоят механизмы исполнения вложенных запросов?
- В каких случаях возможно объединение запросов?
- Какие типы объектов встречаются в реляционных базах данных?
- Что такое правила ограничений?
- Является ли представление одним из видов таблиц?
- Для чего строятся индексы в базах данных?
- Какие события могут инициировать начало работы триггера?
- Где хранятся и кем вызываются на исполнение хранимые процедуры?

Общие задания

1. Создание ER-диаграммы по текстовому описанию предметной области

Создайте ER-диаграмму по текстовому описанию следующей предметной области:

Вы создали компанию, которая обеспечивает для жителей окрестных домов следующие услуги: выгул собак и причесывание кошек.
Разработайте структуру хранения данных так, чтобы можно было работать со следующей информацией:

- Списки владельцев животных (ФИО, адрес, телефон, питомец).
- Списки кошек (кличка, возраст, порода) и собак (кличка, порода, время выгула).
- Списки исполнителей (ФИО, адрес, телефон, специализация).
- Заказы на неделю (владелец, питомец, исполнитель, дата, отметка о выполнении, оценка работы).
- Пакет заказов на день.
- Отчеты для владельцев животных.
- Рейтинг исполнителей.

---

2. Оптимизация скрипта

Ознакомьтесь с предложенным скриптом создания и заполнения демонстрационной базы данных, внесите свои предложения по приведению структуры к третьей нормальной форме.

3. Доработка скрипта

Дополните структуру двумя новыми сущностями: *прививка* и *вид прививки*: (“Мы должны знать даты постановки питомцам прививок и иметь сканы документов, подтверждающих факты вакцинации”). Доработайте скрипт создания базы данных, добавив необходимые операторы создания таблиц и правил целостности, операторы добавления тестовых данных в новые таблицы.

4. Простые запросы к одной таблице

Напишите следующие запросы к таблице A0_Pet:

Table - dbo.A0_Pet | Summary

Pet_ID	Nick	Breed	Age	Description	Pet_Type_ID	Owner_ID
1	Bobik	unknown	3		1	1
2	Musia		12		2	1
3	Katok		2	crazy guy	2	1
4	Apelsin	poodle	5		1	2
5	Partizan	rottweiler	5	very big	2	2
6	Daniel	spaniel	14		1	3
7	Model		5		3	4
8	Markiz	poodle	1		1	5
9	Zombi	Siamese	7	wild	2	6

1. Данные на Партизана.
2. Клички и породы всех питомцев с сортировкой по возрасту.
3. Питомцы, имеющие хоть какое-нибудь описание.
4. Средний возраст пуделей.
5. Количество владельцев.

5. Простые запросы к связанным таблицам

Напишите следующие запросы к таблицам демонстрационной базы данных:

1. Данные на Партизана (включая вид животного).
2. Список всех собак с кличками, породой и возрастом.
3. Средний возраст кошек.
4. Время и исполнители невыполненных заказов.
5. Список хозяев собак (имя, фамилия, телефон).
6. Все виды питомцев и клички представителей этих видов (внешнее соединение).

6. Группирование данных

Напишите следующие запросы к таблицам демонстрационной базы данных:

1. Сколько имеется питомцев в возрасте 1 год, 2 года, и т.д..
2. Сколько имеется котов, собак и т.д. в возрасте 1 год, 2 года, и т.д.
3. Виды питомцев, средний возраст которых меньше шести лет:
4. Фамилии сотрудников, выполнивших более пяти заказов

7. Запросы сложной структуры

Напишите следующие запросы к таблицам демонстрационной базы данных:

1. Все оценки по заказам, исполнителями которых являлись студенты.
(используйте IN (SELECT...))
2. Фамилии исполнителей, не получивших еще ни одного заказа.
Последовательность отладки:
- id исполнителей, у которых нет заказов
(используйте NOT IN (SELECT...))
- фамилии этих исполнителей
(присоедините еще одну таблицу)

3. Список заказов (вид услуги, время, фамилия исполнителя, кличка питомца, фамилия владельца).

(используйте псевдонимы)

4. Общий список всех комментариев, имеющихся в базе. (“Хватит захламлять базу, посмотрите, что вы пишите в комментариях!”).

(используйте UNION)

8. Модификация данных

Напишите шесть операторов модификации данных в демонстрационной базе:

1. Добавление питомцев и хозяев (два вида синтаксиса).
2. Изменение данных о заказах (два вида синтаксиса).
3. Удаление данных об исполнителях (два вида синтаксиса).

9. Создание представлений

Создайте три различных представления (view) в демонстрационной базе:

1. Питомцы и их хозяева.
2. Заказы за последний месяц.
3. Вся информация о собаках (редактируемое).

10. Создание триггеров и хранимых процедур

Создайте триггеры на откат любых изменений информации в базе данных и параметризованную процедуру добавления хозяина с питомцем.

Задания для самостоятельной работы

Задание 1. Создание ER-диаграммы по индивидуальной постановке задачи

Базовые понятия: *База данных, Реляционная модель, Таблица, Связь, Нормализация*.

Вспомогательные материалы - презентация DataBase_01_02.

Рекомендуемый графический редактор - MS Visio.

Создайте ER-диаграмму по текстовому описанию индивидуальной предметной области, полученному у преподавателя.

Требования к диаграмме:

- нужно создать 5-7 таблиц, связанных между собой, причем желательно, чтобы три основные из них представляли собой связку, похожую на связки [хозяин-заказ-питомец] или [студент-оценка-предмет], т.е. реализовали связь многие-ко-многим через более-менее полноценную сущность (чтобы потом легче было строить типовые учебные запросы);

- позаботьтесь о соответствии разработанной структуры третьей нормальной форме;

- наименования таблиц и полей следует писать латиницей (из спецсимволов использовать только подчерк), длина - до 32 символов;

- нужно создать несколько числовых полей, чтобы была возможность суммировать, вычислять среднее и проч. (например, количество, сумма, возраст), советую не увлекаться датами и не использовать в основных таблицах сложных типов данных.

Замечание: Если в описании вашей предметной области недостаточно информации, чтобы построить диаграмму, удовлетворяющую всем перечисленным требованиям, придумайте какую-нибудь правдоподобную дополнительную функциональность (или обратитесь за помощью к преподавателю). Если описание, наоборот, избыточное, ограничьтесь наиболее понятной функциональностью.

Сохраните диаграмму:

- для себя (чтобы при необходимости можно было подправить ее)

- для преподавателя - как картинку, в файле с именем 01_XX_erd.jpg. (где XX - код вашей предметной области) в папке вашей группы.

Покажите диаграмму преподавателю. Не приступайте к выполнению следующего задания, пока преподаватель не одобрит эту диаграмму.

Задание 2. Создание скрипта создания базы данных

Базовые понятия: *Таблица, Связь, Нормализация, SQL, Data Manipulation Language, Data Definition Language*.

Вспомогательные материалы - презентация DataBase_01_02.

1. После одобрения преподавателем вашего предыдущего задания (er-диаграммы) напишите скрипт создания схемы базы данных, соответствующей вашей диаграмме. В этом же скрипте должны быть и операторы заполнения таблиц тестовыми данными.

   Предостережения, советы и требования:

   - для полей, по которым строятся ключи (PK и FK) рекомендуется использовать тип данных integer
   - типы связанных полей (PK и FK) должны совпадать
   - для начала следует избегать нестандартных типов данных (под стандартными имеются в виду integer, decimal, varchar, datetime)
   - тестовые данные д.б. более-менее осмысленными, кол-во записей в каждой таблице - от 3 до 20
   - обратите внимание, что последовательность операторов удаления таблиц не может быть случайной (не забудьте отладить и эту часть скрипта)

   Временно сохраните скрипт в любом месте под любым именем в формате текстового файла (.txt или .sql).

2. Создайте с помощью скрипта свою схему базы данных под MS SQL Server, тем самым отлаживая скрипт.

   Запуск программы-клиента:
   All Programs -> Microsoft SQL Server 2008 -> SQL Server Management Studio

   Параметры соединения с сервером (Connect to Server):

   Server name: mathsrvdb.ad.pu.ru
   Authentification: SQL Server Authentification
   Login: ******
   Pasword: *******

   За подробностями - к преподавателю.

3.

Сдайте отлаженный скрипт: сохраните его в файле с именем 02_XX_script.sql (где XX - код вашей предметной области) в папке \\MATHSRVFS\POM\<ваша фамилия>\

Задание 3. Создание простых запросов к одной таблице

Базовые понятия: *SQL, Data Manipulation Language, оператор SELECT.*

Вспомогательные материалы - презентация DataBase_03.

Придумайте несколько простых запросов к разным своим таблицам, отладьте их в среде MS SQL Server и сдайте в виде текстового файла (03_XX.sql, где XX - код вашей предметной области), содержащего формулировки запросов и операторы select, например:

```
==================================================

-- 1. Фамилии студентов, у которых неизвестен телефон

SELECT name FROM student WHERE phone IS NULL OR phone='';

-- 2. Средняя стипендия студенток

SELECT AVG(grant_st) FROM student WHERE sex = 'ж';

...

==================================================
```

Требования:

- Запросов должно быть не менее пяти.
- В одном запросе - обращение только к одной таблице.
- Запросы должны быть разнообразными - в рамках материалов презентации DataBase_03, в них обязательно должны присутствовать все следующие конструкты:

  DISTINCT
NULL
COUNT и\или другие функции агрегирования
BETWEEN
LIKE
NOT и\или другие логические операторы
IN
ORDER BY
DESCENDING

- Не увлекайтесь выводом ссылочных полей (id) - разве что для отладки, выводите в основном значимую информацию.

- Результат запроса не должен быть пустым, при необходимости добавьте дополнительные тестовые данные (и вставьте соответствующие INSERT в текст сданного задания)

- Не забудьте написать текстовые формулировки, поясняющие суть запросов.

Задание 4. Создание простых запросов к связанным таблицам

Базовые понятия: *SQL, Data Manipulation Language, оператор SELECT.*

Вспомогательные материалы - презентация DataBase_04.

Придумайте несколько запросов, каждый из которых обращается более чем к одной таблице, отладьте их в среде MS SQL Server и сдайте в виде текстового файла (04_XX.sql), содержащего формулировки запросов и операторы select (так же, как в предыдущем задании).

Требования:

- Запросов должно быть не менее пяти.

- В каждом запросе - обращение более чем к одной таблице.

- Запросы должны быть разнообразными - в рамках материалов презентации DataBase_04:

- Используйте оба варианта синтаксиса соединения таблиц:
  ... FROM student, marks WHERE student.student_id=marks.student_id ...
  ... FROM student JOIN marks ON student.student_id=marks.student_id ...

- Хотя бы в одном запросе используйте внешнее соединение двух, трех или более таблиц (для выразительности результата добавьте новые данные, например, нового клиента, у которого еще нет ни одного заказа, и вставьте соответствующие INSERT в текст сданного задания).

- Приветствуется использование фильтров / сортировок / функций агрегирования.

- Не забудьте написать текстовые формулировки, поясняющие суть запросов.

Задание 5. Создание запросов на группирование данных

Базовые понятия: *SQL, Data Manipulation Language, оператор SELECT.*

Вспомогательные материалы - презентация DataBase_05.

Придумайте несколько запросов на группирование данных (используйте дополнительно функций Transact SQL), отладьте их в среде MS SQL Server и сдайте в виде текстового файла (05_XX.sql), содержащего текстовые формулировки и сами операторы select (так же, как в предыдущих заданиях).

Требования:

- Запросов должно быть не менее пяти.

- В не менее чем трех запросах нужно задействовать группирование, в том числе и на соединениях таблиц, не менее чем в одном из них - фильтр на группы - см. презентацию DataBase_05 (при необходимости добавьте дополнительно тестовых данных - в этом случае не забудьте вставить в 05_XX.sql соответствующие операторы insert и упомянуть об этом в сданном задании или просто в задание вставьте эти дополнительные операторы insert).

- В не менее чем двух запросах нужно использовать любые функции Transact SQL (но не стандартные для sql функции агрегирования count, avg, sum, min, max), список функций см. в MS SQL Server Management Studio, в панели Object Explorer (Обозреватель объектов):
  Databases\2016_371\Programmability\Functions\System Functions

- Не забудьте написать текстовые формулировки, поясняющие суть запросов.

Задание 6. Создание запросов сложной структуры

Базовые понятия: *SQL, Data Manipulation Language, Data Definition Language, оператор SELECT.*

Вспомогательные материалы - презентация DataBase_06.

Придумайте несколько "сложных" запросов (см. презентацию DataBase_06), отладьте их в среде MS SQL Server и сдайте в виде текстового файла (06_XX.sql), содержащего текстовые формулировки и сами операторы select (так же, как в предыдущих заданиях).

Требования:

- Запросов должно быть не менее четырех, среди них обязательно:
  - вложенный запрос с использованием IN или = ,<, >, ...
  - вложенный запрос с использованием ANY, ALL или EXISTS
  - запрос с использованием псевдонимов
  (любой вариант кроме 2.1. - см. презентацию)
  - объединение запросов
  (если сложный запрос оказался искусственно усложненным вариантом простого запроса - приведите и простой вариант этого запроса)

- Не забудьте написать текстовые формулировки, поясняющие суть запросов.

Задание 7. Модификация данных

Базовые понятия: *SQL, Data Manipulation Language, оператор SELECT, оператор INSERT, оператор UPDATE, оператор DELETE.*

Вспомогательные материалы - презентация DataBase_07.

Придумайте 4 оператора модификации данных, отладьте их в среде MS SQL Server и сохраните в текстовом файле (07_XX.sql), содержащем текстовые формулировки и операторы.

1. Добавление в любую из таблиц новых записей с сохранением последовательной нумерации (с использованием max).

2. Изменение данных в любой из таблиц с использованием подзапроса.

3. Удаление части записей из любой таблицы с использованием подзапроса.

4. Удаление записи и связанных с ней, с предварительным пересозданием правила целостности "каскадное удаление".
   (выберите такие "главную" и "зависимую" таблицы, чтобы на "зависимую" никто не ссылался, иначе придется пересоздать более одного правила целостности).

Задание 8. Создание представлений

Базовые понятия: *SQL, Data Manipulation Language, оператор SELECT, Data Definition Language, Представление.*

Вспомогательные материалы - презентация DataBase_08.

Придумайте представления (view) по своим таблицам, отладьте в среде MS SQL Server операторы их создания и использования, выложите их в текстовый файл 08_XX.sql с пояснениями.

Требования:

- Не менее трех представлений.

- Представления должны быть "потенциально востребованными" - как основа для различных выборок (для вывода в интерфейсе, для отчетов, ...).

- Кроме оператора создания, для каждого представления нужно добавить один-два оператора его использования - выборка, редактирование или удаление данных через представление.

- Можно использовать свои удачные старые запросы, но внесите в них хотя бы минимальные изменения...

Задание 9. Изменение структуры базы данных

Базовые понятия: *SQL, Data Definition Language, Представление, Индекс, Триггер, Хранимая процедура.*

Вспомогательные материалы - презентация DataBase_08.

1. Придумайте и создайте "полезный" триггер для любой из ваших таблиц (снабдите его пояснительным текстом). Напишите также оператор, выполнение которого инициирует работу триггера и оператор(ы), демонстрирующие результат этой работы (например, создаем триггер, который при добавлении студенту любой оценки создает для него еще и бонусную пятерку по физкультуре, тогда нужно написать и оператор добавления кому-нибудь какой-нибудь оценки и окружить его двумя операторами select, показывающими оценки того самого студента до и после выполения оператора insert, инициирующего запуск триггера).

2. Напишите операторы создания двух индексов. Первый должен быть построен по комбинации двух или более полей, которые, вероятно, "будут особо востребованы в запросах", а второй пусть будет предназначен для контроля уникальности каких-либо значений (например, "у студента по предмету не должно быть более одной оценки").

Создайте в базе данных хранимую процедуру (снабдите ее пояснительным текстом). Можно усложнить алгоритм за счет увеличения числа параметров, различных возможностей языка Transact-SQL, конструкции cursor и т.д..

Сдайте результаты в виде текстового файла (09_XX.sql).
Не забудьте написать комментарии для триггера и для процедуры.

(материалы - см. презентацию DataBase_09)

Рекомендуемая литература

1. Дейт К. Введение в системы баз данных. М.: Вильямс, 8-е изд., 2005.

2. Мартин Грубер Понимание SQL, М., 1993.

3. Тернстрем, Вебер, Хотек Microsoft SQL Server 2008. Разработка баз данных. Учебный курс Microsoft, русская редакция, 2010

4. Новиков, Борис Асенович. Настройка приложений баз данных: учебное пособие для вузов / Б. А. Новиков, Г. Р. Домбровская. - СПб.: БХВ-Петербург, 2012.