Федеральное государственное бюджетное образовательное учреждение высшего образования Национальный исследовательский университет

«Московский Энергетический Институт»

**Курсовая работа**

По дисциплине

Программная Инженерия Часть 2

«Программная реализация алгоритма индексирования базы прецедентов»

Выполнил студент группы А-05-18

Чернов Д. С.

Научный руководитель: Варшавский П. Р.

Преподаватель: Маран М.М.

Москва – 2020

Оглавление

[Постановка задачи 3](#_Toc61109095)

[Теоретическая часть 3](#_Toc61109096)

[***Введение*** 3](#_Toc61109097)

[***Прецедент и База Прецедентов*** 3](#_Toc61109098)

[***Тип индексации Базы Прецедентов*** 4](#_Toc61109099)

[***Алгоритм*** 5](#_Toc61109100)

[Описание Базы Прецедентов 6](#_Toc61109101)

[Практическая часть 7](#_Toc61109102)

[***Библиотека prettytable*** 7](#_Toc61109103)

[***Описание программы*** 8](#_Toc61109104)

[***Пример №0 (Аномалии)*** 12](#_Toc61109105)

[***Пример №1*** 12](#_Toc61109106)

[***Пример №2*** 13](#_Toc61109107)

[***Пример №3*** 14](#_Toc61109108)

[Описание функций модуля indexing 16](#_Toc61109109)

[Код 18](#_Toc61109110)

[Список литературы 22](#_Toc61109111)

# **Постановка задачи**

Целью данной курсовой работы будет написание алгоритма индексирования базы прецедентов

# **Теоретическая часть**

## ***Введение***

**Рассуждения на основе прецедентов** (англ. case-based reasoning, CBR) в широком смысле являются методом решения новых проблем на основе уже известных решений. [1]

Существует множество рутинных задач, которые не требуют логического анализа проблемы и которые больше основаны на предыдущем опыте и памяти. По накопленному опыту мы решаем новые задача, которые очень схожи со старыми.

Например, возьмем тривиальный пример «открыть дверь в кофейню». Этот опыт человек получает с детства и в дальнейшем использует его везде, где нужно открыть дверь. Дверь может быть разная, страна может быть другой, однако мы применяем все тот же алгоритм к этому.

Сложнее же реализовать «заказать чашку кофе». К примеру, человеке в чужой стране и не знает языка. Но опять же и тут можно найти подход, основанный на уже накопленных знаниях и преодолеть языковой барьер. [2]

Но как все это можно реализовать в компьютерной модели рассуждений? Мы уже знаем, что воспоминания и приобретенный опыт не так просто свести к набору правил, но можно представить себе некоторую "библиотеку" ситуаций, встречавшихся в прошлом, которые имеют отношение к возникшей проблеме, например "репертуар" указаний шефа, или судебные решения, принятые в прошлом по аналогичным делам, или наброски архитектурных планов для сооружений аналогичного назначения и т.п. Естественно, что такая библиотека должна быть индексирована каким-то разумным способом, чтобы в массиве хранящихся описаний ситуаций можно было довольно быстро распознать аналогичную текущей. Кроме того, понадобится также и некоторый механизм, который позволит адаптировать ранее принятое решение к новой проблеме (текущей ситуации). [2]

Индексация базы прецедентов нужна для повышения эффективности самой базы. Она может ускорить поиск по базе в несколько раз, следовательно ответ будет получаться быстрее. [9]

## ***Прецедент и База Прецедентов***

Прецеденты напоминают книги тем, что содержат определенную специфическую информацию, "вставленную" в некоторый контекст. Содержимое прецедента – это знание, а контекст описывает некоторое состояние внешнего мира, в котором это знание применяется. Однако прецедент содержит знание в такой форме, которая может быть воспринята программой.

Прецедент должен представлять решение проблемы в определенном контексте и описывать то состояние мира, которое получится, если будет принято предлагаемое в нем решение. Это свойство часто можно встретить и в содержимом книг, но, опять же, разница состоит в том, что информация не представлена в форме, удобной для восприятия программой.

Хотя описания прецедентов и варьируются по размеру, они все-таки значительно уступают книгам в этом смысле. Информация в описаниях прецедентов значительно более сжата и представляется на каком-либо формальном языке. [3]

База Прецедентов — это соответственно это база данных таких прецедентов.

***Индексация Базы Прецедентов***

Индексирование — это способ сортировки нескольких записей в нескольких полях. Создание индекса в поле в таблице создает другую структуру данных, которая содержит значение поля, и указатель на запись, к которой она относится. Затем эта структура индекса сортируется, что позволяет выполнять двоичные поиски. [4]

Индексация схоже с оглавлением в книге. К примеру, у нас есть книжка по программированию на языке Python и нам нужно найти раздел, посвящённый циклу for. Мы не будем просматривать каждый лист, мы пойдем в оглавление и посмотрим на какой странице находится нужная нам информация. Так же работает и индексация в БД.

Индексация заметно ускоряет работу с большой базой данных, однако есть и свои недостатки, например что индексы так же занимают память, после добавления новой строчки приходиться пересобрать индексы. [5]

Для одного отношения может быть создано несколько индексов. Кроме того, можно создавать индекс для нескольких отношений, если они содержат одинаковые атрибуты, что позволяет ускорить выполнение операций соединения отношений

## ***Тип индексации Базы Прецедентов***

Индексирование с помощью весовых коэффициентов

Идея заключается в следующем: когда эксперт задает веса параметрам, то, по сути, мы уже получаем структуру индексированной БП. Т.е. параметры, которые имеют наибольший вес, будут являться полями, по которым будет производиться индексирование. Параметры, имеющие небольшой вес, в индексации участия не примут. Таким образом, от эксперта требуется только грамотно задать весовые коэффициенты и значение порога, которое будет определять, примет ли параметр участие в индексировании или нет. Следует отметить, что порядок полей, по которым ведется индексирование, имеет важное значение, поэтому имеет смысл отсортировать поля по степени важности, т.е. по убыванию весовых коэффициентов. [7][8]

## ***Алгоритм***

**Входные данные:**

– непустое множество названий параметров прецедентов, – непустое множество весовых коэффициентов, – пороговое значение.

**Выходные данные:**

– словарь весовых коэффициентов множество параметров, по которым будет производиться индексация.

**Шаг 1.** и переходим к следующему шагу.

**Шаг 2.** Если , то проверяем условие . Если условие выполняется, то добавить пару в словарь . В противном случае перейти к шагу 3.

**Шаг 3.** . Перейти к шагу 2.

**Шаг 4.** Полученный список является списком индексируемых параметров.

**Шаг 5.** Конец (завершение алгоритма).

# **Описание Базы Прецедентов**

В базе прецедентов содержаться описание трёх видов ирисов.

|  |  |  |
| --- | --- | --- |
| A close up of a flower  Description automatically generated with medium confidence  Рисунок : Iris-virginica | A close up of a purple flower  Description automatically generated with medium confidence  Рисунок : Iris-versicolor | A picture containing plant, flower, outdoor, purple  Description automatically generated  Рисунок : Iris-setosa |

Во все базе прецедентов имеются следующие параметры:

* id – порядковый номер
* col1 – длина наружной доли околоцветника (англ. sepal length)
* col2 – ширина наружной доли околоцветника (англ. sepal width)
* col3 – длина внутренней доли околоцветника (англ. petal length)
* col4 – ширина внутренней доли околоцветника (англ. petal width)
* col5 – название вида ириса
* sampleCode – показывает множество train и test
* qualityCode – показывает результат (хорошо/плохо)

+-----+------+------+------+------+-----------------+------------+-------------+

| id | col1 | col2 | col3 | col4 | col5 | sampleCode | qualityCode |

+-----+------+------+------+------+-----------------+------------+-------------+

| 1 | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica | train | good |

| 2 | 6.4 | 2.7 | 5.3 | 1.9 | Iris-virginica | train | good |

| 3 | 6.7 | 3.1 | 4.4 | 1.4 | Iris-versicolor | train | good |

| 4 | 6.5 | 3.0 | 5.5 | 1.8 | Iris-virginica | train | good |

| 5 | 6.7 | 3.0 | 5.0 | 1.7 | Iris-versicolor | train | good |

| 6 | 6.1 | 2.9 | 4.7 | 1.4 | Iris-versicolor | train | good |

| 7 | 5.9 | 3.0 | 4.2 | 1.5 | Iris-versicolor | train | good |

| 8 | 6.8 | 2.8 | 4.8 | 1.4 | Iris-versicolor | train | good |

| 9 | 6.4 | 3.2 | 4.5 | 1.5 | Iris-versicolor | train | good |

| 10 | 4.8 | 3.1 | 1.6 | 0.2 | Iris-setosa | train | good |

| 11 | 6.9 | 3.1 | 5.1 | 2.3 | Iris-virginica | train | good |

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

| 129 | 4.4 | 3.2 | 1.3 | 0.2 | Iris-setosa | test | |

| 130 | 5.0 | 3.3 | 1.4 | 0.2 | Iris-setosa | test | |

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

| 136 | 6.0 | 2.7 | 5.1 | 1.6 | Iris-versicolor | test | |

| 137 | 5.4 | 3.0 | 4.5 | 1.5 | Iris-versicolor | test | |

| 138 | 5.1 | 2.5 | 3.0 | 1.1 | Iris-versicolor | test | |

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

| 148 | 7.9 | 3.8 | 6.4 | 2.0 | Iris-virginica | test | |

| 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica | test | |

| 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica | test | |

+-----+------+------+------+------+-----------------+------------+-------------+

Так же помимо самой базы есть база с технической информацией, мы будем пользоваться весом каждой колонке, на основе этих весов и будет проиндексирована база прецедентов

+----+----------+----------+--------+-------------------+-------------+-------------+-----+

| id | Параметр | Описание | Тип | Единицы\_измерения | Диапазон\_от | Диапазон\_до | Вес |

+----+----------+----------+--------+-------------------+-------------+-------------+-----+

| 1 | col1 | | DOUBLE | | 4.3 | 7.9 | 0.1 |

| 2 | col2 | | DOUBLE | | 2.0 | 4.4 | 0.2 |

| 3 | col3 | | DOUBLE | | 1.0 | 6.9 | 0.1 |

| 4 | col4 | | DOUBLE | | 0.1 | 2.5 | 0.1 |

+----+----------+----------+--------+-------------------+-------------+-------------+-----+

# **Практическая часть**

Для начала посмотрим индексы нашей базы, для этого воспользуемся командой

SHOW INDEX FROM [table\_name]:

+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment | Visible | Expression |

+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| test | 0 | PRIMARY | 1 | id | A | 150 | NULL | NULL | | BTREE | | | YES | NULL |

+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

Как видим, тут есть только PRIMARY индекс, а он по умолчанию есть у всех создаваемых таблиц. Наша задача написать алгоритм, который добавит нужные индексы.

Так же стоит отметить, что индексация будет вестись по все базе прецедентов, т.е. как train так и test. Это поможет еще больше ускорить поиск в дальнейшем, хотя может уходить много времени на переиндексацию.

## ***Библиотека prettytable***

Так как работа происходит с таблицами, для того чтобы выводить таблицы на экран, удобно использовать библиотеку prettytable.

Для начала нужно импортировать PrettyTable так нам кроме этого ничего больше не понадобится.

from prettytable import PrettyTable

Создаем таблицу

table = PrettyTable()

Дальше можно заполнить заголовки, если этого не сделать они заполняться автоматически как «Field 1», «Feild 2» и так далее. Нужно table\_db.field\_names передать массив строк, важно чтобы количество заголовков совпадало с количеством потом передающих в таблицу столбцов.

table.field\_names = ["City name", "Area", "Population", "Annual Rainfall"]

Чтобы заполнить таблицу строчками нужно взывать .add\_row и передать как аргумент массив значений, по размеру который совпадает с массивом заголовков

table.add\_row(["Adelaide", 1295, 1158259, 600.5])  
table.add\_row(["Brisbane", 5905, 1857594, 1146.4])  
table.add\_row(["Darwin", 112, 120900, 1714.7])  
table.add\_row(["Hobart", 1357, 205556, 619.5])  
table.add\_row(["Sydney", 2058, 4336374, 1214.8])  
table.add\_row(["Melbourne", 1566, 3806092, 646.9])  
table.add\_row(["Perth", 5386, 1554769, 869.4])

В итоге у нас получилась вот такая таблица:

+-----------+------+------------+-----------------+

| City name | Area | Population | Annual Rainfall |

+-----------+------+------------+-----------------+

| Adelaide | 1295 | 1158259 | 600.5 |

| Brisbane | 5905 | 1857594 | 1146.4 |

| Darwin | 112 | 120900 | 1714.7 |

| Hobart | 1357 | 205556 | 619.5 |

| Sydney | 2058 | 4336374 | 1214.8 |

| Melbourne | 1566 | 3806092 | 646.9 |

| Perth | 5386 | 1554769 | 869.4 |

+-----------+------+------------+-----------------+

Эта же таблица без заголовков:

+-----------+---------+---------+---------+

| Field 1 | Field 2 | Field 3 | Field 4 |

+-----------+---------+---------+---------+

| Adelaide | 1295 | 1158259 | 600.5 |

| Brisbane | 5905 | 1857594 | 1146.4 |

| Darwin | 112 | 120900 | 1714.7 |

| Hobart | 1357 | 205556 | 619.5 |

| Sydney | 2058 | 4336374 | 1214.8 |

| Melbourne | 1566 | 3806092 | 646.9 |

| Perth | 5386 | 1554769 | 869.4 |

+-----------+---------+---------+---------+

## ***Описание программы***

Запускаем программу и на рис.4 показано как появляется таблица, где на просят выбрать базу прецедентов для индексации. Выбираем таблицу 1.

Shape, rectangle

Description automatically generated

Рисунок 4

На рис.5 мы видим, что программа сообщила нам имя выбранной таблицы, и прости ввести предельное значение веса, по которому будут выбраны колонки для индексации. Нужно отметить, что диапазон допустимых значений предельного значение от 0.1 до 1. Вводим значение 0.2

Shape, rectangle

Description automatically generated

Рисунок 5

На рис.6, нужно ввести согласие на создание индекса, допустимы буквы (‘Y’, ‘y’, ‘N’, n‘’). При вводе другого символа сообщение повторится. Оно будет повторяться до тех пор, пока пользователь не введет корректный символ.

Shape

Description automatically generated

Рисунок 6

Как видно на рис.7 сначала была введена русская буква ‘н’, и сообщение повторилось, а затем уже нужная нам латинская буква ‘y’.

Так же система сообщила что индекс был успешно создан, об этом на говорит сообщение «Indexing has been successfully created!» и вывелась таблица всех индексов, которая выводиться всегда по умолчанию, чтобы можно было посмотреть какие индексы были и какие были созданы.

Индекс, который создает таблица всегда будет называться как база прецедентов плюс, без пробелов, слово Index.

Затем мы можем распечатать больше информации, а именно: саму базу прецедентов и техническое описание к ней.

A picture containing text

Description automatically generated

Рисунок 7

На рис.8-9 показана больше информации. Так как пользователь подтвердил порос, нажав ‘y’. Если бы была нажата ‘n’, то программа бы просто завершилась.

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок

## 

A picture containing text, electronics, computer

Description automatically generated

Рисунок 9

Теперь попробуем выбрать таблицу и ввести этот же вес. На рис.10 показано что программа говорит, что индекс уже был создан, и что если мы хотим его пересоздать, то его нужно удалить. Нажимаем ‘y’

Shape

Description automatically generated

Рисунок 10

На рис.11 показано что программа успешно удалила индекс и вывела соответствующее сообщение на экран «Indexing has been successfully deleted!»

Shape

Description automatically generated

Рисунок 11

Дальше все продолжается так же как и раньше.

## ***Пример №0 (Аномалии)***

Возьмем следующие параметры:

Text

Description automatically generated

Рисунок 12: пример работы программы

**Вывод:** Программа не создала индекс и вывела предупреждение.

## ***Пример №1***

Возьмем следующие параметры:

Таблица описания БП:

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

| id | Параметр | Описание | Тип | Единицы\_измерения | Диапазон\_от | Диапазон\_до | Вес |

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

| 1 | col1 | | DOUBLE | | 4.3 | 7.9 | 0.1 |

| 2 | col2 | | DOUBLE | | 2 | 4.4 | 0.2 |

| 3 | col3 | | DOUBLE | | 1 | 6.9 | 0.1 |

| 4 | col4 | | DOUBLE | | 0.1 | 2.5 | 0.1 |

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

Таблица индексов:

+-------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment | Visible | Expression |

+-------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| test | 0 | PRIMARY | 1 | id | A | 150 | None | None | | BTREE | | | YES | None |

| test | 1 | testIndex | 1 | col2 | A | 23 | None | None | YES | BTREE | | | YES | None |

+-------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

A picture containing text

Description automatically generated

Рисунок 13: пример работы программы

Если взять пороговое значение 0.1 то проиндексируются все колонки:

+-------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment | Visible | Expression |

+-------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| test | 0 | PRIMARY | 1 | id | A | 150 | None | None | | BTREE | | | YES | None |

| test | 1 | testIndex | 1 | col1 | A | 35 | None | None | YES | BTREE | | | YES | None |

| test | 1 | testIndex | 2 | col2 | A | 23 | None | None | YES | BTREE | | | YES | None |

| test | 1 | testIndex | 3 | col3 | A | 143 | None | None | YES | BTREE | | | YES | None |

| test | 1 | testIndex | 4 | col4 | A | 147 | None | None | YES | BTREE | | | YES | None |

+-------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

Text

Description automatically generated with medium confidence

Рисунок 14: пример работы программы

**Вывод:** в первый раз индексация велась по одной колонке, а после понижение порогового значения до минимума индексация велось по все 4-м колонкам.

## ***Пример №2***

Возьмем следующие параметры:

Таблица описания БП:

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

| id | Параметр | Описание | Тип | Единицы\_измерения | Диапазон\_от | Диапазон\_до | Вес |

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

| 1 | col1 | | DOUBLE | | 4.3 | 7.9 | 0.1 |

| 2 | col2 | | DOUBLE | | 2 | 4.4 | 0.2 |

| 3 | col3 | | DOUBLE | | 1 | 6.9 | 0.3 |

| 4 | col4 | | DOUBLE | | 0.1 | 2.5 | 0.4 |

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

Таблица индексов:

+--------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment | Visible | Expression |

+--------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| test\_2 | 0 | PRIMARY | 1 | id | A | 150 | None | None | | BTREE | | | YES | None |

| test\_2 | 1 | test\_2Index | 1 | col3 | A | 43 | None | None | YES | BTREE | | | YES | None |

| test\_2 | 1 | test\_2Index | 2 | col4 | A | 22 | None | None | YES | BTREE | | | YES | None |

+--------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

***A picture containing table

Description automatically generated***

Рисунок 15: пример работы программы

**Вывод**: Тут получилось три индекса из-за порогового значения, т.е. как раз таки вошли колонки с весами 0.3 и 0.4.

## ***Пример №3***

Возьмем следующие параметры:

Так же не обязательно чтобы у коллонок были названия col1-col5, в следующем примере возьмем реальное название колонок. Так де зададим единицу измерение как м, и поставим больший бес длине лепестков.

Таблица описания БП:

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

| id | Параметр | Описание | Тип | Единицы\_измерения | Диапазон\_от | Диапазон\_до | Вес |

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

| 1 | sepal\_length | | DOUBLE | м | 4.3 | 7.9 | 0.2 |

| 2 | sepal\_width | | DOUBLE | м | 2 | 4.4 | 0.1 |

| 3 | petal\_length | | DOUBLE | м | 1 | 6.9 | 0.2 |

| 4 | petal\_width | | DOUBLE | м | 0.1 | 2.5 | 0.1 |

+----+------------------+------------------+--------+-----------------------------------+-----------------------+-----------------------+--------+

Таблица индексов:

+--------+------------+-------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment | Visible | Expression |

+--------+------------+-------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

| test\_3 | 0 | PRIMARY | 1 | id | A | 150 | None | None | | BTREE | | | YES | None |

| test\_3 | 1 | test\_3Index | 1 | sepal\_length | A | 35 | None | None | YES | BTREE | | | YES | None |

| test\_3 | 1 | test\_3Index | 2 | petal\_length | A | 123 | None | None | YES | BTREE | | | YES | None |

+--------+------------+-------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+

**Text

Description automatically generated**

Рисунок 16: пример работы программы

**Вывод:** здесь индексация видеться по длине, это так де видно из таблицы. Пороговое зданиче тут 0.2 следовательно создалось только два индекса где

# **Описание функций модуля indexing**

Функция которая находит колонки по которым потом будет проиндексирована БП

# function to find right weights  
def findColumn\_1(weights, num, Names):  
 ind = {}  
 for i in range(len(weights)):  
 if weights[i] >= num:  
 ind[Names[i]] = weights[i]  
  
 return ind

Функция самой индексации БП

# creating an index for bd  
def indexing(StrNames, Weights\_mas, BPName, num, cursor, connection):  
 indexDescription = BPName + 'Index'  
 ind = findColumn\_1(Weights\_mas, num, StrNames)  
 if (len(ind) == 0):  
 print("Warning! Index will not be created due to minimum requirement has not met! \nMaximum weight: ", max(Weights\_mas), '\nValue: ', num)  
 else:  
 col = ', '.join([str for str in ind.keys()])  
  
 try:  
 if col != '':  
 cursor.execute('CREATE INDEX {0} ON {1} ({2})'.format(indexDescription, BPName, col))  
 connection.commit()  
 print('Indexing has been successfully created!')  
   
  
  
 except Exception as e:  
 print('Exception has occured: ', make\_string\_from\_list(make\_list\_from\_tuple(e.args)))  
  
 printTableIndex(cursor, BPName)

Проверка на то существует ли индекс для данной БП

# checking if the bd has already been indexed  
def isIndexed(data, BPName):  
 indexDescription = BPName + 'Index'  
  
 for item in data:  
 if item[2] == indexDescription:  
 return True  
 return False

Удаление индекса БП

# deleting index if there is one  
def delete\_index(BPName, cursor, connection):  
 indexDescription = BPName + 'Index'  
 try:  
 cursor.execute('DROP INDEX {0} ON {1}'.format(indexDescription, BPName))  
 connection.commit()  
 print('Indexing has been successfully deleted!')  
 except Exception as e:  
 print('Exception has occured: ', make\_string\_from\_list(make\_list\_from\_tuple(e.args)))

Распечатка таблицы индексов БП с помощью библиотеки prettytable

# print the table of all existing indexes  
def printTableIndex(cursor, BPName):  
 # Printing index table  
 table\_db = PrettyTable()  
 table\_data = make\_list\_from\_tuple(get\_data\_from\_base(cursor, "SHOW INDEX FROM {0}".format(BPName)))  
 table\_db.field\_names = ['Table', 'Non\_unique', 'Key\_name', 'Seq\_in\_index', 'Column\_name', 'Collation',  
 'Cardinality', 'Sub\_part', 'Packed', 'Null', 'Index\_type', 'Comment',  
 'Index\_comment',  
 'Visible', 'Expression']  
  
 for item in table\_data:  
 table\_db.add\_row(item)  
 print(table\_db)

# **Код**

main.py

from indexing import \*  
import pymysql  
  
  
# ==========================[Connecting to BD]=============================  
connection = pymysql.connect(host='localhost',  
 user='root',  
 password='Yxv-v9K-vLs-NFt',  
 db='table\_storage')  
cursor = connection.cursor()  
# ===========================[Choosing Table]==============================  
table\_db = PrettyTable()  
table\_data = make\_list\_from\_tuple(get\_data\_from\_base(cursor, "SHOW TABLES"))  
table\_data\_matrix = []  
  
count = 0  
for item in table\_data:  
 if str(item[0]) == 'users' or 'Description' in str(item[0]):  
 continue  
 table\_data\_matrix.append([count, item[0]])  
 count += 1  
  
table\_db.field\_names = ['Number', 'Tables']  
for item in table\_data\_matrix:  
 table\_db.add\_row(item)  
print(table\_db)  
  
ans\_range = [str(item) for item in range(len(table\_data))]  
while True:  
 ans = input('Choose table: ')  
 if ans in ans\_range:  
 break  
  
table\_name = table\_data\_matrix[int(ans)][1]  
print('Table name: {0}'.format(table\_name))  
  
ans\_range = [str(item / 10.0) for item in range(1, 11)]  
while True:  
 ans = input('Choose max weight: ')  
 if ans in ans\_range:  
 break  
num = float(ans)  
# ==========================[Reading From DB ]=============================  
table\_columns\_description\_tuples = get\_data\_from\_base(cursor, "DESCRIBE {0}".format(table\_name))  
parametrs\_and\_answer\_names\_list = get\_parametrs\_and\_answer\_names\_list(table\_columns\_description\_tuples)  
  
parametrs\_description\_tuples = get\_data\_from\_base(cursor, "SELECT \* FROM {0}".format(table\_name + 'Description'))  
  
parameters\_weights = [str[7] for str in parametrs\_description\_tuples]  
  
# =========================[Modifying From DB ]============================  
  
info = make\_list\_of\_lists\_from\_tuple\_of\_tuples(get\_data\_from\_base(cursor, 'SHOW INDEX FROM {0}'.format(table\_name)))  
if isIndexed(info, table\_name):  
 ans\_range = ['Y', 'y', 'N', 'n']  
 while True:  
 ans = input('Database has already been indexed. Do you want to delete it ? [Y/N]: ')  
 if ans in ans\_range:  
 break  
  
 if ans == 'Y' or ans == 'y':  
 delete\_index(table\_name, cursor, connection)  
  
info = make\_list\_of\_lists\_from\_tuple\_of\_tuples(get\_data\_from\_base(cursor, 'SHOW INDEX FROM {0}'.format(table\_name)))  
if not isIndexed(info, table\_name):  
 ans\_range = ['Y', 'y', 'N', 'n']  
 while True:  
 ans = input('Do you want to create index ? [Y/N]: ')  
 if ans in ans\_range:  
 break  
  
 if ans == 'Y' or ans == 'y':  
 indexing(parametrs\_and\_answer\_names\_list, parameters\_weights, table\_name, num, cursor, connection)  
else:  
 print("Error: The database '{0}' already has been indexed!".format(table\_name))  
  
# =========================[Print Information]=============================  
ans\_range = ['Y', 'y', 'N', 'n']  
while True:  
 ans = input('Do you want to see more information ? [Y/N]: ')  
 if ans in ans\_range:  
 break  
  
if ans == 'Y' or ans == 'y':  
 # Printing Case Base  
 table\_data = make\_list\_from\_tuple(get\_data\_from\_base(cursor, "SELECT \* FROM {0}".format(table\_name)))  
  
 table\_db = PrettyTable()  
  
 fields = make\_list\_from\_tuple(table\_columns\_description\_tuples)  
 table\_db.field\_names = [str[0] for str in fields]  
  
 for item in table\_data:  
 table\_db.add\_row(item)  
 print(table\_db)  
  
 # Printing Description Table  
 table\_db = PrettyTable()  
 table\_data = make\_list\_from\_tuple(parametrs\_description\_tuples)  
 fields = make\_list\_from\_tuple(get\_data\_from\_base(cursor, "DESCRIBE {0}".format(table\_name + 'Description')))  
 table\_db.field\_names = [str[0] for str in fields]  
  
 for item in table\_data:  
 table\_db.add\_row(item)  
 print(table\_db)  
  
connection.close()

indexing.py

from dbLoad import \*  
from prettytable import PrettyTable  
  
# creating an index for bd  
def indexing(StrNames, Weights\_mas, BPName, num, cursor, connection):  
 indexDescription = BPName + 'Index'  
 ind = findColumn\_1(Weights\_mas, num, StrNames)  
 if (len(ind) == 0):  
 print("Warning! Index will not be created due to minimum requirement has not met! \nMaximum weight: ", max(Weights\_mas), '\nValue: ', num)  
 else:  
 col = ', '.join([str for str in ind.keys()])  
  
 try:  
 if col != '':  
 cursor.execute('CREATE INDEX {0} ON {1} ({2})'.format(indexDescription, BPName, col))  
 connection.commit()  
 print('Indexing has been successfully created!')  
  
  
  
 except Exception as e:  
 print('Exception has occured: ', make\_string\_from\_list(make\_list\_from\_tuple(e.args)))  
  
 printTableIndex(cursor, BPName)  
  
# function to find right weights  
def findColumn\_1(weights, num, Names):  
 ind = {}  
 for i in range(len(weights)):  
 if weights[i] >= num:  
 ind[Names[i]] = weights[i]  
  
 return ind  
  
# checking if the bd has already been indexed  
def isIndexed(data, BPName):  
 indexDescription = BPName + 'Index'  
  
 for item in data:  
 if item[2] == indexDescription:  
 return True  
 return False  
  
# deleting index if there is one  
def delete\_index(BPName, cursor, connection):  
 indexDescription = BPName + 'Index'  
 try:  
 cursor.execute('DROP INDEX {0} ON {1}'.format(indexDescription, BPName))  
 connection.commit()  
 print('Indexing has been successfully deleted!')  
 except Exception as e:  
 print('Exception has occured: ', make\_string\_from\_list(make\_list\_from\_tuple(e.args)))  
  
# print the table of all existing indexes  
def printTableIndex(cursor, BPName):  
 # Printing index table  
 table\_db = PrettyTable()  
 table\_data = make\_list\_from\_tuple(get\_data\_from\_base(cursor, "SHOW INDEX FROM {0}".format(BPName)))  
 table\_db.field\_names = ['Table', 'Non\_unique', 'Key\_name', 'Seq\_in\_index', 'Column\_name', 'Collation',  
 'Cardinality', 'Sub\_part', 'Packed', 'Null', 'Index\_type', 'Comment',  
 'Index\_comment',  
 'Visible', 'Expression']  
  
 for item in table\_data:  
 table\_db.add\_row(item)  
 print(table\_db)

dbLoad.py

import pymysql  
import numpy as np  
  
# converting list to string  
def make\_string\_from\_list(splitted\_list):  
 string\_with\_comma\_separated\_elements = ', '.join(str(element) for element in splitted\_list)  
  
 return string\_with\_comma\_separated\_elements  
  
  
# getting columns name as string  
def get\_parametrs\_and\_answer\_names\_string(table\_columns\_description\_tuples):  
 parametrs\_and\_answer\_names\_list = [str[0] for str in table\_columns\_description\_tuples if  
 (str[0] != 'id') and (str[0] != 'sampleCode') and (  
 str[0] != 'qualityCode')]  
 parametrs\_and\_answer\_names\_string = make\_string\_from\_list(parametrs\_and\_answer\_names\_list)  
  
 return parametrs\_and\_answer\_names\_string  
  
# getting columns name as list   
def get\_parametrs\_and\_answer\_names\_list(table\_columns\_description\_tuples):  
 parametrs\_and\_answer\_names\_list = [str[0] for str in table\_columns\_description\_tuples if  
 (str[0] != 'id') and (str[0] != 'sampleCode') and (  
 str[0] != 'qualityCode')]  
  
 return parametrs\_and\_answer\_names\_list  
  
# converting array of tuples to list array  
def make\_list\_from\_tuple(splitted\_tuple):  
 result\_list = [elem for elem in splitted\_tuple]  
  
 return result\_list  
  
# executing some queries  
def get\_data\_from\_base(cur, query):  
 cur.execute(query)  
 data\_from\_base = cur.fetchall()  
  
 return data\_from\_base  
  
# creating connection to database  
def create\_connection():  
 con = pymysql.connect(host='localhost',  
 user='root',  
 password='Yxv-v9K-vLs-NFt',  
 db='table\_storage')  
 return con  
  
# converting matrix of tuples to lists matrix  
def make\_list\_of\_lists\_from\_tuple\_of\_tuples(splitted\_tuple):  
 arr = [np.array(row) for row in splitted\_tuple]  
 return np.array(arr)

# **Список литературы**

1. <https://ru.wikipedia.org/wiki/Рассуждения_на_основе_прецедентов>
2. <http://samoychiteli.ru/document24188.html>
3. <https://im-cloud.ru/blog/chto-takoe-indeksy-bazy-dannyh-dlja-nachinajushhih/>
4. <https://coderoad.ru/1108/Как-работает-индексация-баз-данных>
5. <https://www.youtube.com/watch?v=8p7dzVSr4XU>
6. Чухраев И.В., Жукова И.В. Оптимизация работы с информацией в базах данных //Международный научный журнал «Инновационная наука». 2016. № 4. С. 206-208.
7. Пирогов В.Ю. Информационные системы и базы данных. Организация и проектирование. СПб.: Изд-во БХВ-Петербург. 2009. 528 с.
8. <https://www.sciencedirect.com/science/article/pii/S240589631500049X>
9. <http://alumni.media.mit.edu/~jorkin/generals/papers/Kolodner_case_based_reasoning.pdf>
10. <https://oatao.univ-toulouse.fr/5002/1/Negny_5002.pdf>
11. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.940&rep=rep1&type=pdf>
12. <https://pypi.org/project/prettytable/>