**Задание 1. СУБД**

**Кейс**

Крупная строительная компания, которая также занимается проектированием и девелопментом, решила создать правильную архитектуру для работы с данными. Ниже представлены задачи, которые необходимо решить для каждой предметной области.

Какие типы СУБД, на ваш взгляд, лучше всего подойдут для решения этих задач и почему?

* 1. Бюджетирование проектов с дальнейшим формированием финансовых аналитических отчётов и прогнозирования рисков. СУБД должна гарантировать целостность и чёткую структуру данных.

Ответ: Во первых стоит определиться с архитектурой так как субд должна гарантировать целостность и четкую структуру данных, а так же учитывая что мы имеем дело с крупной строительной компанией я думаю что нам необходима Клиенто-серверная субд которая как раз гарантирует согласованность и целостность данных благодаря своей структуре взаимодействия.

Из всех моделей мы будем использовать именно реляционную для конкретного случая так как она представляет собой четкую фиксированную структуру на основе математических понятий которая в свою очередь как раз таки и покроет задачи по формированию финансовых аналитических отчетов и прогнозирования рисков

Учитывая тот факт что мы используем реляционный тип субд в нашем случае мы будем использовать mySQL или postgres sql или SQL server, а соответственно по теореме CAP мы попадаем только в подкласс CA симбиоз Согласованности данных и доступности

Если бы задача ставилась при учете нынешних условий и требованиям безопасности со стороны ФСТЭК а крупная строительная компания откровенно говоря не хотела бы тратить и выделять большое финансирование на конкретный проект я бы сразу убрал возможность использования SQL Server так как во 1х  
SQL Server принадлежит и требует лицензии со стороны своего непосредственного разработчика Microsoft

Во 2х ввиду того что SQL Server не является программным обеспечением с открытым исходным кодом и не предназначен (в виду требований ФСТЭК) для работы на отечественных ОС разработанных на базе linux мой выбор бы лежал в сторону postgresSQL или mySQL.  
  
1.1.\* Хеширование стало занимать длительно время, какое API можно использовать для ускорения работы?  
  
Ответ: Если мы под API воспринимаем понятия Application Programming interface доступный функционал настроек допустим в mySQL то для решения данной задачи можно использовать:

1) Проверка настроек индексов для столбцов которые мы используем для хеширования

2) Оптимизация запросов

3) Использование кэширования для хеширования данных во избежания повторного вычисления хеша для одних и тех же данных

4) Пересмотр выбора алгоритма хеширования некоторые из них могут быть более эффективными  
  
все отмеченные моменты выше как раз соответствуют решению вопроса конкретно только через использование API но стоит отметить что есть моменты в которых используется решение через усовершенствование аппаратной части на которой работает наша СУБД а именно:  
5) Аппаратное ускорение  
6) Распределенные вычисления  
7) Использование параллелизма

1.2. Под каждый девелоперский проект создаётся отдельный лендинг, и все данные по лидам стекаются в CRM к маркетологам и менеджерам по продажам. Какой тип СУБД лучше использовать для лендингов и для CRM? СУБД должны быть гибкими и быстрыми.

Ответ: Бытует мнение что не нужно хранить лендинг в бд так как каждый раз для редактирования лендинга в бд надо каждый раз входить в бд

Но если всетаки – надо то смотрим в сторону гибких и быстрых субд из гибких и быстрых нам подходит nosql выбираем его

Сам по себе лендинг представляет собой веб-страницу для конкретной маркетинговой и рекламной кампании

Так зачем нам субд для лендинга?

1. СУБД должна хранить информацию используемую на лендинге   
   изображения, текстовые блоки, другие элементы контента при чем не только статические элементы но и динамические такие как баннеры изображение товаров и тд.
2. Информация “Обратной связи”

Вся информация оставляемая посетителями лендинга в процессе взаимодействия

По Видам мы должны учесть что наша субд должна быть достаточно гибкой для хранения различной информации она должна обладать одновременно следующими видами:  
1) Ключ-значение (для хранение изображений, специализированных файловых систем, кэшей)

2) Документо-ориентированность (для управления контентом и каталогов)

3)Матричность (для индексации веб-страниц и других задач связанных с обработкой больших данных)  
  
по трем параметрам нам подходит MongoDB но она запрещена в использовании на территории РФ и белоруссии поэтому нам прийдется искать опен-соурс решения

1) документо-ориентированность выбираем CouchDB

2) матричность – Cassandra

3)Ключ – значения – RocksDB  
  
но нужно учесть тот факт что нам прийдется управлять одновременно сразу несколькими базами данных и возможно будут иметься проблемы с согласованностью между ними

Что же касательно использования базы данных для CRM тут все просто мы будем использовать графовую СУБД так как нам необходимо анализировать предпочтения пользователей и идентифицировать их по группам для последующей работы над тем чтобы наших лидов превратить в клиентов  
Например мы можем использовать: Neo4j  
  
1.2.\* Можно ли эту задачу закрыть одной СУБД? И если да, то какой именно СУБД и какой реализацией?  
  
Ответ: Из опенсорс проектов есть вариант с ArangoDB онаобладает всеми необходимыми нам видами для закрытия поставленных задач а так же поддерживает ACID-транзакции что является важным для CRM-систем

* 1. Отдел контроля качества решил создать базу по корпоративным нормам и правилам, обучающему материалу и так далее, сформированную согласно структуре компании. СУБД должна иметь простую и понятную структуру.

Ответ: Простую и понятную структуру с использованием таблиц и связей имеют реляционные СУБД поэтому в конкретном случае мы будем использовать их

* 1. Департамент логистики нуждается в решении задач по быстрому формированию маршрутов доставки материалов по объектам и распределению курьеров по маршрутам с доставкой документов. СУБД должна уметь быстро работать со связями.

Ответ: Реляционные базы будут достаточно громоздкими и не смогут обеспечить быстроту работы со связами если говорить о скорости выполнения и скорее всего маршруты всегда будут видоизменятся, а это в свою очередь говорит о том что субд должны быть гибкими то в конкретном случае лучше всего смотреть в сторону noSQL.

### Задание 2. Транзакции

2.1. Пользователь пополняет баланс счёта телефона, распишите пошагово, какие действия должны произойти для того, чтобы транзакция завершилась успешно. Ориентируйтесь на шесть действий.

1)Определение текущего баланса счета на банковской карте - достаточно ли тех средств которые пользователь хочет послать на счет телефона?  
2) Указание счета телефона на который будут переведены деньги   
3) Отправка перевода

4)Создание отдельной сессии для конкретной передачи сведений о транзакции

5)Внесение сведений о транзакции на отдельный сервер. В случае если транзакция не достигает своего адресата и не было получено информации об обратной связи о получении средств транзакция временно приостанавливается и данная информация хранится на сервере. Транзакция является завершенной только при достижении финансовых средств адресату.

6) Фиксация транзакции перевода с банковского счета на счет телефона  
7) Определение баланса счета телефона сколько располагалось средств на телефоне до начала транзакции  
8) Прибавление к счету средств которые были учтены до транзакции

### Задание 3. SQL vs NoSQL

3.1. Напишите пять преимуществ SQL-систем по отношению к NoSQL.

Ответ:

1) SQL в сравнении с noSQL имеет простую и понятную структуру

2) SQL в сравнении с noSQL обеспечивает согласованность данных т.е.

Данные объединенные в четкую структуру с начала сессии и по окончанию своей работы остаются между собой согласованы и созависимыми.

3) SQL имеет совместимость с популярными языками программирования что позволяет прикручивать к SQL различные интересности

4) SQL легко изучить и использовать новичкам в сравнении с noSQL

5) Базы данных SQL обычно более сложны чем базы данных NoSQL и следуют правилам ACID обеспечивая безопасность транзакции - все или ничего

### Задание 4. Кластеры

Необходимо производить большое количество вычислений при работе с огромным количеством данных, под эту задачу выделено 1000 машин.

На основе какого критерия будете выбирать тип СУБД и какая модель распределённых вычислений здесь справится лучше всего и почему?

Ответ:

Стоит задаться вопросом о масштабируемости в случае с sql требуется задаваться вопросами о горизонтальной масштабируемости тогда как в случае с nosql речь идет о вертикальной масштабируемости скорее всего если нам выделяется под решение 1000 машин и это является пределом то скорее всего мы будем смотреть в сторону noSQL решений, но опять же все зависит от поставленной задачи большое количество вычислений при работе с огромным количеством данных можно производить как SQL так и noSQL СУБД тем не менее учитывая что 1000 машин это предел мы будем смотреть в сторону noSQL так как её легче масштабировать и она является достаточно гибкой для работы с большим количеством вычислений при работе с огромным количеством данных так же нужно учесть тот факт что в конкретном случае вообще в задачи отсутствует сервер

То есть мы напрочь исключаем существования центрального узла поэтому:  
из всех существующих моделей распределенных вычислений мы выделяем следующие:  
1)Модель Распределеной Репликации  
2)Модель Шардинга  
3)Модель Peer-to-Peer

Модель Распределеной Репликации мы отклоняем так как мы ограничены 1000 машинами

Модель шардинга и модель Peer-to-Peer – более выгодный вариант но нам так же в ограниченных условиях необходимо как то обеспечивать хоть какую то сохранность и безопасность данных  
  
Если же мы говорим о модели CAP то одним из важнейших моментов будет являться устойчивость к разделению так как у нас нет основного сервера и все взаимодействие будет построено на обращении данных между узлами либо PA либо PC на выбор все таки на мой взгляд нужно смотреть в сторону систем класса CP так как именно она обеспечивает целостный результат и способна функционировать в условиях распада.  
  
поэтому мой ответ будет таков:   
Мы будем использовать noSQL СУБД по модели CP – Согласованность данных и устойчивость к разделению