1) Здравствуйте, уважаемая комиссия, и все присутствующие в аудитории. Меня зовут Савченко Егор, я являюсь студентом Факультета Информационных технологий.
   Тема моей работы - “Разработка учебной системы оценки сложности планов SQL-запросов”

2) Язык SQL является декларативным. При составлении запроса пользователь лишь дает некоторую характеристику желаемому множеству данных, но не описывает конкретные действия по его получению.
   СУБД внутренними механизмами преобразует декларативный запрос в функциональный набор действий (или план запроса). В результате исполнения которого получается требуемое множество данных.
   На формирование конкретного плана запроса влияют несколько внутренних факторов, таких как типы данных, статистика использования таблиц и индексы.

3) Еще раз заострю внимание на определениях. Запрос - лишь описание множества. Тогда как план - конкретная последовательность действий по получению этого множества.

4) На слайде представлен запрос и два возможных плана его исполнения. Стоит обратить внимание на на очередность операций фильтрации данных (обозначено точками).
   Несмотря на то, что оба плана приведут к одному множеству данных, стоимость исполнения этих двух планов будет отличаться.

5) СУБД предоставляют возможность просмотра плана запроса, однако на этом все. На построение плана нельзя никак повлиять. И обычно СУБД старается построить наиболее оптимальный план.

6) Целью моей работы является реализация учебного инструмента по построению, выполнению и оцениванию планов запросов, который поддерживает основные операции по выборке и агрегации данных.
   А также инструмент должен следовать основным концепциям обработки реляционных данных, которые используются в промышленных движках СУБД.

   Инструмент изначально задумывался как учебный, поэтому к нему не выдвигаются требования о высокой производительности и возможности обработки больших объемов данных.

7) На Факультете информационных технологий на первом году обучения есть дисциплина «Декларативное программирование». Одна из ее задач - это познакомить студента с реляционной моделью и основами SQL.
   В рамках этого курса есть блок задач, который студент выполняет на специально подготовленном инструменте — программе на языке Haskell.
   Эта программа позволяет студенту выступить в роли планировщика и потренироваться в построении планов запросов.

В принципе с программой все хорошо, однако есть ряд недостатков - отсутствие некоторых значимых операций (не все виды Join-ов реализованы). И есть сложность расширения в основном связанные с целевой платформой.

Разрабатываемое в рамках моей работы программное решение планируется в качестве более функциональной замены существующему.

8) Для осуществления поставленной цели были поставлены следующие задачи:
   Исследование существующих методов оценивания сложности
   Исследование принципов обработки и организации реляционных данных промышленными реализациями стандарта SQL
   Анализ требований (определение необходимого множество типов данных и операций)
   Разработка обоснованного архитектурного решения
   Определение необходимого набора инструментов реализации
   Реализация учебной системы

9) Выдвигаемые к системе требования в основном исходят из предметной области и способствуют наиболее приближенному к реальным системам баз данных поведению.
   Ключевые моменты:
   Реализация клиентской часть не должна быть привязана к конкретному ЯП, а система в целом - к конкретной реализации базы данных.
   Алгоритм подсчета сложности должен быть детерминированным и не зависеть от окружения.
   Необходимо поддерживать основные операции по обработке данных, индексы, ленивость и кэширование, а также должны присутствовать подзапросы.

10) На слайде - пример использования системы. Пользователь сформировал два различных плана, которые приведут к одному множеству данных, а система их оценила. Здесь для наглядности указано только цифровое обозначение, чуть дальше будет показан полный вывод программы.

11) Концептуально система выглядит следующим образом: Есть клиент, реализованный на каком-либо языке программирования. Пользователь составляет план запроса с помощью некоторого интерфейса.
   При запуске клиента запускается ядро обработки реализованное на java (оно читает необходимые структуры из базы данных, такие как заголовки таблиц, колонок и их типы, размеры таблиц. Данные из таблиц на этом этапе не считываются). Клиент посредством программного канала вызывает ядро, строятся необходимые структуры, после чего происходит обработка плана и клиенту возвращается полученное реляционное множество и характеризующих сложность объект.

12) В качестве основного инструмента реализации была выбрана Java и соответствующий ей набор инструментов (JDBC и Jackson). JDBC позволяет довольно просто использовать другую реализацию бд, путем использования соответствующего драйвера. Также был реализован клиент на haskell.

SQL обладает слабой смешанной типизацией (может показаться, что было бы лучше выбрать слабо типизированный язык), но в SQL присутствуют довольно нетривиальные преобразования типов, поэтому лучше не полагаться на реализации взаимодействия типов конкретного языка, а явно специфицировать взаимодействие типов.

13) План запроса представляет собой упорядоченный набор операций по обработке данных.
    На самом деле план запроса это дерево. В листьях всегда находится операция From - она формирует изначальный поток данных.
    Концептуально каждая операция переводит данные из одного состояние в другое.

14) На слайде представлен перечень реализованных операций над данными. Операция - это обертка над Java Stream Api. При составлении плана запроса операции можно чередовать в почти любой последовательности.

15) Выражение - одна из ключевых частей реализации. В SQL две операции операции содержат выражения. (Join и Where).
    Выражения задаются пользователем в виде дерева и могут быть довольно сложными: например содержать подзапросы.

16) Выражения имеют следующую иерархию классов. Здесь видно четкое разделение на значения и операции.. В таблицах данных могут встречаться только значения, а при использовании операции join также и различные операции.

17) Ленивость - наиболее важная часть реализации, так как все настоящие движки баз данных обрабатывают запросы лениво. В SQL есть операции как поддерживающие ленивость, так и операции которые требуют полной обработки потока данных. На слайде можно обратить внимание, насколько значительно падает сложность плана запроса при добавлении ограничения в 10 записей.

18) Тут перед лимитом была добавлена операция упорядочивания, которая не поддерживает ленивость. И результат сразу изменился

19) Подсчет сложности реализован в виде прямого подсчета элементарных операций обработки данных. На слайде представлен процесс подсчета сложности для выражения, для операций алгоритм схожий: (например GroupBy учитывает количество записей, и количество столбцов по которым происходит группировка)

20) Пример пользовательского плана запроса и результат подсчета сложности этого плана.

21) В результате была реализована учебная система, позволяющая сформировать и выполнить план запроса, получить результат в виде таблицы данных, а также характеризующего сложность исполнения этого плана объект.
Система поддерживает основные операции по выборке и агрегации данных, подзапросы, кеширование, а также обрабатывает данные лениво.