1. Добрый день, уважаемая комиссия.
2. Была поставлена цель разработать метод построения дерева синтаксического анализа языка SQL на основе графических примитивов.
   Для решения поставленной цели были выделены задачи, представленные на слайде.
3. Для анализа больших объемов данных бизнес аналитики создают отчеты, на основе которых принимаются важные стратегические решения. На сегодняшний день для построения отчета аналитику необходимо тесное взаимодействие с инженером данных и его сопровождение на всех этапах построения отчета, в том числе и написании etl-процессов, (так как в компетенцию аналитика зачастую не входит написание кода на языках отличных от SQL).
4. Для сокращения времени разработки необходимо предоставить аналитику графический инструмент, который избавит его от написания кода.
   Существуют готовые инструменты для решения описанной проблемы такие как sas dis, Informatica Power Center… На левой стороне слайда представлен запрос в виде строки на правой стороне визуализирован тот же запрос в виде графических примитивов. Все проанализированные инструменты имеют возможность изменять графических представленный запрос в виде строки, что накладывает на них обязательное условие лексического анализа.
5. Приведены типы запросов. Очень простой запрос включает в себя только выборку из таблицы. Простой запрос учитывает фильтрацию данных. Нормальный запрос содержит объединение таблиц и увеличенный объем фильтрации. Сложный запрос содержит подзапрос, группировку, фильтрацию группировки и сортировку. Данные типы запросов используются далее в исследовании построения АСТ.
6. Существующие ETL-инструменты принимают на вход строку, после чего преобразуют ее в поток токенов, далее с использованием существующей грамматики парсер строит дерево синтаксического анализа. Далее АСТ проходит анализ и создание минимизированного плана запроса, который далее отправляется на выполнение. Поскольку ETL-инструменты основаны на графических элементах, а не на строке, можно не проводить лексический анализ, а изначально методу передавать поток токенов.
7. Изначально пользователь должен расположить графических элементы на холсте, далее данные элементы преобразуются в поток токенов и передаются методу построения АСТ.
8. Программное обеспечение содержит модуль пользовательского интерфейса, который использует модуль преобразования граф. Элементов. К набору токенов для получения потока токенов, который передает модулю построения АСТ. Модель построения АСТ использует для построения модуль ANTLR. ANTLR - это инструмент для генерации синтаксических анализаторов. Данный инструмент не позволяет возможности использовать парсер без лексера, поэтому была произведена модификация инструмента для достижения поставленной цели.
9. Была использована спецификация СУБД 602SQL, которая базируется на спецификации SQL2. Разработанная грамматика описана в форме, близкоф к форме БНФ. На слайде показано описание одного и того же правила.

10. Было разработано 60 правил для грамматики, а также 81 тип токенов. Все разработанные правила рассчитаны на извлечение данных. Не разработана грамматика для изменения данных, такие правила как обновления, удаление, вставка и подобные.
11. По разработанной грамматике было построено следующее дерево синтаксического анализа для простого примера, содержащего только выборку из таблицы показано на данном слайде. Дерево содержит атомы - это листья, содержащие основную информацию и синтаксическая категория - остальные узлы.
12. На данном слайде показана часть того же запроса, только добавлено where-условие. Разработанная грамматика учитывает приоритет операций. В данном запрос имеется операции AND и OR. Операция AND имеет более высокий приоритет, поэтому в дереве она расположена ниже, нежели чем OR.
13. Было произведено исследование построение АСТ от сложности запроса. Результаты показали, что с ростом сложности запроса росло время построения АСТ, Но самое главное, что время построения АСТ на основе токенов меньше, чем время построение АСТ на основе строки.
14. В заключение поставленная цель была достигнута, задачи были решены.
15. Направление дальнейшего развития можно рассмотреть расширение грамматики, добавление запросов на изменение данных и реализация оставшихся этапов построения и выполнения запроса.

Перейдем к демонстрации программного обеспечения.

Плохое дерево: select name from order by name
Хорошее дерево: select name from users

Дерево успешно построилось - демонстрация завершена.