УДК 004 EDN ITWNYZ

# Разработка обучающей системы «Основы SQL» с элементами лексического и синтаксического анализа запросов

**Д.Ю. Гаськов, А.С. Дорофеев***

Иркутский национальный исследовательский технический университет, ул. Лермонтова, 83, Иркутск, 664074, Россия

*E-mail: dorbaik@ex.istu.edu

**Аннотация.** Статья посвящена краткому описанию разработанной обучающей системы с элементами лексического и синтаксического анализа входного текста, представляющего собой SQL-запрос. Описана общая функциональность системы, схема данных, дано определение грамматики, некоторые правила грамматики. Представлены результаты работы приложения. Создание подсистем, использующих проверку правильности ввода входной цепочки на языке SQL с помощью синтаксического анализа, может использоваться в обучающих системах для получения навыка разработки SQL-запросов к базам данных, лучшего понимания структуры и синтаксиса языка.

**Ключевые слова:** лексический анализ, синтаксический анализатор, формальная грамматика, JavaScript.

# Development of a training system "SQL basics" with elements of lexical and syntactic analysis of queries

**D.Y. Gaskov, A.S. Dorofeev***

Irkutsk National Research Technical University, 83, Lermonov st., Irkutsk, 664074, Russia

*E-mail: dorbaik@ex.istu.edu

**Abstract.** The article is devoted to a brief description of the learning system using lexical and syntactic analysis of the input text, which is a SQL query. The general functionality of the system is described, the data scheme is described, the grammar is defined, some grammar rules are given. Suggestions for the results of the application. The creation of subsystems that use SQL input string validation using parsing can be implemented in educational methods to gain skills in developing SQL queries to the database, the highest understanding of the structure and syntactic language.

**Keywords:** scanner, parser, formal grammar, JavaScript.

## 1. Введение

Structured Query Language (SQL) – это язык структурированных запросов, который используется для работы с базами данных; применяется почти во всех сферах деятельности, где требуется обработка запросов пользователей. Это единственное средство «программирования», стандартизированное для всех стран. Разработчики, работающие с СУБД, создают свою продукцию, применяя SQL или SQL-интерфейс. В настоящее время с SQL сталкивается большинство программистов, осуществляющих разработку веб-приложений или оболочек сайта [1].

Таким образом, знание синтаксиса языка SQL, умение формировать запросы к базам данных является необходимым условием любого IT-специалиста.

## 2. Цель исследования

В настоящей работе предлагается создание обучающей системы с элементами лексического и синтаксического анализа введенной входной цепочки символов на языке SQL [2].

По своему изначальному предназначению лексический анализатор используется для разбиения входной цепочки на лексемы (неделимые единица языка). Лексемами могут служить операторы, константы, идентификаторы. Лексический анализатор (сканер) выполняет проверку входной цепочки символов на правильность написания и соответствие заданному регулярному выражению.

Цепочка лексем, являющаяся результатом работы лексического анализатора, поступает в синтаксический анализатор (парсер), который выполняет проверку на правильность построения выражений по заранее описанным правилам языка.

Сканер и парсер служат основой построения компилятора и интерпретатора различных языков.

## 3. Общая функциональность системы и база данных

Описание общей функциональности системы реализовано на языке моделирования UML и представлено на рисунке 1.


**Рисунок 1.** Общая функциональность системы.

Из рисунка видны роли пользователей системы и их основные функции. Следует отметить, что здесь в качестве роли также представлена Система, которая выполняет проверку входной цепочки на соответствие заданной грамматике языка.

Физическая модель данных изображена на рисунке 2 [2].

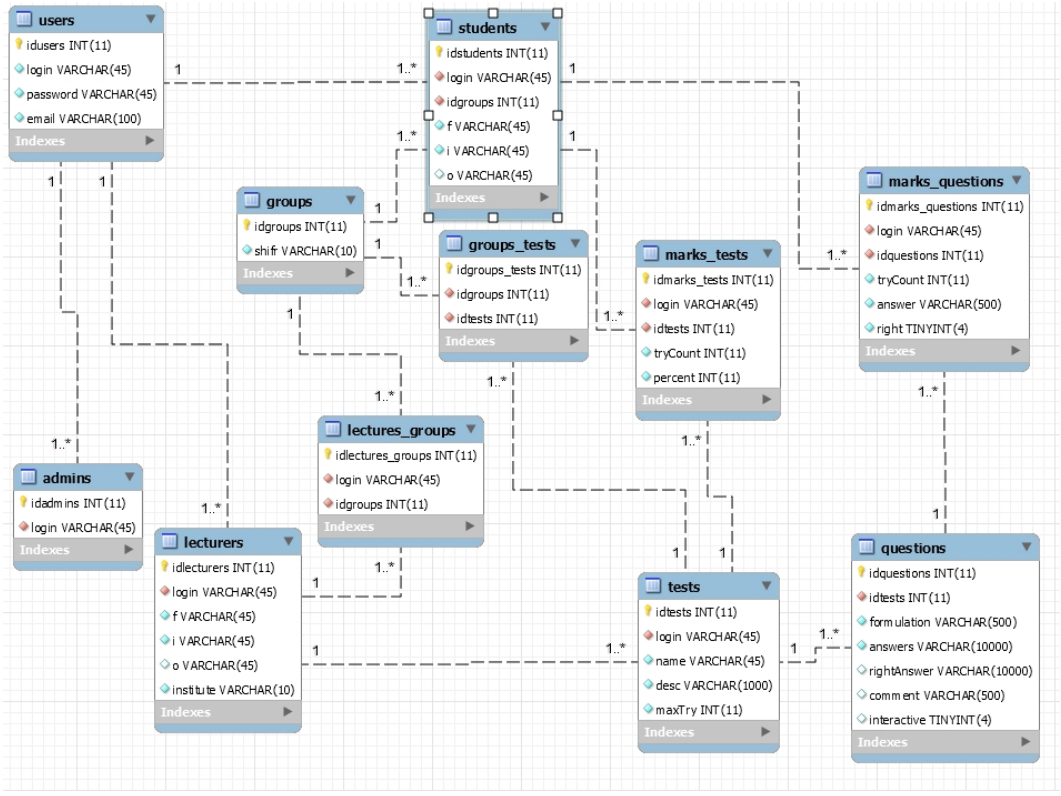


**Рисунок 2.** Физическая модель данных.

## 3. Грамматика языка. Реализация анализатора

Язык – это заданный набор символов и правил, устанавливающих способы комбинации этих символов для записи правильных цепочек. Основой любого языка является алфавит – множество допустимых символов V.

Любой язык задается одним из способов: либо перечислением всех допустимых цепочек языка (что маловероятно), указанием способа порождения цепочек языка – есть заданием грамматики языка (описание набора правил, с помощью которых строятся цепочки языка), определением метода распознавания цепочек языка (предусматривает построение некоторого логического устройства (распознавателя) – автомата, на вход которого поступает цепочка, и автомат на выходе делает вывод о принадлежности или не принадлежности этой цепочки языку).

В данной работе используется второй способ для реализации синтаксического анализатора – путем задания грамматики.

Основой любого языка являются лексика, синтаксис и семантика языка. Лексика – совокупность слов (лексем/неделимых единиц) языка; синтаксис – правила, определяющие допустимые конструкции языка; семантика – содержание языка, то есть значение его предложений [3, 4].

В работе не затрагивается семантический блок, так как цель создания реализуемого анализатора – проверка на правильность ввода входной цепочки

Лексический блок реализуется с помощью регулярных выражений. Одну лексему можно назвать терминалом (неделимая единица алфавита).

Синтаксический блок реализуется за счет описания грамматики языка.

На рисунке 3 представлена структура построения и принцип работы синтаксического анализатора в работе.

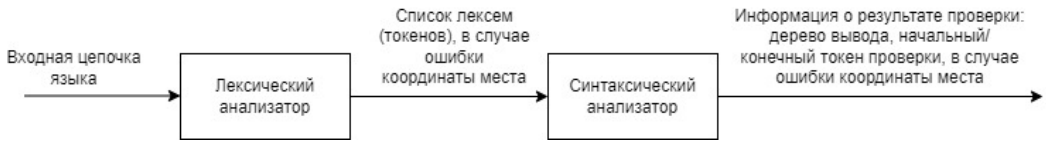


**Рисунок 3.** Структура синтаксического анализатора.

Любая контекстно-зависимая грамматика представляет собой совокупность множества терминальных и нетерминальных символов, начального нетерминального символа, множества правил грамматики [5].

Описание грамматики, используемой в синтаксическом анализаторе:

- главный нетерминал: singleStatement;
- множество терминальных символов: ADD, ADMIN, ALL, ALTER, ANALYZE, AND, EQ, NEQ, LT, LTE, GT, GTE, FROM, UPDATE, DELETE, SELECT, TABLE, IDENTIFIER, EXPONENT, DIGIT, LETTER и так далее;
- множество нетерминальных символов: standaloneExpression, standaloneRoutineBody, statement, query, with, tableElement, columnDefinition, likeClause, properties, property, sqlParameterDeclaration, и так далее.

С помощью конструкции, реализованной в виде регулярных выражений (рисунок 4), у правил грамматики была устранена зависимость от регистра символов входной цепочки (fragment A: [aA]; fragment B: [bB]; … fragment Z: [zZ];).

```
ADD: A D D;
ADMIN: A D M I N;
ALL: A L L;
ALTER: A L T E R;
ANALYZE: A N A L Y Z E;
AND: A N D;
ANY: A N Y;
ARRAY: A R R A Y;
```

**Рисунок 4.** Пример использования фрагментов.

На рисунках 5-7представлены некоторые из правил грамматики [2].

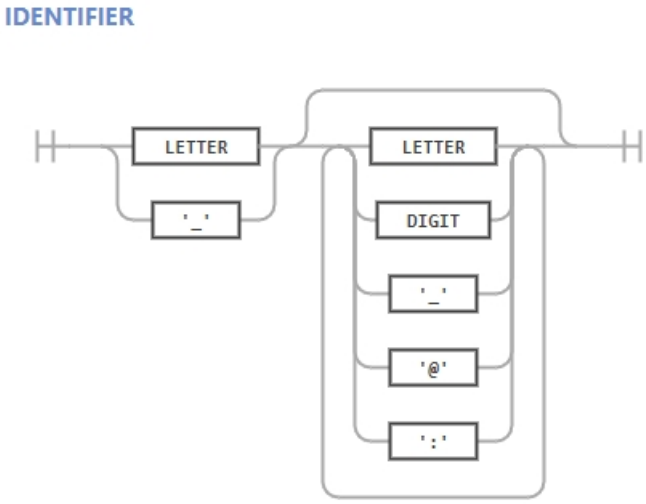


**Рисунок 5.** Диаграмма правила «Идентификатор».

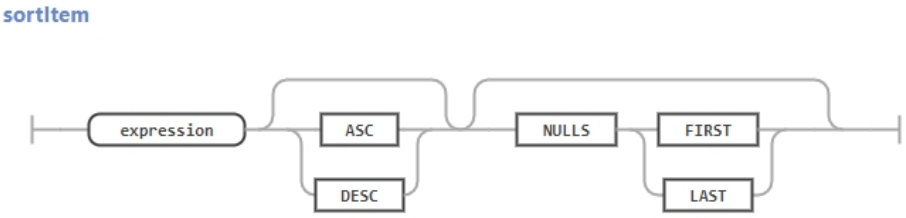


**Рисунок 6.** Диаграмма правила «Блок сортировки».

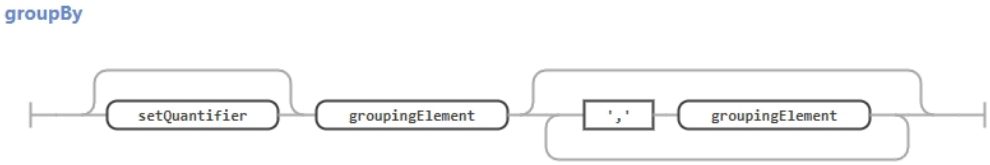


**Рисунок 7.** Диаграмма правила «Блок группировки».

## 4. Конвертация в модули JavaScript

Полученная грамматика с помощью приложения ANTLR [7] при заданных параметрах конвертируется в следующие модули JavaScript:

- лексический анализатор;
- синтаксический анализатор;
- слушатель грамматики, используемый как вспомогательный модуль в синтаксическом анализаторе;

Анализатор в работе встраивается в контроллер antlrController. Функция check_input_string имеет следующий алгоритм (рисунок 8).

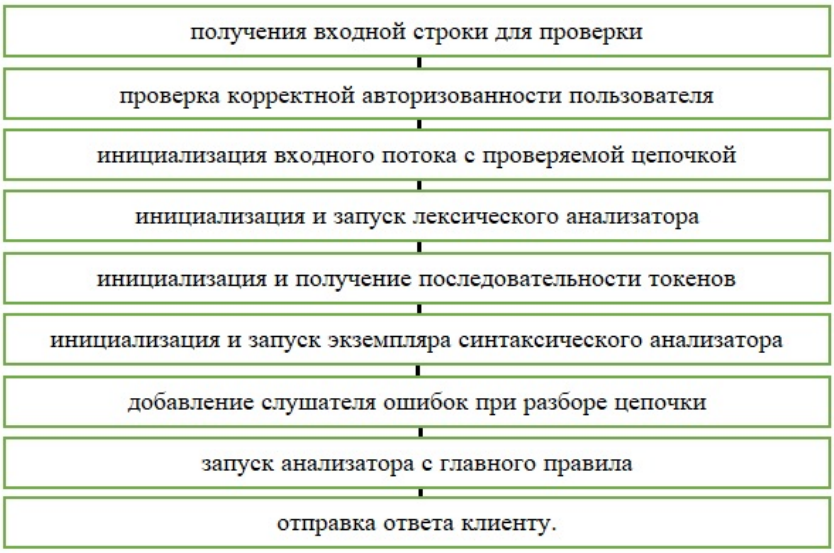


**Рисунок 8.** Алгоритм функции check_input_string».

Если запуск анализатора с главного правила успешен, то строится корректный ответ по результатам проверки; при неверной цепочке – уведомление пользователя информацией о месте найденной ошибки.

На рисунке 9 представлен текст инициализации анализатора [6], рисунок 10 содержит код проверки входной строки и отправки ответа клиенту.

```
let input_stream = new antlr.InputStream(data.input);
let lexer = new Lexer(input_stream);
let tokens = new antlr.CommonTokenStream(lexer);

let parser = new Parser(tokens);
let result_pars;
parser.buildParseTrees = true;
parser.removeErrorListeners();
parser.addErrorListener({
    syntaxError: (recognizer, offendingSymbol, line, column, msg, err) => {
        result_pars = `Ошибка: line ${line}, col ${column}: ${msg.split("expecting")[0]}`;
        //console.error(`line ${line}, col ${column}: ${msg}`);
    }
});
```

**Рисунок 9.** Инициализация анализатора.

```
try {
    let result = parser.singleStatement();
    if (!result_pars) {
        result_pars = "Синтаксический анализ проведен без ошибок\nНачальный токен: "+ result.start.text + "\nКонечный токен: "+ result
    }
    return response.status(200).send(result_pars);
} catch (error) {
    return response.status(800).send("Ошибка при проверке входной строки, повторите попытку позже!");
}
```

**Рисунок 10.** Проверка входной строки и отправка ответа клиенту.

## 5. Примеры работы приложения

На рис. 11-12 представлены примеры окон интерфейса с правильной и ошибочной строкой запроса.

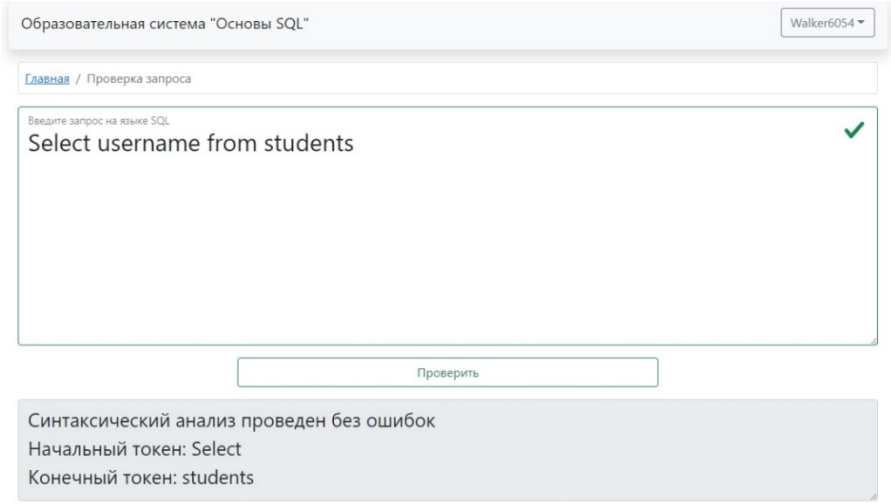


**Рисунок 11.** Успешная проверка входной цепочки.

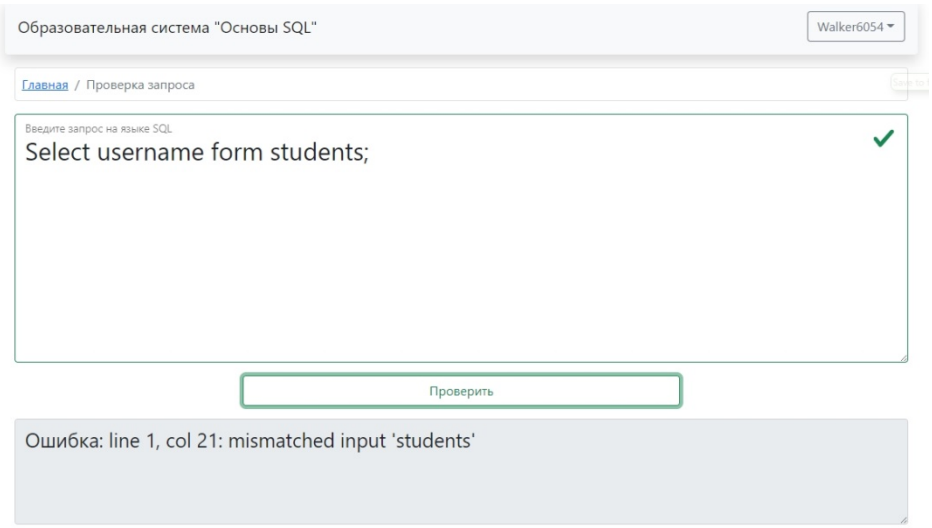


**Рисунок 12.** Задание неверной входной цепочки.

Из рисунков видно, что текст запроса не должен заканчиваться символом «;».

## 6. Выводы

SQL является простым и лёгким в изучении языком, который применяется многими разработчиками для обеспечения функциональности приложений, тестировщиками программного обеспечения, аналитиками для анализа и преобразования данных, администраторами для поддержания работоспособности среды. Язык достаточно универсален и структурирован за счет устоявшихся стандартов. Взаимодействие с базами данных посредством SQL производится достаточно быстро даже при больших объемах данных (Big Data) [8].

Создание подсистем, использующих проверку правильности ввода входной цепочки на языке SQL с помощью синтаксического анализа, может использоваться в обучающих системах для получения навыка разработки SQL-запросов к базам данных, лучшего понимания структуры и синтаксиса языка.

## Список литературы

1. Язык программирования SQL в 2022 году: стоит ли его изучать, перспективы на будущее [Электронный ресурс] // Айтистанция [сайт]. URL: https://itstan.ru/programmirovanie/yazyk-programmirovaniya-sql-stoit-li-ego-izuchat.html (дата обращения: 10.02.23).
2. Гаськов, Д.Ю. Разработка обучающей системы «Основы SQL». Выпускная квалификационная работа [Электронный ресурс] / Д.Ю. Гаськов // Коллекция Руконт [сайт]. – URL: https://lib.rucont.ru/efd/793068/info (дата обращения: 10.02.23).

3. Сосинская, С.С. Трансляторы и программные системы: учебное пособие / С.С. Сосинская, Р.С. Дорофеев. – 2-е изд., стер. – Старый Оскол: ТНТ, 2018. – 192 с.
4. Гордеев, А.В. Системное программное обеспечение [Текст] / А.В. Гордеев, А.Ю. Молчанов. – СПб.: Питер, 2001. – 736 с.
5. Контекстно-зависимая грамматика [Электронный ресурс] // Википедия: [сайт]. URL: https://ru.wikipedia.org/wiki/Контекстно-зависимая_грамматика (дата обращения: 20.01.23).
6. Presto/SqlBase, грамматика языка запросов SQL [Электронный ресурс] // Github: [сайт]. – URL: https://github.com/prestodb/presto/blob/master/presto-parser/src/main/antlr4/com/facebook/presto/sql/parser/SqlBase.g4 (дата обращения: 12.02.23).
7. ANTLR [Электронный ресурс] // ANTLR: [сайт]. URL: https://www.antlr.org (дата обращения: 21.01.23).
8. SQL: кому нужен и в чём его польза? [Электронный ресурс] // Институт ИТ и бизнес-администрирования: [сайт].
– URL: https://www.instituteiba.by/courses/it/oracle/column-expert/sql-who-needs-and-how-does-it-help/ (дата обращения: 10.02.23).