Домашние задания

Домашнее задание 1. Установка и использование СУБД

1. Установите систему управления реляционными базами данных.
2. Узнайте, как в вашей СУБД исполнять SQL в интерактивном режиме.
3. Узнайте, как в вашей СУБД исполнять SQL в пакетном режиме.
4. Разберитесь, как в вашей СУБД осуществляется поддержка русского языка.
5. Создайте базу данных и наполните ее в соответствии с примерами из презентации.

Ожидаемая структура проекта

1. Текстовая часть
    1. Описание предметной области с кратким описанием неочевидных сущностей и атрибутов.
    2. Предварительное разбиение на отношения (может отсутствовать).
    3. Для каждого отношения: определение функциональных зависимостей, нормализация до 5НФ, денормализация (при необходимости).
    4. Модель сущность-связь.
    5. Физическая модель (должна соответствовать ERM) с указанием типов для доменов.
2. Часть на SQL
    - ddl.sql – описание таблиц и индексов.
    - insert.sql – добавление тестовых данных.
    - select.sql – запросы на получение данных и представления.
    - update.sql – запросы на изменение данных, хранимые процедуры и триггеры.

В рамках проекта:

- Выберите тему проекта.
    - Тема должна быть уникальной.
    - Тема должна быть достаточно сложной.
    - Нельзя брать темы: обучение в университете; торговля (как товарами, так и билетами); соревнования по программированию.

Домашнее задание 2. Моделирование БД «Университет»

Спроектируйте базу данных «Университет», позволяющую хранить информацию о факультетах, студентах, группах, преподавателях, дисциплинах и оценках.

1. Составьте модель сущность-связь.
2. Преобразуйте модель сущность-связь в физическую модель.
3. Запишите физическую модель на языке SQL. Запись должна включать объявления ограничений.
4. Создайте базу данных по спроектированной модели.
5. Запишите операторы SQL, заполняющие базу тестовыми данными. Достаточно 2–3 записей на таблицу, если они в полной мере демонстрируют особенности БД.

Примечания

1. Не требуется поддержка:
    - нескольких университетов;
    - дисциплин по выбору;
    - дисциплин с необычным распределением по группам (таких как физическая культура и иностранный язык);
    - переводов между группами;
    - исторических данных;
    - нескольких оценок по одной дисциплине.
2. Многосеместровые дисциплины считаются по семестрам, например:
    - Математический анализ (семестр 1);
    - Математический анализ (семестр 2).

Форма для сдачи ДЗ. Проверка проводится в полуавтоматическом режиме, поэтому строго соблюдайте указанные форматы.

В рамках проекта:

- Сделайте предварительную схему для БД проекта на основе моделей:
    - модель сущность-связь;
    - физическая модель;
    - определение схемы на SQL.

Домашнее задание 3. Функциональные зависимости в БД «Университет»

Дано отношение с атрибутами *StudentId*, *StudentName*, *GroupId*, *GroupName*, *GroupFacultyId*, *CourseId*, *CourseName*, *LecturerId*, *LecturerName*, *LecturerFacultyId*, *Mark*, *FacultyId*, *FacultyName*, *FacultyDeanId*.

1. Найдите функциональные зависимости в данном отношении.
2. Найдите все ключи данного отношения.
3. Найдите замыкание множеств атрибутов:
    1. *GroupId*, *CourseId*;
    2. *StudentId*, *CourseId*;
    3. *StudentId*, *LecturerId*;
    4. *StudentId*, *LecturerFacultyDeanId*;
    5. *GroupName*, *LecturerId*.
4. Найдите неприводимое множество функциональных зависимостей для данного отношения.

Примечания

1. Не требуется поддержка:
    - нескольких университетов;
    - дисциплин по выбору;
    - дисциплин с необычным распределением по группам (таких как физическая культура и иностранный язык);
    - переводов между группами;
    - нескольких оценок по одной дисциплине.
2. Многосеместровые дисциплины считаются по семестрам, например: *Математический анализ (семестр 1)*, *Математический анализ (семестр 2)*.

Форма для сдачи ДЗ. Проверка проводится в полуавтоматическом режиме, поэтому строго соблюдайте указанные форматы.

В рамках проекта:

1. Определите набор атрибутов, необходимых для проекта, и определите отношения на них.
2. Найдите функциональные зависимости полученных отношений.
3. Найдите все ключи полученных отношений.
4. Найдите неприводимые множества функциональных зависимостей для полученных отношений.

Домашнее задание 4. Нормализация БД «Университет»

Дано отношение с атрибутами *StudentId*, *StudentName*, *GroupId*, *GroupName*, *GroupFacultyId*, *GroupFacultyName*, *GroupFacultyDeanId*; *CourseId*, *CourseName*, *LecturerId*, *LecturerName*, *LecturerFacultyId*, *LecturerFacultyName*, *LecturerFacultyDeanId*, *Mark*. и функциональными зависимостями:

- StudentId → StudentName, GroupId, GroupName;
- GroupId → GroupName, GroupFacultyId;
- GroupName → GroupId;
- CourseId → CourseName;
- LecturerId → LecturerName, LecturerFacultyId;
- StudentId, CourseId → Mark;
- GroupId, CourseId → LecturerId;
- GroupFacultyId → GroupFacultyName, GroupFacultyDeanId;
- GroupFacultyName → GroupFacultyId;
- LecturerFacultyId → LecturerFacultyName, LecturerFacultyDeanId;
- LecturerFacultyName → LecturerFacultyId.

1. Инкрементально приведите данное отношение в пятую нормальную форму.
2. Постройте соответствующую модель сущность-связь.
3. Постройте соответствующую физическую модель.
4. Реализуйте SQL-скрипты, создающие схему базы данных.
5. Создайте базу данных по спроектированной модели.
6. Заполните базу тестовыми данными.

Форма для сдачи ДЗ

В рамках проекта:

1. Приведите схему базы в пятую нормальную форму.
2. Если итоговая схема не будет в НФ-5, то обоснуйте принятое решение.
3. Запишите определения таблиц на языке SQL.
4. Запишите на языке SQL наполнение таблиц тестовым данными.

Домашнее задание 5. Реляционная алгебра

Структура базы данных «Университет»:

- *Faculties*(*FacultyId*, *FacultyName*, *DeanId*)
- *Groups*(*GroupId*, *GroupName*, *GroupFacultyId*)
- *Students*(*StudentId*, *StudentName*, *GroupId*)
- *Courses*(*CourseId*, *CourseName*)
- *Lecturers*(*LecturerId*, *LecturerName*, *LecturerFacultyId*)
- *Plan*(*GroupId*, *CourseId*, *LecturerId*)
- *Marks*(*StudentId*, *CourseId*, *Mark*)

Составьте выражения реляционной алгебры и соответствующие SQL-запросы, позволяющие получать

1. Информацию о студентах
    1. С заданным идентификатором (*StudentId*, *StudentName*, *GroupId* по *:StudentId*).
    2. С заданным ФИО (*StudentId*, *StudentName*, *GroupId* по *:StudentName*).
2. Полную информацию о студентах
    1. С заданным идентификатором (*StudentId*, *StudentName*, *GroupName* по *:StudentId*).
    2. С заданным ФИО (*StudentId*, *StudentName*, *GroupName* по *:StudentName*).
    3. Из заданной группы (*StudentId*, *StudentName*, *GroupName* по *:GroupName*).
    4. С заданного факультета (*StudentId*, *StudentName*, *GroupName* по *:FacultyName*).
    5. С факультета, заданного деканом (*StudentId*, *StudentName*, *GroupName* по *:LecturerName*).
3. Информацию о студентах с заданной оценкой по дисциплине
    1. С заданным идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseId*).
    2. С заданным названием (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseName*).
    3. Которую у него вёл лектор заданный идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerId*).
    4. Которую у них вёл лектор, заданный ФИО (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerName*).
    5. Которую вёл лектор, заданный идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerId*).
    6. Которую вёл лектор, заданный ФИО (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:LecturerName*).
4. Информацию о студентах не имеющих оценки по дисциплине
    1. Среди всех студентов (*StudentId*, *StudentName*, *GroupId* по *:CourseName*).
    2. *Тут был дубль задачи, пункт оставлен для сохранения нумерации.*
    3. Среди студентов факультета (*StudentId*, *StudentName*, *GroupId* по *:CourseName*, *:FacultyName*).
    4. Среди студентов, у которых есть эта дисциплина (*StudentId*, *StudentName*, *GroupId* по *:CourseName*).
5. Для каждого студента ФИО и названия дисциплин
    1. Которые у него есть по плану (*StudentName*, *CourseName*).
    2. Есть, но у него нет оценки (*StudentName*, *CourseName*).
    3. Есть, но у него не 4 или 5 (*StudentName*, *CourseName*).
    4. Вёл преподаватель (*StudentName*, *CourseName* по *:LecturerName*).
    5. Вёл преподаватель с :FacultyName (*StudentName*, *CourseName* по *:FacultyName*).
    6. Вёл преподаватель другого факультета (*StudentName*, *CourseName*).
    7. Вёл декан (*StudentName*, *CourseName*).
6. Идентификаторы студентов по преподавателю
    1. Имеющих хотя бы одну оценку у преподавателя (*StudentId* по *:LecturerName*).
    2. Не имеющих ни одной оценки у преподавателя (*StudentId* по *:LecturerName*).
    3. Имеющих оценки по всем дисциплинам преподавателя (*StudentId* по *:LecturerName*).
    4. Имеющих оценки по всем дисциплинам преподавателя, которые он вёл у этого студента (*StudentId* по *:LecturerName*).
7. Группы и дисциплины, такие что все студенты группы имеют оценку по этой дисциплине
    1. Идентификаторы (*GroupId*, *CourseId*).
    2. Названия (*GroupName*, *CourseName*).

Составьте SQL-запросы, позволяющие получать

8. Суммарный балл

1. Одного студента (*SumMark* по *:StudentId*).
2. Каждого студента (*StudentName*, *SumMark*).
3. Каждой группы (*GroupName*, *SumMark*).

9. Средний балл
    1. Одного студента (*AvgMark* по *:StudentId*).
    2. Каждого студента (*StudentName*, *AvgMark*).
    3. Каждой группы (*GroupName*, *AvgMark*).
    4. Средний балл средних баллов студентов каждой группы (*GroupName*, *AvgAvgMark*).
10. Для каждого студента: число дисциплин, которые у него были, число сданных дисциплин и число несданных дисциплин (*StudentId*, *Total*, *Passed*, *Failed*).

Тестовый полигон

Технические особенности проверки.

- Сдача — в PCMS. Если у вас нет аккаунта в PCMS, либо доступа к ДЗ, обратитесь к Николаю Викторовичу Ведерникову.
- Проверяться и оцениваться будет **последняя** посланная версия.
- Проверка разделена на 4 фазы:
    1. пустые таблицы (синтаксис и набор столбцов);
    2. таблицы с не более чем одной записью;
    3. таблицы с простыми данными;
    4. таблицы со сложными данными.
- В случае проблем с синтаксисом или набором столбцов вы будете получать *Presentation Error*.
- Реляционная алгебра проверяется одним тестом на фазу, движком из тестового полигона.
- SQL проверяется тремя тестами на фазу — с разными СУБД. Первая СУБД — SQLite, как на тестовом полигоне.
- Известные спецэффекты:
    - SQLite поддерживает только left join. right и outer join делаются через него.
    - Все вложенные запросы надо именовать, даже если вы не будете использовать это имя:

      ```
      select ... from ... (select ... ) SubQueryName ...
      ```

    - Используйте данные из минимально возможного набора таблиц.

Домашнее задание 6. Реляционное исчисление

Составьте запросы в терминах языков Datalog и SQL для базы данных «Университет», позволяющие получать

1. Информацию о студентах
    1. С заданным ФИО (*StudentId*, *StudentName*, *GroupId* по *:StudentName*).
    2. Учащихся в заданной группе (*StudentId*, *StudentName*, *GroupId* по *:GroupName*).
    3. Учащихся на заданном факультете (*StudentId*, *StudentName*, *GroupId* по *:FacultyName*).
    4. С заданной оценкой по дисциплине, заданной идентификатором (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseId*).
    5. С заданной оценкой по дисциплине, заданной названием (*StudentId*, *StudentName*, *GroupId* по *:Mark*, *:CourseName*).
2. Полную информацию о студентах
    1. Для всех студентов (*StudentId*, *StudentName*, *GroupName*).
    2. Учащихся на заданном факультете (*StudentId*, *StudentName*, *GroupName* по *:FacultyName*).
    3. С факультета, заданного деканом (*StudentId*, *StudentName*, *GroupName* по *:LecturerName*).
    4. Студентов, не имеющих оценки по дисциплине, заданной идентификатором (*StudentId*, *StudentName*, *GroupName* по *:CourseId*).
    5. Студентов, не имеющих оценки по дисциплине, заданной названием (*StudentId*, *StudentName*, *GroupName* по *:CourseName*).
    6. Студентов, не имеющих оценки по дисциплине, у которых есть эта дисциплина (*StudentId*, *StudentName*, *GroupName* по *:CourseId*).
    7. Студентов, не имеющих оценки по дисциплине, у которых есть эта дисциплина (*StudentId*, *StudentName*, *GroupName* по *:CourseName*).
3. Студенты и дисциплины, такие что у студента была дисциплина (по плану или есть оценка)
    1. Идентификаторы (*StudentId*, *CourseId*).
    2. Имя и название (*StudentName*, *CourseName*).
    3. Имя и название, преподаватель того же факультета (*StudentName*, *CourseName*).
    4. Имя и название, преподаватель другого факультета (*StudentName*, *CourseName*).
4. Студенты и дисциплины, такие что дисциплина есть в его плане, и у студента долг по этой дисциплине
    1. Долгом считается отсутствие оценки (*StudentName*, *CourseName*).
    2. Долгом считается оценка не выше 2 (*StudentName*, *CourseName*).
    3. Долгом считается отсутствие оценки или оценка не выше 2 (*StudentName*, *CourseName*).
5. Идентификаторы студентов по преподавателю
    1. Имеющих хотя бы одну оценку у преподавателя (*StudentId* по *:LecturerName*).
    2. Не имеющих ни одной оценки у преподавателя (*StudentId* по *:LecturerName*).
    3. Имеющих оценки по всем дисциплинам преподавателя (*StudentId* по *:LecturerName*).

    4. Имеющих оценки по всем дисциплинам преподавателя, которые он вёл у этого студента (*StudentId* по *:LecturerName*).
6. Группы и дисциплины, такие что все студенты группы имеют оценку по предмету
    1. Идентификаторы (*GroupId*, *CourseId*).
    2. Названия (*GroupName*, *CourseName*).

Примечания

1. В Datalog итоговым считается последнее объявленное отношение.
2. Текущая реализация Datalog не поддерживает рекурсивные определения.
3. В SQL-запросах нельзя использовать * join.
4. SQL проверяется *четырьмя* тестами на фазу — с разными СУБД. Первая СУБД — SQLite, как на тестовом полигоне.

В рамках проекта:

1. Определите запросы (в том числе, агрегирующие), необходимые для работы проекта.
2. Реализуйте запросы на языке SQL.

Домашнее задание 7. Изменение данных

Реализуйте указанные запросы, представления, проверки и триггеры на языке SQL.

1. Напишите запросы, удаляющие студентов
    1. Учащихся в группе, заданной идентификатором (*GroupId*).
    2. Учащихся в группе, заданной названием (*GroupName*).
    3. Учащихся на факультете (*FacultyName*).
    4. Без оценок.
    5. Имеющих 3 и более оценки.
    6. Имеющих 3 и менее оценки.
2. Напишите запросы, удаляющие должников (здесь и далее, долг определяется по отсутствию оценки)
    1. Студентов, с долгами.
    2. Студентов, имеющих 2 и более долга.
    3. Студентов, имеющих 2 и более долга, учащихся на факультете (*FacultyName*).
    4. Студентов, имеющих не более 3 долгов.
3. Напишите запросы, обновляющие данные студентов
    1. Изменение имени студента (*StudentId*, *StudentName*).
    2. Перевод студента из группы в группу по индентификаторам (*StudentId*, *GroupId*, *FromGroupId*).
    3. Перевод всех студентов из группы в группу по идентификаторам (*GroupId*, *FromGroupId*).
    4. Изменение имени всех студентов группы (*GroupName*, *StudentName*).
    5. Перевод всех студентов из группы в группу по названиям (*GroupName*, *FromGroupName*).
    6. Перевод всех студентов из группы в группу, только если целевая группа существует (*GroupName*, *FromGroupName*).
4. Напишите запросы, подсчитывающие статистику по оценкам
    1. Число оценок студента (столбец *Students.Marks*) (*StudentId*).
    2. Число оценок каждого студента (столбец *Students.Marks*).
    3. Число оценок каждого студента факультета (столбец *Students.Marks*) (*FacultyName*).
    4. Пересчет числа оценок каждого студента, с учётом новых оценок из таблицы *NewMarks*, структура которой такая же как у таблицы *Marks* (столбец *Students.Marks*).
5. Напишите запросы, подсчитывающие статистику по студентам
    1. Число сданных дисциплин каждого студента (столбец *Students.Marks*).
    2. Число долгов студента (столбец *Students.Debts*) (*StudentId*).
    3. Число долгов каждого студента (столбец *Students.Debts*).
    4. Число долгов каждого студента группы (столбец *Students.Debts*) (*GroupName*).
    5. Число долгов каждого студента у деканов (столбец *Students.Debts*) (*GroupName*).
    6. Число оценок и долгов каждого студента (столбцы *Students.Marks*, *Students.Debts*).
6. Напишите запросы, обновляющие оценки, с учетом данных из таблицы *NewMarks*, имеющей такую же структуру, как таблица *Marks*
    1. Проставляющий новую оценку только если ранее оценки не было.
    2. Проставляющий новую оценку только если ранее оценка была.
    3. Проставляющий максимум из старой и новой оценки только если ранее оценка была.
    4. Проставляющий максимум из старой и новой оценки (если ранее оценки не было, то новую оценку).
7. Работа с представлениями
    1. Создайте представление *StudentMarks* в котором для каждого студента указано число оценок (*StudentId*, *Marks*).
    2. Создайте представление *AllMarks* в котором для каждого студента указано число оценок, включая оценки из таблицы *NewMarks* (*StudentId*, *Marks*).
    3. Создайте представление *Debts* в котором для каждого студента, имеющего долги указано их число (*StudentId*, *Debts*).
    4. Создайте представление *StudentDebts* в котором для каждого студента указано число долгов (*StudentId*, *Debts*).

8. Целостность данных.

    Обратите внимание, что задания из этого раздела надо посылать в PCMS, но они будут проверяться только вручную после окончания сдачи. То есть в PCMS вы получите *+* за любое решение.

    В комментарии перед каждым запросом укажите название и версию использованной СУБД.

    1. Добавьте проверку того, что у студентов есть оценки только по дисциплинам из их плана (*NoExtraMarks*).
    2. Добавьте проверку того, что все студенты каждой группы имеют оценку по одному и тому же набору дисциплин

(*SameMarks*).
3. Создайте триггер *PreserveMarks*, не позволяющий уменьшить оценку студента по дисциплине. При попытке такого изменения оценка изменяться не должна.

Примечания

1. Некоторые базы данных не понимают, что * join ... using (Column) должен оставлять один экземпляр Column и не требовать указывать для него таблицу.

В рамках проекта:

1. Определите модифицирующие запросы, необходимые для работоспособности проекта.
2. Запишите эти запросы на языке SQL.
3. Определите представления, необходимые для работоспособности проекта.
4. Запишите эти представления на языке SQL.

Домашнее задание 8. Индексирование

1. Определите, какие индексы требуется добавить к таблицам базы данных «Университет» на основе запросов из ДЗ-5, 6 и 7.
2. Пусть частым запросом является определение среднего балла студентов факультета по дисциплине. Как будет выглядеть запрос и какие индексы могут помочь при его исполнении?
3. Придумайте три запроса, требующих новых индексов и запишите их. Если в результате, некоторые из старых индексов станут бесполезными, удалите их.

При выполнении задания считайте, что ФЗ соответствуют полученным в ДЗ-3 и 4.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите индексы и их типы, необходимые для эффективного исполнения запросов.
2. Запишите определения индексов на языке SQL.

Домашнее задание 9. Хранимые процедуры

В базе данных Airline информация о рейсах самолётов задана в виде таблиц

```
Flights(
    FlightId integer,
    FlightTime timestamp,
    PlaneId integer,
    -- Дополнительные столбцы, при необходимости
)
Seats(
    PlaneId integer,
    SeatNo varchar(4), -- 123A
    -- Дополнительные столбцы, при необходимости
)
```

Реализуйте запросы к базе данных Airline с применением представлений, хранимых процедур и функций. При необходимости, вы можете создать дополнительные таблицы, представления и хранимые процедуры.

Возможность бронирования должна автоматически отключаться за трое суток до начала рейса. Продажа мест должна автоматически отключаться за три часа до начала рейса. Также должна быть предусмотрена возможность отключения бронирования и продаж вручную.

1. Администрирование.
    1. RegisterUser(UserId, Pass) — зарегистрировать нового пользователя. Возвращает *истину*, если удалось и *ложь* — в противном случае.
    2. ManageFlight(UserId, Pass, FlightId, SellAllowed, ReservationAllowed) — изменить настройки рейса. Примечание: автоматические настройки имеют более высокий приоритет.
2. Покупка и бронирование.
    1. FreeSeats(FlightId) — список мест рейса, доступных для продажи и для бронирования.
    2. Reserve(UserId, Pass, FlightId, SeatNo) — бронирует место на сутки начиная с момента бронирования. Возвращает *истину*, если удалось и *ложь* — в противном случае.
    3. ExtendReservation(UserId, Pass, FlightId, SeatNo) — продлевает бронь места на сутки начиная с момента продления. Возвращает *истину*, если удалось и *ложь* — в противном случае.
    4. BuyFree(FlightId, SeatNo) — покупает свободное место. Возвращает *истину*, если удалось и *ложь* — в противном случае.
    5. BuyReserved(UserId, Pass, FlightId, SeatNo) — покупает забронированное место (пользователи должны совпадать). Возвращает *истину*, если удалось и *ложь* — в противном случае.
3. Статистика.
    1. FlightsStatistics(UserId, Pass) — статистика по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест.
    2. FlightStat(UserId, Pass, FlightId) — статистика по рейсу: возможность бронирования и покупки, число свободных, забронированных и проданных мест.
4. CompressSeats(UserId, Pass, FlightId) — оптимизирует занятость мест в самолете. В результате оптимизации, в начале самолета должны быть купленные места, затем — забронированные, а в конце — свободные. Примечание: клиенты, которые уже

выкупили билеты, также должны быть пересажены.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите хранимые процедуры и функции, необходимые для работы проекта.
2. Реализуйте хранимые процедуры (функции) на языке SQL.

Домашнее задание 10. Транзакции

Спланируйте транзакции и выберите их уровни изоляции для базы данных Airline.

1. Для каждой хранимой процедуры из предыдущего домашнего задания выберите минимальный допустимый уровень изоляции транзакций (с обоснованием).
2. Реализуйте сценарий работы:
    1. Запрос списка свободных мест.
    2. Отображение списка свободных мест пользователю.
    3. Бронирование или покупка места, выбранного пользователем.

Форма для сдачи ДЗ

В рамках проекта:

1. Определите минимальный уровень изоляции транзакций, необходимый для каждого запроса и хранимой процедуры.