АНАЛИЗ СУЩЕСТВУЮЩИХ НАПРАВЛЕНИЙ ПРИ РАЗРАБОТКЕ БАЗ ДАННЫХ

Сорока Антон Сергеевич – студент, кафедра компьютерной инженерии, Донецкий государственный технический университет, г. Донецк

Аннотация

Сорока А.С Завадская Т.В. Анализ существующих направлений при разработке баз данных.
В данной статье рассмотрены реляционные и нереляционные базы данных, когда и зачем нужно приметь их. Проведен сравнительный анализ преимуществ и недостатков между ними.

Ключевые слова: SQL, NoSQL, базы данных.

Введение. Каждая фирма или компания, или будь то обычный сайт должны хранить и обрабатывать информацию различного рода. Для облегчения систематизации данных были разработаны базы данных, которые занимают неотъемлемую часть в хранении и обработки полученной информации. Потребность в них растет т.к., в наше время большая часть времени и работы опирается на интернет, в следствии чего появляются новые базы данных, которые можно разделить на: реляционные и не реляционные.

Постановка проблемы. Одной из проблем при работе с серверной частью – это выбор использования нужной базы данных(БД), которая обеспечит быструю и верную работу с получаемыми данными. В среде IT существует должность архитектор БД, но не всегда данного специалиста нанимаю, т.к. он дорого обходится, а маленькой компания может обойтись небольшой БД, которую может сделать и обычный IT специалист. Сейчас очень много различных баз данных и трудно выбрать, что для какой задачи лучше подойдет.

Реляционные и нереляционные. Существует два основных направления в БД: реляционные (SQL) и нереляционные (NoSQL). Основные различия в них в том, как спроектированы они, какие типы данных используют и как хранят информацию.

Реляционные БД представляют собой реальные объекты (содержимое корзины товаров, данные о работнике и т.п.), которые структурированно хранятся и формат задан на этапе проектирования. Одни из популярных реляционных систем управления баз данных (СУБД) это:

- SQLite – файловая БД, которая встраивается в приложения.
- MySQL – самая популярная СУБД, которая легка в изучении и имеет множество различных драйверов (PDO, MySQLi).
- PostgreSQL – Продвинутая СУБД, которая пытается соответствовать SQL стандарту ANSI/ISO.

Нереляционные БД хранят в себе информацию виде иерархических структур данных (например, документные БД), с произвольным количеством атрибутов. Например, то что реляционные БД будет разбито на иерархию таблиц, в нереляционной может хранится в целой сущности. В нереляционные СУБД популярны следующие:

- MongoDB – документированная СУБД с открыт исходным кодом. Имеет классификацию NoSQL, использует JSON-подобные документы.

- CouchDB – документо-ориентированная СУБД, не требующая описания системы данных.
- Voldemort – это СУБД имеет распределенное хранилище «ключ-значение», которая не имеет единой точки сбоя и нет центрального пункта кординации.

Выбор СУБД. При выборе между SQL и NoSQL следует представлять, какие данные будут храниться. Для сравнения можно взять и рассмотреть преимущества и недостатки у MySQL и MongoDB.

MySQL – довольно старая СУБД, но очень популярна в использовании. Спомощью функций данной СУБД уже мало, что делают т.к. не ее поддерживают, а используют драйвера такие как PDO и MySQLi, которые дают доступ к ней и улучшают ее функции.

Функции mysql являются процедурами и используют ручное экранирование. Данные функции уже не используют, т.к. с появлением PHP 5 начали возникать проблемы различного рода. Разработка была остановлена на Mysql 4.1.3, это расширение не поддерживает транзакции, имеет уязвимости при подстановке в запрос. Mysql функциям пришли на замену: MySQLi и PDO.

MySQLi является заменой функций mysql, с объектно-ориентированными и процедурными версиями. Он поддерживает подготовленные заявления. В отличие от PDO является прямым наследником, API Mysql. Имеет следующие улучшения: объектно-ориентированный интерфейс, поддержка множественных операторов, поддержка транзакций, расширенная поддержка отладки.

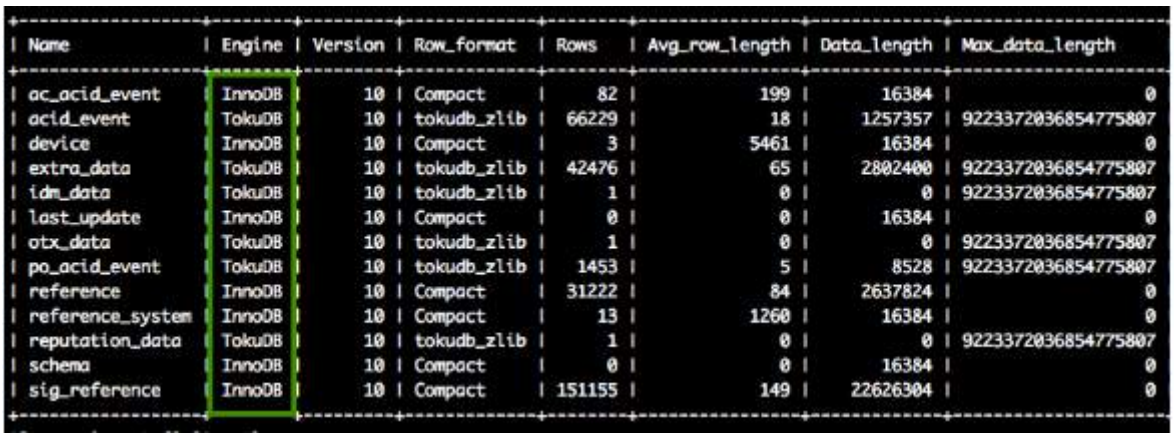
PDO (PHP Data Objects) - это общий уровень абстракции базы данных с поддержкой MySQL среди многих других баз данных. Он предоставляет подготовленные заявления и значительную гибкость в отношении того, как данные возвращаются. Очень хорошо подходит, если нужно сменить базу данных. Имеет возможность работать с несколькими БД, что позволяет “забыть” какую базу использовали, за нас это будут делать специальные драйвера. Из методов борьбы с sql-инъекциями появилось prepared. Prepared statement — это заранее скомпилированное SQL-выражение, которое может быть многократно выполнено путём отправки серверу лишь различных наборов данных.

Преимущества:

- Строго структурированные данные.
- Соответствие требованиям Atomicity, Consistency, Isolation, Durability.
- Обеспечения целостности БД.
- Если нужно простое отображение данных для пользователя (Рис. 1 пример вывода в MySQL).

Недостатки:

- Медленный доступ к данным.
- Трудоемкость разработки.
- Не всегда предметную область можно представить введи таблиц.



Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Max_data_length
ac_acid_event	InnoDB	10	Compact	82	199	16384	0
acid_event	TokuDB	10	tokudb_zlib	66229	18	1257357	922337203685477580
device	InnoDB	10	Compact	3	5461	16384	0
extra_data	TokuDB	10	tokudb_zlib	42476	65	2802400	922337203685477580
idm_data	TokuDB	10	tokudb_zlib	1	0	0	922337203685477580
last_update	InnoDB	10	Compact	0	0	16384	0
otx_data	TokuDB	10	tokudb_zlib	1	0	0	922337203685477580
po_acid_event	TokuDB	10	tokudb_zlib	1453	5	8528	922337203685477580
reference	InnoDB	10	Compact	31222	84	2637824	0
reference_system	InnoDB	10	Compact	13	1260	16384	0
reputation_data	TokuDB	10	tokudb_zlib	1	0	0	922337203685477580
schema	InnoDB	10	Compact	0	0	16384	0
sig_reference	InnoDB	10	Compact	151155	149	22626304	0

Рисунок 1 пример вывода данных в MySQL

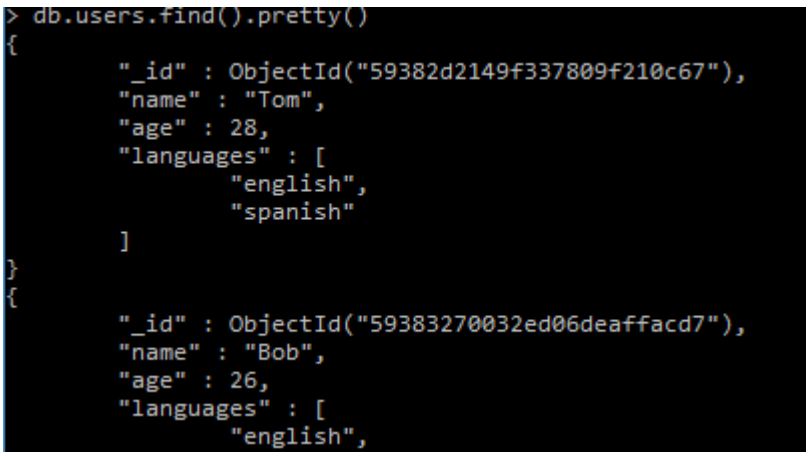
MongoDB – очень быстра в разработке, что дает возможность менять все местами. Нет необходимости привязки поля к формату, а также имеется возможность реализации схемы данных в приложении. Схема данных реализуется с помощью JSON-документ, позволяя разработчику не раскладывать по таблицам различные данные. Такие нереляционные СУБД хорошо подходят к недолгому хранению информации и быстры в работе с данными.

Достоинства:

- Простата работы.
- Простой синтаксис запросов, что приводит к меньшему количеству ошибок.
- Отсутствие SQL.
- Нет ограничений на тип данных.
- Скорость работы и масштабируемость.

Недостатки:

- Отображение данных для пользователя затруднительны (Рис. 1 пример вывода в MySQL).
- Сильная привязка к одной СУБД.
- Трудности перехода с одной нереляционные СУБД на другую
- Множества инструментов для работы с БД, необходимо разрабатывать самому.



```
> db.users.find().pretty()
{
        "_id" : ObjectId("59382d2149f337809f210c67"),
        "name" : "Tom",
        "age" : 28,
        "languages" : [
                "english",
                "spanish"
        ]
}
{
        "_id" : ObjectId("59383270032ed06deaffacd7"),
        "name" : "Bob",
        "age" : 26,
        "languages" : [
                "english",
```

Рисунок 2 пример вывода данных в MongoDB

Вывод. Можно начать работу с любой СУБД, а позже уточнив требования у заказчика, перейти на другую СУБД. Для экономии времени разумно планировать приблизительно, что будет в БД. Например для построения банковской структуры следует выбирать реляционную СУБД, т.к. она надежней хранит данные. В тех случая, когда данные должны хранится небольшой период, то лучше использовать нереляционную СУБД. В качестве примера можно привести стрименговую платформу, которая удаляет данные через определенное время. Рассмотрев SQL и NoSQL можно вывести признаки проектов для выбора СУБД .

Для реляционной (SQL) СУБД:

- Имеется логические данные, которые определены заранее. К примеру данные о заказе: код товара, цена, название, контент, изображения.
- Очень важна целостность данных.
- Нужна технология которая хорошо устоялась, имеет опытную техническую поддержку и долгая в использовании.

Для нереляционной (NoSQL) СУБД:

- Информация о данные мало известна, нет четкого понимая что будет хранится. Возможно будут появятся новые данные с развитием проекта.
- Цель проекта может поменяться или корректироваться со временем.
- Основное требование к БД скорость и масштабируемость.

Так как нет идеальной системы управления базами данных, очень часто используют совместную работу SQL и NoSQL, но в таких случаях приходится чем то жертвовать при разработки для устроения ошибок.