Московский Государственный Университет им. М. В. Ломоносова
Механико-математический факультет
Кафедра вычислительной математики

Якушев Дмитрий
группа 509

Дипломная работа

Создание модели сущность-связь процесса поиска зависимостей структура-свойство для построения репозитория QSAR моделей.

Научный руководитель
д. ф.-м. н.
Кумсков М. И.

Москва, 2009

В работе предложена, проанализирована, и разработана модель сущность-связь процесса поиска зависимостей структура-свойство, на основе которой реализован репозиторий QSAR моделей.

Модель сущность-связь представляет собой объединение двух моделей: процесса расчета по QSAR модели и сохранения промежуточных результатов расчетов этих моделей.

На ее основе при помощи методов прямого проектирования(forward engineering) из программного продукта IBM Rational Rose Professional 7.0.0.0 Data Modeler Edition был сформирован скрипт на языке SQL для создания базы данных.

Для хранения элементов QSAR моделей помимо базы данных была использована система управления версиями.

Для наполнения базы данных и системы управления версиями посредством сети internet был разработан web-interface. Для безопасной загрузки и выгрузки информации из системы был дополнительно введен безопасный ftp-сервер.

Вся эта система была развернута на машине, находящейся в Институте Органической Химии им. Н.Д. Зелинского.

Содержание

1 Общая постановка задачи "структура-свойство".

1.1 Методы распознавания образов.

Прежде чем рассматривать саму задачу "структура-свойство", которую предстоит решать, коснемся совокупности математических методов под названием "распознавание образов". Задача Quantitative Structure Activity Relationships является всего лишь одной из частных проблем данной области. Поэтому необходимо рассмотреть, какие существуют методы теории распознавания.

1.1.1 Постановка задачи распознавания образов.

Исходной информацией являются описания объектов, ситуаций, предметов, явлений или процессов S в виде векторов значений признаков S = $(x_1(S), x_2(S), ..., x_n(S))$, где признаки x_i, i = 1, ..., n, характеризуют различные стороны-свойства S. У объектов S существует "основное свойство"y(S), которое для части объектов $S_1, S_2, ..., S_m$ предполагается известным, а для части объектов нет. Задача распознавания (прогноза, идентификации, "классификации с учителем") состоит в определении значения свойства y(S) по информации $S_1, S_2, ..., S_m, y(S_1), y(S_2), ..., y(S_m)$ (обучающей или эталонной выборке). Признаки могут быть числовыми (задающими степень выраженности какого-либо свойства), бинарными ("есть"или "нет"свойство), номинальными (обозначающими наличие различных свойств без числовой оценки - пол, цвет, и т.д.).

1.1.2 Основные методы теории распознавания.

1. Алгоритмы распознавания, основанные на вычислении оценок. Распознавание осуществляется на основе сравнения распознаваемого объекта с эталонными по подмножествам признаков различной мощности при использовании процедур голосования. Оптимальные параметры решающего правила и процедуры голосования находятся из решения задачи оптимизации модели распознавания: находятся такие значения параметров, при которых точность распознавания является максимальной [4, 5, 6].

2. Алгоритмы голосования по тупиковым тестам. Сравнение распознаваемого объекта с эталонными осуществляется по различным "информативным"подмножествам признаков. В качестве подобных подсистем признаков используются тупиковые тесты (аналоги тупиковых тестов для веществennозначных признаков) различных случайных подтаблиц исходной таблицы эталонов [4, 5, 6, 7]

3. Алгоритмы голосования по логическим закономерностям. По обучающей выборке вычисляются множества логических закономерностей каждого класса - наборы признаков и интервалы их значений, свойственные каждому классу. При распознавании нового объекта вычисляется число логических закономерностей каждого класса, выполняющихся на распознаваемом объекте. Каждое отдельное "выполнение"считается "голосом"в пользу соответствующего класса. Объект относится в тот класс, нормированная сумма "голосов"за который является максимальной. Настоящий метод позволяет оценивать веса признаков, логические корреляции признаков, строить логические описания классов, находить минимальные признаковые подпространства [8, 9, 10].

4. Алгоритмы статистического взвешенного голосования. По данным обучающей выборки находятся статистически обоснованные логические закономерности классов. При распознавании новых объектов вычисляется оценка вероятности принадлежности объекта к каждому из классов, которая является взвешенной суммой "голосов"[6, 11, 12, 13, 14].

5. Линейная машина. Для каждого класса находится некоторая линейная функция. Распознаваемый объект относится в тот класс, функция которого принимает максимальное значение на данном объекте. Оптимальные линейные функции классов находятся в результате решения задачи поиска максимальной совместной подсистемы системы линейных неравенств, которая формируется по обучающей выборке - находится специальная кусочно-линейная поверхность, правильно разделяющая максимальное число элементов обучающей выборки [15, 16].

6. Линейный дискриминант Фишера. Классический статистический метод построения кусочно-линейных поверхностей разделяющих клас-

сы. Благоприятными условиями применимости линейного дискриминанта Фишера являются выполнение следующих факторов: линейная отделимость классов, дихотомия, "простая структура"классов, невырожденность матриц ковариаций, отсутствие выбросов [15].

7. Метод k ближайших соседей. Классический статистический метод. Распознаваемый объект относится в тот класс, из которого он имеет максимальное число соседей. Оптимальное число соседей и априорные вероятности классов оцениваются по обучающей выборке [15].

8. Нейросетевая модель распознавания с обратным распространением. Модификация метода обучения нейронной сети распознаванию образов (метод обратного распространения ошибки). В качестве критерия качества текущих параметров нейронной сети используется гибридный критерий, учитывающий как сумму квадратов отклонений значений выходных сигналов от требуемых, так и количество ошибочных классификаций на обучающей выборке [17].

9. Метод опорных векторов (support vector machine). Метод построения нелинейной разделяющей поверхности с помощью опорных векторов. В новом признаковом пространстве (спрямляющем пространстве) строится линейная разделяющая поверхность. Построение данной поверхности сводится к решению задачи квадратичного программирования [18].

10. Алгоритмы решения задач распознавания коллективами различных распознающих алгоритмов. Задача распознавания решается в два этапа. Сначала применяются независимо различные алгоритмы теории распознавания образов. Далее находится автоматически оптимальное коллективное решение с помощью специальных методов-"корректоров"[4, 5, 19, 20].

11. Методы кластерного анализа. "Алгоритмы иерархической группировки"; "кластеризация по методу минимизации суммы квадратов отклонений"; "метод k-средних. Возможно решение задачи классификации как при заданном, так и неизвестном числе классов [3,15].

12. Алгоритм построения коллективных решений задачи классификации. Задача классификации решается в два этапа. Сначала находится набор различных решений (в виде покрытий или разбиений)

при фиксированном числе классов с помощью различных алгоритмов теории распознавания образов. Далее находится оптимальная коллективная классификация в результате решения специальной дискретной оптимизационной задачи [21].

1.2 Quantitative Structure Analysis Relationships.

1.2.1 Общая постановка QSAR-задачи, основные определения и термины.

Дадим определение так называемого QSAR/QSPR-анализа (QSAR/QSPR - Quantitative Structure Analysis/Property Relationships, задача "структура-свойство"). Он является самым распространенным методом установления количественных соотношений между структурой и активностью соединений и представляет собой статистический подход к проблеме.

Меченый молекулярный граф G = E, V - помеченный граф, вершины которого интерпретируются как атомы молекулы, а ребра - как валентные связи между парами атомов. Метки вершин и ребер (числа или символы) кодируют атомы и связи различной химической природы. В качестве меток вершин могут быть использованы любые характеристики соответствующих атомов (например, трехмерные координаты, символ химического элемента, заряд ядра, поляризуемость, атомный вес, атомный радиус и др.), а в качестве меток ребер - любые характеристики соответствующих связей (кратность, длины, порядки связей, полученные из квантово-химических расчетов, и т.д. [22]).

Задача "структура-свойство" Пусть задана обучающая (или эталонная) выборка - база данных из N химических соединений, где:

1. i-ое соединение представлено меченым молекулярным графом G_i, имеющим укладку в трехмерном пространстве (т.е., для каждой вершины в качестве меток заданы ее трехмерные координаты);

2. либо i-ое соединение отнесено к C_i - одному из K классов активности (например, "активных", "слабоактивных", "неактивных"веществ) согласно исследуемому свойству, либо для него задано численное значение исследуемого свойства A_i.

Необходимо построить классифицирующую функцию F, получающую в качестве аргумента произвольный молекулярный граф с метками того же типа, и "наилучшим образом"относящую это соединение к одному из классов активности, либо "наилучшим образом"предсказывающую численное значение исследуемого свойства. Какая из классифицирующих функций "лучше", позволяет определить функционал качества φ(F). Например, в качестве функционала качества можно использовать процент верно классифицированных функцией F молекул из обучающей выборки:

$\varphi(F) = 1 - \frac{\sum \varepsilon_i}{N}$, где $\varepsilon_i = \{0, F(G_i) = c_I, 1$ - иначе$\}$, (1)

или, в случае, когда функция должна предсказывать численное значение свойства,

$\varphi(F) = 1 - \frac{\sum (F(G_i) - A_i)^2}{\sum A_i^2}$. (2)

Поставленную таким образом задачу поиска классифицирующей функции будем называть задачей "структура-свойство"или QSAR-задачей.

Дескриптором будем называть какое-либо свойство, численное значение которого может быть вычислено для произвольного молекулярного графа G.

Алфавитом дескрипторов будем называть множество всех дескрипторов, используемых для анализа обучающей выборки, обозначенных различными символьными метками.

Пусть алфавит дескрипторов состоит из M элементов.

Вектором признаков молекулярного графа G будем называть вектор , где x_j- значение j-ого дескриптора, вычисленное для G.

Матрицей "молекула-признак"(матрицей признаков) для рассматриваемой обучающей выборки будем называть матрицу размера N x M, в i-ой строке которой стоит вектор признаков i-ого соединения.

1.2.2 Основные этапы решения QSAR-задачи.

В вышеописанных терминах задача "структура-свойство"разбивается на две части:

1. этап описания: Исходя из формата молекулярных графов (типа меток вершин и ребер) выбирается алфавит дескрипторов A. На основе этого алфавита строится отображение из множества молекулярных графов в признаковое пространство и формируется матрица "молекула-признак"для обучающей выборки.

2. этап поиска модели функциональной зависимости: В результате анализа матрицы "структура-свойство"на признаковом пространстве строится модель функциональной зависимости - классифицирующая функция F с наилучшей прогностической способностью, т.е. с наибольшим значением функционала качества.

1.2.3 Этап описания молекулярного графа.

Рассмотрим два основных типа дескрипторов, на основе которых чаще всего строятся вектора признаков для молекулярных графов.

Топологические дескрипторы. Данные дескрипторы строятся по топологической матрице.

Топологическая матрица A = (a_{ij}) меченого молекулярного графа строится следующим образом: в ней элемент a_{ii} - это метка i-ой вершины, a_{ij} - метка ребра (i, j). Очевидно, что при изменении нумерации вершин графа получается, вообще говоря, другая матрица. Это является существенным недостатком описания химических структур в терминах матриц. Эта проблема привела к термину "инвариант графа".

Инвариант графа - это такое число (или функция, если метка графа - символы), вычисляемое по матрице графа A, которое не зависит от нумерации вершин графа. **Молекулярный граф** называется простым в следующем случае:

- метки вершин равны нулю
- если атомы связаны химической связью, то им сопоставляется ребро с меткой "1", в противном случае метка ребра равна нулю,

т.е. простой граф отображает только наличие связей между вершинами.

Топологическими индексами (ТИ) называют инварианты простых графов. Часто это определение переносится и на инварианты меченых графов, которые могут отражать не только топологию молекулы, но и элементы электронного и пространственного строения.

Топологические индексы и широко используются как дескрипторы при решении задачи "структура-свойство". Популярность данного подхода к описанию молекулярной структуры связана с простотой и быстротой вычисления ТИ, возможностью учитывать при их построении элементы электронного и пространственного строения, я также наличием

огромного количества удачных корреляций вида "ТИ - свойство". Однако такой подход имеет и очевидный недостаток: он не позволяет различать разные конфигурации молекул и не учитывает их конформационные особенности.

Ниже приведены примеры некоторых наиболее популярных ТИ.

Индекс Рандича X определяется по следующей формуле:

$$\mathrm{X} = \sum (v_i v_j)^{-1/2},$$

где v_i - степень i-ой вершины графа; суммирование проводится по всем ребрам графа.

Индекс X был обобщен Киром и Холлом, которые ввели определение так называемого индекса связности m-ого порядка (индекс Кира-Холла)

$$^{m}X^{v} = \sum (\delta_i \delta_j \delta_k)^{-1/2}, m \geq 1, \delta = \frac{(Z_i^v - h_i)}{(Z_i - Z_i^v - 1)},$$

где Z_i - общее число, а Z_i^v - число валентных электронов i-ого атома; h_i - число атомов водорода, связанных с i-ым атомом; суммирование происходит по всем цепочкам, состоящим из m ребер графа; i, j, k - номера вершин, образующих соответствующую цепочку.

Индекс Винера W определяется по следующей формуле:

$$W = 0.5 \sum d_{ij},$$

где d_{ij} - кратчайшее расстояние между i-ой и j-ой вершинами. Матрица $\mathrm{D} = (d_{ij})$ называется матрицей топологических расстояний.

Индекс Хосойя Z определяется формулой:

$$\mathrm{Z} = \sum p_k,$$

где p_k - число способов выбрать в графе k ребер ($k \geq 1$) так, что никакие два из них не являются смежными.

Индекс Балабана (для ациклических графов (деревьев)) определяется по следующей формуле:

$$B = \sum r_i^2,$$

где r_i - число вершин дерева, отсекаемых на i-ом шаге процедуры обрезки деревьев. Данная процедура состоит из последовательного удаления вершин степени 1 исходного дерева и инцедентных им ребер. Кроме перечисленных широко используются также семейства информационных топологических индексов[24] и топологических индексов спектрального типа[25].

Фрагментарные (структурные) дескрипторы. Использование данного типа дескрипторов основано на выделении в молекулах структурных фрагментов.

Структурным фрагментом молекулярного графа называется группа вершин с заданными условиями на их метки или метки их связей.

После выделения фрагментов каждому фрагменту сопоставляется структурный дескриптор, значение которого соответствует либо наличию или отсутствию данного фрагмента в молекулярном графе, либо количеству повторений фрагмента. В первом случае получаем дескриптор, принимающий логические значения, во втором - целые неотрицательные. Структурные фрагменты и соответствующие им дескрипторы подразделяют на два типа:

2D-дескрипторы В данном случае не учитываются значения валентных углов и евклидовых расстояний между атомами, не учитывается трехмерная структура фрагмента, важны только связи между атомами. Структурные 2D-фрагменты обычно имеют вид цепочек связанных атомов с определенными метками вершин и ребер, образующих данную цепочку.

3D-дескрипторы Дескрипторы этого типа учитывают трехмерную структуру фрагмента и обычно представляют собой множество вершин с заданными условиями на расстояния между ними и на их метки.

В общем виде метод состоит из следующих этапов:

1. Проводится дополнительная классификация атомов (вершин молекулярного графа) на основе их локальных свойств (заряда, эксцентриситета вершины, каких-либо топологических свойств). В результате этого метка каждой вершины заменяется на другую, содержащую информацию о локальных свойствах.

2. В молекулах выбираются структурные фрагменты (атомы, цепочки связанных атомов, группы атомов).

3. Каждому структурному фрагменту сопоставляется символьное имя - тип фрагмента (например, если рассматриваются цепочки атомов, то "именем"цепочки может служить объединение символьных меток входящих в нее атомов).

4. Множества фрагментов для всех молекулярных графов выборки объединяются.

5. Для каждого молекулярного графа и каждого фрагмента находим значение соответствующего структурного дескриптора (либо количество повторений, либо наличие/отсутствие в молекулярном графе).

В итоге, получаем матрицу "молекула-признак", состоящую из "структурных спектров"молекул.

1.2.4 Этап поиска функциональной зависимости.

Напомним, что после формирования матрицы "структура-свойство"для обучающей выборки необходимо построить классифицирующую функцию $F(x_1, x_2, , x_M), (x_1, x_2, , x_M)$ - вектор признаков молекулярного графа. Причем, построенная функция должна обеспечивать лучшее значение функционала качества. Обычно вид классифицирующей функции F заранее задается (например, функция может быть линейной, квадратичной и др.) и зависит от ряда параметров, которые определяются по обучающей выборке соединений. Чаще всего в качестве F используется линейная функция. Получаемое уравнение называют линейной регрессионной моделью. Заметим, что полученная формулировка задачи на этапе поиска функциональной зависимости полностью совпадает с формулировкой задачи распознавания. Поэтому для нахождения классифицирующей функции F можно использовать любые методы распознавания и классификации, описанные в начале главы. Следует отметить, что в задаче "структура-свойство"число дескрипторов M, как правило, значительно превышает число молекул в обучающей выборке (M » N), что затрудняет анализ матрицы "молекула-признак". Для того чтобы сократить число дескрипторов, необходимо рассматривать лишь наиболее информативные из них, т.е. те, которые потенциально будут значимы при построении классифицирующей функции на признаковом пространстве. Это можно проделать как на этапе описания (например, при эволюционном формировании дескрипторов), так и на этапе анализа матрицы признаков. В данной работе предлагается отбирать наиболее информативные дескрипторы путем взаимодействия этих двух этапов - использования результатов этапа анализа на этапе описания.

1.3 Выводы.

QSAR-задача (задача поиска зависимостей "структура-свойство") представляет собой одну из задач области распознавания образов. В практических решениях этой задачи часто можно выделить общие этапы. Актуальной выглядит проблема сохранения и дальнейшего использования(в виде составных частей других решений либо в качестве готового независимого инструмента) уже реализованных этапов. Необходимо создать систему, которая бы позволила пользователям, находящимся удаленно, использовать созданные ранее решения частично либо целиком.

2 Постановка задачи.

2.1 Описание проблемы.

В процессе научных исследований очень часто требуется проведение каких-либо расчетов, основной целью которых ставится получение результата по их завершении. После получения результатов все внимание обычно уделяется им: их анализируют, сравнивают, на их основе делают прогнозы и производят дальнейшие вычисления. А инструмент, с помощью которого они были получены, остается в стороне. Он обычно хранится у конечного исполнителя, и, как правило, такие инструменты даже не прикрепляют к отчетам по исследованию. Действительно, зачем нам пробирка, в которой был проведен опыт, если важно только ее содержимое? С течением времени о них все забывают, они теряются и изменяются. И вот, спустя какое-то время, когда требуется повторить такой расчет или опыт, найти человека, получившего результаты, часто просто невозможно, а вместе с этим невозможно восстановить технологический процесс получения этих результатов. В таких случаях работа по воспроизведению результатов начинается новым сотрудником "с нуля". Это иногда приводит к тому, что результаты предыдущего и нового экспериментов не совпадают. Возможно, затем даже строится теория, для опровержения предыдущей. Вся эта цепочка возникла из-за того, что детали проведения расчетов и опытов были утеряны со временем.

Для поиска зависимостей структура-свойство существует множество способов, каждый из которых представляет собой программную реализацию одной из моделей с определенными параметрами в виде цепочки последовательных расчетов, где результат предыдущего этапа расчетов является входными данными для следующего. Неточность на любом из этапов может привести к существенным отличиям между конечными результатами.

2.2 Постановка задачи.

Целью данного диплома является разработка средства, с помощью которого возможно долговременное хранение реализованных моделей по поиску зависимостей структура-свойство. А так же результатов отдельных расчетов, произведенных на основе этих моделей. В связи с частым

использованием общих частей у нескольких моделей для разработки такого средства также необходимо выделить в моделях общие части для их независимого хранения. Это позволит избежать дублирования информации и облегчит реализацию новых моделей за счет использования частей существующих моделей.

Переформулировать цель диплома можно следующим образом: "Разработка модели сущность-связь процесса поиска зависимостей структура-свойство и создание на ее основе репозитория для QSAR моделей".

Репозиторий - место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.[1]

Модель Сущность-Связь (ER-модель) (англ. entity-relationship model или entity-relationship diagram) - это модель данных, позволяющая описывать концептуальные схемы. Она предоставляет графическую нотацию, основанную на блоках и соединяющих их линиях, с помощью которых можно описывать объекты и отношения между ними какой-либо другой модели данных. В этом смысле ER-модель является метамоделью данных, то есть средством описания моделей данных.[2]

2.3 Разбиение общей задачи на подзадачи.

Исходя из последней формулировки цели диплома, можно выделить несколько последовательных подзадач, на которые разбивается основная задача и которые должны быть решены в рамках диплома:

1. Разработка модели сущность-связь процесса поиска зависимостей структура-свойство.

2. Создание базы данных результатов расчетов QSAR моделей на основе разработанной модели.

3. Построение репозитория QSAR на основе созданной базы данных.

В рамках первой подзадачи необходимо выделить у моделей общие части, определить порядок и тип связей, привязать эти части к их программной реализации в виде скриптов с конфигурационными файлами. На основе выделенных сущностей создать ER-модель.

[1] http://ru.wikipedia.org/wiki/Репозиторий.
[2] http://ru.wikipedia.org/wiki/ER-модель_данных.

Скриптовый язык (англ. scripting language, в русскоязычной литературе принято название язык сценариев) - язык программирования, разработанный для записи "сценариев", последовательностей операций, которые пользователь может выполнять на компьютере.[3]

Скрипт - последовательность действий, записанная на скриптовом языке.

Во вторую подзадачу входит введение дополнительных сущностей в созданную ER-модель, необходимых для отображения промежуточных и конечных результатов работы скриптов, а также создание базы данных, обеспечивающей хранение информации о моделях и результатах расчетов на основе этой ER-модели.

База данных(БД) - структурированный организованный набор данных, описывающих характеристики каких-либо физических или виртуальных систем.[4]

Последняя подзадача состоит в разработке на основе созданной базы данных репозитория, отвечающего требованиям, сформулированным ниже, продиктованным сценариями использования репозитория.

2.4 Первичная формулировка требовании к репозиторию.

В нашем случае под репозиторием будем понимать систему, обладающую следующими свойствами:

1. Долговременное хранение реализованных моделей и результатов, полученных на основе этих моделей.

2. Безопасный доступ к хранимым в системе данным из сети Internet для их анализа.

3. Возможность независимого использования хранимых данных.

Первое требование суть основная причина создания репозитория: требуется хранить без изменений в течение долгого времени модели и результаты, для возможности их использования в будущем.

[3]http://ru.wikipedia.org/wiki/Скриптовый_язык
[4]http://ru.wikipedia.org/wiki/База_данных.

Второе требование состоит из нескольких частей. Во-первых, для более оперативного использования хранимой в системе информации требуется обеспечить доступ к системе посредством сети Internet как наиболее массового и доступного средства обмена данными в электронном формате. Вторая часть требования основана на соображениях компьютерной безопасности, используемых в вероятных местах развертывания системы. Под последними в основном понимаются кафедра вычислительной математики механико-математического факультета МГУ и лаборатория Института Органической Химии им. Н.Д. Зелинского.

Третье требование продиктовано необходимостью обеспечить возможность использования составной частей модели независимо от остальных в качестве готового этапа для новой модели, а также возможность проведения расчетов целиком пользователем, не являющимся автором этой модели, на своем компьютере.

2.5 Обзор существующих решений схожих задач.

2.5.1 Сохранение кодов при разработке приложении.

Со схожей задачей сохранения промежуточных результатов работы зачастую сталкиваются в рамках разработки сложного программного обеспечения. Под сложностью в данном случае я понимаю то, что над таким продуктом работает группа разработчиков, что практически гарантирует возникновение ситуации, при которой несколько человек вносят изменения в одну часть проекта. Тогда важно сохранить изменения по отдельности для более оперативного воспроизведения, локализации и устранения ошибки. В этом случае в любое время можно точно воспроизвести из различных компонентов, входящих в продукт, ту версию, в которой была обнаружена ошибка. Дополнительно такой метод снижает риск потери внесенных изменений. За основу репозитория берется система управления версий.

Система управления версиями, система версионного контроля (от англ. Version Control System или Revision Control System) - программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более

ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое. [5]

Чаще всего система управления версиями представляет собой сервер. Доступ к информации, хранимой этим сервером, осуществляется через клиентскую часть.

Преимущества систем управления версиями:

1. Позволяют создавать разные варианты одного документа, т. н. ветки, с общей историей изменений до точки ветвления и с разными - после нее.

2. Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.

3. Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.

4. Контролируют права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие.

5. Сохраняются только изменения относительно предыдущей версии, что позволяет экономить ресурсы по сравнению с хранением всех версии документа целиком.

6. Некоторые системы управления версиями дают возможность заблокировать файл в хранилище. Блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла (например, средствами файловой системы) и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

7. Централизованное хранилище.

8. Масштабируемость.

Недостатки систем управления версиями: при отсутствии механизмов блокировки могут возникать ситуации, связанные с одновременным внесением изменений двумя и более пользователями.

[5] http://ru.wikipedia.org/wiki/Система_управления_версиями.

Большинство систем может автоматически объединить (слить) изменения, сделанные разными разработчиками. Однако такое автоматическое объединение изменений, обычно, возможно только для текстовых файлов и при условии, что изменялись разные (непересекающиеся) части этого файла. Если автоматическое объединение выполнить не удалось, система может предложить решить проблему вручную.

Примеры программных продуктов: Subversion(SVN), Microsoft Visual SourceSafe, Rational ClearCase, CVS.

2.5.2 Резервное копирование.

Резервное копирование (англ. backup) - процесс создания копии данных на носителе (жестком диске, дискете и т. д.), предназначенном для восстановления данных в оригинальном месте их расположения в случае их повреждения или разрушения.[6]

Этот метод представляет собой создание резервной копии или архива исходных файлов с дальнейшим их помещением в специализированное хранилище, которое обеспечивает долговременное сохранение данных. Часто используется для сохранения информации на домашних компьютерах или в маленьких фирмах.

Преимущества резервного копирования:

1. Сложность ПО, обеспечивающего резервное копирование, как правило, значительно ниже, чем необходимого для организации системы управления версиями.

2. Независимость внесения и сохранения изменений (конфликтные ситуации, вероятные при работе с системой управления версий принципиально невозможны при уникальности имен резервных копий).

3. Применим для сохранения не только набора файлов, но и всей системы целиком.

Недостатки резервного копирования:

1. Неэкономное использование ресурсов (дискового пространства при хранении данных, а также ресурсов компьютера при создании резервной копии).

[6] http://ru.wikipedia.org/wiki/Резервное_копирование.

2. Сложности учета изменений, внесенных разными пользователями.

3. Контроль за правами пользователей должен осуществляться сторонними средствами.

Примеры программных продуктов: архиваторы zip и rar, Acronis True Image Home, IBM Tivoli Storage Manager, Symantec Norton Ghost.

2.5.3 Хранение документов в базе данных.

Многие крупные предприятия выбрали базу данных за основу системы хранения внутреннего документооборота.

Преимущества использования базы данных:

1. Централизованное хранилище.

2. Наличие встроенного механизма резервного копирования базы данных как дополнительного уровня обеспечения долговременного хранения данных.

3. Масштабируемость.

4. Возможность написания внутреннего ПО для работы с базой данных.

5. Сохранение целостности данных за счет транзакционного метода внесения изменений.

6. Контролируются права доступа пользователей, разрешается или запрещается чтение или изменение данных, в зависимости от того, кто запрашивает это действие.

7. Быстрый поиск за счет возможности индексации данных.

Индексация - процесс обработки входящей информации для базы данных с формированием индекса.

Индекс (лат. index - список, реестр, указатель) - число (а иногда символ или набор символов) указатель места элемента в совокупности

или показатель активности, производительности, развития или изменения чего-либо.[7]

Индекс (англ. index) - объект базы данных, создаваемый с целью повышения производительности выполнения запросов. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному значению путем последовательного просмотра таблицы строка за строкой может занимать много времени. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и, таким образом, позволяет находить нужную строку по заданному значению. Ускорение работы с использованием индексов достигается в первую очередь за сччт того, что индекс имеет структуру, оптимизированную под поиск - например, балансированного дерева.[8]

Недостатки использования базы данных:

1. Требуется разработка и внедрение самой базы данных.
2. Сложности учета изменений, внесенных разными пользователями.
3. Неэкономное использование дискового пространства за счет хранения всех версий документа целиком, а так же связанное с поддержкой структуры самой базы данных.

Примеры программных продуктов: MicroSoft SQL Server, Oracle, Postgres, MySQL.

2.6 Выводы.

Для создания репозитория QSAR требуется построить ER-модель процесса поиска зависимостей "структура-свойство", на основе которой существующие модели можно будет разбить на общие части. На основе разбиения следует определить наиболее соответствующую методику сохранения данных и разработать на ее основе механизм репозитория.

[7] http://ru.wikipedia.org/wiki/Индекс
[8] http://ru.wikipedia.org/wiki/Индекс_(базы_данных)

3 Методика решения задачи.

3.1 Описание алгоритма решения.

Разобьем данные, которые необходимо хранить, на метаданные и собственно сами данные.

3.1.1 Способ хранения метаданных.

Метаданные, в общем случае[9]

1. информация о данных.
2. информация об информации. Пример: Имя автора правки в тексте. Этот термин в широком смысле слова используется для любой информации о данных: именах таблиц, колонок в таблице в реляционных базах данных, номер версии в файле программы (т.е как информативная часть в бинарном файле) и т. п.
3. структурированные данные, представляющие собой характеристики описываемых сущностей для целей их идентификации, поиска, оценки, управления ими.
4. данные из более общей формальной системы, описывающей заданную систему данных.

В нашем случае будем придерживаться третьего варианта определения. Понятие данных будем определять исключением из всей информации, которую требуется хранить в системе, метаданных, которые уже определены.

Свойства и характерисики метаданных:

1. Исходя из постановки задачи, можно сделать предположение, что объем метаданных будет составлять малую часть от объема всей информации.
2. Также зачастую поиск необходимой информации в системе будет ограничиваться поиском по метаданным.

[9]http://ru.wikipedia.org/wiki/Метаданные

В данном случае будет оправдано использование базы данных для хранения метаданных. Структура этой базы будет построена на основе ER-модели, полученной в результате объединения ER-моделей процесса поиска зависимостей "сущность-связь"и процесса хранения результатов расчетов, сделанных на основе таких моделей.

3.1.2 Способ хранения данных.

Исходя из постановки задачи, данные будут делиться, в свою очередь, еще на 2 категории: результаты расчетов(в том числе и промежуточные) и инструменты для расчетов(скрипты).

Первые представляют собой файл значительного объема, который никогда не будет меняться, т.к. для минимизации рисков, связанных с изменением хранимой информации, будет возможно только добавление новых файлов с результатами. Для сохранения таких данных эффективно резервное копирование с возможностью архивирования. А доступ к хранилищу архивов предоставлять на основе ftp-сервера.

FTP (англ. File Transfer Protocol - протокол передачи файлов) - протокол, предназначенный для передачи файлов в компьютерных сетях. FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер.[10]

ftp-сервер - сервер, обеспечивающий доступ к информации на основе протокола FTP.

Сетевой протокол - набор правил, позволяющий осуществлять соединение и обмен данными между двумя и более включенными в сеть устройствами.[11]

Инструменты расчетов в основном представляют собой текст скрипта. Для них характерно следующее:

1. Они достаточно часто изменяются и дополняются различными пользователями.

2. Т.к. это текст, то они гораздо меньше(по объему занимаемого дискового пространства), чем результаты.

[10] http://ru.wikipedia.org/wiki/FTP
[11] http://ru.wikipedia.org/wiki/Сетевой_протокол

3. Требуется независимое хранение каждого скрипта, т.к. один и тот же скрипт может участвовать в нескольких расчетах.

Исходя из этого, для хранения скриптов следует воспользоваться системой управления версиями.

3.1.3 Организация доступа к данным.

Каждый из используемых методов хранения данных имеет свой собственный интерфейс для доступа к данным из сети internet. Но для централизации все системы вводиться web-интерфейс(в виде сайта), доступ к которому будет предоставляться на основе данных о пользователе из базы данных. Те же данные будут использоваться сайтом для связи с остальными частями системы.

3.1.4 Итоговая схема системы.

В ходе разработки было выделено четыре составных части системы:

1. База данных.
2. FTP-сервер.
3. Система управления версиями.
4. Web-интерфейс.

Общая схема взаимодействия этих частей представлена на рис.1 в Приложении.

3.2 Особенности и отличия предлагаемого решения от существующих.

Данное решение, являясь составным, позволяет использовать преимущества приведенных выше обычных методов хранения информации, устраняя большую часть их недостатков за счет разграничения зоны действия каждой из частей.

Существующие решения также могут использовать внутри себя части других. Например, система управления версиями может хранить данные в сжатом виде на сервере с настроенным резервным копированием

Предложенная система обладает следующими преимуществами, по сравнению с вариантом использования только одного из представленных методов хранения информации:

1. Данные занимают меньше места по сравнению с сохранением их в базу данных.

2. Ведется централизованная база для сохранения информации о хранимых данных, в отличие от системы управления версиями и резервного копирования данных.

3. Обеспечивается контроль за правами доступа к системе, в отличие от резервного копирования данных.

4. Возможно распределение хранимых данных на различные независимые подсистемы за счет использования ftp-сервера.

При создании сложных систем из более простых наряду с возможностью устранения части недостатков последних приобретаются недостатки, которыми ни одна их составных частей не обладала:

1. Плохая масштабируемость комплексного решения за счет жесткости межкомпонентных связей.

2. Понижение общего уровня безопасности системы(в том числе устойчивости к внешним воздействиям) за счет дополнительных узлов.

3.3 Выводы.

Из-за наличия информации, сильно отличающейся по структуре и размеру, репозиторий QSAR следует представлять как комплекс, включающий в себя в качестве составных частей все основные способы хранения информации.

4 Реализация решения.

4.1 Выбор ПО для реализации.

Главным критерием при выборе ПО в качестве составных частей системы является его бесплатность. Дополнительно, ПО должно обеспечивать нужный уровень безопасности (это требование дублирует общее требование к репозиторию).

4.1.1 Выбор сервера базы данных.

Рынок серверов баз данных сейчас представлен в основном следующим ПО:

1. MicroSoft SQL Server.
2. Oracle Database.
3. IBM DB2.
4. Postgres.
5. MySql.

Первые три разработаны крупными компаниями(MicroSoft, Oracle и IBM соответственно). Они представляют собой комплексное решение, включающее в себя помимо самого сервера базы данных приложения для ее проектирования и администрирования(например Queue Analyzer, входящий в состав MSSql). Но лицензия на данные программные продукты стоит достаточно дорого. Поэтому мы будем рассматривать в качестве кандидатов только Postgres и MySql. В сети internet это сравнение и обсуждение этого сравнения проводится достаточно часто и с разных сторон. В них часто используется следующий термин:

Система управления базами данных (СУБД) - специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных. Для создания и управления информационной системой СУБД необходима в той же степени, как для разработки программы на алгоритмическом языке необходим транслятор.[12]

[12]http://ru.wikipedia.org/wiki/СУБД

Вот различные стороны сравнения, на результаты которых я буду опираться при выборе:

1. "Мы же тестируем пригодность СУБД для реализации конкретного проекта, а именно - блогохостинга с большим количеством активных пользователей, которые не только ведут свои дневники, пишут и читают комментарии, но и читают записи своих друзей.".[13]

2. "Мы (разработчики MySQL) и разработчики PostgreSQL постоянно заняты улучшением наших СУБД, поэтому обе системы являются серьезными альтернативами любым коммерческим СУБД. Приведенное ниже сравнение проводилось в MySQL AB. Мы старались быть как можно более точными и объективными, однако, зная MySQL наизусть, мы не можем похвастаться таким же знанием возможностей PostgreSQL, поэтому в чем-то могли и ошибиться".[14]

3. "Тестировалось железо, но получившиеся выводы и графики очень интересны и показательны".[15][16] [17] [18]

Обобщим результаты этих сравнений и обсуждений[19]. MySql имеет следующие преимущества по сравнению с Postgres:

1. Более высокая производительность обработки небольших запросов, в которых не производится большое кол-во соединений таблиц.

2. Более высокая производительность при небольшом количестве запросов за счет оптимизации работы на одноядерных системах(большое количество запросов требует использования всех доступных ядер).

3. Реализован полнотекстовый поиск.

4. Реализован кэш запросов, что увеличивает скорость обработки часто повторяющихся запросов.

[13] http://www.samag.ru/art/07.2007/07.2007_02.html
[14] http://program.rin.ru/razdel/html/490-2.html
[15] http://www.linux.org.ru/view-message.jsp?msgid=1677254
[16] http://tweakers.net/reviews/657
[17] http://tweakers.net/reviews/657/6
[18] http://tweakers.net/reviews/646/9
[19] http://xpoint.ru/forums/computers/dbms/misc/thread/41185.xhtml

5. Более высокая производительность в случае использования с web-сайтом и их расположении на одной машине.

6. Обновление MySQL проходит совершенно "безболезненно". При модернизации MySQL нет нужды в копировании/восстановлении данных, что приходится делать при установке большинства обновлений PostgreSQL.

Postgres имеет следующие преимущества по сравнению с MySql:

1. Более простая возможность использования хранимых процедур.

2. Поддержка большого количества языков для написания функций(хотя в последнее время это преимущество почти исчезло).

3. Более высокая производительность при обработке большого количества запросов.

4. Хорошо отлаженный механизм транзакций.

5. В некоторых случаях PostgreSQL оказывается ближе к ANSI SQL

6. Почти линейно масштабируется до 16 процессоров.

В нашем случае в систему будет включен web-сайт, в основном будет небольшой объем запросов, но часто будет производиться поиск по тексту(по описанию молекулы или сripта например). Планируется устанавливать систему на рабочую станцию, из чего следует, что поддержка большого количества ядер не обязательна.

В итоге видно, что следует использовать MySql в качестве сервера базы данных.

4.1.2 Выбор системы управления версиями.

Из предложенных в пункте 2.5.1 примеров такого ПО только SVN и CVS являются бесплатными. Несмотря на то, что остальные два примера используются в крупных компаниях и хорошо зарекомендовали себя, их использование в нашем случае невозможно из-за высокой цены. Остается пойти по пути фирм среднего размера, чьи финансы не позволяют приобрести лидеров в этом секторе, выбрать между SVN и CVS.

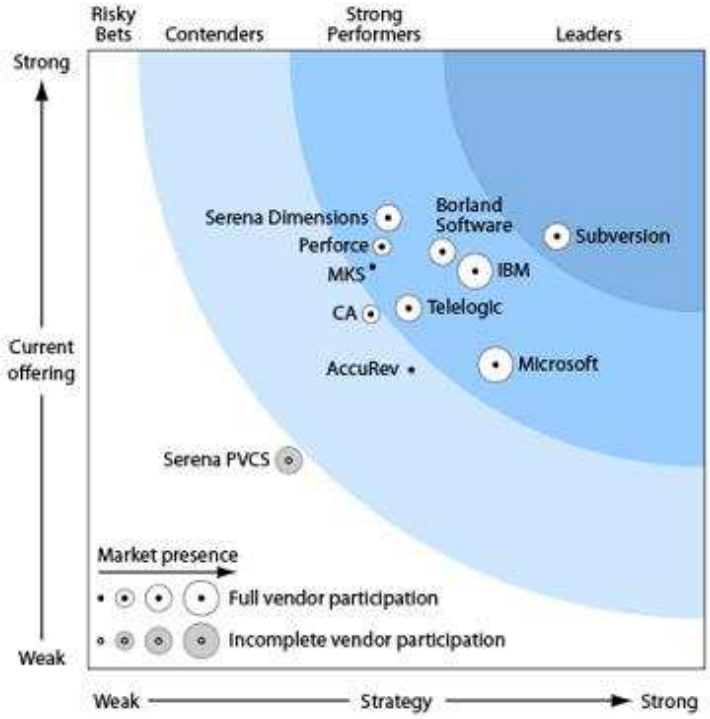
SVN является потомком CVS, но т.к. в истории не всегда следующие поколение было лучше предыдущего, то и в ПО стоит детально сравнить их между собой.

CVS (Concurrent Versions System, "Система Конкурирующих Версий") - программный продукт, относящийся к разряду систем управления версиями (англ. version control system). Хранит историю изменений определчнного набора файлов, как правило исходного кода программного обеспечения, и облегчает совместную работу группы людей (часто - программистов) над одним проектом.[20]

Subversion - свободная централизованная система управления версиями. Subversion разработана специально для замены устаревшей системы CVS, распространенной открытой системы управления версиями. Subversion обладает всеми основными функциями CVS (хотя некоторые из них выполняет другими способами) и свободна от ряда еч недостатков.[21]

В 2007 году независимая компания Forrester Research, сравнивая преимущества и недостатки различных систем, оценила Subversion как "единоличного лидера в категории Standalone Software Configuration Management (SCM) и сильного участника в категории Software Configuration and Change Management (SCCM)"[22] Результаты исследования в виде диаграммы:

[20]http://ru.wikipedia.org/wiki/CVS
[21]http://ru.wikipedia.org/wiki/Subversion
[22]http://www.collab.net/forrester_wave_report/index.html



Проанализируем результаты этого исследования, а затем сравним SVN и CVS между собой(т.к. CVS в нем не участвовала). Сравнение проводилось на основе следующих критериев(приведу только интересующие нас, т.к. например, поддержка различных платформ не важна для нас):

- Производительность.
- Безопасность.
- Легкость администрирования(в том числе простота развертки).
- Цена.

По каждому критерию выставлялась оценка от 0 до 5, чем больше - тем лучше. Приведем результаты в виде таблицы.

Критерий	IBM	MicroSoft	Subversion
Производительность	3.02	3.14	2.52
Безопасность	2.93	1.49	2.43
Администрирование	1.70	2.30	5.00
Цена	1.80	4.50	5.00

Не все критерии одинаково важны в нашем случае, поэтому проведем взвешенное сравнение, где цена(как самый важный критерий) будет составлять 50%, администрирование и безопасность(второстепенные критерии) по 20%, а производительность(в виде того, что первое время система будет установлена на обычном десктопе) в 10%.

Итоговый рейтинг сравниваемых программных продуктов: IBM - 2.13, MicroSoft - 3.32, Subverion - 4.24. Они слегка отличаются от общих результатов исследования, но лидер не изменился.

Наши предположения по поводу выбора между SVN и остальными продуктами оправдались. Но стоит все-таки сравнить SVN и CVS.

Преимуществами SVN называют[23]:

- Атомарность внесения изменений.
- Поддержка переименования файлов.
- Поддержка символьных ссылок.

Символьная ссылка (также симлинк от англ. Symbolic link, символическая ссылка) - специальный файл в UNIX-подобных операционных системах, для которого в файловой системе не хранится никакой информации, кроме одной текстовой строки. Эта строка трактуется как путь к файлу, который должен быть открыт при попытке обратиться к данной ссылке. Символьная ссылка занимает ровно столько места на файловой системе, сколько требуется для записи ее содержимого (нормальный файл занимает как минимум один блок раздела)[24].

Нет никаких предпосылок, чтобы использовать родителя вместо потомка, поэтому за основу системы управления версиями будет взят SVN.

[23]http://en.wikipedia.org/wiki/Comparison_of_revision_control_software
[24]http://ru.wikipedia.org/wiki/Симлинк

4.2 Описание построенной модели сущность-связь.

Исходя из описания основных этапов решения QSAR-задачи(п.1.2.2) мы будем работать только первым этапом - описанием молекулярного графа, который представляет собой перевод описания молекулы(молекулярного графа) в вид, удобный для поиска зависимости(строку матрицы "структура-признак").

4.2.1 Модель сущность-связь процесса поиска зависимостей типа "структура-свойство".

Процесс поиска зависимостей типа "структура-свойство"описан в п.1.2.3. На основе этого описания мы и будем строить ER-модель. Требуется определиться с уровнем детализации, на котором будет создана ER-модель.

В рамках ER-модели мы будем использовать 2 понятия:

1. Сущность - отражение объекта или процесса(действия), имеющего место в реальном мире, на модели.

2. Связь - отражение взаимодействия сущностей на нашей модели(например, что объект в реальном мире инициирует какое-либо действие).

Первое предложенное решение состояло в следующем: требовалось взять наиболее сложное(по количеству этапов) решении задачи QSAR. Построить максимально детальную ER-модель, т.е. каждый этап выделить в отдельный объект. Связь между этими объектами устанавливалась в той последовательности, в которой они выполнялись в этом решении. Остальные решения накладывались на эту модель и, в случае отсутствия какого-либо этапа, просто опускали его. Т.к. каждый этап представлял собой не только расчетный скрипт, но и набор параметров, с которым этот скрипт был запущен(например порог для вычислений), для удобства использования скриптов с разными параметрами, было решено разбить объект, соответствующий этапу, две сущности: скрипт и файл конфигурации скрипта.

Файл конфигурации(скрипта) - сущность, представляющая собой совокупность всех параметров, которые принимает на вход соответствующий скрипт.

Созданная таким способом ER-модель обладала следующими преимуществами:

1. Максимальная детализация обеспечивала максимальную выгоду блочному механизму создания новых решений.

2. Точная классификация и описание каждого этапа.

Эта ER-модель обладала следующими недостатками:

1. Могли возникать этапы, которые не включал в себя базовое(на основе которого формировалась ER-модель) решение.

2. У различных решений порядок этапов мог различаться.

Оба этих случая были очень вероятны, если первый можно было устранить введением нового объекта, то второй недостаток был очень сложнопреодолим.

Исходя из этого, второе предложенное решение состояло в следующем. Для сохранения максимальной выгоды блочного механизма решено было выделить скрипт и файл конфигураций в отдельные сущности ScriptList, элементом которого являлся один скрипт, и ConfigList, элементом которого являлся один файл конфигураций. Для всего решения целиком вводилась сущность "цепочка расчетов"(или просто цепочка) - Chains.

Цепочка расчетов(цепочка) - сущность, соответствующая решению QSAR-задачи, элементом которой является конкретное решение.

Для описания одного этапа решения была введена сущность ScriptsInChains. Она содержала в себе информацию о том, какой скрипт, на каком этапе(по счету) и с каким конфигурационным файлом должен быть запущен в рамках определенного решения QSAR-задачи.

4.2.2 Модель сущность-связь для сохранения результатов.

Некоторые этапы являются очень дорогими(с точки зрения затрат времени). В случае, если у двух решений этапы до определенного моменты точно совпадают, то выгодно сохранять некоторые промежуточные результаты одного решения, для того, чтобы можно было не тратить время на их повторное получение, в случае запуска второго решения на тех же исходных данных.

Первый возникший вариант представлял собой отдельную сущность, результаты, которая связывалась с сущностью ScriptsInChains. Но в этом

случае таблица, соответствующая этой сущности в базе данных со временем стала бы очень большой, что существенно замедлило бы поиск(основную операцию, ради которой эту сущность вводят в модель) по ней. Решение этой проблемы было следующим: на основе описания п.1.2.3 были 5 основных этапов решений. Это:

- Предварительное преобразование описания молекулы. Несмотря на общий формат файла, для хранения описания молекулы, разные решения использовали разные части в описании. Например, часть старых решений не учитывала заряды на атомах.
- Поиск особых точек. В описании первого этапа решения QSAR-задачи мы отталкиваемся от существующего меченого графа. Для его создания из описания молекулы в пространстве выделяют особые точки(узлы графа).
- Классификация особых точек. Здесь выделенным особым точкам(узлам графа) присваивают соответствующие метки, а затем разделяют узлы графа на классы, исходя из присвоенных им меток.
- Построение дескрипторов. Суть этого процесса достаточно подробно описана в п.1.2.3.
- Построение матрицы "структура-признак"на основе сформированных дескрипторов.

Такое разделение позволяет бы классифицировать результаты для более быстрого поиска по ним.

Итоговая схема ER-модели процесса поиска зависимостей и промежуточных результатов расчетов представлена на рис.2 приложения А.

4.2.3 Общая ER-модель репозитория с учетом нужд web-интерфейса.

Результаты любых расчетов основываются на входных данных. Поэтому система должна содержать информацию и о них тоже. Исходя из того, что модель, как правило обрабатывает не одну молекулу(описание), а множество(выборку) в ER-модель были введены следующие 3 сущности:

1. Молекул(Molecules) элементом которой являлось описание молекулы.

2. Выборка(Selections) соответствовала выборке.

3. Для отражения информации о включении молекулы в выборку была введена сущность MoleculesInSelection.

Так же расчеты в рамках нашей QSAR-задачи всегда ограничивались поиском зависимостей только по одному свойству - обладанию некоторой химической активностью(далее будет использоваться термин активность). Для отражения этого факта были введены две сущности: ActivityList - список всех типов активностей, зарегистрированных в системе и связанных с определенными выборками, и связанная с ней Activity содержащая значение(цифровое) активности для определенной молекулы.

Схема этой части модели находится на рис.3 в приложении А. Схема связи двух частей ER-модели - рис.4 в приложении А.

Для поддержки web-интерфейса были введены еще две сущности. Users - пользователь системы. Т.к. планировалось загружать только обработанные выборки была создана связь пользователя с сущностью молекула, по которой однозначно восстанавливался автор результатов. Для удобства использования существующих скриптов в качестве частей новой цепочки была введена дополнительная сущность Files_Format_List, которая соответствовала описаниям типов файлов, которые используют скрипты в качестве входных или выходных данных.

Схема связей с этими дополнительными сущностями содержит рис.5 приложения А.

4.3 Интеграция компонентов системы.

На основе ER-модели, созданной в IBM Rational Rose Professional Data Modeler Edition v.7.0.0.0, средствами forward engineering был сформирован скрипт на языке SQL, который позволял создавать эту база данных на серверах.

forward engineering - применение методов инженерного проектирования в программировании, когда после создания чертежа(проекта), на

его основе создаются экземпляры(или основа экземпляра, костяк) конечного продукта. В нашем случае применение методов forward engineering на основе ER-модели позволят создать скрипт на языке SQL(ANSI SQL), для формирования пустой базы данных с таблицами, соответствующими сущностям, с внешними ключами, соответствующими связям между сущностями.

Первичный ключ (англ. primary key) - понятие теории реляционных баз данных, минимальное множество атрибутов, являющееся подмножеством заголовка данного отношения, составное значение которых уникально определяет кортеж отношения. На практике термин первичный ключ обозначает поле (столбец) или группу полей таблицы базы данных, значение которого (или комбинация значений которых) используется в качестве уникального идентификатора записи (строки) этой таблицы.[25]

Внешний ключ (англ. foreign key) - понятие теории реляционных баз данных. Внешним ключом называется поле таблицы, предназначенное для хранения значения первичного ключа другой таблицы с целью организации связи между этими таблицами.[26]

Полное описание базы можно прочитать в приложении С.

Для понижения уязвимости базы к SQL-injections большинство полей в таблице были максимально ограничены в размерах.

Внедрение SQL-кода (англ. SQL injection) Ч один из распространнных способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.[27]

Установка системы осуществлялась на машину под управлением ОС freebsd-7.1Beta с установленными:

- Сервером баз данных MySQL v.5.2.
- Сервером apache, сконфигурированым для поддержки сервера Subversion(исходя из инструкции по установке SVN).
- Сервером Subversion, установленным по инструкции[28] с возможностью доступа по протоколу http[29] только локально.

[25]http://ru.wikipedia.org/wiki/Первичый_ключ
[26]http://ru.wikipedia.org/wiki/Внешний_ключ
[27]http://ru.wikipedia.org/wiki/SQL_инъекция
[28]http://www.opennet.ru/tips/info/1069.shtml
[29]http://ru.wikipedia.org/wiki/http

- Сервером lftp, работающим только безопасными соединениями(зашифрованными с помощью сертификатов openssl[30]).

Скрипт был выполнен на сервере MySQL. В таблицу Users были внесены первоначальные пользователи системы. На основе данных из этой таблицы разрешался или запрещался доступ к lftp(выдавался или не выдавался сертификат безопасности x.509 для создания безопасного соединения) и web-интерфейсу(процедура авторизации пользователя для доступа на сайт). Все операции с SVN совершались от имени пользователя svn, который имел права только для работы с этим сервером.

Данные меры безопасности отвечали необходимым требованиям к репозиторию.

4.4 Описание интерфейса взаимодействия с пользователем.

Для взаимодействия с пользователем(в том числе и его авторизации на сайте) был разработан web-интерфейс, полное описание которого можно прочитать в приложении D.

Все пользователи, которым разрешался доступ в систему, были разделены на 3 категории:

1. Обычный пользователь, обладающий правами на подачу заявки на регистрацию, на изменение своих регистрационных данных, на выгрузку готовой цепочки расчетов и внесения в систему результатов расчетов,.

2. Привилегированный пользователь, обладающий, кроме прав обычного пользователя, возможностью загрузки новых скриптов, формирования новых цепочек и повышения прав обычного пользователя до привилегированного.

3. Администратор система, обладающий всеми правами привилегированного пользователя, а также возможностью подтверждать регистрацию новых пользователей и изменять уровень привилегий и регистрационных данных у не-администраторов.

[30]http://ru.wikipedia.org/wiki/Openssl

Полное описание сценариев использования для каждого типа пользователей можно прочитать в приложении С.

4.5 Выводы.

Данная реализация описанной в главе 3 методики решения задачи по построению репозитория QSAR отвечает требованиям, сформулированным в главе 2. Она обеспечивает возможность независимого использования удаленным пользователем сохраненных цепочек расчетов целиком или их частей, для реализации новых методов решения QSAR-задачи и для организации процесса расчетов по уже существующим решениям.

5 Результаты.

Был развернут тестовый репозиторий. При помощи прямого SQL запроса локально в базу данных была внесена запись первого пользователя - администратора системы. С использованием прав этого пользователя помощи web-интерфейса в репозиторий были помещены примеры рабочих скриптов на языке Matlab: main, write_parameters, filter, mgua_rounding, find_clustmin, clusters , write_clusters и get_cluster. Особенность этих скриптов состояла в том, что они не имели конфигурационных файлов. Скрипты main и mgua_roundingm имели по две версии. На осонве загруженных материалов было сформировано две цепочки:

- main(v.1) -> write_parameters -> filter -> mgua_rounding(v.1) -> filter -> find_clustmin -> clusters -> write_clusters -> get_cluster -> mgua_rounding(v.1)

- main(v.2) -> write_parameters -> filter -> mgua_rounding(v.2) -> filter -> find_clustmin -> clusters -> write_clusters -> get_cluster -> mgua_rounding(v.2)

После этого через web-интерфейс была подана заявка на регистрацию тестового пользователя. Администратор системы подтвердил заявку, вручную выслав сформированный пароль на указанный e-mail. В результате, в ответ на запрос тестового пользователя о выгрузке одной из этих цепочек система формировала архив(*.zip), в который входили все участвующие в цепочке скрипты(несмотря на то, что в цепочку один и тот же скрипт входил несколько раз, все скрипты были в единственном экземпляре) соответствующих версий.

В ходе этих операций был выявлен недостаток разработанного механизма формирования архива с цепочкой для выгрузки в случае использования различных версий одного и того же скрипта на разных этапах цепочки.

6 Дальнейшие пути развития системы.

На текущем этапе возможны несколько направлений дальнейшего улучшения системы.

Во-первых, не до конца проработан механизм взаимодействия с пользователем. В основу системы заложена возможность организации процесса расчетов не на основе исходных описаний молекул, а на основе сохраненных результатах совершенных ранее расчетов. Это можно реализовать в виде специфического клиента, устанавливаемого на клиентской машине. Так же для этого потребуется доработка текущего механизм формирования итогового архива, например, при помощи помещения каждого этапа в отдельную директорию. Это позволило бы устранить проблему, выявленную на этапе тестирования и описанную в главе 5.

Во-вторых, возможно создание интерфейса, позволяющего производить скрининг открытых баз данных химических соединений для отбора перспективных молекул в плане той или иной активности(например, скрининг баз, содержащих описание потенциальных антиканцерогенных соединений, на предмет обнаружения лекарств от рака).

Скрининг (от англ. screening кпросеиваниењ) Ч общее название методов специальных проверок, обследований, применяемых в медицине, биохимии, а также в бизнесе и т. п.[31]

В-третьих, в случае дальнейшего развития системы может появиться необходимость ее расширения, в то время как текущая организация(несколько составных частей, расположенных на одной машине, могут препятствовать развертке системы на кластере) будет создавать для этого некоторые сложности.

В-четвертых, в случае установки системы на мощную вычислительную технику, возможна разработка методики организации процесса вычислений, на основе предоставленных пользователем данных, на нашем сервере(пользователь будет иметь возможность загрузить на сервер описание молекул и запустить существующие цепочки расчетов прямо на нашем сервере).

[31] http://ru.wikipedia.org/wiki/Скрининг

7 Библиография.

1. Журавлев Ю.И., Рязанов В.В., Сенько О.В. "Распознавание". Математические методы. Программная система. Практические применения. - М.: ФАЗИС, 2006.

2. Скворцова М.И., Станкевич И.В., Палюлин В.А., Зефиров Н.С. Концепция молекулярного подобия и ее использование для прогнозирования свойств химических соединений. (в публикации)

3. Сошникова Л.А., Тамашевич В.Н., Уебе Г., Шефер М. Многомерный статистический анализ в экономике.

4. Журавлев Ю. И. Избранные научные труды. - М.: Магистр, 1998.

5. Журавлев Ю. И. Об алгебраическом подходе к решению задач распознавания и классификации. - М.: Наука, 1978, вып. 33.

6. Ryazanov V.V., Sen'ko O.V., Zhuravlev Yu.I. Mathematical Methods for Pattern Recognition: Logical, Optimization, Algebraic Approaches. - Proceedings of the 14th International Conference on Pattern Recognition. Brisbane, Australia, August 1998.

7. Дмитриев А. Н., Журавлев Ю. И., Рязанов В.В., Чернявский Г.М. Разработка системы оперативного прогнозирования сельскохозяйственного урожая на территории РФ. - В кн.: Доклады 10-ой Всероссийской конференции "Математические методы распознавания образов (ММРО-10) Москва, 2001.

8. Ryazanov V.V. Recognition Algorithms Based on Local Optimality Criteria. - Pattern Recognition and Image Analysis, 1994, vol. 4, no. 2.

9. Ryazanov V.V. About some approach for automatic knowledge extraction from precedent data. - Proceedings of the 7th International Conference "Pattern Recognition and Image Processing", Minsk, May 21-23, 2003, vol. 2.

10. Ryazanov V.V., Vorotnichkin V.A. Discrete Approach for Automatic Knowledge Extraction from Precedent Large-scale Data, and Classification. - Proceedings of the 16th International Conference on Pattern Recognition. Quebec, Canada, 11-15 August 2002.

11. Кузнецов В.А., Сенько О.В., Кузнецова А.В. и др. Распознавание нечетких систем по методу статистически взвешенных синдромов и его применение для иммуногематологической нормы и хронической патологии. - Химическая физика, 1996, 15(1).

12. Сенько О.В. Использование процедуры взвешенного голосования по системе базовых множеств в задачах прогнозирования. - Журнал вычислительной математики и математической физики, 1995, 35(10).

13. Kuznetsova A.V., Sen'ko O.V., Matchak G.N., Vakhotsky V.V., Zabotina T.N., Korotkova O.V. The prognosis of Survivance in Solid Tumor Patients Based on Optimal Partitions of Immunological Parameters Ranges. - Journal Theoretical Medicine, 2000, vol. 2.

14. Ryazanov V.V., Sen'ko O.V., Zhuravlev Yu.I. Methods of Recognition and Prediction Based on Voting Procedures. - Pattern Recognition and Image Analysis, 1999, vol. 9, no. 4.

15. Дуда Р., Харт П. Распознавание образов и анализ сцен. - М.:Мир, 1976.

16. Обухов А.С., Рязанов В.В. Применение релаксационных алгоритмовпри оптимизации линейных решающих правил. - В кн.: Доклады 10-ой Всероссийской конференции "Математические методы распознавания образов (ММРО-10) Москва, 2001.

17. Уоссермен Ф. Нейрокомпьютерная техника. - М.:Мир, 1992.

18. Christopher J.C. Burges A Tutorial on Support Vector machines for Pattern Recognition. - Data Mining and Knowledge Discovery 2, 1998.

19. Kuncheva L.I. Combining pattern classifiers Methods and Algorithms. - Wiley, 2004.

20. Vetrov D.P. On the Stability of the Pattern Recognition Algorithms. - Pattern Recognition and Image Analysis, 2003, vol. 13, no. 3.

21. Рязанов В.В. О построении оптимальных алгоритмов распознавания и таксономии (классификации) при решении прикладных задач. - В кн.: Распознавание, классификация, прогноз: Математические методы и их применение. - М.: Наука, 1998, вып. 1.

22. O.Ivanciuc, S.L.Taraviras, D.Cabrol-Bass - Journal of Chemical Information and Computer Sciences, 40

23. Кумсков М.И., Смоленский Е.А., Пономарева Л.А., Митюшев Д.Ф., Зефиров Н.С. Системы структурных дескрипторов для решения задач "структура-свойство". - Доклады Академии Наук, 1994, 336.

24. В. Магнусон, Д. Харрис, С. Бейсак В кн. Химические приложения топологии и теории графов. (Ред. Р. Кинг) - М.:Мир, 1987.

25. Станкевич М.И., Станкевич И.В., Зефиров Н.С. Успехи химии, 57

A Приложение. Схемы.

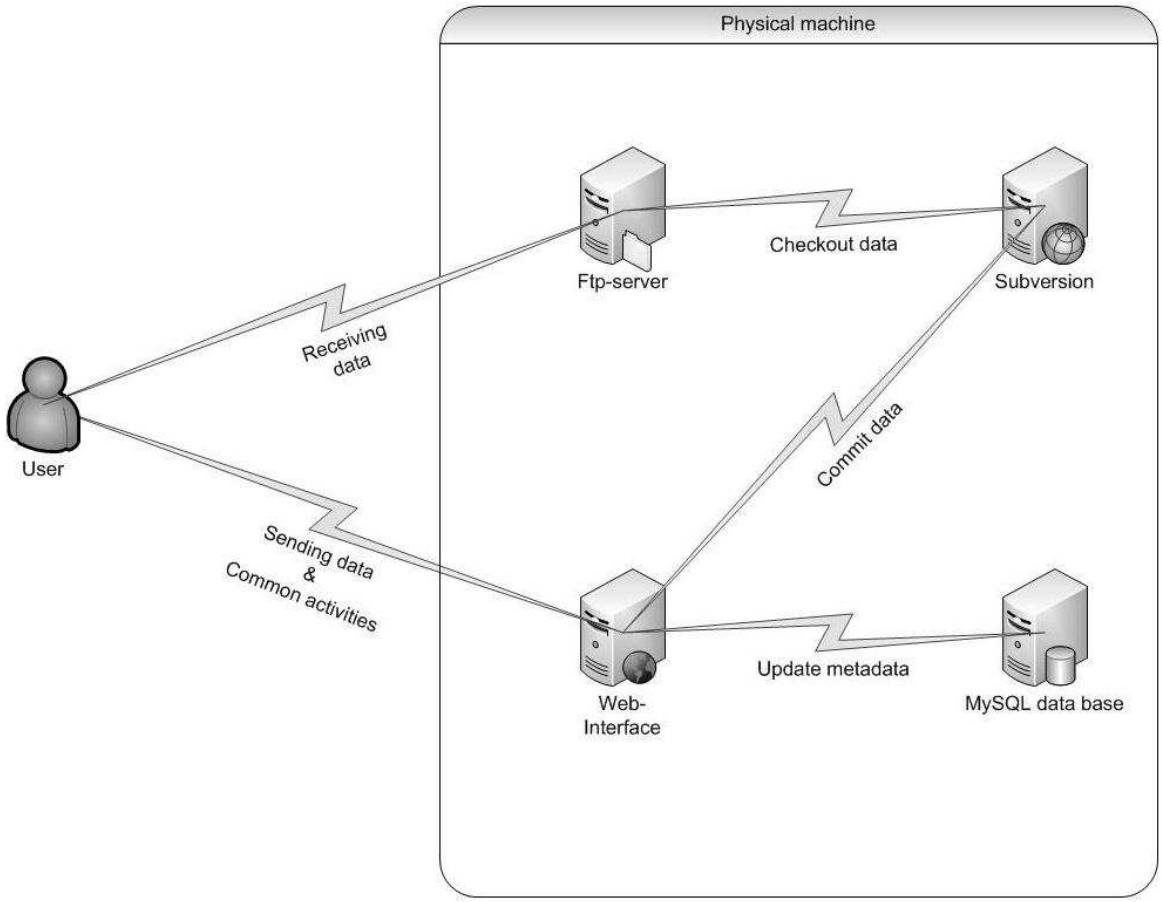


Рис.1 Схема взаимодействия частей системы.

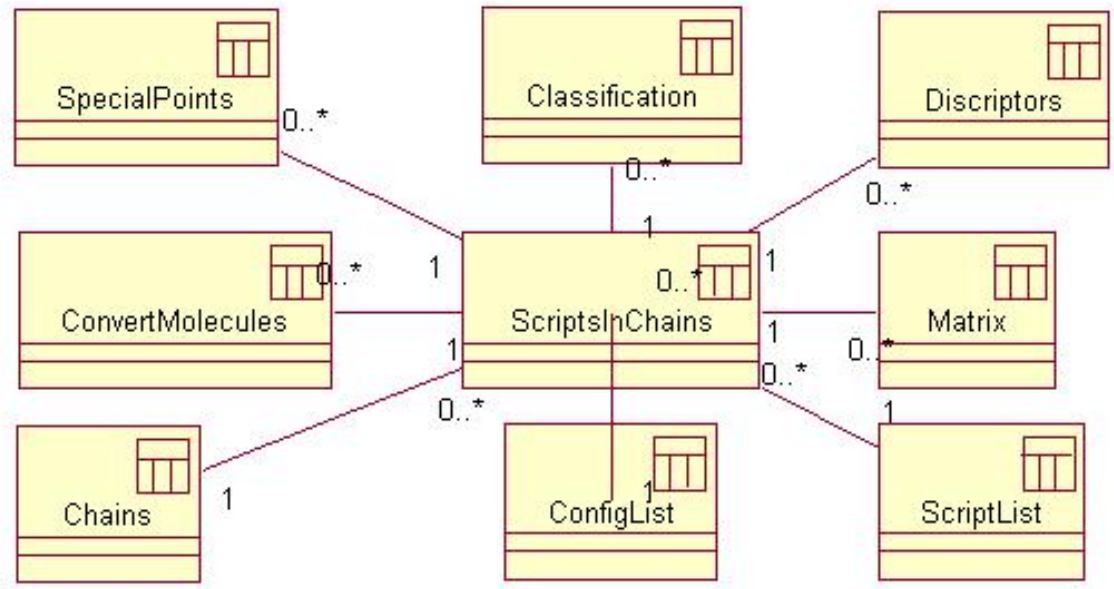


Рис.2 Схема ER-модели процесса поиска зависимостей и сохранения промежуточных результатов.

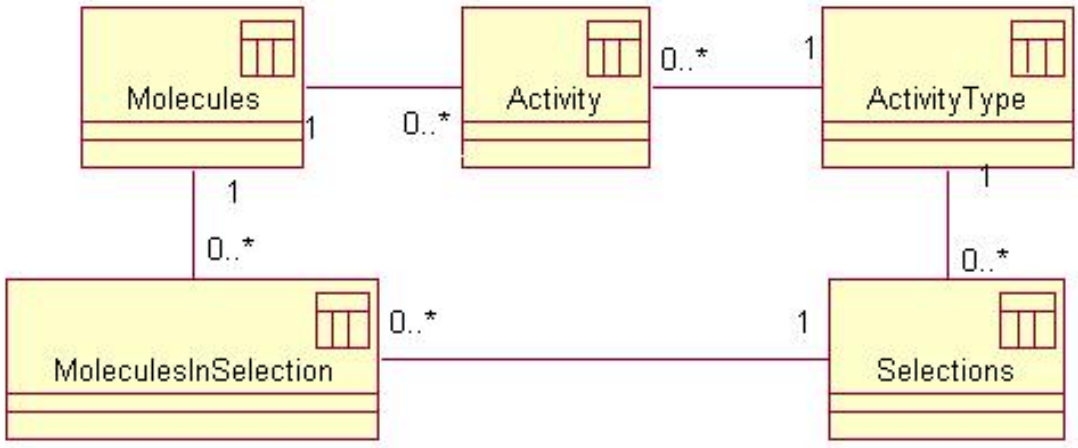


Рис.3 Схема ER-модели процесса сохранения описания молекул.

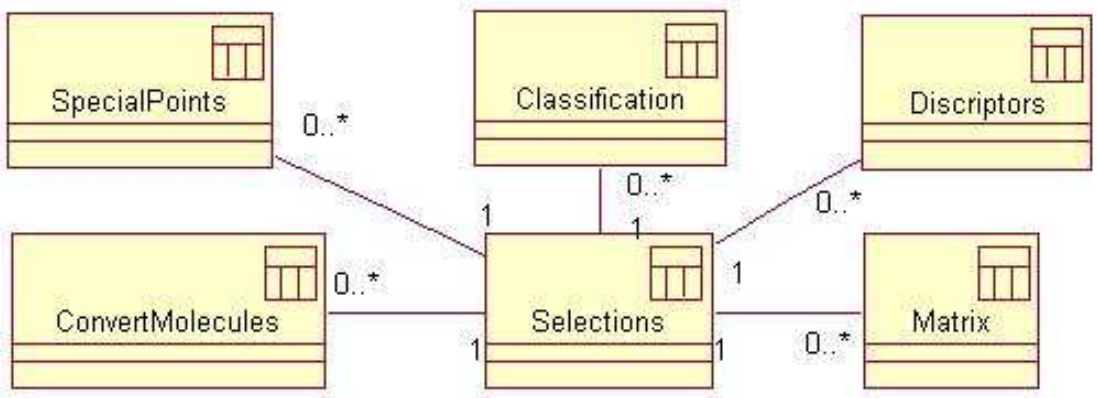


Рис.4 Схема ER-модели на месте связи двух частей.

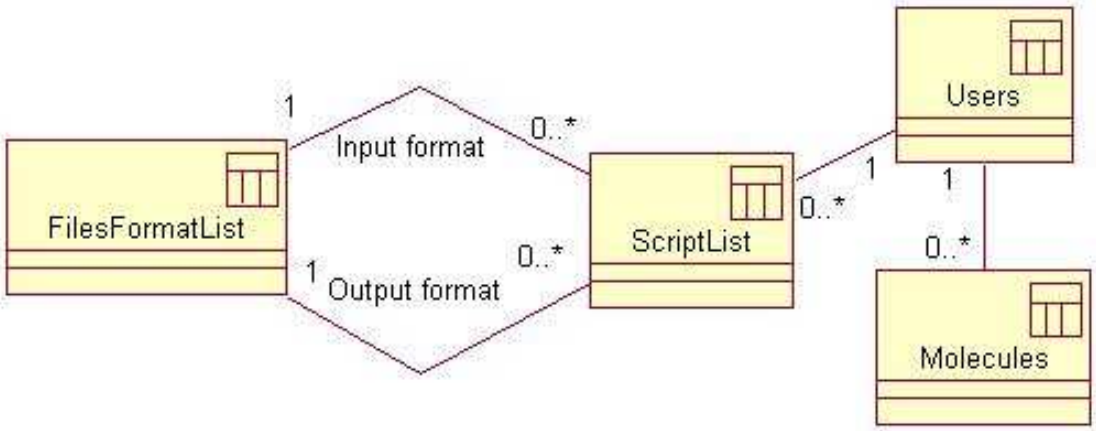


Рис.5 Схема ER-модели с дополнительными сущностями.

В Приложение. Описание базы данных.

Таблица Users

Назначение таблицы.

Предназначена для хранения информации о пользователях репозитория QSAR: имени пользователя в системе, пароля пользователя(храниться в незашифрованном виде), e-mail пользователя, на который отправляются уведомления, информации о том, что пользователь является администратором системы или привилегированным. Также является таблицей, содержащей информацию о пользователях, имеющих доступ на ftp-сервер.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер пользователя, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле UserName - имя пользователя в системе, состоящее из 40 символов(так же является именем для входа в систему).

3. Поле FirstName - фамилия пользователя для обращения, состоит из 40 символов.

4. Поле SecondName - имя пользователя для обращения, состоит из 40 символов.

5. Поле Password - пароль пользователя для доступа в систему, состоит из 40 символов.

6. Поле Email - e-mail пользователя для получения уведомлений

7. Поле Admin - флаг, является ли данный пользователь администратором системы.

8. Поле Premium - флаг, является ли данный пользователь привилегированным.

Таблица Molecules

Назначение таблицы.

Предназначена для хранения информации об описаниях молекул: номенкулатурное название молекулы, smiles для данной молекулы, путь до

файла с описанием, дату внесения описания в систему. Также таблица содержит ссылку на информацию о пользователе, внесшего это описание в систему.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер молекулы, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле User_ID - идентификационный номер пользователя, внесшего молекулу в систему(системное поле, является внешним ключем для связи с полем ID таблицы Users).

3. Поле Molecule Name - номенкулатурное название молекулы, состоит из 1000 символов.

4. Поле Path To File - полный путь до файла с описанием на сервере, состоящий из 255 символов.

5. Поле Input Date - дата внесения описания молекулы в систему.

6. Поле Smiles - содержит значение smiles данной молекулы, состоит из 20000 символов

Таблица ActivityType

Назначение таблицы.

Предназначена для хранения информации о типах активностей, расчеты которых могут быть произведены при помощи данных, содержащихся в системе. Также таблица содержит ссылку на информацию о пользователе, внесшего информацию об активности в систему.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер типа активности, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Activity Name - имя типа активности в системе, состоящее из 40 символов.

3. Поле User_ID - идентификационный номер пользователя, внесшего тип активности в систему(системное поле, является внешним ключем для связи с полем ID таблицы Users).

4. Поле Input Date - дата внесения типа активности в систему.

Таблица Activity

Назначение таблицы.

Предназначена для хранения значений активности заданного типа для молекул, чье описание было введено в систему в качестве обучающей выборки(значение активности взято из внешнего источника) или для которых расчеты были произведены при данных, содержащихся в системе. Для активностей, принадлежащих описанию молекулы из обучающей выборки, обязательно наличие данных об источнике значения активности. В случае активности, чье значение было рассчитано при помощи данных системы, в качестве информации об источнике в таблицу будет помещаться значение "QSAR".

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер активности, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Molecules_ID - идентификационный номер описания молекулы, значением активности которого является данная(системное поле, является внешним ключем для связи с полем ID таблицы Molecules).

3. Поле ActivityType_ID - идентификационный номер типа активности, которому принадлежит данное значение(системное поле, является внешним ключем для связи с полем ID таблицы ActivityType).

4. Поле Value - значение активности, действительное число.

5. Поле Info - данные об источнике информации об активности, строка, состоящая из 256 символов.

Таблица Selections

Назначение таблицы.

Предназначена для хранения информации о выборках, сформированных пользователями: ссылки на описания молекул, включенных в выборку, ссылку на тип активности, расчет значений которой будет производиться при обработке данной выборки,а также имя выборки в системе.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер выборки, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Molecules_ID - идентификационный номер молекулы, значением активности которой является данная(системное поле, является внешним ключем для связи с полем ID таблицы Molecules).

3. Поле ActivityType_ID - идентификационный номер типа активности, для расчета которой сформирована выборка(системное поле, является внешним ключем для связи с полем ID таблицы ActivityType).

4. Поле Selection Name - имя выборки в системе, состоящее из 40 символов.

5. Поле Input Date - дата формирования выборки.

Таблица MoleculesInSelections

Назначение таблицы.

Это вспомогательная таблица. Он предназначена для хранения информации о вхождении описания молекулы в ту или иную выборку.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер соответствия, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Molecules_ID - идентификационный номер молекулы(системное поле, является внешним ключем для связи с полем ID таблицы Molecules).

3. Поле Selections_ID - идентификационный номер выборки(системное поле, является внешним ключем для связи с полем ID таблицы Selections).

Таблица Chains

Назначение таблицы.

Предназначена для хранения списка цепочек расчетов, сформированных пользователями системы.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер цепочки, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Name - имя цепочки в системе, состоящий из 70 символов.

Таблицы ConvertMolecules

Назначение таблицы.

Предназначена для хранения информации о промежуточных результатах преобразования mol файла в формат, удобный для поиска особых точек: ссылки на этап цепочки расчетов, на котором этот результат был получен, ссылки на выборку, которая обрабатывалась в данном расчете.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер результата, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Path To File - полный путь до файла с результатом на сервере, состоящий из 255 символов. Может быть пустым, в случае если сохранялась только информация о промежуточных результатах.

3. Поле ScriptsInChains_ID - идентификационный номер скрипта в цепочке, с помощью которой этот результат был получен результат(системное поле, является внешним ключем для связи с полем ID таблицы ScriptsInChains).

4. Поле Selection_ID - идентификационный номер выборки, которая обрабатывалась(системное поле, является внешним ключем для связи с полем ID таблицы Selections).

5. Поле Input Date - дата внесения результатов в систему.

Таблицы SpecialPoints

Назначение таблицы.

Предназначена для хранения информации о промежуточных результатах поиска особых точек: ссылки на этап цепочки расчетов, на котором этот результат был получен, ссылки на выборку, которая обрабатывалась в данном расчете.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер результата, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Path To File - полный путь до файла с результатом на сервере, состоящий из 255 символов. Может быть пустым, в случае если сохранялась только информация о промежуточных результатах.

3. Поле ScriptsInChains_ID - идентификационный номер скрипта в цепочке, с помощью которой этот результат был получен результат(системное поле, является внешним ключем для связи с полем ID таблицы ScriptsInChains).

4. Поле Selection_ID - идентификационный номер выборки, которая обрабатывалась(системное поле, является внешним ключем для связи с полем ID таблицы Selections).

5. Поле Input Date - дата внесения результатов в систему.

Таблицы Classification

Назначение таблицы.

Предназначена для хранения информации о промежуточных результатах маркировки и кластеризации особых точек: ссылки на этап цепочки расчетов, на котором этот результат был получен, ссылки на выборку, которая обрабатывалась в данном расчете..

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер результата, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Path To File - полный путь до файла с результатом на сервере, состоящий из 255 символов. Может быть пустым, в случае если сохранялась только информация о промежуточных результатах.

3. Поле ScriptsInChains_ID - идентификационный номер скрипта в цепочке, с помощью которой этот результат был получен результат(системное поле, является внешним ключем для связи с полем ID таблицы ScriptsInChains).

4. Поле Selection_ID - идентификационный номер выборки, которая обрабатывалась(системное поле, является внешним ключем для связи с полем ID таблицы Selections).

5. Поле Input Date - дата внесения результатов в систему.

Таблицы Discriptors

Назначение таблицы.

Предназначена для хранения информации о промежуточных результатах формирования списка дескрипторов: ссылки на этап цепочки расчетов, на котором этот результат был получен, ссылки на выборку, которая обрабатывалась в данном расчете.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер результата, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Path To File - полный путь до файла с результатом на сервере, состоящий из 255 символов. Может быть пустым, в случае если сохранялась только информация о промежуточных результатах.

3. Поле ScriptsInChains_ID - идентификационный номер скрипта в цепочке, с помощью которой этот результат был получен результат(системное поле, является внешним ключем для связи с полем ID таблицы ScriptsInChains).

4. Поле Selection_ID - идентификационный номер выборки, которая обрабатывалась(системное поле, является внешним ключем для связи с полем ID таблицы Selections).

5. Поле Input Date - дата внесения результатов в систему.

Таблицы Matrix

Назначение таблицы.

Предназначена для хранения информации о промежуточных результатах формирования матрицы из дескрипторов: ссылки на этап цепочки расчетов, на котором этот результат был получен, ссылки на выборку, которая обрабатывалась в данном расчете.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер результата, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Path To File - полный путь до файла с результатом на сервере, состоящий из 255 символов. Может быть пустым, в случае если сохранялась только информация о промежуточных результатах.

3. Поле ScriptsInChains_ID - идентификационный номер скрипта в цепочке, с помощью которой этот результат был получен результат(системное поле, является внешним ключем для связи с полем ID таблицы ScriptsInChains).

4. Поле Selection_ID - идентификационный номер выборки, которая обрабатывалась(системное поле, является внешним ключем для связи с полем ID таблицы Selections).

5. Поле Input Date - дата внесения результатов в систему.

Таблица FilesFormatList

Назначение таблицы.

Предназначена для хранения информации о типах файлов, используемых в расчетах: условное имя для данного типа файлов и шаблон, под который подпадает имя файлов данного типа.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер типа файла, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Name - имя типа файла в системе, состоящий из 25 символов.

3. Поле Templates - шаблон названия файла, состоящий из 40 символов.

Таблица ScriptList

Назначение таблицы.

Предназначена для хранения информации о скриптах, внесенных в систему, с помощью которых совершаются расчеты на тех или иных этапах цепочки: имя скрипта, имя для выгрузки из репозитория, версия(в рамках скриптов с одинаковым названием), ссылки на входные и выходные типы файлов, ссылки на пользователя, внешего скрипт в систему. Также содержит информацию о том, является ли данный скрипт исполнимым под OS Linux.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер скрипта, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Name - имя скрипта в системе, состоящий из 25 символов.

3. Поле SVN Name - имя для загрузки скрипта из системы контроля версий, состоящий из 255 символов.

4. Поле Version - версия скрипта, существующая в системе контроля версий, целое положительное число.

5. Поле Input File Format_ID - формат файлов с входными данными(системное поле, является внешним ключем для связи с полем ID таблицы FilesFormatList).

6. Поле Output File Format_ID - формат файлов с выходными данными(системное поле, является внешним ключем для связи с полем ID таблицы FilesFormatList).

7. Поле Linux Script - флаг, обозначающий возможность выполнения скрипта под ОС linux.

8. Поле Input Date - дата внесения скрипта в систему.

9. Поле User_ID - идентификационный номер пользователя, внесшего скрипт в систему(системное поле, является внешним ключем для связи с полем ID таблицы Users).

Таблица ConfigList

Назначение таблицы.

Предназначена для хранения информации о конфигурационных файлах к скриптам, с параметрами(данными) из которых осуществляется запуск этапов из цепочки расчетов.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер конфигурационного файла, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Name - имя конфигурационного файла в системе, состоящий из 40 символов.

3. Поле SVN Name - имя для загрузки конфигурационного файла из системы контроля версий, состоящий из 40 символов.

4. Поле Version - версия конфигурационного файла, существующая в системе контроля версий, целое положительное число.

Таблица Chains

Назначение таблицы.

Предназначена для хранения информации о цепочках расчетов, сформированных и зарегистрированных в системе.

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер цепочки, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле Name - имя цепочки в системе, состоящий из 40 символов.

Таблица ScriptsInChains

Назначение таблицы.

Предназначена для хранения информации о вхождении определенного расчетного скрипта в цепочку на определенном ее этапе: ссылки на скрипт, с помощью которого будет производиться расчет на данном этапе цепочки, ссылки на конфигурационный файл, с параметрами из которого скрипт будет производить расчет в рамках данной цепочки, ссылки на цепочку, в рамках которой будет производиться расчет, строку для запуска скрипта, и номер этапа расчета в цепочке .

Описание полей таблицы.

1. Поле ID - уникальный идентификационный номер соответствия, состоящий из 40 символов(системное поле, первичный ключ).

2. Поле ConfigList_ID - идентификационный номер конфигурационного файла с которым связан скрипт(системное поле, является внешним ключем для связи с полем ID таблицы ConfigList).

3. Поле Script_ID - идентификационный номер скрипта(системное поле, является внешним ключем для связи с полем ID таблицы Scripts).

4. Поле Chain_ID - идентификационный номер цепочки, к которой принадлежит скрипт(системное поле, является внешним ключем для связи с полем ID таблицы Chainы).

5. Поле Etap Count - порядок скрипта в цепочке, натуральное число.

C Приложение. Описание сценариев использования системы.

C.1 Пользователь

Регистрация нового пользователя в системе

Процесс регистрации в системе представляет собой заполнение полей на странице регистрации системы с последующим занесением информации в базу данных таблицу Users. Обязательными для заполнения являются следующие поля: имя пользователя в системе (login, уникален в рамках системы), фамилия и имя пользователя для обращений, e-mail пользователя для получения уведомлений. Возможно установление отметки о типе ОС, установленной на компьютере пользователя.

Изменение личной информации

Пользователь имеет возможность изменить свои данные, введенные при регистрации посредством формы аналогичной регистрационной. После подтверждения изменений новые данные вносятся в базу данных в таблицу Users.

Загрузка файла с описанием молекулы.

На соответствующей странице пользователю предлагается выбрать файл формата *.mol(группу файлов или файл формата sdf с последующим его разбиением на mol файлы) на компьютере пользователя, которые он хочет загрузить в систему. После формирования списка и успешной загрузки в систему в базе данных в таблице Molecules формируются записи о сохраненных описаниях молекул: уникальный идентификационный номер молекулы(UID), имя файла, содержащего описание, путь на сервере до файла с описанием, дата внесения информации в базу, ссылка на идентификационный номер пользователя, внесшего описание в систему. В случае внесения описания для обучающих выборок для каждой молекулы необходимо выбрать типы активностей, которыми она обладает, значения этих активностей, а так же источник информации об этой активности. После подтверждения данные заносятся в базу данных в таблицу Activity.

Редактирование меток для файлов с описанием.

На соответствующей странице пользователь имеет возможность установить (изменить, удалить) метку (набор меток), представляющих собой

положительное целое число для файлов с описанием, загруженных в систему. После подтверждения изменений данные заносятся в базу данных в таблицу Molecules в поле Marks как последовательные числа, соответствующие имеющейся метке, разделенные запятой.

Создание выборки.

Посредством соответствующей формы из загруженных в систему описаний молекул пользователь может сформировать выборку, отметив описания для включения в выборку (возможно формирование выборки по описаниям, обладающими определенными метками), задав имя выборки (уникальное для данного пользователя), выбрав из списка тип активности для проверки. После подтверждения формирования выборки в базу данных в таблицу Selections заносятся данные: уникальный идентификационный номер выборки, ссылка на идентификационный номер типа активности, связанной с выборкой, дата внесения информации в базу, идентификационный номер пользователя, внесшего выборку в систему. Так же делаются записи в таблице связи между описанием молекул и выборками, в которые описание включено - MoleculesInSelection.

Запуск расчета для выборки.

Пользователь имеет возможность запуска расчетов для сформированных им выборок. На соответствующей странице пользователю предлагается выбрать цепочку скриптов (соответствующую указанной для выборки активности), посредством которой он хочет произвести расчет. После подтверждения начала расчета из системы контроля версий во временную директорию выгружаются скрипты и конфигурационные файлы (согласно данным из базы), соответствующие выбранной цепочке. Туда же помещаются файлы с описанием молекул из выборки. Формируется xml для запуска управляющего скрипта. Осуществляется запуск управляющего скрипта, в процессе работы которого осуществляется логирование процесса расчета.

Сохранение результатов расчетов.

В случае успешного завершения расчетов пользователь имеет возможность сохранить результаты в базе данных. Процесс сохранения представляет собой занесение информации по логу, сформированному в процессе расчета. В таблицу ConvertMolecules вносятся данные о промежуточных результатах преобразования молекулы из формата mol в формат, удобный для поиска особых точек. В таблицу SpecialPoints вносятся данные о промежуточных результатах поиска особых точек.В таблицу

Classification вносятся данные о промежуточных результатах маркировки и кластеризации особых точек. В таблицу Discriptors вносятся данные о промежуточных результатах построения списка дескрипторов. В таблицу Matrix вносятся данные о промежуточных результатах построения матрицы. Данные о результатах представляют собой уникальный идентификационный номер, путь к файлу с результатами, идентификационный номер цепочки скриптов, с помощью которой эти результаты были получены, идентификационный номер выборки, которая обрабатывалась, дату внесения в результатов.

С.2 Привилегированный пользователь

Обладает всеми возможностями, что и пользователь, плюс следующими.

Регистрация типа активности.

На соответствующей странице привилегированный пользователь имеет возможность зарегистрировать новый тип активности. В ходе этого процесса привилегированный пользователь должен заполнить следующие обязательные поля: имя типа активности (уникально в рамках системы), ссылка на общую информацию о данном типе активности или сама информация. После подтверждения регистрации типа активности в базу данных в таблицу Type Activity заносятся следующие данные: уникальный идентификационный номер типа активности, имя типа активности, информация, дата внесения информации в базу, ссылка на идентификационный номер пользователя, внесшего тип активности в систему.

Загрузка новых скриптов (версий скриптов).

На соответствующей странице привилегированному пользователю предлагается выбрать файл для загрузки в качестве нового скрипа (новой версии существующего). Так же для загрузки скрипта в систему требуется указать имя скрипта в системе (для новой версии уже существующего необходимо выбрать имя из списка) и тип ОС, под которой этот скрипт исполняется, формат входных и выходных данных. После подтверждения загрузки происходит добавление информации о скрипте в базу данных в таблицу Scripts: уникальный идентификационный номер скрипта, имя в системе, версия, формат входных и выходных данных, тип ОС, под которой этот скрипт исполняется, дата внесения информации в базу, идентификационный номер пользователя, внесшего скрипт в

систему. Сам файл помещается в систему контроля версии под соответствующим именем и версией.

Формирование цепочки скриптов для расчетов.

Пользователь имеет возможность сформировать цепочку из существующих в системе скриптов посредством выбора из ниспадающего списка имен скриптов с указанием версии. Так же требуется задать имя для цепочки (уникально в рамках системы). После подтверждения формирования производиться проверка совпадения типа выходных данных скрипта с типом входными данными следующего в цепочке. В случае несовпадения типов данных выдается предупреждение. Привилегированный пользователь имеет возможность согласиться с предупреждением и вернуться на страницу формирования цепочки или же отклонить предупреждение и сформировать цепочку. В случае успеха проверки сразу начинается формирование цепочки - в базу данных в таблицу Chains заносятся данные: уникальный идентификационный номер цепочки, имя цепочки, дата внесения информации в базу, идентификационный номер пользователя, внесшего цепочку в систему. Так же в таблицу ScriptsInChain вноситься информация о связи цепочки и скриптов с указанием последовательности скриптов в цепочке.

С.3 Администратор

Изменение данных пользователей.

Администратор системы имеет возможность изменять данные зарегистрированных пользователей на соответствующей странице сайта.

Управление привилегиями пользователей.

Администратор системы имеет возможность изменять статус привилегированности пользователя.

D Приложение. Описание web-interface.

Список форм(страниц):

1. Главная страница.
2. Регистрация пользователя.
3. Список пользователей.
4. Загрузка скрипта в систему.
5. Выгрузка цепочки.
6. Формирование новой цепочки.
7. Форма отправки сообщения.

D.1 Описание страниц.

D.1.1 Главная страница.

Эта страница представляет собой вводную страницу проекта. Она должна включать в себя форму для входа зарегистрированных пользователей в систему, содержащую два текстовых поля, "login"и "password", кнопку "Войти", а также ссылку на страницу регистрации. По нажатию на кнопку "Войти"должен происходить поиск записи в таблице Users со значением login в поле UserName. В случае если такая запись найдена и поле "Password"в ней равно значению поля "password"формы, то вход в в систему считается успешным. В случае успешного входа пользователя в систему, должен осуществляться переход на обновленную главную страницу, где вместо формы для входа, должно отображаться приветствие вида: "Здравствуйте, <Имя пользователя> <Фамилия пользователя>". На обновленной главной странице должны быть доступны ссылки на страницы список пользователей и выгрузка цепочки. Если вход осуществен привилегированным пользвателем или администратором системы должны быть также доступны ссылки на страницы загрузка скрипта в систему и формирование новой цепочки. В случае несовпадения поля "Password"найденой записи и значения поля "password"формы должен

происходить переход на главную страницу с отображением сообщения о несовпадении пароля и формой входа в систему. В случае отсутствия записи в таблице должен происходить переход на главную страницу с отображением сообщения о несовпадении имени пользователя и формой входа в систему.

D.1.2 Регистрация пользователя.

Эта страница представляет собой форму, содержащую информацию о пользователе системы.

В случае первичной регистрации она должна содержать следующие пустые поля: login, имя, фамилия, e-mail. Также на странице должна находиться кнопка "Регистрация". Возможно также наличие переключателя, в котором пользователь сможет указать тип ОС, установленной на его компьютере(Windows/linux). По нажатию на кнопку "Регистрация"должен происходить поиск записи в таблице Users со значением login в поле UserName. В случае отсутствия записи данные из этой формы должна создаваться новая запись в базе данных в таблице Users: login записывается в поле UserName, имя - FirstName, фамилия - SecondName, e-mail - email. Поле "Password"таблицы Users должно заполняться случайно сформированным паролем из 8 символов, который должен отправлятсья по указанному пользователем e-mail'у(возможно не автоматическое отправление, а подтверждение регистрации администратором системы). В случае существования записи со значением login в поле UserName должен осуществляться переход на страницу регитсрации пользователя с сообщением о существовании пользователя с указанным и именем и предложением выбрать другое имя для регистрации в системе.

В случае перехода на данную страницу пользоваетлем, совершившим вход в систему, на странице должны отображаться пустые поля password, new password, retype password. Также дожны отображаться поля login, имя, фамилия, e-mail, данные для заполнения которых беруться из таблицы Users, полей UserName, FirstName, SecondName, email соответственно. Должна присутствовать кнопка "Изменить", по нажатию на которую должно совершаться сравнение значения поля password формы с полем "Password"таблицы Users для данного пользователя. В случае различных значений этих полей совершать переход на страницу регистрации пользователя с сохранением введенных данных и с отобра-

жением предупреждения о несовпадении пароля и предложением ввести его еще раз. В случае совпадения значений этих полей проверять значения полей new password, retype password. Если оба этих поля пусты, то данные из этой формы должны обновлять запись в базе данных в таблице Users у которой UserName совпадает именем, под которым совершен вход в систему: имя - FirstName, фамилия - SecondName, e-mail - email. Если значения этих полей несовпадают совершать переход на страницу регистрации пользователя с сохранением введенных данных и с отображением предупреждения о несовпадении нового пароля и предложением ввести его еще раз. В случае, когда оба этих поля совпадают и не пусты данные из этой формы должны обновлять запись в базе данных в таблице Users у которой UserName совпадает именем, под которым совершен вход в систему: имя - FirstName, фамилия - SecondName, e-mail - email, new password - Password.

D.1.3 Список пользователей.

Данная страница должна представлять собой таблицу, отображающую записи таблицы Users.

Если переход на данную страницу осуществлен обычным пользователем, то поле UserName должно отображаться ввиде текста, поле Premium должно отображаться ввиде флага, недоступного для изменения, поле Admin должно отображаться ввиде флага, недоступного для изменения. По нажатию на имя пользователя производиться проверка совпадения этого имени с именем, с которым осуществлен вход в систему. В случае совпадения этих имен осуществляется переход на страницу регистрации данного пользователя. Иначе, производиться переход на страницу отправки сообщений где адресатом выступает выбранный пользователь.

Если переход на данную страницу осуществлен привилегированным пользователем, то поле UserName должно отображаться ввиде текста, поле Premium должно отображаться ввиде флага, недоступного для изменения в случае если значение этого поля равно единице и доступного для изменения в случае если значение этого поля равно нулю, поле Admin должно отображаться ввиде флага, недоступного для изменения. Также должна присутсвовать кнопка "Сохранить изменения"по нажатию на которую происходит сравнение записей таблицы формы, у которых установленны флаги в графе Premium и записей таблицы Users,

поля Premium которых имеют значение 1. В случае обнаружения отличий, в базу данных в таблицу Users в поле Premium для соответствующих записей вносится значение 1. По нажатию на имя пользователя производиться проверка совпадения этого имени с именем, с которым осуществлен вход в систему. В случае совпадения этих имен осуществляется переход на страницу регистрации данного пользователя. Иначе, производиться переход на страницу отправки сообщений где адресатом выступает выбранный пользователь.

Если переход на данную страницу осуществлен администратором системы, то поле UserName должно отображаться ввиде текста, поле Premium доступного для изменения, поле Admin должно отображаться ввиде флага, недоступного для изменения в случае если значение этого поля равно единице и доступного для изменения в случае если значение этого поля равно нулю . Также должна присутсвовать кнопка "Сохранить изменения"по нажатию на которую происходит сравнение записей таблицы формы, у которых установленны флаги в графе Admin и записей таблицы Users, поля Admin которых имеют значение 1. В случае обнаружения отличий, в базу данных в таблицу Users в поле Admin для соответствующих записей вносится значение 1. Также по нажатию на кнопку "Сохранить изменения"происходит сравнение записей таблицы формы и записей таблицы Users по полю Premium. В случае отличий в базу данных в соответствующую запись вносятся изменеия: в поле Premium вносится значение 1, если присутствует флаг в поле Premium в таблице формы и 0 иначе. По нажатию на имя пользователя производиться проверка совпадения этого имени с именем, с которым осуществлен вход в систему. В случае совпадения этих имен осуществляется переход на страницу регистрации данного пользователя. Иначе, производиться переход на страницу отправки сообщений где адресатом выступает выбранный пользователь.

D.1.4 Загрузка скрипта.

На данной странице в виде таблицы должна отображаться информация о скриптах, загруженных данным пользователем(значение поле UserName таблицы Users должно совпадать с именем пользователя, с которым совершен вход в систему, а также значение поля ID таблицы Users должно совпадать со значением поля User_ID записи в таблице Script List). Необ-

ходимыми для отображения в виде текста являются следующие поля таблицы ScrtiptList: Name, Version, Input Date. Поле Linux Script должно отображаться ввиде флага, недоступного для изменения. На данной странице должна быть форма с данными о новом загружаемом скрипте. Текстовое поле "Имя скрипта"обозначающее название скрипта в системе. Флаг Linux Script, обозначающий возможность запуска данного скрипта под управлением ОС семейства Linux. Два ниспадающих списка Input file format и Output file format, соответствующие входным и выходным форматам файлов для данного скрипта, содержащие значения полей Name и Templates записей из таблицы FilesFormatList. Поле в которое посредством выбора через проводник можно внести путь к файлу на компьютере пользователя, который необходимо загрузить. Должна присутствовать кнопка "Загрузить", по нажатию на которую совершается проверка значений, внесенных в поля данной формы. В случае, если какое-то поле не заполнено, совершаем переход на страницу загрузки скрипта с сохранением внесенных пользователем данных. Если все поля заполнены, то загружаем указанный пользователем файл на сервер системы и проверяем на наличие в базе данных в таблице Script List записи с совпадающим с полем "Имя скрипта"формы, и значение User_ID которой совпадает со значением ID записи из таблицы Users, поле UserName которой совпадает с именем пользователя, с которым совершен вход в систему. Если такой записи в таблице Script List не обнаружено, тогда создаем новую запись в это таблице, где значение поля Name таблицы берется из значения "Имя скрипта"формы, в поле Version таблицы записывается значение 1. Если в таблице Script List такая запись найдена, то для каждой такой записи сравниваем загруженный пользователем файл с файлом, хранящимся под именем "SVN Name"в системе контроля версий. Если обнаружено точное совпадение, то возвращаем пользователя на страницу Загрузка скрипта с выделением в таблице формы той записи из базы данных, с которой произошло точное совпадение. Иначе создаем новую запись в это таблице, где значение поля Name таблицы берется из значения "Имя скрипта"формы, в поле Version таблицы записывается максимальное значение поля Version записей в таблице Script List, для которых поле Name совпадает со значением "Имя скрипта"из формы, + 1. В обоих случаях в качестве значении полей "Input File Format_ID"и "Output File Format_ID"берутся значения из поля ID записи в таблице FilesFormatList, для которой поля Name и Template совпадают с полями Input file format и Output file format соответственно. В

поле Linux script записи помещается значение, соответствующее значению флага Linux script формы: 0 - если флаг не установлен, 1 - иначе. В поле Input Date вносится текущая дата по времени сервера. В поле User_ID вносится значение поля ID записи из таблицы Users, значение поля UserName которой совпадает с именем пользователя, с которым совершен вход в систему.