Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

КУРСОВАЯ РАБОТА (ПРОЕКТ)

по дисциплине «Проектирование и архитектура программных систем»
наименование дисциплины

на тему Разработка Системы Управления Базами Данных

Направление подготовки (специальность) 09.03.04
(код, наименование)
Программная инженерия

Автор работы (проекта) Е. А. Сологуб _____
(инициалы, фамилия) (подпись, дата)

Группа ПО-23б

Руководитель работы (проекта) А. А. Чаплыгин _____
(инициалы, фамилия) (подпись, дата)

Работа (проект) защищена _____
(дата)

Оценка _____

Члены комиссии

_____ _____
подпись, дата фамилия и. о.

_____ _____
подпись, дата фамилия и. о.

_____ _____
подпись, дата фамилия и. о.

Курск 2025 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ (ПРОЕКТ)

Студента Сологуба Е.А., шифр 22-06-0610, группа ПО-23б

1. Тема «Разработка Системы Управления Базами Данных » .

2. Срок предоставления работы к защите «13» января 2025 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

1) проанализировать внутренее строение системы управления базами данных;

2) спроектировать архитектуру программного интерфейса системы управления базами данных;

3) сконструировать и протестировать программную систему управления базами данных.

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для системы управления базами данных являются: введенные данные пользователем, такие как: название таблицы, название колонок, данные для заполнения, условия;

2) Выходными данными для программной системы являются: сформированные данные после обработки их функциями программного интерфейса системы управления базами данных.

4. Содержание работы (по разделам):

4.1. Введение.

4.1. Анализ предметной области.

4.2. Техническое задание: основание для разработки, назначение разработки, требования к программной системе, требования к оформлению документации.

4.3. Технический проект: общие сведения о программной системе, проект данных программной системы, проектирование архитектуры программной системы, проектирование пользовательского интерфейса программной системы.

4.4. Рабочий проект: спецификация компонентов и классов программной системы, тестирование программной системы, сборка компонентов программной системы.

4.5. Заключение.

4.6. Список использованных источников.

5. Перечень графического материала:

Руководитель работы (проекта)		А. А. Чаплыгин
	(подпись, дата)	(инициалы, фамилия)
Задание принял к исполнению		Е. А. Сологуб
	(подпись, дата)	(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 27 страницам. Работа содержит 4 иллюстрации, 1 таблицу, 11 библиографических источников и 4 листа исходного кода. Количество приложений – 2. Фрагменты исходного кода представлены в приложении А. Графические компоненты представлены в приложении Б.

Перечень ключевых слов: база данных, пользователь, система, система управления базами данных, данные, структура, макросы, функции, методы.

Объектом разработки является системы управления базами данных. Целью курсовой работы является улучшения понимания работы системы управления базами данных и создание своей собственной базы данных.

При создании программного продукта применялась технология функционального программирования.

ABSTRACT

The volume of work is 27 pages. The work contains 4 illustrations, 1 table, 11 bibliographic sources and 4 sheets of listing. Number of appendices – 2. Source code fragments in Appendix A. Source graphic components in Appendix B.

List of keywords: database, user, system, database management system, data, structure, macros, functions, methods.

The object of development is a database management system. The purpose of the course work is to improve the understanding of the database management system and to create your own database.

The technology of functional programming was used in the creation of the software product.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

СУБД – система управления базами данных.

ПО – программное обеспечение.

РП – рабочий проект.

ТЗ – техническое задание.

ТП – технический проект.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

ВВЕДЕНИЕ

В современном мире данные являются одним из самых ценных ресурсов, что делает их обработку и управление ключевыми задачами для различных сфер деятельности. СУБД обеспечивают эффективное хранение, обработку, управление и доступ к информации, играя важную роль в бизнесе, науке, образовании и других областях.

Разработка СУБД представляет собой сложный процесс, включающий анализ предметной области, проектирование архитектуры системы и её реализацию с использованием современных технологий программирования. Такая система позволяет автоматизировать многие рутинные процессы, улучшить производительность и сократить вероятность ошибок при работе с большими объемами данных.

Главной задачей профессионально построенной системы управления базами данных является корректное хранение данных и быстрый доступ к ним.

Цель настоящей работы – создание функциональной системы управления базами данных, ориентированной на практическое применение в реальных условиях. СУБД, которая обеспечивает поддержку основных операций с данными, включая их добавление, удаление, обновление и поиск. Для достижения поставленной цели необходимо решить *следующие задачи:*

– провести анализ предметной области;

– разработать концептуальную модель СУБД;

– реализовать функционал СУБД средствами современных технологий программирования;

– провести тестирование и анализ работы системы.

Структура и объем работы. Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 27 страницам.

Во введении сформулирована цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

В первом разделе на стадии описания технической характеристики предметной области приводится сбор информации о устройстве системы управления базами данных.

Во втором разделе на стадии технического задания приводятся требования к разрабатываемой СУБД.

В третьем разделе на стадии технического проектирования представлены проектные решения для СУБД.

В четвертом разделе приводится список макроссов и функций, использованных при разработке СУБД, производится тестирование разработанной системы.

В заключении излагаются основные результаты работы, полученные в ходе разработки.

В приложении А представлены фрагменты исходного кода. В приложении Б представлены графические компоненты.

1 Анализ предметной области

Характеристика и использование СУБД.

Системы управления базами данных (СУБД) начали формироваться с середины XX века. Первая коммерческая СУБД, известная как Integrated Data Store (IDS), была разработана в 1960-х годах инженером Чарльзом Бахманом. Она представляла собой навигационную СУБД, где данные организовывались в виде сетевых структур.

В 1970 году революционным шагом стало предложение Эдгара Кодда, который разработал реляционную модель данных, опубликованную в его работе ”A Relational Model of Data for Large Shared Data Banks”. На основе этой модели в 1980-х годах появились первые реляционные СУБД, такие как IBM System R, Oracle и другие, которые до сих пор составляют основу многих современных систем.

Предметная область включает в себя процессы хранения, обработки, извлечения и анализа данных, которые актуальны для различных организаций и сфер деятельности. СУБД используется для управления информацией в самых разных контекстах, таких как коммерческие компании, научные учреждения, государственные структуры и образовательные организации. В основе всех этих процессов лежит необходимость упорядоченного хранения данных, которые могут быть представлены в виде таблиц, графов или других структур. Поэтому ключевыми аспектами анализа предметной области являются:

1. Определение типов данных. Для разработки СУБД важно учитывать, какие данные будут храниться в системе.

2. Выявление пользователей и их ролей. Анализ должен определить, кто будет работать с СУБД, какие операции они смогут выполнять, и какие уровни доступа к данным им необходимы.

3. Требования к безопасности данных. Важно предусмотреть защиту информации от несанкционированного доступа, утечек и потерь.

4. Интеграция с другими системами. Если СУБД должна взаимодействовать с другими программными продуктами, необходимо определить требования к интерфейсам и протоколам обмена данными.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки является задание на курсовую работу «Разработка системы управления базами данных».

2.2 Цель и назначение разработки

Основной задачей курсовой работы является разработка и тестирование собственной СУБД.

Посредством разработки и тестирование системы управления базами данных планируется улучшить понимание о внутренних процессах работы СУБД и разработать собственную систему управления базами данных.

Задачами данной разработки являются:

– реализация создания и удаления таблиц;

– реализация чтения таблиц по определенным условиям;

– реализация обновления таблицы;

– реализация создания копий и сохранения таблиц в отдельный текстовый файл.

2.3 Требования пользователя к интерфейсу СУБД

Система управления базами данных должна включать в себя:

– создание и удаление таблиц;

– работу с данными внутри таблиц;

– сохранение таблиц в отдельные файлы.

2.4 Моделирование вариантов использования

Для разрабатываемой системы управления базами данных была реализована модель, которая помогает понять использование СУБД.

На основании анализа предметной области в программе должны быть реализованы следующие прецеденты:

1. Добавление данных.

2. Обновление данных.
3. Удаление данных
4. Чтение и фильтрация данных.
5. Экспорт таблицы.

На рисунке 2.1 изображена диаграмма прецедентов для СУБД.

2.5 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Необходимо спроектировать и разработать СУБД, которая должна способствовать хранению и фильтрации данных.

Система управления базами данных представляет собой набор взаимосвязных функций и макроссов, которые отвечают за работу с таблицами. БД представляет собой хэш-таблицу со структурами таблиц пользователей, которые имеют такие поля: название, списки колонок, списки столбцов.

3.2 Обоснование выбора технологии проектирования

На сегодняшний день информационный рынок, поставляющий программные решения в выбранной сфере, предлагает множество продуктов, позволяющих достигнуть поставленной цели – разработки СУБД.

3.2.1 Описание используемых технологий и языков программирования

В процессе разработки СУБД используются программные средства и языки программирования. Каждое программное средство и каждый язык программирования применяется для круга задач, при решении которых они необходимы.

3.2.2 Язык программирования Lisp

Lisp - это один из старейших языков программирования высокого уровня, созданный в 1958 году Джоном Маккарти. Первоначально он был разработан для исследований в области искусственного интеллекта и быстро стал одним из самых популярных языков для решения задач, связанных с обработкой данных, символическим вычислением и логическим выводом.

3.2.2.1 Основные особенности языка программирования Lisp

1. Символьная обработка.

2. Простота синтаксиса.
3. Поддержка функционального программирования.
4. Макросы.
5. Множество диалектов.

3.3 Диаграмма компонентов и схема обмена данными между файлами компонента

Диаграмма компонентов описывает особенности физического представления разрабатываемой системы. Она позволяет определить архитектуру системы, установив зависимости между программными компонентами, в роли которых может выступать как исходный, так и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы, а также зависимости между ними. На рисунке 3.1 изображена диаграмма компонентов для проектируемой системы.

4 Рабочий проект

4.1 Структуры, используемые при разработке СУБД

Можно выделить следующий список структур и их методов, использованных при разработке СУБД (таблица 4.1).

Таблица 4.1 – Описание структур, используемых в СУБД

Название структуры	Модуль, к которому относится структура	Описание структуры	Методы
1	2	3	4
table	main	table – основная структура приложения. Имеет такие поля как: name, rows, columns.	Нет.

4.2 Описание функций системы управления базами данных

1. create-table – функция для создания таблицы.
2. insert-into – функция для вставки данных в таблицу.
3. make-select-from – макрос для выборки данных по различным критериям.
4. delete-from – функция для удаления данных из таблицы.
5. drop-table – функция для удаления таблицы.
6. backup-table – функция для создании копии таблицы в другую таблицу.
7. save-table-to-file – функция для записи таблицы в другой файл.
8. make-update – макрос для обновления данных по различным критериям.

4.3 Автоматизированное тестирование СУБД

Один из автоматизированных тестов функций СУБД представлен на рисунке 4.1.

Вызов теста может осуществляться с разными входными данными. Пример представлен на рисунке 4.2.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана система управления базами данных на языке программирования Lisp, ориентированной на практическое применение в реальных условиях.

Основные результаты работы:

1. Проведен анализ предметной области.
2. Разработана концептуальная модель системы управления базами данных. Определены требования к системе.
3. Осуществлено проектирование СУБД. Разработана архитектура взаимодействия СУБД с пользователем.
4. Реализована и протестирована СУБД. Проведено автоматизированное тестирование.

Все требования, объявленные в техническом задании, были полностью реализованы, все задачи, поставленные в начале разработки проекта, были также решены.

Готовый рабочий проект представлен программным интерфейсом системы управления базы данных для интеграции в различные проекты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сайбель Питер, С.П. Практическое использование Common Lisp / С.П. Сайбель Питер. – М. СПб. : Питер, 2017. – 615 с. – ISBN 978-5-97060-538-7. – Текст : непосредственный.

2. Грэм Пол, Г. П. ANSI Common Lisp / Г. П. Грэм Пол. – Москва : Бомбора, 2017. – 498 с. – ISBN 978-5-93286-206-3.. – Текст : непосредственный.

3. Шилдс, Ш. У. SQL: быстрое погружение / Ш. У. Шилдс. – М. СПб. : Питер, 2022. – 254 с. – ISBN 978-5-4461-1835-9.. – Текст : непосредственный.

4. Моргунов, Е. П. PostgreSQL. Основы языка SQL / Е. П. Моргунов. – Москва : БХВ, 2022. – 336 с. – ISBN 978-5-9775-4022-3. – Текст : непосредственный.

5. Роберт, М. Чистая архитектура. Искусство разработки программного обеспечения / М. Роберт. – М. СПб. : Питер, 2022. – 592 с. – ISBN 978-5-4461-0772-8. – Текст : непосредственный.

6. Head First. Паттерны проектирования. 2-е издание / Эрик Фримен, Элизабет Робсон, Кэтти Сьерра, Берт Бейтс. – М. СПб. : Питер, 2021. – 640 с. – ISBN 978-5-4461-1819-9. – Текст : непосредственный.

7. Бхаргава, Адитья Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих / Адитья Бхаргава. – М. СПб. : Питер, 2022. – 288 с. – ISBN 9785446109234. – Текст : непосредственный.

8. Бондарь, А.Г. Microsoft SQL Server 2022 / А.Г. Бондарь. – Москва : БХВ, 2022. – 528 с. – ISBN 978-5-9775-1805-5. – Текст : непосредственный.

9. Фридман, Д.П. THE LITTLE SCHEMER: Чудесное функциональное программирование / Д.П. Фридман. – Москва : Наука и техника, 2024. – 230 с. – ISBN 978-5-93700-234-1. – Текст : непосредственный.

10. Metanit : сайт. – URL: https://metanit.com/sql/mysql/2.1.php (дата обращения: 11.01.2025)

11. GitHub : сайт. – URL: https://github.com/sqlalchemy/sqlalchemy (дата обращения: 11.01.2025)

ПРИЛОЖЕНИЕ А

Фрагменты исходного кода программы

main.lsp

```
(defpackage :DBMS-package
(:use :cl))

(defstruct table
name
rows
columns)

(defvar *database* (make-hash-table :test 'equal))

; Функции для работы с таблицей
(defun create-table (name &optional columns)
(progn
(unless (gethash name *database*)
(setf (gethash name *database*) (make-table :name name :columns columns :
   rows nil)))
1))

(defun insert-into (table-name row-data)
(let ((table (gethash table-name *database*)))
(if table
(let* ((columns (table-columns table))
(column-names (get-column-names columns)))
;;Проверка, что все переданные ключи есть в колонках таблицы, ключи
   приводятся из  строкового формата в знаковый
(if (every (lambda (col) (member (intern (string-upcase (string col)))
   column-names :test 'eq)) (mapcar #'car row-data))
(progn
(push row-data (table-rows table)))
;; (format t "Rows ~a inserted into~%" row-data))
(format t "ERROR: Some columns is not exist in table ~a~%" table-name)))
(format t "ERROR: Table ~a not found!~%" table-name))))

(defun select-all (table-name)
(let ((table (gethash table-name *database*)))
(if table
(table-rows table)
(progn
(format t "ERROR: Table ~a not found!~%")
nil))))


(defun delete-from (table-name cond-pair)
(let ((table (gethash table-name *database*)))
(if table
(let ((new-rows (remove-if (lambda (row) (every (lambda (condit) (equal (
   cdr condit) (cdr (assoc (car condit) row)))) cond-pair)) (table-rows
   table))))
(setf (table-rows table) new-rows)
```

```
45  (format t "Rows matching ~a deleted from table ~a.~%" cond-pair table-name)
        new-rows)
46  (format t "ERROR: Table ~a not found!~%" table-name))))
47
48  (defun drop-table (table-name)
49  (if (gethash table-name *database*)
50  (progn
51  (remhash table-name *database*)
52  (format t "Table ~a dropped!~%" table-name)
53  1)
54  (progn
55  (format t "ERROR: Table ~a not found!~%" table-name)
56  nil)))
57
58  (defun backup-table (table-name backup-name)
59  (let ((table (gethash table-name *database*)))
60  (if table
61  (progn
62  (setf (gethash backup-name *database*)
63  (make-table    :name (table-name table)
64  :columns (copy-list (table-columns table))
65  :rows (copy-list (table-rows table))))
66  (progn
67  (format t "Backup of table ~a success, create backup table ~a!~%"
        table-name backup-name)
68  1))
69  (progn
70  (format t "ERROR: Table ~a nor found!~%" table-name)
71  nil))))
72
73  (defun save-table-to-file (table-name file-path)
74  (let ((table (gethash table-name *database*)))
75  (if table
76  (with-open-file (stream file-path :direction :output
77  :if-exists :supersede
78  :if-does-not-exist :create)
79  (let ((columns (table-columns table)))
80  ;; Запись заголовков
81  (format stream "~{~A~^,~}~%" (mapcar #'car columns))
82  ;; Запись строк
83  (dolist (row (table-rows table))
84  (format stream "~{~A~^,~}~%"
85  (mapcar (lambda (col) (cdr (assoc (car col) row)))  columns))))
86  1)
87  (format t "ERROR: Table ~a not found!~%" table-name))))
88
89  (defun table-exist (table-name)
90  (not (null (gethash table-name *database*))))
91
92  (defun get-column-names (columns)
93  (mapcar (lambda (col) (intern (string (car col)))) columns))
94
95  (defun filter (predicate rows)
```

```
  "Фильтрует строки rows, оставляя только те, которые удовлетворяют предикату
      predicate."
  (remove-if-not predicate rows))

  (defun gen-vars (table)
  (mapcar (lambda (col)
  `(,col (cdr (assoc ',col row))))
  (mapcar #'car (table-columns table))))

  (defmacro create-condition (conditions)
  `(lambda (row)
  (every (lambda (condition)
  (cond
  ;; Условие вида (< age 30)
  ((and (listp condition)
  (symbolp (car condition))
  (symbolp (cadr condition)))
  (let* ((col (cadr condition))
  (op (car condition))
  (val (caddr condition))
  (row-val (cdr (assoc col row))))
  (case op
  ((=) (equal row-val val))
  ((>) (> row-val val))
  ((<) (< row-val val))
  ((>=) (>= row-val val))
  ((<=) (<= row-val val))
  ((/=) (not (equal row-val val)))
  (otherwise (error "Unsupported operator: ~a~%" op)))))

  ;; Условие вида (name . "Egor")
  ((and (consp condition)
  (symbolp (car condition)))
  (let ((key (car condition))
  (value (cdr condition)))
  (equal value (cdr (assoc key row)))))

  ((or (stringp condition) (not (consp condition)))
  (error "Unsupported condition format: ~a~%" condition))

  (t (error "Unsupported condition format: ~a~%" condition))))
  ,conditions)))

  (defmacro make-select-from (table-name columns &body body)
  `(let ((table (gethash ,table-name *database*)))
  (if table
  (let ((table-columns (mapcar #'car (table-columns table))))
  ;; Проверка наличия всех выбранных столбцов
  (if (every (lambda (col)
  (member (intern (string-upcase (string col))) table-columns :test 'eq))
  ,columns)
  (let* ((filtered-rows
  (remove-if-not (lambda (row)
  (let () ;;,(gen-vars table) (id (cdr (assoc 'id row)))
```

```
,@body)) ;; Выполняем действия полученные из body
(table-rows table))))
;; Вывод столбцов
(mapcar (lambda (row)
(remove-if-not (lambda (pair)
(member (car pair) ,columns :test 'equal))
row))
filtered-rows))
(progn
(format t "ERROR: Columns ~a not found in table ~a~%" ,columns ,table-name
    )
nil)))
(progn
(format t "ERROR: Table ~a not found!~%" ,table-name)
nil))))

(defmacro make-update (table-name updates &body body)
`(let* ((table (gethash ,table-name *database*))
(rows (table-rows table)))
(let ((updated-rows
(mapcar
(lambda (row)
(let ,(mapcar (lambda (col)
`(,col (cdr (assoc ',col row))))
(mapcar #'car (table-columns table)))
(if (progn ,@body) ; Выполняем условия
(let ((new-row (copy-list row)))
(dolist (update ',updates)
(let ((col (car update))
(val (cdr update)))
(setf (cdr (assoc col new-row)) val)))
new-row)
row))) ; Если условие не выполнено, строка остаётся неизменной
rows)))
(setf (table-rows table) updated-rows)
1)))
```

unittest.lsp

```
(load "./main.lsp")
(format t "Файл main.lsp загружен~%")
;(in-package :DBMS-package)

(defun test-create-table (table-name &optional columns expected-result)
(let ((result (create-table table-name columns)))
(as-eq "test-create-table" result expected-result)))

(defun test-insert-into (table-name columns row expected-row)
(create-table table-name columns)
(insert-into table-name row)
(let ((table (gethash table-name *database*)))
(if table
(let ((actual-row (first (table-rows table))))
(as-eq "test-insert-into" actual-row expected-row))
(format t "ERROR: Table ~a не найдена в базе данных~%" table-name))))
```

```
17
18  (defun test-select-from (table-name columns rows selected-columns
        expected-result &optional condition)
19  (create-table table-name columns)
20  (dolist (row rows)
21  (insert-into table-name row))
22  (let ((result (select-from table-name selected-columns condition)))
23  (as-eq "test-select-from" result expected-result)))
24
25  (defun test-select-all (table-name columns rows expected-result)
26  (create-table table-name columns)
27  (dolist (row rows)
28  (insert-into table-name row))
29  (let ((result (select-all table-name)))
30  (as-eq "test-select-all" result expected-result)))
31
32  (defun test-drop-table (table-name columns expected-result)
33  (create-table table-name columns)
34  (let ((result (drop-table table-name)))
35  (as-eq "test-drop-table" result expected-result)))
36
37  (defun test-update-data (table-name columns rows id field-name new-value
        expected-result)
38  (create-table table-name columns)
39  (dolist (row rows)
40  (insert-into table-name row))
41  (let ((result (update table-name id field-name new-value)))
42  (as-eq "test-update" result expected-result)))
43
44  (defun test-backup-table (table-name columns rows backup-name
        expected-result)
45  (create-table table-name columns)
46  (dolist (row rows)
47  (insert-into table-name row))
48  (let ((result (backup-table table-name backup-name)))
49  (as-eq "test-backup-table" result expected-result)))
50
51  (defun test-save-table-to-file (table-name columns rows path
        expected-result)
52  (create-table table-name columns)
53  (dolist (row rows)
54  (insert-into table-name row))
55  (let ((result (save-table-to-file table-name path)))
56  (as-eq "test-save-table-to-file" result expected-result)))
57
58  (defun test-create-condition ()
59  (let* ((rows '(((id . 1) (name . "Alice") (age . 25))
60  ((id . 2) (name . "Egor") (age . 30))
61  ((id . 3) (name . "Alex") (age . 35))))
62  ;; Условие: возраст < 30
63  (condition1 (create-condition '((< age 30))))
64  ;; Условие: возраст между 20 и 40
65  (condition3 (create-condition '((>= age 30) (/= name "Alex")))))
66  (format t "Rows matching condition1: ~a~%" (remove-if-not condition1 rows))
```

```
(format t "Rows matching condition3: ~a~%" (remove-if-not condition3 rows))
   ))

(defun mk-sel-from ()
(create-table 'tab1 '((id integer) (name string) (age integer)))
(insert-into 'tab1 '((id . 1) (name . "alex") (age . 25)))
(insert-into 'tab1 '((id . 2) (name . "egor") (age . 54)))
(insert-into 'tab1 '((id . 3) (name . "vova") (age . 14)))
(insert-into 'tab1 '((id . 4) (name . "daniil") (age . 18)))
(insert-into 'tab1 '((id . 5) (name . "alex") (age . 27)))

(let ((result (make-select-from 'tab1 '(name age)
(and (string= (cdr (assoc 'name row)) "alex")
(> (cdr (assoc 'age row)) 20)))))
(format t "Result make-select-from: ~a~%" result)))

(defun mk-update-test ()
(create-table 'tab1 '((id integer) (name string) (age integer)))
(insert-into 'tab1 '((id . 1) (name . "alex") (age . 25)))
(insert-into 'tab1 '((id . 2) (name . "egor") (age . 54)))
(insert-into 'tab1 '((id . 3) (name . "vova") (age . 14)))
(insert-into 'tab1 '((id . 4) (name . "daniil") (age . 18)))
(insert-into 'tab1 '((id . 5) (name . "alex") (age . 27)))

(make-update 'tab1 '((age . 20)) (< (cdr (assoc 'age row)) 20))
(let ((result (make-select-from 'tab1 '(id name age))))
(format t "Result make-update: ~a~%" result)))


(defun as-eq (test-name result expected)
(if (equal result expected)
(format t "~a: PASSED~%" test-name)
(format t "~a: FAILED - ожидалось ~a, но было полученно ~a~%" test-name
   expected result)))

(defun run-unittest ()
(format t "Начало тестирования...~%~%")
(test-gen-vars)
(test-create-condition)
;; Тестирование создания таблицы.
(test-create-table 'test-table01 '((id . 1) (name . "adm")) 1)
;; Тестирование вставки данных в таблицу.
(test-insert-into 'users '((id integer) (name string) (age integer))
'((id . 1) (name . "adm") (age . 24))
'((id . 1) (name . "adm") (age . 24)))
(test-drop-table 'test-table111 '((id integer) (name string) (age integer))
    1)
;; Тестирование чтения данных из таблицы без условия.
(test-select-from 'test-table111
'((id integer) (name string) (age integer))
'(((id . 1) (name . "egor") (age . 19))
((id . 2) (name . "admin") (age . 24)))
'(name)
'(((name . "admin")) ((name . "egor")))
```

```
118   nil)
119   ;; Тестирование чтения данных из таблицы с условием.
120   (test-drop-table 'test-table1211 '((id integer) (name string)) 1)
121   (test-select-from 'test-table1211
122   '((id integer) (name string) (age integer))
123   '(((id . 1) (name . "egor") (age . 19))
124   ((id . 2) (name . "admin") (age . 24))
125   ((id . 3) (name . "alex") (age . 26)))
126   '(name)
127   '(((name . "admin")))
128   '((> age 23) (/= name "alex")))
129   ;; Тестирование удаления таблицы.
130   (test-drop-table 'table1 '((id integer) (name string)) 1)
131   ;; Тестирование чтения всех данных из таблицы.
132   (test-select-all 'table1 '((id integer) (name string))
133   '(((id . 1) (name . "adm"))
134   ((id . 2) (name . "adm1")))
135   '(((id . 2) (name . "adm1"))
136   ((id . 1) (name . "adm"))))
137   ;; Тестирование функции обновления данных по id
138   (test-drop-table 'table999 '((id integer) (name string) (age integer)) 1)
139   (test-update-data 'table999 '((id integer) (name string) (age integer))
140   '(((id . 1) (name . "egor") (age . 19))
141   ((id . 2) (name . "admin") (age . 24)))
142   1
143   'name
144   "update-egor"
145   '((id . 1) (name . "update-egor") (age . 19)))
146   ;; Тестирование функции создания копии таблицы
147   (test-backup-table 'table2213 '((id integer) (name string) (age integer))
148   '(((id . 1) (name . "egor"))
149   ((id . 2) (name . "alex"))
150   ((id . 3) (name . "asd")))
151   'backup-table2213
152   1)
153   ;; Тестирование функции записи таблицы в CSV файл
154   (test-drop-table 'table2214 '((id integer) (name string) (age integer)) 1)
155   (test-save-table-to-file 'table2214 '((id integer) (name string) (age
          integer))
156   '(((id . 1) (name . "alex") (age . 23))
157   ((id . 2) (name . "egor") (age . 24))
158   ((id . 3) (name . "vova") (age . 25))
159   ((id . 4) (name . "daniil") (age . 12)))
160   "output.txt"
161   1)
162   (test-drop-table 'tab1 '((id integer) (name string) (age integer)) 1)
163   (mk-sel-from)
164   (test-drop-table 'tab1 '((id integer) (name string) (age integer)) 1)
165   (mk-update-test)
166
167   (format t "~%Тестирования завершенно.~%"))
```

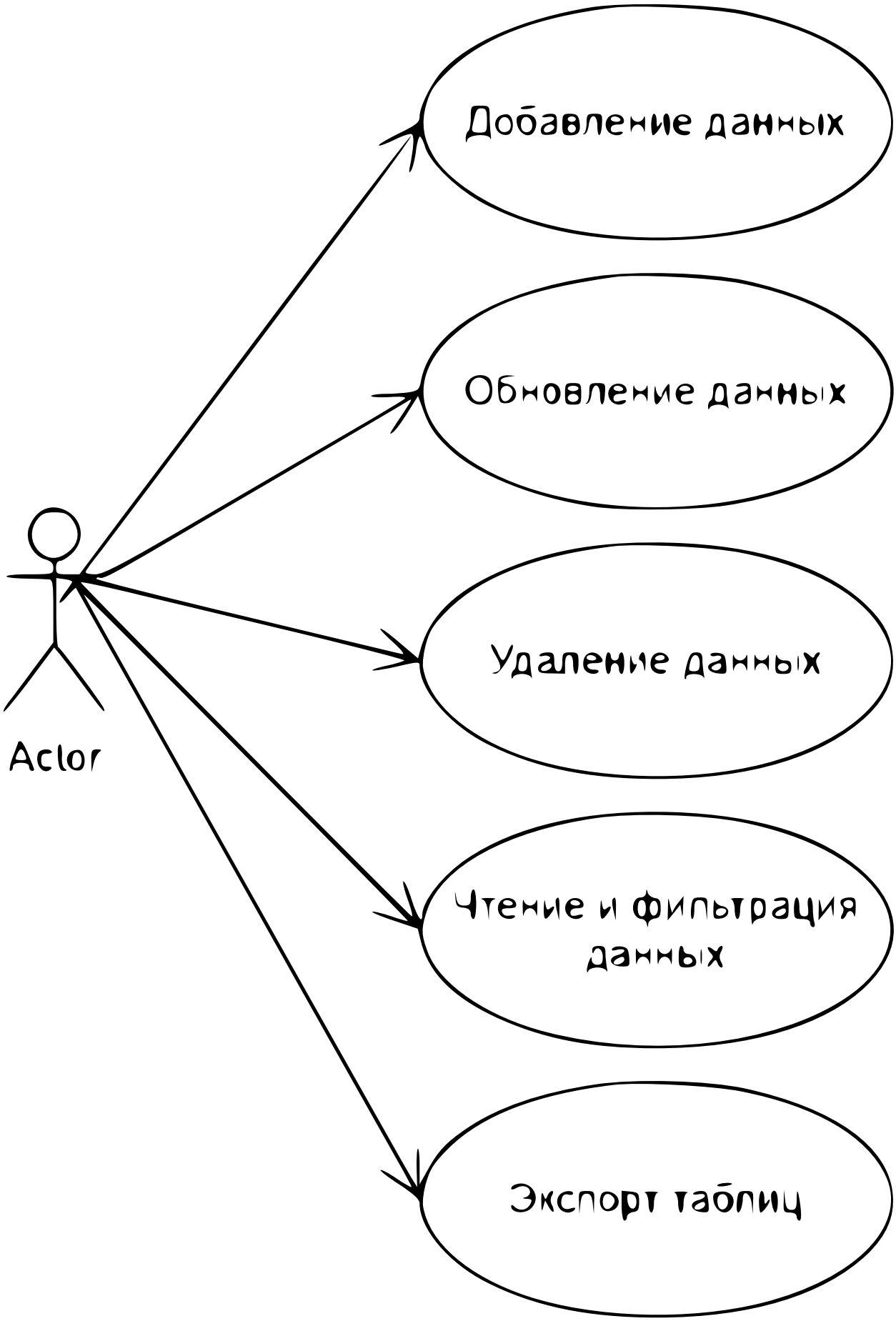

Рисунок 2.1 – Диаграмма прецедентов

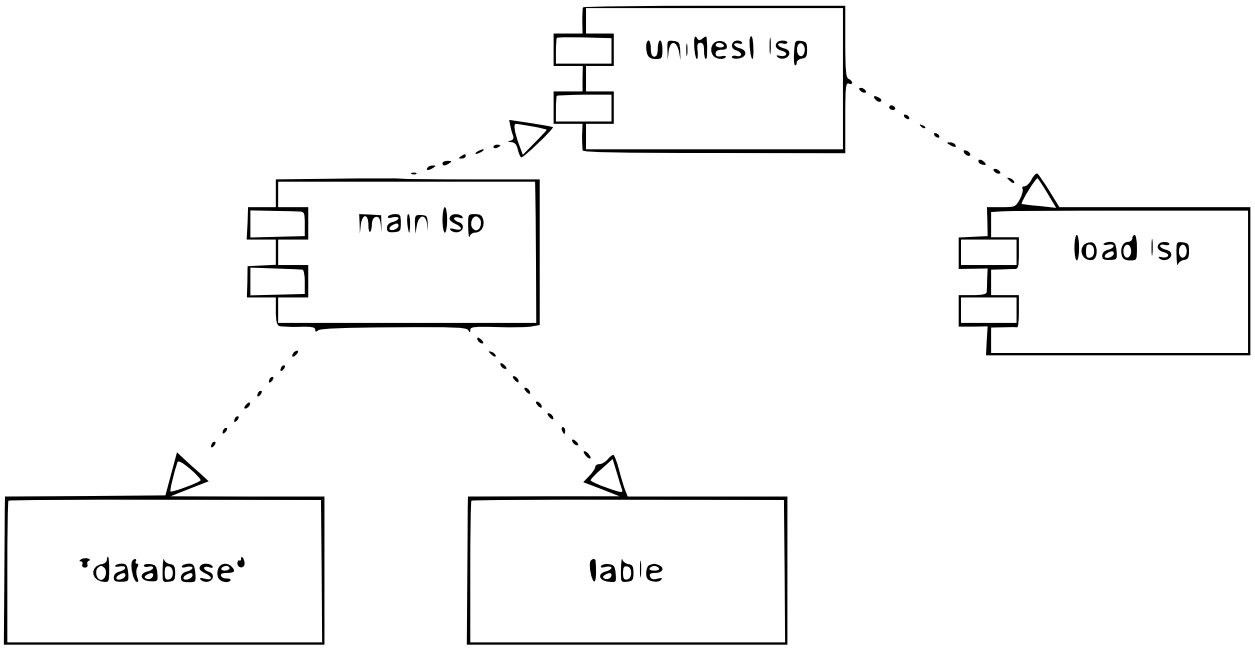

Рисунок 3.1 – Диаграмма компонентов

```
(defun test-select-from (table-name columns rows selected-columns
   expected-result &optional condition)
(create-table table-name columns)
(dolist (row rows)
(insert-into table-name row))
(let ((result (select-from table-name selected-columns condition)))
(as-eq "test-select-from" result expected-result)))
```

Рисунок 4.1 – Автоматизированный тест функции

```
    (test-select-from 'test-table1211
    '((id integer) (name string) (age integer))
    '(((id . 1) (name . "egor") (age . 19))
    ((id . 2) (name . "admin") (age . 24))
    ((id . 3) (name . "alex") (age . 26)))
    '(name)
    '((name . "admin")))
    '((> age 23) (/= name "alex")))
```

Рисунок 4.2 – Вызов автоматизированного теста функции