**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»**

**им. В.И. Ульянова (Ленина)**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Факультет**  **Направление** |  |
|  |  |
| **Профиль** |  |
|  |  |
| **Кафедра** |  |

*К защите допустить:*

**Заведующий кафедрой**

**ВЫПУСКНАЯ**

**КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

***Тема: “Разработка аналитического сервиса на примере базы данных stackoverflow.com”***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Студент* |  |  |  |  |
|  |  | *подпись* |  | *Фамилия И.О.* |
| *Руководитель* |  |  |  |  |
|  | *(Уч. степень, уч. звание)* | *подпись* |  | *Фамилия И.О.* |
| *Консультанты* |  |  |  |  |
|  | *(Уч. степень, уч. звание)* | *подпись* |  | *Фамилия И.О.* |
|  |  |  |  |  |
|  | *(Уч. степень, уч. звание)* | *подпись* |  | *Фамилия И.О.* |

Санкт-Петербург

2015

**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»**

**им. В.И. Ульянова (Ленина)**

|  |
| --- |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | УТВЕРЖДАЮ |
| Факультет |  | | |  |  | Заведующий кафедрой |
| Кафедра |  | | |  |  |  |
|  |  | | | *подпись* |  | *Фамилия И.О.* |
|  |  |  |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2015 г. | | |

**ЗАДАНИЕ**

**на выпускную квалификационную работу бакалавра**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  | Группа № |  |

|  |  |  |
| --- | --- | --- |
| 1. Тема работы |  | |
|  | | |
|  | | |
|  |  | |
| 2. Исходные данные (технические требования) | |  |
|  | | |
|  | | |
|  | | |

|  |  |
| --- | --- |
| 3. Содержание работы |  |
|  | |
|  | |
|  | |
|  | |

|  |  |  |
| --- | --- | --- |
| 4. Перечень отчетных материалов | |  |
|  | | |
|  | | |
|  |  | |
| 5. Дополнительные разделы |  | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Дата выдачи задания |  | Дата представления  работы к защите |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2015 г. |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2015 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Руководитель |  |  |  |  |
|  | *(Уч. степень, уч. звание)* | *подпись* |  | *Фамилия И.О.* |
| Студент |  |  |  |  |
|  |  | *подпись* |  | *Фамилия И.О.* |
|  |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2015 г. | | |

**Содержание**

[**Введение**](#h.gjdgxs)

[**Глава 1. Оперативная аналитическая обработка данных (OLAP)**](#h.30j0zll)

[1.1 Определение и требования OLAP-систем](#h.1fob9te)

[1.2 Классификация OLAP - систем по типу доступа к БД](#h.3znysh7)

[1.3 Сравнение OLAP - серверов](#h.2et92p0)

[**Глава 2. Обработка файлов копии базы данных stackoverflow.com.**](#h.tyjcwt)

[2.1 Выбор системы управления базами данных](#h.3dy6vkm)

[2.2 Структура БД в выбранной СУБД](#h.1t3h5sf)

[2.3 Подготовка данных из копии базы данных stackoverflow.com для вставки в БД](#h.4d34og8)

[2.4 Вставка данных в БД](#h.2s8eyo1)

[**Глава 3. Анализ базы данных stackoverflow.com**](#h.17dp8vu)

[3.1 Структура OLAP куба](#h.3rdcrjn)

[3.2 Построение модели классов и физической ER - модели для OLAP](#h.26in1rg)

[3.3 Добавление данных в БД для OLAP куба](#h.lnxbz9)

[3.4 Создание OLAP куба и анализ данных](#h.35nkun2)

[**Глава 4. Разработка веб - приложения для отображения результатов анализа данных**](#h.1ksv4uv)

[4.1 Технологии разработки веб - приложения](#h.44sinio)

[**Список использованных источников**](#h.2jxsxqh)

[**Приложение А**](#h.z337ya)

[**Приложение В**](#h.3j2qqm3)

[**Приложение С**](#h.4i7ojhp)

# **Введение**

“Кто владеет информацией, тот владеет миром”. Но сегодня информация это не просто набор данных - это результат анализа.

Актуальность дипломной работы обуславливается тем, что современная корпоративная база данных любого предприятия в современном мире содержит набор таблиц, хранящих записи об объектах либо фактах (например, о пользователях, записях пользователей). Как правило, каждая запись в подобной таблице описывает какой-то конкретный факт или объект. Совокупность большого количества таких записей, накопленных за несколько лет, может стать источником дополнительной, гораздо более ценной информации, которую нельзя получить на основе одной конкретной записи, а именно, сведений о закономерностях, тенденциях или взаимозависимостях между данными. Объемы данных настолько внушительны, что человеку просто не по силам проанализировать их самостоятельно, хотя необходимость проведения такого анализа вполне очевидна, ведь в этих "сырых данных" заключены знания, которые могут быть использованы при принятии решений. Для того чтобы провести анализ данных, используются специальные средства, которые помогают выявить скрытые тенденции, строить стратегию развития, находить новые пути развития.

Такой анализ нужен прежде всего, аналитикам, руководителям компаний и лицам, принимающим ключевые бизнес решения.

Кроме централизации и структурирования хранимых данных предприятия требуется инструмент для просмотра и визуализации хранимой информации. Традиционные отчеты лишены гибкости. К ним нельзя применить фильтрацию, чтобы получить желаемое представление данных. Конечно, можно вызвать программиста, и он сделает новый отчет достаточно быстро - скажем, в течение часа. Но в современных условиях ведения бизнеса этого недостаточно. Оперативность в данном случае — один из факторов успеха. Нужен такой инструмент, который позволил бы визуализировать данные быстро, просто и удобно. В качестве такого инструмента и выступает OLAP.

В качестве инструмента для обеспечения обычного пользователя компьютера всей необходимой информацией для принятия правильного решения в нужный момент без изучения работы OLAP сервера может выступать аналитический сервис.

На основе вышеизложенного целью дипломного проекта следует определить разработку аналитического сервиса.

Для решения поставленной цели необходимо рассмотреть ряд сопутствующих задач:

1. Рассмотреть оперативную аналитическую обработку данных (OLAP).
2. Произвести обработку файлов копии базы данных stackoverflow.com.
3. Произвести анализ базы данных stackoverflow.com.
4. Разработать веб-приложение для отображения результатов анализа данных.

Объектом исследования является база данных stackoverflow.com.

Предметом исследования является аналитический сервис для базы данных stackoverflow.com.

# **Глава 1. Оперативная аналитическая обработка данных (OLAP)**

## 1.1 Определение и требования OLAP-систем

OLAP (On-Line Analytical Processing) - технология оперативной аналитической обработки данных, использующая методы и средства для сбора, хранения и анализа многомерных данных в целях поддержки процессов принятия решений.

Многомерный подход возник практически одновременно и параллельно с реляционным, но только начиная с 1993 г. к нему был проявлен всеобщий интерес. В 1993 г. появилась статья одного из основоположников реляционного подхода Кодда, в которой он сформулировал 12 требований к средствам реализации OLAP и произвел анализ недостатков реляционного подхода [1, с.53].

Позднее эти требования были переработаны в тест FASMI (Fast Analysis of Shared Multidimensional Information – Быстрый Анализ Разделяемой Многомерной Информации), который определяет требования к продуктам OLAP. Это определение состоит из пяти требований:

 Fast (Быстрый) - означает, что система должна обеспечивать почти мгновенный отклик на большинство запросов;

 Analysis (Анализ) - возможность осуществления любого логического и статистического анализа, характерного для данного приложения, и его сохранения в доступном для конечного пользователя виде;

 Shared (Разделяемый) - многопользовательский доступ к данным с поддержкой соответствующих механизмов блокировок и средств авторизованного доступа;

 Multidimensional (Многомерный) - данные должны представляться в виде многомерных кубов;

 Information (Информация) - приложение должно иметь возможность обращаться к любой нужной информации, независимо от ее объема и места хранения[1, с. 57-58].

Концепция OLAP - систем основана на следующих понятиях (Рисунок 1):

1. Многомерный куб – это многомерная структура, состоящая из множества ячеек и хранящая взаимосвязанные данные, описывающие предметную область или её составляющую.

2. Показатель (мера) - это величина (обычно числового типа), которая собственно и является предметом анализа. Например, показателями являются величина закупок, величина продаж, посещаемость сайта и т.д. Один OLAP-куб может обладать одним или несколькими показателями.

3. Измерение (dimension) - это множество объектов одного или нескольких типов, организованных в виде иерархической структуры и обеспечивающих информационный контекст числового показателя. Измерение принято визуализировать в виде ребра многомерного куба. Например, измерение “товар” может состоять из следующих значений: “бытовая техника”, “компьютеры”, “мобильные телефоны” и т.д.

4. Ячейка (cell) - является атомарной структурой куба и при отображении располагается внутри него[2].

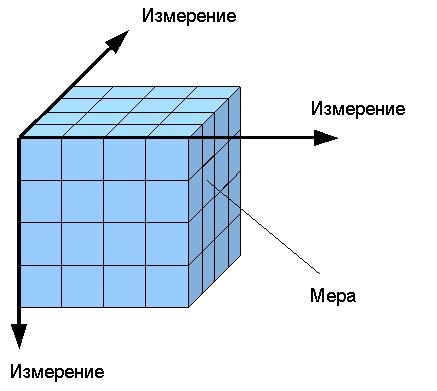


Рисунок 1 - Многомерная модель данных.

## 1.2 Классификация OLAP - систем по типу доступа к БД

На данный момент существует три основных модели OLAP отличающихся по способу организации база дынных:

* многомерный OLAP (MOLAP, Multidimensional OLAP);
* реляционный OLAP (ROLAP, Relational OLAP);
* гибридный OLAP (HOLAP, Hybrid OLAP).

Многомерный OLAP

В многомерных OLAP-системах структура куба хранится в многомерной базе данных. В той же базе данных хранятся предварительно обработанные агрегаты и копии листовых значений. В связи с этим все запросы к данным удовлетворяются многомерной системой баз данных, что делает MOLAP-системы исключительно быстрыми.

Архитектура MOLAP требует большего объема дискового пространства из-за хранения в многомерной базе копий листовых данных.

Достоинства MOLAP-систем:

* все данные хранятся в многомерных структурах, что существенно повышает скорость обработки запросов;
* доступны расширенные библиотеки для сложных функций оперативного анализа;
* обработка разреженных данных выполняется лучше, чем в ROLAP.

Недостатки:

* необходимы специальные инструменты для формирования кубов и их пересчёта в случае изменения базовых значений;
* сложно изменять измерения без повторной агрегации.

Реляционный OLAP

В реляционных OLAP-системах структура куба данных хранится в реляционной базе данных. Меры самого нижнего уровня остаются в реляционной витрине данных, служащей источником данных для куба. Предварительно обработанные агрегаты также хранятся в реляционной таблице.

Благодаря реляционным таблицам, архитектура ROLAP позволяет хранить большие объемы данных.

Достоинства этого класса систем:

* возможность манипулирования большими объемами данных; объем данных могут ограничивать только лежащие в основе ROLAP системы реляционных баз данных, подход ROLAP сам по себе не ограничивает объем данных;
* безопасность и администрирование обеспечивается реляционными СУБД.

Недостатки:

* Меньшая производительность, чем у MOLAP и HOLAP. Для обеспечения сравнимой с MOLAP производительности реляционные системы требуют тщательной проработки схемы БД и специальной настройки индексов;
* функциональность систем ограничивается возможностями SQL, так как аналитические запросы пользователя транслируются в SQL-операторы выборки;
* сложно пересчитывать агрегированные значения при изменениях начальных данных.

Гибридный OLAP

В гибридных OLAP сочетаются черты ROLAP и MOLAP, отсюда и название – гибридный. В моделях HOLAP используются преимущества и минимизируются недостатки обеих архитектур.

В HOLAP-системах структура куба и предварительно обработанные агрегаты хранятся в многомерной базе данных. Это позволяет обеспечить быстрое извлечение агрегатов из структур MOLAP. Значения нижнего уровня иерархии в HOLAP остаются в реляционной витрине данных, которая служит источником данных для куба.

К достоинствам подхода можно отнести комбинирование технологии ROLAP для разреженных данных и MOLAP для плотных областей, а к недостаткам – необходимость поддерживания MOLAP и ROLAP [3].

Так как анализ будет производиться над данными из реляционной БД, то следует выбрать реляционный OLAP (ROLAP).

## 1.3 Сравнение OLAP - серверов

В приведенных таблицах сравниваются основные и технические характеристики основных OLAP-серверов.

Таблица 1 - Общая информация

|  |  |  |
| --- | --- | --- |
| OLAP - сервер | Производитель | Лицензия |
| Microsoft Analysis Services | Microsoft | - |
| Essbase | Oracle | - |
| TM1 | IBM | - |
| Mondrian OLAP Server | Pentaho | свободное программное обеспечение |
| Palo | Jedox | - |
| Oracle OLAP Option | Oracle | - |
| Microstrategy OLAP Services | Microstrategy | - |
| SAS OLAP Server | SAS Institute | - |
| icCube OLAP Server | MISConsulting SA | свободное программное обеспечение |

Таблица 2 - Модель хранения данных

|  |  |  |  |
| --- | --- | --- | --- |
| OLAP - сервер | MOLAP | ROLAP | HOLAP |
| Microsoft Analysis Services | Да | Да | Да |
| Essbase | Да | Да | Да |
| TM1 | Да | Нет | Нет |
| Mondrian OLAP Server | Нет | Да | Нет |
| Palo | Да | Нет | Нет |
| Oracle OLAP Option | Да | Да | Да |
| Microstrategy OLAP Services | Да | Да | Нет |
| SAS OLAP Server | Да | Да | Да |
| icCube OLAP Server | Да | Нет | Нет |

Таблица 3 - API и язык запросов

|  |  |  |  |
| --- | --- | --- | --- |
| OLAP - сервер | XML для анализа | MDX | Пользовательские функции |
| Microsoft Analysis Services | Да | Да | Да |
| Essbase | Да | Да | Да |
| TM1 | Да | Да | Да |
| Mondrian OLAP Server | Да | Да | Да |
| Palo | Да | Да | Да |
| Oracle OLAP Option | Нет | Да | Да |
| Microstrategy OLAP Services | Да | Да | Да |
| SAS OLAP Server | Да | Да | Нет |
| icCube OLAP Server | Да | Да | Да |

Таблица 4 - Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OLAP - сервер | Windows | Linux | Unix | z/OS |
| Microsoft Analysis Services | Да | Нет | Нет | Нет |
| Essbase | Да | Да | Да | Нет |
| TM1 | Да | Да | Да | Нет |
| Mondrian OLAP Server | Да | Да | Да | Да |
| Palo | Да | Да | Да | Нет |
| Oracle OLAP Option | Да | Да | Да | Да |
| Microstrategy OLAP Services | Да | Да | Да | Нет |
| SAS OLAP Server | Да | Да | Да | Да |
| icCube OLAP Server | Да | Да | Да | Да |

[4, с. 148-151]

Для выполнения аналитической обработки данных был выбран Mondrian OLAP Server, по следующим критериям:

* поддерживает модель хранения данных ROLAP;
* бесплатно распространяемое программное обеспечение.

В данной главе мы ознакомились с основами OLAP.

Мы узнали следующее:

* OLAP - технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу.
* Приложения с OLAP-функциональностью должны предоставлять пользователю результаты анализа за приемлемое время, осуществлять логический и статистический анализ, поддерживать многопользовательский доступ к данным, осуществлять многомерное концептуальное представление данных и иметь возможность обращаться к любой нужной информации.

Кроме того, мы выбрали Mondrian OLAP Server для аналитической обработки данных, который позволяет анализировать большие и сложные объемы данных в режиме реального времени. Поддерживает язык запросов MDX, а также спецификации XML for Analysis. Для хранения данных может использовать любую систему управления базами данных с поддержкой SQL.

# **Глава 2. Обработка файлов копии базы данных stackoverflow.com.**

## 2.1 Выбор системы управления базами данных

Рассмотрим две основных свободно распространяемых СУБД:

* MySQL - самая распространённая СУБД
* PostgreSQL - наиболее продвинутая СУБД

MySQL

MySQL - это полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб-приложениями. Обучиться использованию этой СУБД довольно просто, так как на просторах интернета вы легко найдете большее количество информации.

Несмотря на то, что в ней не реализован весь SQL функционал, MySQL предлагает довольно много инструментов для разработки приложений.

Преимущества MySQL

* Простота в работе - установить MySQL довольно просто.
* Богатый функционал - MySQL поддерживает большинство функционала SQL.
* Масштабируемость - MySQL легко работает с большими объемами данных и легко масштабируется.
* Скорость - упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки MySQL

* Известные ограничения - по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях.
* Проблемы с надежностью - из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности.
* Многопоточность - параллельные операции чтения-записи могут создать проблемы.

PostgreSQL

PostgreSQL является наиболее профессиональным СУБД, чем MySQL. Она свободно распространяемая и максимально соответствует стандартам SQL.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID).) Благодаря мощным технологиям Postgre очень производительна.

Хотя PostgreSQL и не может похвастаться большой популярностью в отличии от MySQL, существует довольно большое число приложений облегчающих работу с PostgreSQL. Сейчас довольно легко установить эту СУБД используя стандартные менеджеры пакетов операционных систем.

Достоинства PostgreSQL

* Открытое ПО соответствующее стандарту SQL - PostgreSQL - бесплатное ПО с открытым исходным кодом.
* Надежное и защищенное хранение данных, которыми она управляет.
* Большое сообщество - существует довольно большое сообщество в котором вы запросто найдёте ответы на свои вопросы
* Предлагает множество современных возможностей: комплексные запросы, внешние ключи, триггеры, представления (views), транзакционная целостность, многоверсионное управление параллельным доступом.
* Возможности PostgreSQL могут быть расширены пользователем путём добавления новых типов данных, функций, операторов, агрегатных функций, индексных методов.
* PostgreSQL предлагает 4-ре типа индексов. Каждый тип индекса имеет свой алгоритм реализации, что позволяет существенно увеличить быстродействие.

Недостатки PostgreSQL

* Популярность - по своей природе, популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество [5].

На основании выше изложенного в качестве СУБД выбран PostgreSQL.

## 2.2 Структура БД в выбранной СУБД

Модель классов

Пользователи (users) могут создавать посты (posts). Пост в свою очередь может быть или вопросом (questions) или ответом на вопрос (answers). Пользователи отвечая на вопросы и задавая их получают определённые значки (badges). Также пользователи могут оставлять комментарии к постам (comments) и голосовать (votes).

Класс “votes” имеет cледующие типы голосов: acceptedByOriginator, upMod, downMod, offensive, favorite, close, reopen, bountyStart, bountyClose, deletion, undeletion, spam, moderatorReview, approveEditSuggestion.

Класс “tags” включает в себя название тега (name), а также сколько вопросов (count) создано с таким тегом.

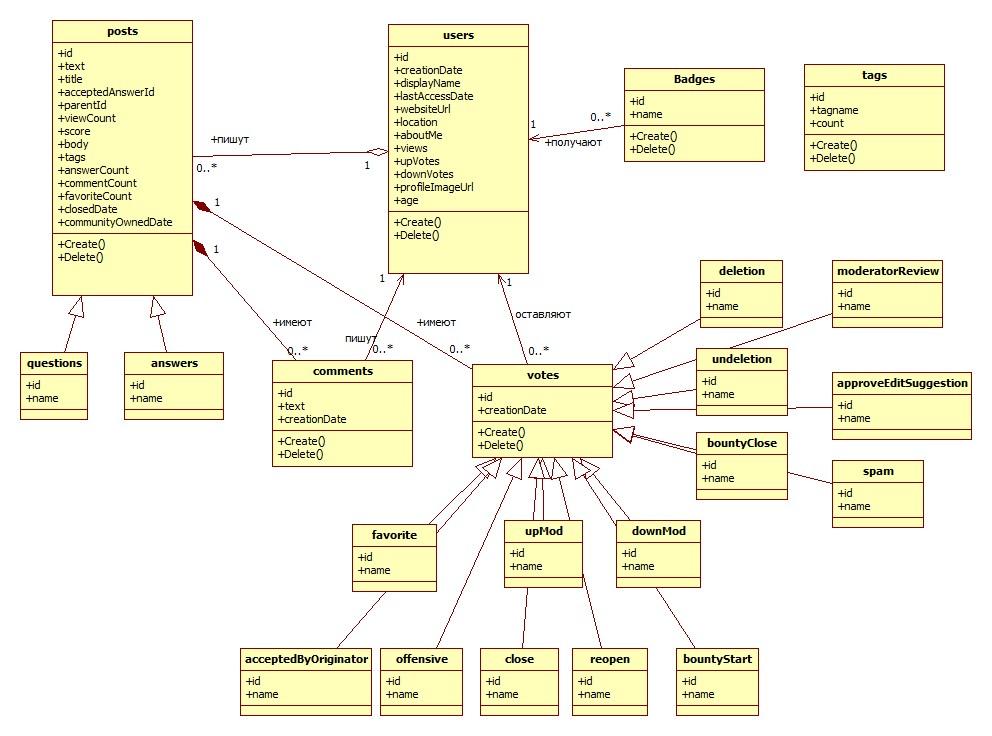


Рисунок 2 - Диаграмма классов БД stackoverflow.

Физическая ER - модель

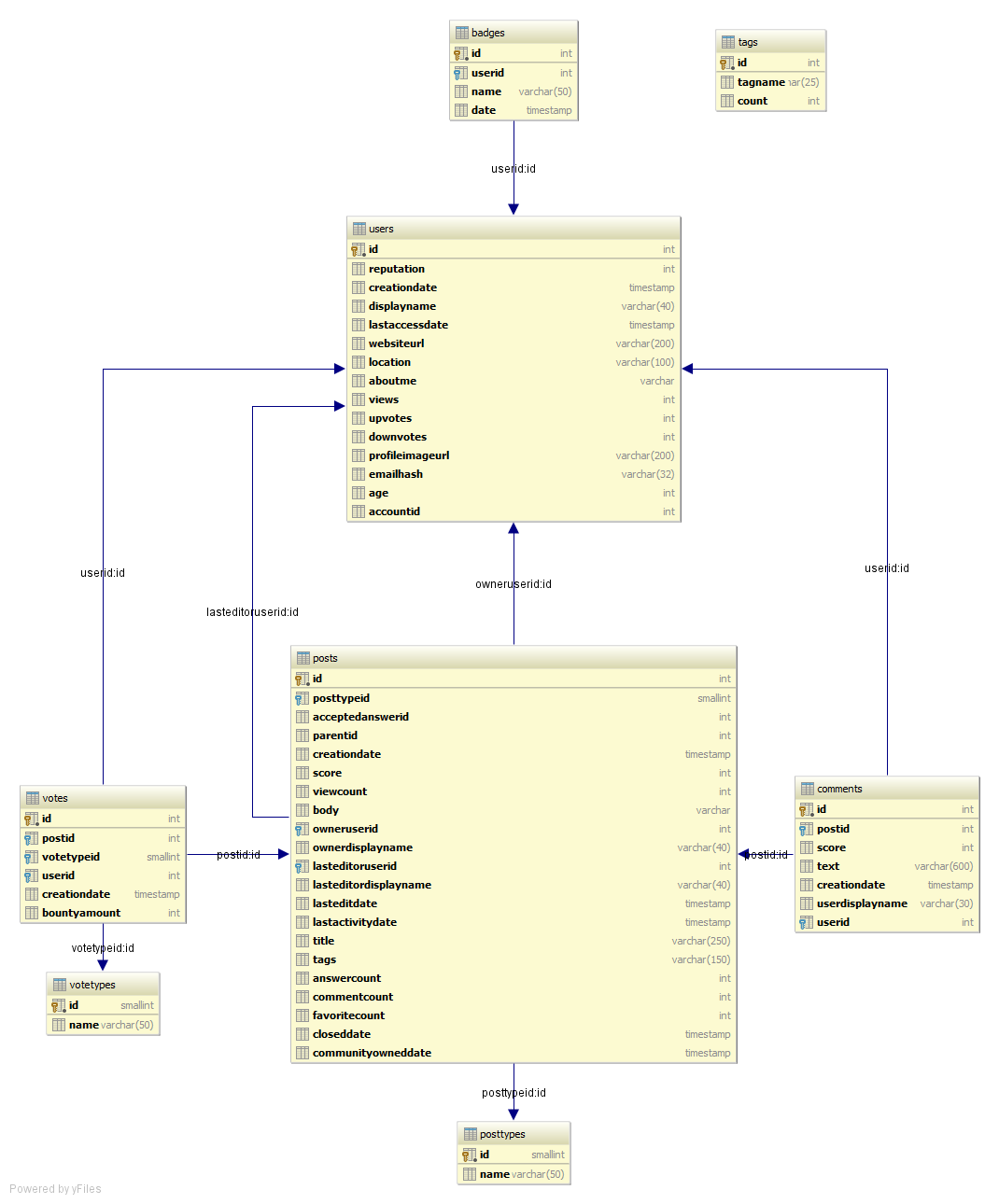


Рисунок 3 - Физическая ER - модель БД stackoverflow

Описание таблиц

Таблица posts:

Id - идентификатор поста

PostTypeId - тип поста, например, вопрос или ответ

AcceptedAnswerId - есть только если тип поста – вопрос

ParentID - есть только если тип поста – ответ, тогда в это поле записывается

id вопроса

CreationDate - дата создания поста

Score - оценка

ViewCount - количество просмотров

Body - тело поста

OwnerUserId - id пользователя, создавшего пост. Присутствует, только если пользователь не был удален.

OwnerDisplayName - имя пользователя, который создал пост

LastEditorUserId - id пользователя, который последним редактировал пост

LastEditorDisplayName="Rich B"

LastEditDate - дата последнего редактирования поста

LastActivityDate - дата недавней активности на пост

Title- заголовок

Tags - теги (ключевые слова)

AnswerCount - количество ответов

CommentCount - количество комментариев

FavoriteCount - количество голосов типа favorite

ClosedDate - присутствует только если пост закрыт

CommunityOwnedDate - присутствует, только если пост сообщества вики

(wiki)

Таблица users:

Id - идентификатор пользователя

Reputation - репутация

CreationDate - дата регистрации

DisplayName - имя

LastAccessDate - последние посещение

WebsiteUrl - url сайта

Location - расположение (откуда пользователь)

AboutMe - информация о пользователе

Views - количество просмотров (сколько человек посещали данного пользователя)

UpVotes and DownVotes - Количество голосов, что этот пользователь оставил

EmailHash - не всегда имеется

Age - возраст

AccountId - id пользователя в StackExchange

Таблица comments:

PostId - id поста

Score - оценка комментария

Text - текст

CreationDate – дата создания комментария

UserDisplayName - имя пользователя, создавшего комментарий

UserId – id пользователя

Таблица voteTypes:

Id - идентификатор типа голоса

Name - название типа голоса

Таблица votes:

PostId - id поста

VoteTypeId - тип голоса

UserId – id пользователя, присутствует только если тип голоса 5 или 8

CreationDate -дата, когда проголосовали

BountyAmount - сумма, только если тип голоса 8 или 9

Таблица badges:

Id - идентификатор значка

UserId - id пользователя, получившего значок

Name - название значка

Date - дата получения пользователем значка

Таблица tags:

Id - идентификатор тега

TagName - название тега

Count - количество вопросов с этим тегом

## 2.3 Подготовка данных из копии базы данных stackoverflow.com для вставки в БД

Для обработки данных использовался Apache Spark.

Apache Spark — это фреймворк для анализа больших массивов данных.

Благодаря представлению информации в виде так называемых упругих распределенных наборов данных (RDD - Resilient Distributed Dataset), осуществляется быстрый и простой доступ к данным через прикладные интерфейсы на Python, Scala и Java. Кроме того, такая технология позволяет запрашивать данные и одновременно запускать сложные алгоритмы их анализа[6].

Большинство операций над RDD не приводит к каким-либо вычислениям, а только создаёт очередную обёртку, обещая выполнить операции только тогда, когда они понадобятся.

В качестве основного языка реализации был выбран язык программирования Scala. Scala - это мультипарадигмальный язык программирования, спроектированный кратким и типобезопасным для простого и быстрого создания компонентного программного обеспечения, сочетающий возможности функционального и объектно-ориентированного программирования.

Ниже показан код для обработки данных из файла типа xml:

val conf = new SparkConf().setMaster("local[4]").setAppName("test")

val sc = new SparkContext(conf)

val rdd = sc.textFile("D:\\stackexchange\\Users.xml")

rdd.map(line => {

val map = scala.collection.mutable.Map[String,String]()

val regex = """([A-z]\*?)=(".\*?")""".r

for (p <- regex.findAllIn(line)) p match {

case regex(key, value) => map += key -> value.replace("\"", "")

}

val id = map.getOrElse("Id","null")

val reputation = map.getOrElse("Reputation","null")

val creationDate = if(map.getOrElse("CreationDate","null").equals("null")) "null"

else "'%s'".format(map.get("CreationDate").get.replace("T", " ").

replaceAll("\\.\\d{3}", ""))

val displayName = map.getOrElse("DisplayName","null")

val lastAccessDate = if(map.getOrElse("LastAccessDate","null").equals("null")) "null"

else "'%s'".format(map.get("LastAccessDate").get.replace("T", " ").

replaceAll("\\.\\d{3}", ""))

val websiteUrl = map.getOrElse("WebsiteUrl","null")

val location = map.getOrElse("Location","null")

val aboutMe = map.getOrElse("AboutMe","null").replace("'", "")

val views = map.getOrElse("Views","null")

val upVotes = map.getOrElse("UpVotes","null")

val downVotes = map.getOrElse("DownVotes","null")

val profileImageUrl = map.getOrElse("ProfileImageUrl","null")

val emailHash = map.getOrElse("EmailHash","null")

val age = map.getOrElse("Age","null")

val accountId = map.getOrElse("AccountId","null")

"INSERT INTO users values (%s,%s,%s,'%s',%s,'%s','%s','%s',%s,%s,%s,'%s','%s',%s,%s);".

format(id, reputation, creationDate, displayName, lastAccessDate, websiteUrl, location, aboutMe, views, upVotes, downVotes, profileImageUrl, emailHash, age, accountId)

}).saveAsTextFile("D:\\stackexchange\\users")

}

Вычисления начинаются только на последней строчке: Spark видит, что нужно материализовать результаты, и для этого начинает применять операции к данным.

Каждая строчка считывается из файла и обрабатывается следующим образом: из строки выделяются подстроки подходящие под следующее регулярное выражение: ([A-z]\*?)=(".\*?"). Полученные подстроки попадают в коллекцию “Ключ-Значение”, после чего в зависимости от колонок таблицы вытаскиваются нужные значения по ключу, если в коллекции не оказалось такого ключа записывается null. Все полученные значения записываются в строку шаблон для вставки в СУБД , после чего эта строка добавляется в файл.

## 2.4 Вставка данных в БД

Для доступа к БД (stackoverflow), а также для выполнения к ней SQL запросов использовался JDBC.

**JDBC Driver** — (Java DataBase Connectivity — соединение с базами данных на Java) — платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE [7].

Ниже приведён код для считывания строк из файла и их добавления в БД:

def main(args: Array[String]) {

val tableName = Array("users","posts","votes","comments")

for (table <- tableName) {

addData(table)

}

def addData(tableName: String) = {

val connection = getConnection

Directory("D:\\insert\\" + tableName).files.foreach(file=>{

file.lines.foreach(line=> {

var statement = connection.createStatement()

try {

statement.executeUpdate(line)

}catch {

case \_ =>

}

})

})

}

}

def getConnection = {

Class.forName("org.postgresql.Driver") // загрузка JDBC драйвера

val url = "jdbc:postgresql://localhost/stackoverflow"

val user = "postgres"

val password = "123"

val connection = DriverManager.getConnection(url, user, password)

//открывает соединение с базой данных и обеспечит дальнейшее обращение к ней

connection

}

В данной главе была выбрана СУБД - PostgreSQL.

Мы узнали следующее:

PostgreSQL - это продукт с открытым исходным кодом, который можно бесплатно использовать, модифицировать и распространять для любых целей, включая личные, коммерческие или академические. PostgreSQL поддерживает большую часть стандарта SQL и предлагает множество современных возможностей.

Также были подготовлены данные из копии базы данных stackoverflow.com для вставки в СУБД и добавлены в PostgreSQL. Для обработки данных использовался Apache Spark.

# **Глава 3. Анализ базы данных stackoverflow.com**

## 3.1 Структура OLAP куба

Конечной целью использования OLAP является анализ данных и представление результатов этого анализа в виде, удобном для восприятия и принятия решений. Основная идея OLAP заключается в построении многомерных кубов, которые будут доступны для пользовательских запросов.

Исходные данные для построения OLAP-куба хранятся в реляционных базе данных stackoverflowForOlap. Данные в stackoverflowForOlap копируются из оригинальной базы данных stackoverflow.

Cтруктура stackoverflowForOlap существенно отличается от структуры оригинальной БД stackoverflow. Так как структура денормализована (это позволяет повысить скорость выполнения запросов), поэтому она может допускать избыточность данных.

Основными составляющими структуры БД для OLAP являются таблица фактов (fact table) и таблицы измерений (dimension tables).

Таблица фактов

Таблица фактов является основной таблицей хранилища данных. Как правило, она содержит сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться. Обычно говорят о четырех наиболее часто встречающихся типах фактов. К ним относятся:

* факты, связанные с транзакциями (Transaction facts). Они основаны на отдельных событиях (типичными примерами которых являются телефонный звонок или снятие денег со счета с помощью банкомата);
* факты, связанные с «моментальными снимками» (Snapshot facts). Основаны на состоянии объекта (например, банковского счета) в определенные моменты времени, например на конец дня или месяца. Типичными примерами таких фактов являются объем продаж за день или дневная выручка;
* факты, связанные с элементами документа (Line-item facts). Основаны на том или ином документе (например, счете за товар или услуги) и содержат подробную информацию об элементах этого документа (например, количестве, цене, проценте скидки);
* факты, связанные с событиями или состоянием объекта (Event or state facts). Представляют возникновение события без подробностей о нем (например, просто факт продажи или факт отсутствия таковой без иных подробностей).

Таблица фактов, как правило, содержит уникальный составной ключ, объединяющий первичные ключи таблиц измерений. Чаще всего это целочисленные значения либо значения типа «дата/время» — ведь таблица фактов может содержать сотни тысяч или даже миллионы записей, и хранить в ней повторяющиеся текстовые описания, как правило, невыгодно — лучше поместить их в меньшие по объему таблицы измерений. При этом как ключевые, так и некоторые неключевые поля должны соответствовать будущим измерениям OLAP-куба. Помимо этого таблица фактов содержит одно или несколько числовых полей, на основании которых в дальнейшем будут получены агрегатные данные.

Таблицы измерений

Таблицы измерений содержат как минимум одно описательное поле (обычно с именем члена измерения) и, как правило, целочисленное ключевое поле (обычно это суррогатный ключ) для однозначной идентификации члена измерения. Если будущее измерение, основанное на данной таблице измерений, содержит иерархию, то таблица измерений также может содержать поля, указывающие на «родителя» данного члена в этой иерархии.

Каждая таблица измерений должна находиться в отношении «один ко многим» с таблицей фактов. [1, c. 63-65]

Скорость роста таблиц измерений должна быть незначительной по сравнению со скоростью роста таблицы фактов.

Куб создаётся из соединения таблиц с применением схемы звезды или схемы снежинки.

Схема звезда

Схема типа звезды (Star Schema) - схема реляционной базы данных, служащая для поддержки многомерного представления содержащихся в ней данных.

1. Таблица фактов (fact table), которая сильно денормализована. Является центральной в схеме, может состоять из миллионов строк и содержит суммируемые или фактические данные, с помощью которых можно ответить на различные вопросы.
2. Несколько денормализованных таблиц измерений (dimensional table). Имеют меньшее количество строк, чем таблицы фактов, и содержат описательную информацию. Эти таблицы позволяют пользователю быстро переходить от таблицы фактов к дополнительной информации.
3. Таблица фактов и таблицы размерности связаны идентифицирующими связями, при этом первичные ключи таблицы размерности мигрируют в таблицу фактов в качестве внешних ключей. Первичный ключ таблицы факта целиком состоит из первичных ключей всех таблиц размерности.
4. Агрегированные данные хранятся совместно с исходными.

Преимущества:

Благодаря денормализации таблиц измерений упрощается восприятие структуры данных пользователем и формулировка запросов, уменьшается количество операций соединения таблиц при обработке запросов. Некоторые промышленные СУБД и инструменты класса OLAP умеют использовать преимущества схемы "звезда" для сокращения времени выполнения запросов.

Недостатки:

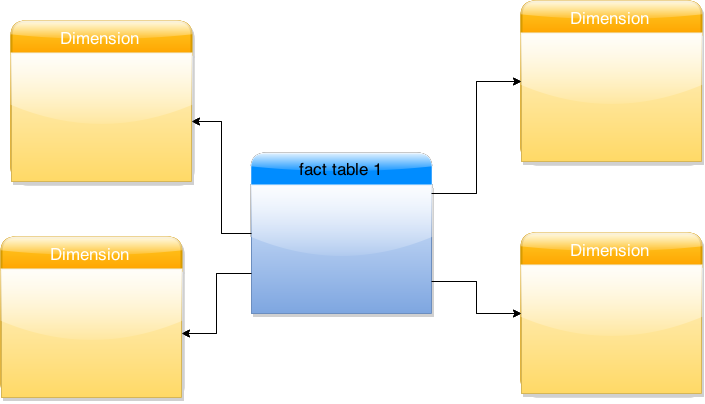
Денормализация таблиц измерений вносит избыточность данных, возрастает требуемый для их хранения объем памяти. 

Рисунок 4 - Схема звезда

Схема снежинка

Схема типа снежинки (Snowflake Schema) - схема реляционной базы данных, служащая для поддержки многомерного представления содержащихся в ней данных, является разновидностью схемы типа "звезда" (Star Schema).

1. Присутствует таблица фактов (fact table), которая сильно денормализована.
2. Несколько таблиц измерений (dimensional table), которые нормализованы в отличие от схемы "звезда". Имеют меньшее количество строк, чем таблицы фактов, и содержат описательную информацию. Эти таблицы позволяют пользователю быстро переходить от таблицы фактов к дополнительной информации. Первичные ключи в них состоят из единственного атрибута (соответствуют единственному элементу измерения).
3. Таблица фактов и таблицы размерности связаны идентифицирующими связями, при этом первичные ключи таблицы размерности мигрируют в таблицу фактов в качестве внешних ключей. Первичный ключ таблицы факта целиком состоит из первичных ключей всех таблиц размерности.
4. В схеме "снежинка" агрегированные данные могут храниться отдельно от исходных.

Преимущества:

Нормализация таблиц измерений в отличие от схемы "звезда" позволяет минимизировать избыточность данных.

Недостатки:

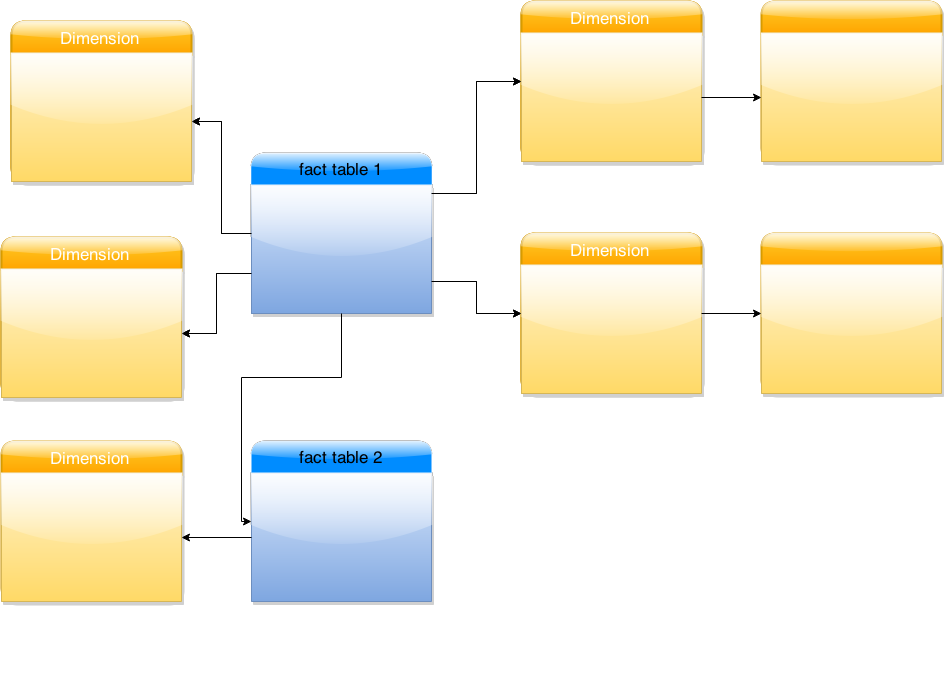
За нормализацию таблиц измерений иногда приходится платить временем выполнения запросов[8]. 

Рисунок 5 - Схема снежинка

## 3.2 Построение модели классов и физической ER - модели для OLAP

Модель классов

Пользователи (users) могут создавать посты (posts). У каждого пользователя есть дата регистрации на сайте stackoverflow.com (dates). У каждого поста есть дата его создания (dates). Пользователи отвечая на вопросы и задавая их получают определённые значки (badges). Также пользователи могут оставлять комментарии к постам (comments) и голосовать (votes).

У каждого комментария есть дата его создания (dates).

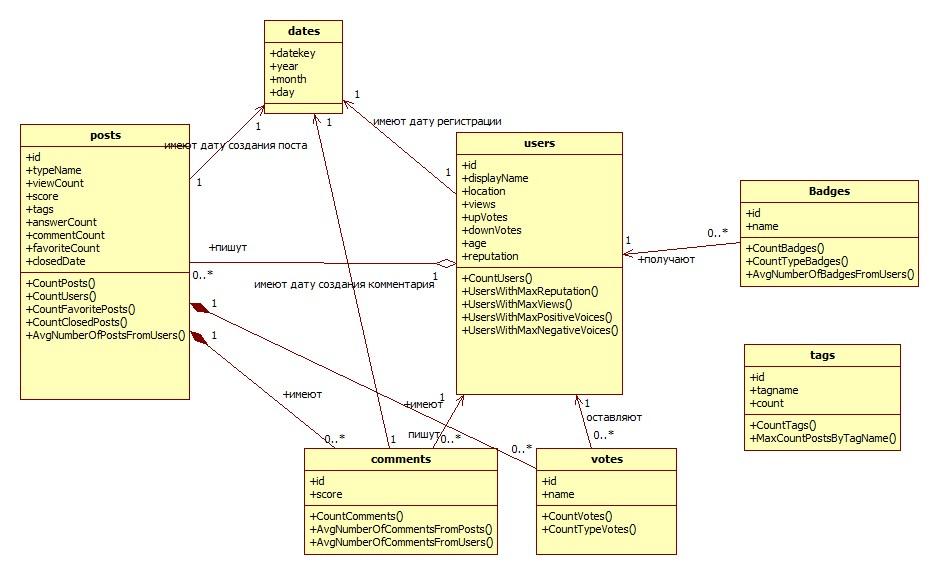
Класс “votes” включает в себя название типа голоса. Класс “tags” включает в себя название тега (name), а также сколько вопросов (count) создано с таким тегом. 

Рисунок 6 - Модель классов БД stackoverflowForOlap

Физическая ER - модель

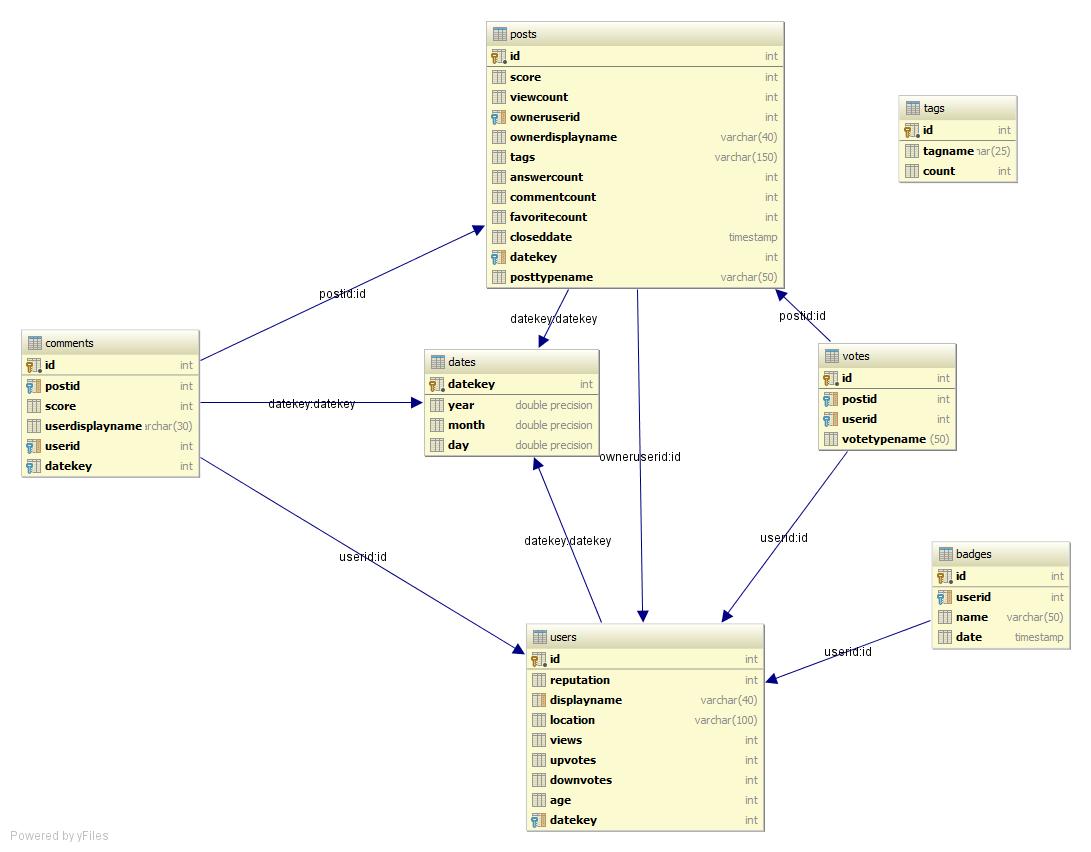


Рисунок 7 - Физическая ER - модель для OLAP

Таблицы фактов: posts, tags, votes, comments, users.

Таблицы измерений: users (является таблицей измерений для таблицы фактов posts) , dates.

Спроектированная модель является схемой звезда. Так как в данной схеме таблицы фактов связаны с таблицами измерений, но таблицы измерений не имеют связи с другими таблицами измерений. Физическая - ER модель для каждого куба находится в “Приложении В”.

Описание таблиц votes, users, posts, tags, badges, comments представлено во второй главе (2.2 Структура БД в выбранной СУБД).

Описание таблицы dates:

datekey - идентификатор даты

year - год

month - месяц

day - день

## 3.3 Добавление данных в БД для OLAP куба

При изменении, удалении или добавлении данных в stackoverflow, происходит изменение, удаление или добавление данных в stackoverflowForOlap. Это реализовано с помощью триггеров для таблиц users, posts, comments, votes, badges, tags. Примеры скриптов находятся в “Приложении А”.

Добавление данных в таблицу dates:

INSERT INTO dates(year, month, day) WITH date\_series AS (

SELECT DATE (GENERATE\_SERIES (DATE '2008-01-01', DATE '2015-04-01' , '1 day' ))

AS date )

SELECT

EXTRACT(YEAR FROM date)

, EXTRACT(MONTH FROM date)

,EXTRACT(DAY FROM date)

FROM date\_series;

## 3.4 Создание OLAP куба и анализ данных

В первой главе в качестве сервера для аналитической обработки данных былл выбран Mondrian OLAP Server. С ноября 2005 года входит как программный компонент в BI-пакет Pentaho BI Suite, как компонент фигурирует под наименованием *Pentaho Analysis Services Community Edition* (бесплатная реализация с открытым кодом).

Pentaho BI Server позволяет пользователям создавать свои собственные отчеты.

Какие инструменты нужны для создания отчётов?

1. Mondrian - Pentaho Schema workbench - позволяет создавать и проверять Mondrian OLAP схемы куба визуально, поддерживает язык запросов MDX (Multidimensional Expressions). Кубы и измерения определяются путем создания схемы XML.

Архитектура Mondrian состоит из четырех уровней:

* уровень представления,
* уровень измерений,
* главный уровень,
* физический уровень.

Уровень представления определяет, как будет отображаться информация, и как пользователь будет взаимодействовать с системой для определения запросов. Уровень измерений анализирует, проверяет и выполняет MDX запросы. Главный уровень поддерживает кэш с агрегированными данными. Уровень хранения – реляционная СУБД [2].

1. Saiku Analytics - средство OLAP-анализа.

Для Pentaho Saiku Analytics является инструментом для создания отчётности. Загрузить и установить Saiku можно из Pentaho Market Place в Penthao Server.

Создание OLAP куба с помощью Schema workbench

Для организации кубов нужна схема (в виде XML) представления куба. В схеме описывается, к какой таблице происходит построение куба, производится описание измерений, меры, расчет дополнительных полей.

Пример схемы:

<Schema name="cube-stackoverflow" description="cube-stackoverflow">

<Dimension type="TimeDimension" visible="true" highCardinality="false" name="Date">

<Hierarchy name="Date" visible="true" hasAll="true" primaryKey="datekey">

<Table name="dates" schema="public">

</Table>

<Level name="Year" visible="true" column="year" type="Integer"

uniqueMembers="false" levelType="TimeYears" hideMemberIf="Never">

</Level>

<Level name="Month" visible="true" column="month" nameColumn="month"

uniqueMembers="false" levelType="TimeMonths" hideMemberIf="Never">

</Level>

<Level name="Day" visible="true" column="day" nameColumn="day"

uniqueMembers="false" levelType="TimeDays" hideMemberIf="Never">

</Level>

</Hierarchy>

</Dimension>

<Dimension type="StandardDimension" visible="true" name="User-age">

<Hierarchy name="User-age" visible="true" hasAll="true" primaryKey="id">

<Table name="users" schema="public">

</Table>

<Level name="Age" visible="true" column="age" uniqueMembers="false" >

</Level>

</Hierarchy>

</Dimension>

<Dimension type="StandardDimension" visible="true" name="Users-UserId">

<Hierarchy name="Users-UserId" visible="true" hasAll="true" primaryKey="id">

<Table name="users" schema="public">

</Table>

<Level name="UserId" visible="true" column="id" uniqueMembers="true">

</Level>

</Hierarchy>

</Dimension>

<Cube name="data\_users" caption="data\_users" visible="true" description="data\_users" cache="true" enabled="true">

<Table name="users" schema="public"> </Table>

<DimensionUsage source="User-age" name="Age" caption="age" visible="true" foreignKey="age">

</DimensionUsage>

<DimensionUsage source="Users-UserId" name="Id" caption="Id" visible="true" foreignKey="id"

</DimensionUsage>

<DimensionUsage source="Date" name="Date" caption="Date" visible="true" foreignKey="datekey">

</DimensionUsage>

<Measure name="count\_users" column="id" aggregator="count">

</Measure>

</Cube>

Описание тегов

*<Schema name="cube-stackoverflow" description="cube-stackoverflow">*

*<Schema>* - тег, необходимый для описания схемы функционирования OLAP куба на сервере Pentaho. В схеме могут быть описаны права пользователей, меры, кубы, виртуальные кубы и т.д.

*name="cube-stackoverflow"* - название схемы.

*description="cube-stackoverflow"* - описание схемы.

*<Cube>* - тег, в котором происходит описание куба.

*<Table name="users" schema="public">* - это таблица фактов, по которой происходит построение куба.

*<Dimension>* - тег измерений. В измерении указывается информация, по какому полю будет задаваться соответствие с таблицей фактов.

Выделим два вида измерений: внутренние и внешние.

Внутренние измерения задаются между тегами *<cube> ...</cube>*.

Внешние измерения для кубов делают, если в схеме будет более одного куба.

Внешние измерения описываются точно так же, как и внутренние, только располагаются между тегами *<Schema>...</Schema>*. Для использования внешнего измерения в кубе нужно сделать на него ссылку. Ссылка делается между тегами куба *<cube> </cube>* тегом

*<DimensionUsage source="Users-UserId" name="Id" caption="Id" visible="true" foreignKey="id">*

*name="Id"* - название измерения.

*source="Users-UserId"* - название источника измерения, которое добавляется в куб.

*foreignKey="id"* - внешний ключ таблицы фактов.

Внутреннее измерение

*<Dimension type="StandardDimension" visible="true" name="Users-UserId">*

*<Hierarchy name="Users-UserId" visible="true" hasAll="true" primaryKey="id">*

*<Table name="users" schema="public">*

*</Table>*

*<Level name="UserId" visible="true" column="id" uniqueMembers="false">*

*</Level>*

*</Hierarchy>*

*</Dimension>*

*<Hierarchy>* - это тег, описывающий иерархию, которая находится в таблице. Для описания требуется задать первичный ключ таблицы измерений.

*primaryKey="id"* - первичный ключ таблицы измерений.

*hasAll="true"* - отображает все уровни иерархии.

*<Table name="users" schema="public">* - таблица измерений.

*<Level name="UserId" visible="true" column="id" uniqueMembers="true">*

*<Level>* - уровень иерархии.

*name="UserId"* - название иерархии.

*column="id"* - столбец, по которому строится иерархия.

*uniqueMembers="true"* - все уникальные значения.

*<Measure name="count\_users" column="id" aggregator="count">*

*<Measure>* - тег, который описывает меру (сами данные, по которым проводятся операции, допустимые на множестве).

*name="count\_users"* - название меры.

*column="id"* - название столбца в таблице фактов.

*aggregator="count"* - операция агрегации на множестве.

Анализ данных с помощью Saiku Analytics

Плюсы Saiku Analytics:

1. Автоматическая генерация MDX запросов.
2. Совместимость с другими OLAP кубами (не только Mondrian).
3. Дружелюбный интерфейс.
4. Хорошие аналитические возможности.
5. Поддерживает экспорт результатов анализа данных в Excel, PDF, CSV.

Минусы:

1. Отображения по двум осям.
2. Иногда проблемы с обновлением данных.

С помощью Saiku Analytics получены ответы на следующие вопросы:

1. Сколько пользователей было зарегистрировано на сайте в определённый год? (Рисунок 8)

Время выполнения запроса - 0,01 сек.

2. Сколько на сайте зарегистрировано пользователей в возрастной

категории от 20-ти лет до 50-ти? (Рисунок 9)

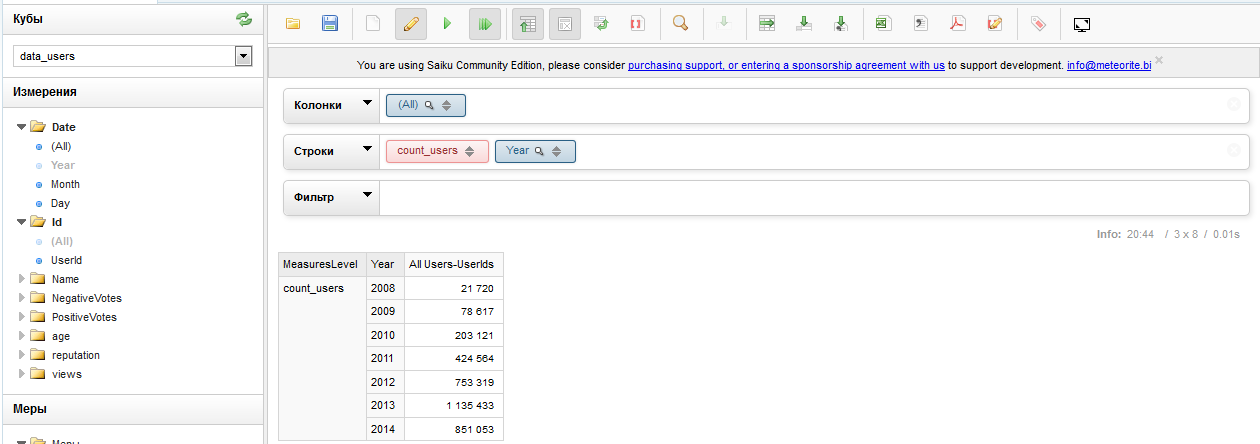


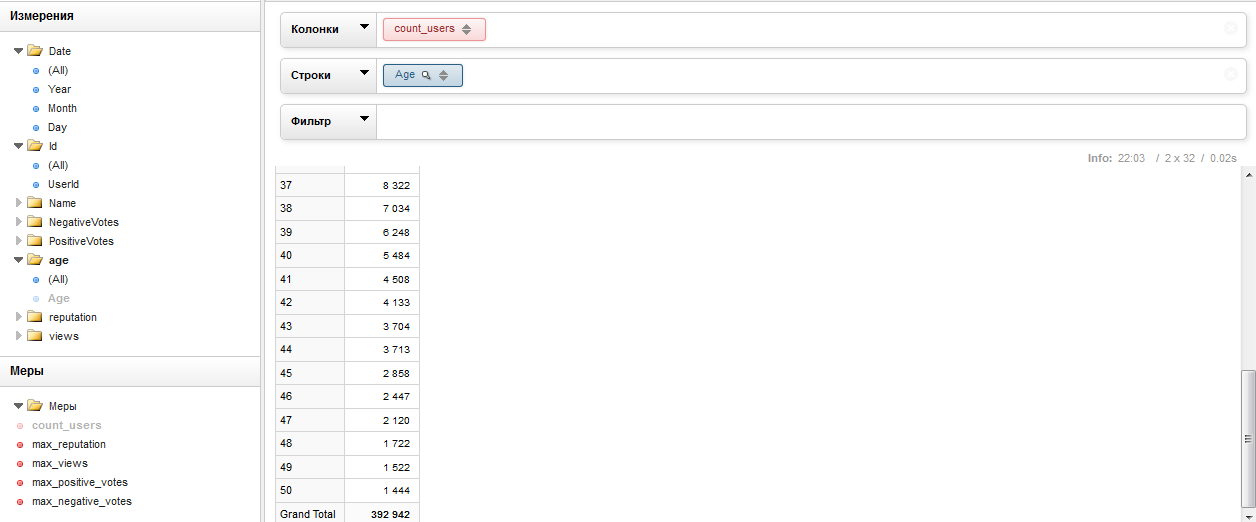
Рисунок 8 - Количество пользователей зарегистрировавшихся на сайте за определённый год 

Рисунок 9 - График: Количество пользователей зарегистрировавшихся на сайте в возрастной категории от 20-ти до 50-ти лет

Дальнейшие результаты анализа данных с помощью Saiku Analytics находятся в “Приложении C”.

В данной главе ознакомились со структурой OLAP куба.

Узнали, что куб создаётся из соединения таблиц с применением схемы звезды или схемы снежинки. В центре схемы звезды находится таблица фактов, которая содержит ключевые факты, по которым делаются запросы. Множественные таблицы с измерениями присоединены к таблице фактов. Эти таблицы показывают, как могут анализироваться агрегированныереляционные данные.

Также с помощью Pentaho Schema workbench и Saiku Analytics была создана схема куба и произведён анализ данных.

# **Глава 4. Разработка веб - приложения для отображения результатов анализа данных**

## 4.1 Технологии разработки веб - приложения

Для создания сайта выбран Play Framework.

Play Framework — это MVC веб-фреймворк для языков программирования Java и Scala. C помощью Play можно быстро разрабатывать сайты разной сложности. Play Framework делает разработку быстрее за счет автоматической перекомпиляции и перезагрузки кода, обеспечивает минимальное потребление ресурсов для высоко масштабируемых приложений.

Для построения графиков использовалась библиотека Chart.js.

Chart.js - это небольшая JavaScript библиотека предназначенная для создания графиков, в том числе и анимированных, объединяющая в себе простоту и легкость использования. Входит в пятнаднадцать самых популярных библиотек с открытым исходным кодом для построения графиков.

Для определения функциональных требований к системе была разработана диаграмма вариантов использования (Рисунок 10).

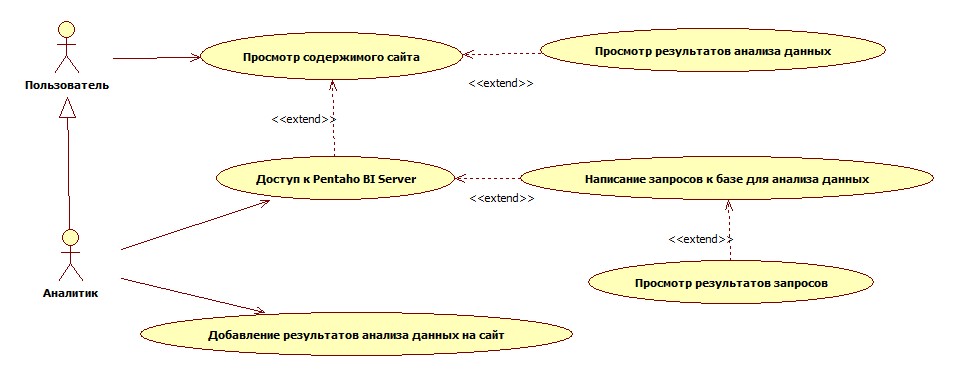


Рисунок 10 - Диаграмма вариантов использования

# **Список использованных источников**

1. Барсегян А.А. Методы и модели анализа данных: OLAP и Data Mining. - М: БХВ-Петербург, 2004. - 336 с.
2. Каширин И.Ю. Интерактивная аналитическая обработка данных в современных OLAP - системах [Электронный ресурс]. -<http://ecsocman.hse.ru/data/2011/11/28/1270195662/2009_2_%D1%81.12-19_%D0%9A%D0%B0%D1%88%D0%B8%D1%80%D0%B8%D0%BD.pdf>.
3. Файзрахманов Е. Г. Классификация OLAP по способу организации хранилища данных. Достоинства и недостатки [Электронный ресурс]. -<http://www.lib.tpu.ru/fulltext/c/2013/C04/063.pdf>
4. Kevin Roebuck. Data Quality: High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors. - М: Emereo Pty Limited, 2011. - 692 с.
5. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных [Электронный ресурс]. — М.: Справочно-информационный интернет-портал “devacademy”, 2014. — Режим доступа:- <http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/> ,свободный.
6. Инструментарий специалиста по большим данным: Apache Spark [Электронный ресурс]. — М.: Справочно-информационный интернет-портал “datareview.info”. — Режим доступа:- <http://datareview.info/article/instrumentariy-spetsialista-po-bolshim-dannyim-apache-spark/> ,свободный.
7. Работа с базами данных с помощью JDBC драйвера [Электронный ресурс]. — М.: Справочно-информационный интернет-портал “DevColibri”. — Режим доступа:- <http://devcolibri.com/477> ,свободный.
8. Многомерное представление данных. Общая схема организации хранилища данных. Характеристики, типы и основные отличия технологий OLAP и OLTP. Схемы звезда и снежинка. Агрегирование [Электронный ресурс]. — М.: Справочно-информационный интернет-портал “НОУ «ИНТУИТ»”,2003 – 2015 . — Режим доступа:- <http://www.intuit.ru/studies/courses/1145/214/lecture/5508?page=2> ,свободный.

# **Приложение А**

Создание триггера для таблицы users базы данных stackoverflowForOlap:

CREATE TRIGGER t\_user

AFTER INSERT OR UPDATE OR DELETE ON stackoverflow.public.users FOR EACH ROW EXECUTE PROCEDURE add\_to\_user ();

Создание триггерной функции для таблицы users базы данных stackoverflowForOlap:

CREATE OR REPLACE FUNCTION add\_to\_user() RETURNS TRIGGER AS $$

DECLARE

id\_user integer;

reputation integer;

displayname varchar(40);

location varchar (100);

views integer;

upvotes integer;

downvotes integer;

age smallint;

datekey integer;

BEGIN

IF TG\_OP = 'INSERT' THEN

id\_user = NEW.id;

reputation = NEW.reputation;

displayname = NEW.displayname;

location = NEW.location;

views = NEW.views;

upvotes = NEW.upvotes;

downvotes = NEW.downvotes;

age = NEW.age;

INSERT INTO stackoverflowForOlap.public.users (id, reputation,displayname,location,views,upvotes,downvotes,age,datekey) values (id\_user, reputation,displayname,location,views,upvotes,downvotes,age,null);

UPDATE stackoverflowForOlap.public.users cu

SET datekey = d.datekey

FROM dates d, stackoverflow.public.users u

WHERE year=date\_part('year',u.creationdate) AND

d.month=date\_part('month',u.creationdate) AND

d.day=date\_part('day',u.creationdate) AND

cu.id= id\_user;

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

id\_user = NEW.id;

reputation = NEW.reputation;

displayname = NEW.displayname;

location = NEW.location;

views = NEW.views;

upvotes = NEW.upvotes;

downvotes = NEW.downvotes;

age = NEW.age;

UPDATE stackoverflowForOlap.public.users set(id, reputation,displayname,location,views,upvotes,downvotes,age) = (id\_user, reputation,displayname,location,views,upvotes,downvotes,age);

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

id\_user = OLD.id;

DELETE FROM stackoverflowForOlap.public.users WHERE id=id\_user;

RETURN OLD;

END IF;

END;

$$ LANGUAGE plpgsql;

Создание триггера для таблицы postsбазы данных stackoverflowForOlap:

CREATE TRIGGER t\_post

AFTER INSERT OR UPDATE OR DELETE ON stackoverflow.public.posts FOR EACH ROW EXECUTE PROCEDURE add\_to\_posts ();

Создание триггерной функции для таблицы posts базы данных stackoverflowForOlap:

CREATE OR REPLACE FUNCTION add\_to\_posts() RETURNS TRIGGER AS $$

DECLARE

id\_post integer;

score integer;

viewcount integer;

owneruserid integer;

ownerdisplayname varchar(40);

tags varchar(150);

answercount integer;

commentcount integer;

favoritecount integer;

closeddate timestamp;

posttypename varchar(50);

BEGIN

IF TG\_OP = 'INSERT' THEN

id\_post = NEW.id;

score = NEW.score;

viewcount = NEW.viewcount;

owneruserid = NEW.owneruserid;

ownerdisplayname = NEW.ownerdisplayname;

tags = NEW.tags;

answercount = NEW.answercount;

commentcount = NEW.commentcount;

favoritecount = NEW.favoritecount;

closeddate = NEW.closeddate;

posttypename = NEW.posttypename;

INSERT INTO stackoverflowForOlap.public.posts (id, score, viewcount, owneruserid,

ownerdisplayname, tags, answercount,

commentcount, favoritecount, closeddate,

datekey, posttypename)

values (id\_post, score,viewcount,owneruserid,

ownerdisplayname,tags,answercount,

commentcount,favoritecount,closeddate,

null,posttypename);

UPDATE stackoverflowForOlap.public.posts cp

SET datekey = d.datekey

FROM dates d, stackoverflow.public.posts p

WHERE year=date\_part('year',p.creationdate) AND

d.month=date\_part('month',p.creationdate) AND

d.day=date\_part('day',p.creationdate) AND cp.id=id\_post;

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

id\_post = NEW.id;

score = NEW.score;

viewcount = NEW.viewcount;

owneruserid = NEW.owneruserid;

ownerdisplayname = NEW.ownerdisplayname;

tags = NEW.tags;

answercount = NEW.answercount;

commentcount = NEW.commentcount;

favoritecount = NEW.favoritecount;

closeddate = NEW.closeddate;

posttypename = NEW.posttypename;

UPDATE stackoverflowForOlap.public.posts set(id, score, viewcount, owneruserid,

ownerdisplayname, tags, answercount,

commentcount, favoritecount, closeddate,

posttypename) = (id\_post, score,viewcount,owneruserid,

ownerdisplayname,tags,answercount,

commentcount,favoritecount,closeddate,

posttypename);

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

id\_post = OLD.id;

DELETE FROM stackoverflowForOlap.public.posts WHERE id=id\_post;

RETURN OLD;

END IF;

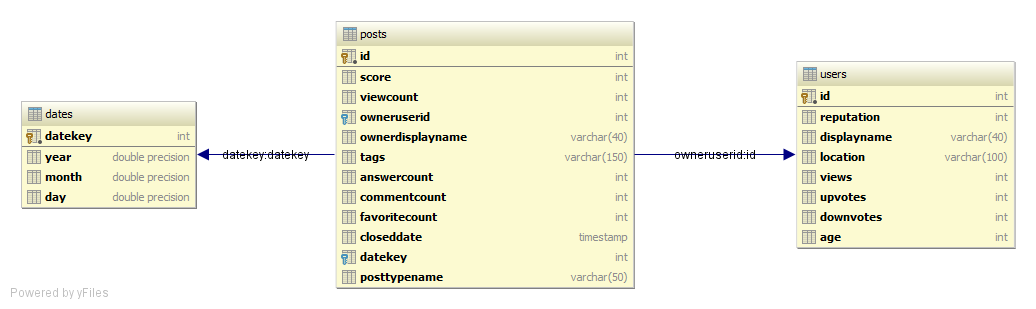
END;

$$ LANGUAGE plpgsql;

Аналогично для таблиц comments, tags, badges, votes.

# **Приложение В**

Ниже представлены физические - ER модели четырёх кубов:

Рисунок - Физическая - ER модель куба data\_posts

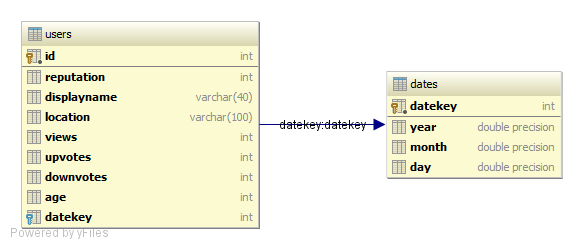


Рисунок - Физическая - ER модель куба data\_usesrs

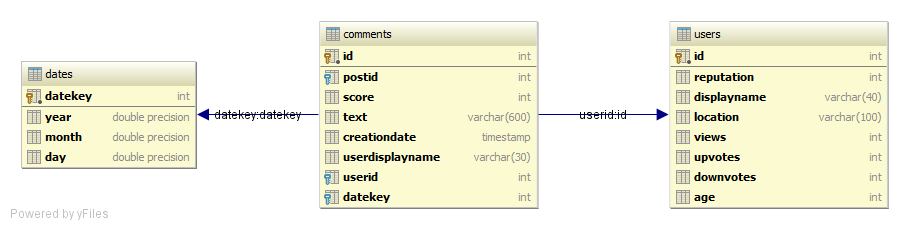
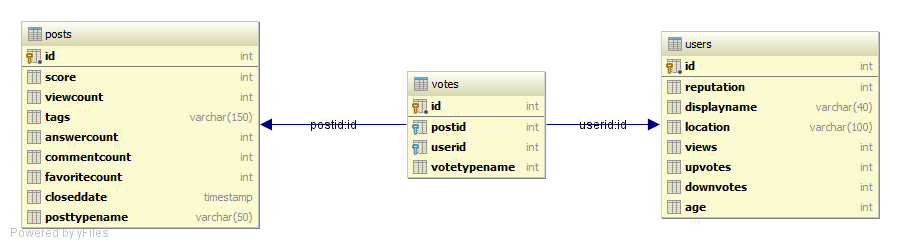
Рисунок - Физическая - ER модель куба data\_comments

Рисунок - Физическая - ER модель куба data\_votes

# **Приложение С**

Проанализировано сколько в среднем у пользователя постов, значков, комментариев:

* 11 постов;
* 6 значков;
* 17 комментариев.

Также проанализировано, что у поста в среднем 1 комментарий.

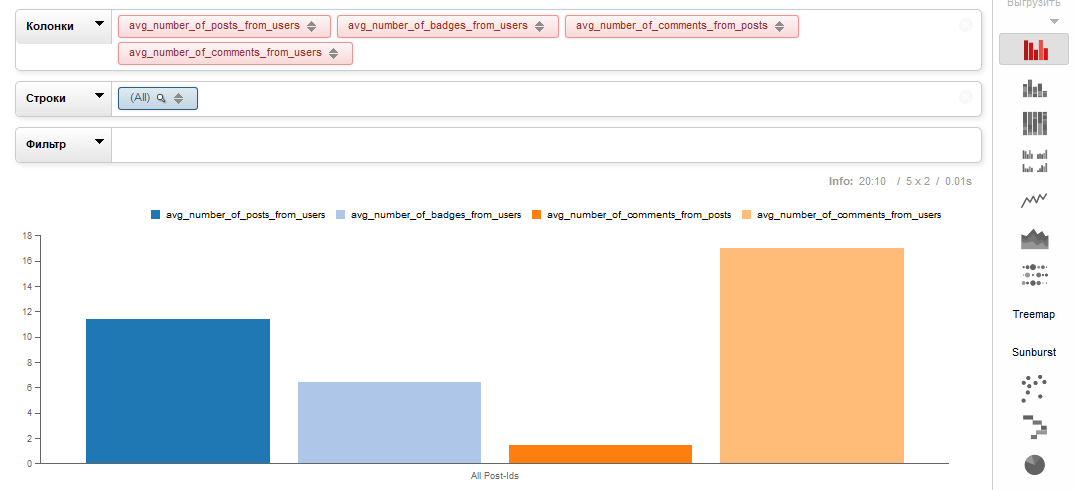
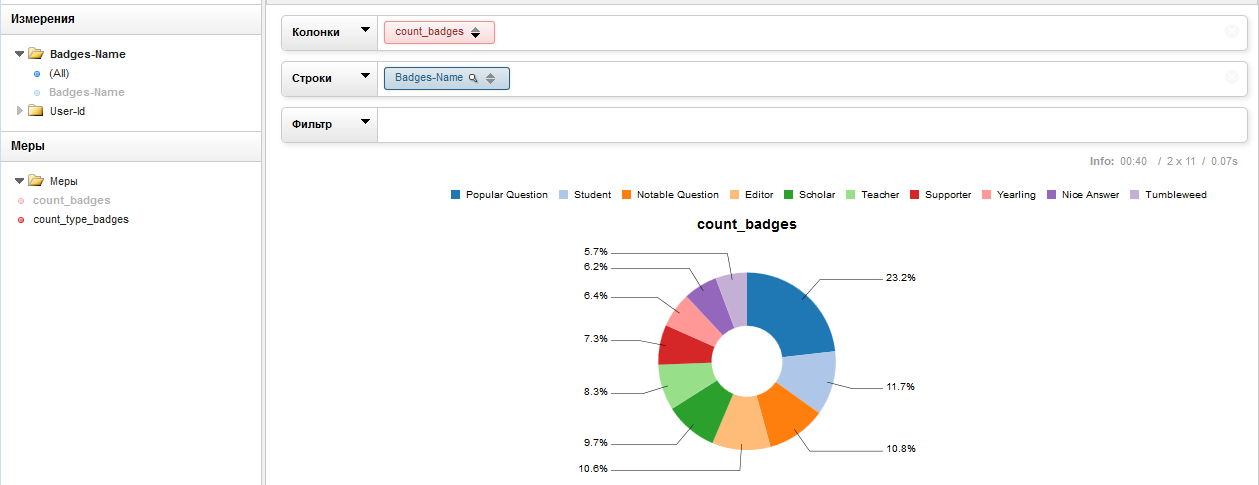


Рисунок - График: Среднее количество постов, значков, комментариев

Рисунок - График: 10 наиболее часто присваиваемых видов значков 

На графике, приведённом ниже, показано количество всех постов, количество постов отмеченных как “favorite” и количество закрытых постов

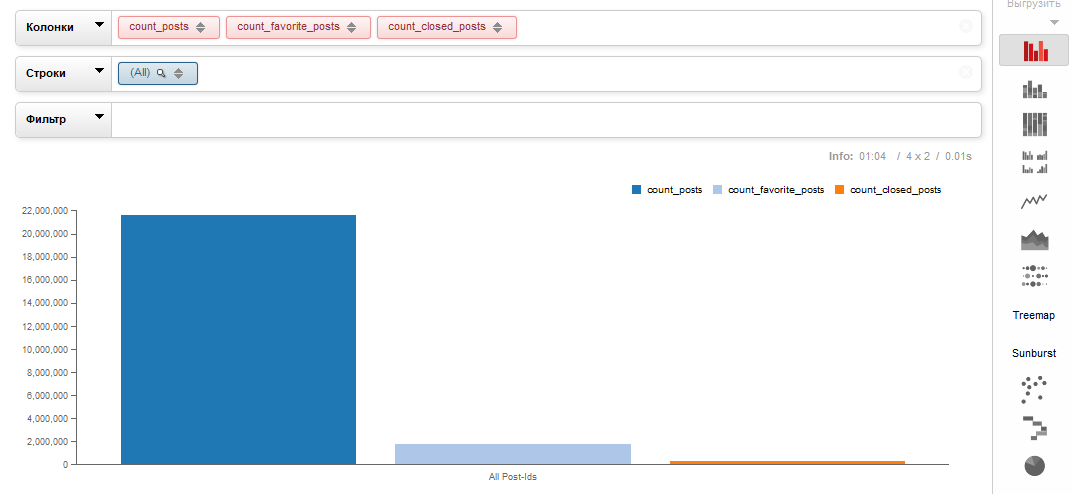
Рисунок - График: Количество всех постов, количество “favorite” постов и количество закрытых постов

Рисунок - Количество пользователей зарегистрировавшихся за определённый месяц- год и отсортированные по возрасту