Лекция 13

Введение в NoSQL-системы

Современные NoSQL-системы.

Модели данных

- Иерархическая
- Сетевая
- Реляционная
- Объектно-ориентированная

NoSQL:

- Документ-ориентированная (MongoDB, CouchDB ,…
- Хранилища «ключ-значение» (riak, redis, ….
- Графовая (Neo4, OrientDB, …
- Столбцовая (Cassandra, Hbase, Hypertable ,…
- др.

Рейтинг СУБД издания DB-Engines (https://db-engines.com/en/ranking)

реляционные СУБД используют 99,5% респондентов

381 systems in ranking, December 2021

Rank Dec 2021	Rank Nov 2021	Rank Dec 2020	DBMS	Database Model	Score Dec 2021	Score Nov 2021	Score Dec 2020
1.	1.	1.	Oracle	Relational, Multi-model	1281.74	+9.01	-43.86
2.	2.	2.	MySQL	Relational, Multi-model	1206.04	-5.48	-49.41
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	954.02	-0.27	-84.07
4.	4.	4.	PostgreSQL	Relational, Multi-model	608.21	+10.94	+60.64
5.	5.	5.	MongoDB	Document, Multi-model	484.67	-2.67	+26.95
6.	6.	↑ 7.	Redis	Key-value, Multi-model	173.54	+2.04	+19.91
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	167.18	-0.34	+6.74
8.	8.	8.	Elasticsearch	Search engine, Multi-model	157.72	-1.36	+5.23
9.	9.	9.	SQLite	Relational	128.68	-1.12	+7.00
10.	↑ 11.	↑ 11.	Microsoft Access	Relational	125.99	+6.75	+9.25
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.20	-1.68	+0.36
12.	12.	12.	MariaDB	Relational, Multi-model	104.36	+2.17	+10.75
13.	13.	13.	Splunk	Search engine	94.32	+2.02	+7.32
14.	↑ 15.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model	83.25	+1.93	+13.76
15.	↓ 14.	15.	Hive	Relational	81.93	-1.38	+11.66
16.	16.	↑ 17.	Amazon DynamoDB	Multi-model	77.63	+0.64	+8.51
17.	↑ 18.	↑ 41.	Snowflake	Relational	71.03	+6.84	+58.12
18.	↓ 17.	↓ 14.	Teradata	Relational, Multi-model	70.29	+0.71	-3.54
19.	19.	19.	Neo4j	Graph	58.03	+0.05	+3.40
20.	↑ 22.	↑ 21.	Solr	Search engine, Multi-model	57.72	+3.87	+6.48
21.	↓ 20.	↓ 20.	SAP HANA	Relational, Multi-model	54.58	-0.95	+2.08
22.	↓ 21.	22.	FileMaker	Relational	53.86	-0.36	+6.16

Рейтинг СУБД 2023 издания DB-Engines

Rank			DBMS	Database Model	Score		
Nov 2023	Oct 2023	Nov 2022			Nov 2023	Oct 2023	Nov 2022
1.	1.	1.	Oracle	Relational, Multi-model	1277.03	+15.61	+35.34
2.	2.	2.	MySQL	Relational, Multi-model	1115.24	-18.07	-90.30
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	911.42	+14.54	-1.09
4.	4.	4.	PostgreSQL	Relational, Multi-model	636.86	-1.96	+13.70
5.	5.	5.	MongoDB	Document, Multi-model	428.55	-2.87	-49.35
6.	6.	6.	Redis	Key-value, Multi-model	160.02	-2.95	-22.03
7.	7.	7.	Elasticsearch	Search engine, Multi-model	139.62	+2.48	-10.70
8.	8.	8.	IBM Db2	Relational, Multi-model	136.00	+1.13	-13.56
9.	9.	↑ 10.	SQLite	Relational	124.58	-0.56	-10.05
10.	10.	↓ 9.	Microsoft Access	Relational	124.49	+0.18	-10.53
11.	11.	↑ 12.	Snowflake	Relational	121.00	-2.24	+10.84
12.	12.	↓ 11.	Cassandra	Wide column, Multi-model	109.17	+0.34	-8.96
13.	13.	13.	MariaDB	Relational, Multi-model	102.09	+2.43	-2.82
14.	14.	14.	Splunk	Search engine	97.32	+4.95	+3.10
15.	15.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model	83.17	+2.24	-0.49
16.	16.	↓ 15.	Amazon DynamoDB	Multi-model	82.24	+1.32	-3.16
17.	17.	↑ 19.	Databricks	Multi-model	77.22	+1.40	+16.33
18.	18.	↓ 17.	Hive	Relational	68.64	-0.54	-13.25
19.	↑ 20.	↑ 22.	Google BigQuery	Relational	59.31	+2.74	+5.18
20.	↓ 19.	↓ 18.	Teradata	Relational, Multi-model	57.33	-1.23	-7.90
21.	21.	21.	FileMaker	Relational	52.44	-0.88	-1.87
22.	↑ 23.	↓ 20.	Neo4j	Graph	49.70	+1.26	-7.60

Что такое «большие данные»?

- «Большие данные» характеризуются объемом, разнообразием и скоростью, с которой структурированные и неструктурированные данные поступают по сетям передачи в процессоры и хранилища, наряду с процессами преобразования этих данных в ценную для бизнеса информацию (Исследовательская компания Gartner)
- **Характеристики больших данных:**
  - **Объем**
  - **Разнообразие** (неструктурированность даных)
  - **Скорость** (поступление и извлечение данных)
  - [Ценность]

Недостатки реляционной модели

- **ACID** свойства (атомарность, согласованность, изолированность, долговечность) не позволяют наращивать производительность реляционных систем

Решение проблемы производительности реляционных СУБД

- Использовать более мощное оборудование (вертикальное масштабирование)
- Оптимизировать запросы, проанализировав планы их исполнения, и создать дополнительные индексы.
- Денормализация схемы БД

- **noSQL решения**

- **newSQL решения**

noSQL решения

- Термин «NoSQL» впервые был использован в 1998 году для описания реляционной базы данных, не использовавшей SQL
- Термин "NoSQL" прижился, но никогда не имел строгого определения. Исходное название семинара относилось к "распределенным нереляционным базам данных с открытым исходным кодом».
- Эти эпитеты относятся к базам данных **Voldemort, Cassandra, Dynomite, HBase, HypertaЬle, CouchDB и MongoDB**.
- Популярность NoSQL стал набирать в 2009 г, в связи с появлением большого количества веб-стартапов, для которых важнейшей задачей является поддержание постоянной высокой пропускной способности хранилища при неограниченном увеличении объема данных.

Классификация NoSQL решений

Хранилища ключ-значение.

Отличительной особенностью является простая модель данных — ассоциативный массив или словарь, позволяющий работать с данными по ключу. Основная задача подобных хранилищ — максимальная производительность, поэтому никакая информации о структуре значений не сохраняется.

Документ-ориентированные хранилища.

- Модель данных подобных хранилищ позволяет объединять множество пар ключ - значение в абстракцию, называемую «документ». Документы могут иметь вложенную структуру и объединяться в коллекции.
- Однако это скорее удобный способ логического объединения, т.к. никакой жесткой схемы у документов нет и множества пар ключ-значение, даже в рамках одной коллекции, могут быть абсолютно произвольными.
- Работа с документами производится по ключу, однако существуют решения, позволяющие осуществлять запросы по значениям атрибутов.

Классификация NoSQL решений

Колоночные(столбцовые) хранилища.

- Основой модели данных является колонка, число колонок для одной таблицы может быть неограниченным
- Этот тип кажется наиболее схожим с традиционными реляционными СУБД. Модель данных хранилищ подобного типа подразумевает хранение значений как неинтерпретируемых байтовых массивов, адресуемых кортежами **<ключ строки, ключ столбца, метка времени>.**
- Колонки по ключам объединяются в семейства, обладающие определенным набором свойств.

Хранилища на графах.

- Подобные хранилища применяются для работы с данными, которые естественным образом представляются графами (например, социальная сеть).
- Модель данных состоит из вершин, ребер и свойств. Работа с данными осуществляется путем обхода графа по ребрам с заданными свойствами.

Хранилища ключей и значений

- Хранилище ключей и значений (КЗ-хранилище) – простейшая из всех рассматриваемых моделей. Как следует из названия, КЗ-хранилище сопоставляет значения ключам, как словарь (или хеш-таблица) в любом популярном языке программирования.
- Некоторые КЗ хранилища допускают в качестве значений составные типы данных, например хеши или списки, но это необязательно.
- Есть реализации, в которых ключи можно перебирать, но это также считается дополнительным бонусом. Примером КЗ-хранилища можно считать файловую систему, если рассматривать путь к файлу как ключ, а его содержимое – как значение.
- Поскольку от КЗ-хранилища требуется так мало, то базы данных этого типа могут демонстрировать невероятно высокую производительность, но в общем случае бесполезны, когда требуются сложные запросы и агрегирование.
- Как и в случае реляционных СУБД, имеется много продуктов с открытым исходным кодом. Из наиболее популярных отметим memcached (и родственные ему memcachedb и membase), Voldemort , redis и riak.

Столбцовые базы данных

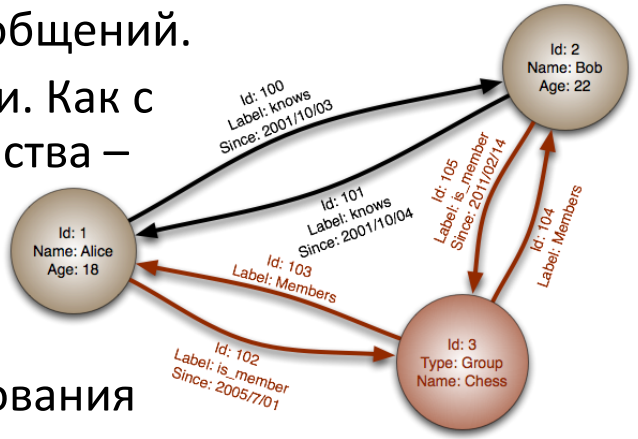
- Столбцовые, или ориентированные на хранение данных по столбцам базы данных получили свое название благодаря одному существенному аспекту дизайна: данные, принадлежащие одному столбцу (в смысле двумерных таблиц) хранятся рядом. Напротив, в строковых базах данных (к числу которых относятся реляционные), рядом хранятся данные, принадлежащие одной строке.
- В столбцовых базах данных добавление нового столбца обходится дешево и производится построчно. В каждой строке набор столбцов может быть разным, возможно даже, что в некоторых строках столбцов вообще нет, и, значит, таблица может быть *разреженной* без накладных расходов на хранение null-значений.
- С точки зрения структуры, столбцовые базы данных занимают промежуточное положение между реляционными СУБД и КЗ-хранилищами.
- На рынке столбцовых баз данных конкуренция меньше, чем среди реляционных СУБД и хранилищ ключей и значений. Наиболее популярны три системы: HBase, Cassandra и Hypertable.

Документо-ориентированные БД

- В документо-ориентированных, или просто документных базах данных хранятся – естественно – документы. В двух словах документ – это некий аналог хеша, в котором имеется поле уникального идентификатора, а в качестве значения могут выступать данные произвольного типа, в том числе другие хеши.
- Документы могут содержать вложенные структуры и обладают высокой гибкостью, что делает их пригодными для применения в разных предметных областях.
- Система налагает немного ограничений на входные данные при условии, что они удовлетворяют базовым требованиям к представимости в виде документа.
- В различных документных базах данных применяются различные подходы к индексированию, формулированию произвольных запросов, репликации, обеспечению согласованности и другим аспектам.
- Для правильного выбора системы нужно хорошо понимать эти различия и их влияние на конкретный сценарий использования. Два основных игрока на поле документных баз данных с открытым исходным кодом – MongoDB и CouchDB.

Графовые базы данных

- **Гра́фовая база данных** — разновидность баз данных с реализацией сетевой модели в виде графа и его обобщений.
- Графовая база состоит из узлов и связей между ними. Как с узлами, так и со связями можно ассоциировать свойства – пары ключ-значение, – в которых хранятся данные.
- Истинная сила графовых баз данных заключается в возможности обхода узлов, следуя связям.
- При использовании графовой модели для моделирования тех же самых данных мы моделируем все объекты как узлы, а отношения между ними - как связи; эти связи имеют тип на направленность.



•N**eo4J**

•Операция, на которой другие базы данных часто сдаются, – это обход данных со ссылками на себя или с другими сложно устроенными связями. Именно здесь достоинства Neo4J проявляются во всем блеске.

•Преимущество графовой базы данных в том и состоит, что обеспечивается быстрый просмотр узлов и связей для поиска нужных данных.

•Такие базы часто используются в социальных сетях и завоевали признание за свою гибкость/

Time Series СУБД

- **Time Series СУБД.** Такие СУБД оптимизированы для хранения данных временных меток или временных рядов. Данные временных рядов могут содержать измерения или события, которые отслеживаются, собираются или объединяются в течение определенного периода времени.
- Это могут быть данные, собранные с датчиков отслеживания движения, метрики JVM из приложений Java, рыночные торговые данные, сетевые данные, ответы API, время безотказной работы процессов и т.д.
- Данные хранятся с отметками времени (это ключевое), которые индексируются и записываются таким образом, чтобы можно было запрашивать данные этих временных рядов намного быстрее, чем при использовании классической реляционной базы данных.
- **Наиболее известные СУБД такого типа:** InfluxDB, Kdb+, Prometheus, TimescaleDB, QuestDB, AWS, Timestream, OpenTSDB, GridDB.
- **Когда выбирать Time series СУБД**: Основная область применения таких СУБД — это системы мониторинга, сбора телеметрии и финансовые системы.

Spatial СУБД

- Этот тип СУБД оптимизирован и предназначен для работы с объектами определенными в геометрическом пространстве. Это могут быть простые объекты (точки, линии, многоугольники) или сложные (3D-объекты, топологические покрытия, линейные сети).
- Реализован набор специальных функций, позволяющих проводить с объектами операции создания, трансформации, измерения (расстояния, площади, объема), вычисления (пересечений / соприкосновений) и выборки по определенным критериям.
- Существуют специальные индексы, оптимизирующие работу с объектами, и специальный стандартизированный SQL/MM язык.
- **Известные представители этого типа СУБД:** Oracle Spatial, Microsoft SQL, SpatialLite, PostGIS (https://sbercloud.ru/ru/warp/blog/postgis-about)
- **Когда выбирать Spatial СУБД**
- Если строите GIS-решения. Если планируете не просто хранить, но и работать с геометрическими объектами на уровне СУБД.
- **Когда не выбирать Spatial СУБД**
- Если планируете просто хранить геометрические объекты в виде координат.

Search engines СУБД

- **Поисковые системы**, предназначенные для поиска содержимого данных. В дополнение к общей оптимизации для этого типа приложений специализация заключается в том, что обычно предлагаются следующие функции:
  - Поддержка сложных поисковых выражений
  - Полнотекстовый поиск
  - Стеммирование (сведение флективных слов к их основанию)
  - Ранжирование и группировка результатов поиска
  - Распределенный поиск для высокой масштабируемости
- **Самые популярные примеры**
    - Elasticsearch
    - Splunk
    - Solr
    - MarkLogic
    - Algolia
-

Search engines СУБД

- Такой тип СУБД используется для организации полнотекстового поиска. Причем поиск может производиться по различным данным — это например, данные из других БД, e-mail, rSS-feed, текст, JSON, XML, CSV, и даже по документам PDF и MS Office.
- У **Search engine** СУБД свои оптимизированные подходы к индексированию данных. В том числе используются так называемые инвертированные индексы, для того, чтобы предоставлять практически real-time поиск.
- В разных СУБД данного типа могут использоваться свои языки запросов, отличающихся друг от друга.
- **Известные СУБД данного типа:** Apache Solr, Elasticsearch, Splunk.
- **Когда выбирать Search engine СУБД:**
- Подходят для организации быстрого полнотекстового поиска по различным источникам данных, как по структурированным, так и по слабо структурированным. Яркий пример — системы сбора логов и поиска по ним.
- **Когда не выбирать Search engine СУБД:**
- Если поиск производится по ограниченному количеству полей структурированных данных.
-

Time Series СУБД

- **СУБД *временных рядов*** - это система управления базами данных, оптимизированная для обработки данных временных рядов: каждая запись связана с меткой времени.
- **Например**, данные временных рядов могут быть получены с помощью датчиков, интеллектуальных счетчиков или RFID-кодов в Интернете вещей или могут отображать биржевые тикеры высокочастотной системы торговли акциями.
- Предназначены для эффективного сбора, хранения и запроса различных временных рядов с большими объемами транзакций. Хотя данными временных рядов можно управлять с помощью других категорий, для решения конкретных задач часто требуются специализированные системы. **Например**, запрос типа "ВЫБЕРИТЕ SENSOR1_CPU_FREQUENCY / SENSOR2_HEAT" объединяет два временных ряда на основе перекрывающихся областей времени для каждого и выводит один составной временной ряд.
- **Наиболее популярные СУБД:**
  - InfluxDB
  - Kdb+
  - Prometheus
  - Graphite
  - TimescaleDB

RDF Stores

- Resource Description Framework(RDF) - это методология описания информации, первоначально разработанная для описания метаданных ИТ-ресурсов. Сегодня он используется гораздо шире, часто в связи с сематической сетью, но также и в других приложениях.
- Модель **RDF** представляет информацию в виде троек в форме **субъект-предикат-объект**.
- СУБД, которые способны хранить и обрабатывать такие тройки, называются хранилищами RDF или тройными хранилищами.
- Хранилища RDF можно рассматривать как подкласс графовых СУБД, интерпретируя предикат как связь между субъектом и объектом в приведенной выше нотации. Однако хранилища RDF предлагают специальные методы, которые выходят за рамки обычных графовых СУБД. Например, SPARQL, похожий на SQL язык запросов для данных RDF, поддерживается большинством хранилищ RDF.
- **Наиболее популярные СУБД:**
  - MarkLogic
  - Virtuoso
  - GraphDB
  - Apache Jena - TDB
  - Amazon Neptune

Многостороннее(multi-model) хранение

- На практике различные базы данных часто используются в сочетании. Все еще нетрудно встретить приложение, где применяется только реляционная СУБД, но со временем все популярнее становятся комбинации разных баз данных, в которых сильные стороны каждой позволяют создать экосистему, которая оказывается более мощной, функциональной и надежной, чем сумма ее частей.
- Эта практика получила название ***многостороннее хранение*** (**polyglot persistence**)
- **Пример**: Teradata (Relational, Document store, Graph, Spatial, Time Series СУБД)

newSQL решения

- **newSQL** - класс современных реляционных СУБД, стремящихся совместить в себе преимущества NoSQL и транзакционные требования классических баз данных.
- Термин был предложен в 2011 году Мэтью Аслетом.
- Примеры СУБД:
  - VoltDB
  - MemSQL
  - SAP HANA
  - OrientDB

Области применения NoSQL СУБД

№	Тип СУБД	Когда выбирать	Популярные СУБД данного типа
1	Реляционные	Нужна транзакционность; высокая нормализация; большая доля операций на вставку	Oracle, MySQL, Microsoft SQL Server, PostgreSQL, IBM DB2, SQLite
2	Объектные	Высокопроизводительная обработка данных, имеющих сложную структуру, с использованием языков объектно ориентированного программирования	MongoDB realm, InterSystems Caché, ObjectStore, Actian NoSQL DB, Objectivity/DB
3	Ключ-значение	Задачи кэширования и брокеры сообщений	redis, Memcached, etcd
4	Документные	Для хранения объектов в одной сущности, но с разной структурой; хранение структур на основе JSON	Couchbase, MongoDB, Amazon DocumentDB
5	Графовые	Задачи подобные социальным сетям; системы оценок и рекомендаций	Neo4j, Amazon Neptune, InfiniteGraph, TigerGraph
6	Колоночные	Хранилища данных; выборки со сложными аналитическими вычислениями; количество строк в таблице превышает сотни миллионов	Vertica, ClickHouse, Google BigQuery, Sybase \ SAP IQ, InfoBright
7	Time series	Системы мониторинга, сбора телеметрии, и финансовые системы, с привязкой к временным меткам или временным рядам	InfluxDB, Kdb+, Prometheus, TimescaleDB, QuestDB, AWS Timestream, OpenTSDB, GridDB
8	Search engine	Системы полнотекстового поиска	Apache Solr, Elasticsearch, Splunk
9	Spatial	GIS-решения, работа с геометрическими объектами	Oracle Spatial, Microsoft SQL, PostGIS, SpatialLite

Характеристики NoSQL баз данных

- 1. Не используется SQL
- 2. Неструктурированные (schemaless)
- 3. Представление данных в виде агрегатов (aggregates).
- 4. Слабые ACID свойства.
- 5. Распределенные системы, без совместно используемых ресурсов (share nothing).
- 6. NoSQL базы в основном оупенсорсные и созданы в 21 столетии.

2. Неструктурированные (schemaless)

в **NoSQL** базах в отличие от реляционных структура данных не регламентирована (или слабо типизированна, если проводить аналогии с языками программирования) — в отдельной строке или документе можно добавить произвольное поле без предварительного декларативного изменения структуры всей таблицы.

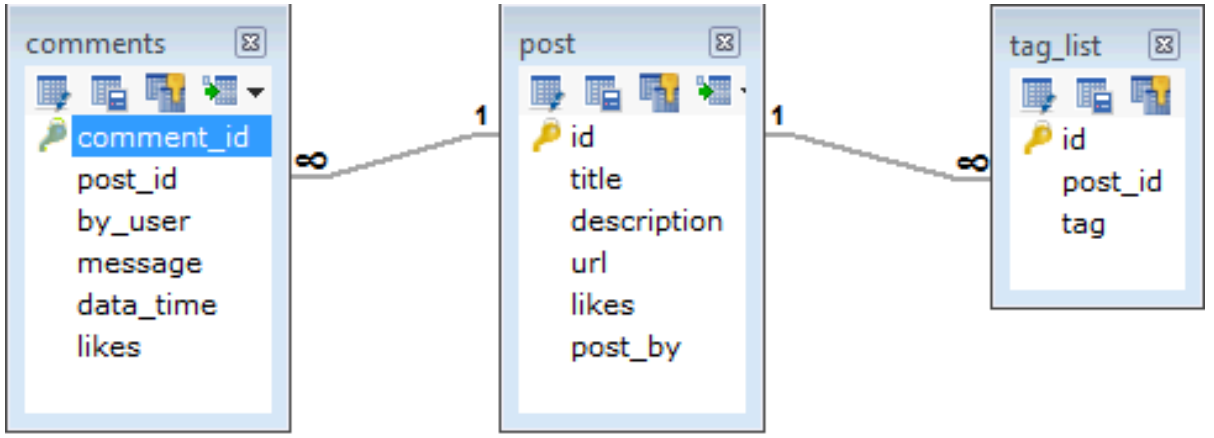
Таким образом, если появляется необходимость поменять модель данных, то единственное достаточное действие — отразить изменение в коде приложения.
Приятное следствие отсутствия схемы — эффективность работы с разреженными (sparse) данными.
Однако в силу отсутствия схемы, колонки не объявляются декларативно и могут меняться/добавляться во время пользовательской сессии работы с базой. Это позволяет в частности использовать динамические колонки для реализации списков.

3. Представление данных в виде агрегатов (aggregates).

- В отличие от реляционной модели, которая сохраняет логическую бизнес-сущность приложения в различные физические таблицы в целях нормализации, NoSQL хранилища оперируют с этими сущностями как с целостными объектами.
- Агрегатно-ориентированные базы данных создают межагрегатные связи, которые труднее обрабатывать, чем внутриагрегатные.
- Графовые базы данных организуют данные в виде графа, состоящего из узлов и ребер; они лучше всего работают с данными, имеющими сложную структуру связей.
- Неструктурированные базы данных позволяют легко добавлять поля в записи, но обычно существует неявная схема, подразумеваемая пользователями этих данных.
- Агрегатно-ориентированные базы данных часто вычисляют материализованные представления, чтобы представить пользователям данные, организованные не так, как в их исходных агрегатах. Для этого часто используются вычисления "отображения-свертка“(map-reduce).

Различия схемы БД в RDBMS и MongoDB

Схема БД в RDBMS



В MongoDB, имеем одну коллекцию и структуру(без схемы) документа:

```
{
 _id: POST_ID
  title: TITLE_OF_POST,
 description: POST_DESCRIPTION,
 by: POST_BY,
 url: URL_OF_POST,
 tags: [TAG1, TAG2, TAG3],
 likes: TOTAL_LIKES,
 comments: [
            { user:'COMMENT_BY', message: TEXT, dateCreated: DATE_TIME, like: LIKES },
            { user:'COMMENT_BY', message: TEXT, dateCreated: DATE_TIME, like: LIKES }
            ]
}
```

Различия схемы БД в MongoDB

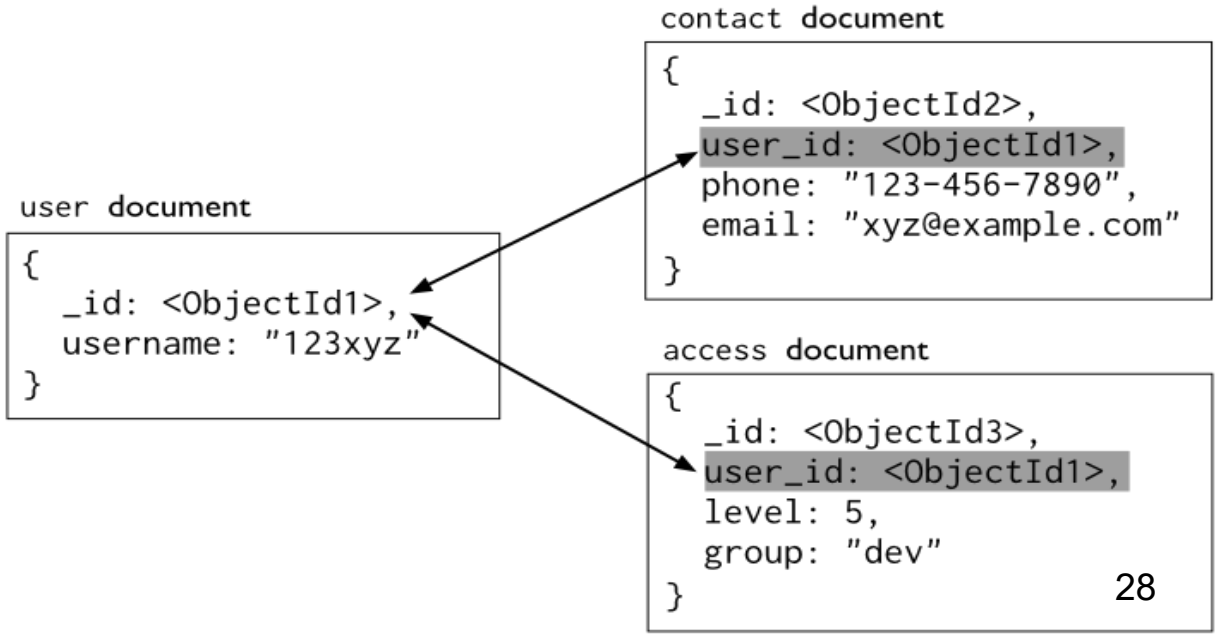
Структура документа с вложенными документами (денормализованная модель)

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
             level: 5,
             group: "dev"
           }
}
```

Embedded sub-document

Embedded sub-document

Пример нормализованной модели с использованием ссылок между документами



Сравнение SQL с NoSQL

Нормализация данных	Данные в виде агрегатов
• Целостность информации при обновлении (меняем запись в одной таблице, а не в нескольких) • Ориентированность на широкий спектр запросов к данных	• Оптимизация только под определенный вид запросов • Сложности при обновлении денормализованных данных
• Неэффективна в распределенной среде • Низкая скорость чтения при использовании объединений (joins) • Несоответствие объектной модели приложения физической структуре данных (impedance mismatch, решается с помощью Hibernate etc.)	• Лучший способ добиться большой скорости на чтение в распределенной среде • Возможность хранить физически объекты в том виде, в каком с ними работает приложение (легче кодировать и меньше ошибок при преобразовании) • Родная (native) поддержка атомарности на уровне записей

Основы MongoDB

1. MongoDB — концептуально то же самое, что обычная, привычная нам база данных (или в терминологии Oracle — схема).Внутри MongoDB может быть ноль или более баз данных, каждая из которых является контейнером для прочих сущностей.
2. База данных может иметь ноль или более «коллекций». Коллекция настолько похожа на традиционную «таблицу», что можно смело считать их одним и тем же.
3. Коллекции состоят из нуля или более «документов». Опять же, документ можно рассматривать как «строку».
4. Документ состоит из одного или более «полей», которые — как можно догадаться — подобны «колонкам».
5. «Индексы» в MongoDB почти идентичны таковым в реляционных базах данных.
6. «Курсоры» отличаются от предыдущих пяти концепций, но они очень важны (хотя порой их обходят вниманием)

Моделирование данных

создадим сотрудника

db.employees.insert({_id: ObjectId("4d85c7039ab0fd70a117d730"), name: 'Leto'})

Теперь добавим пару сотрудников и сделаем Leto их менеджером:

db.employees.insert({_id: ObjectId("4d85c7039ab0fd70a117d731"), name: 'Duncan', manager: ObjectId("4d85c7039ab0fd70a117d730")});

db.employees.insert({_id: ObjectId("4d85c7039ab0fd70a117d732"), name: 'Moneo', manager: ObjectId("4d85c7039ab0fd70a117d730")});

Чтобы найти всех сотрудников, принадлежащих Leto, выполним просто:

db.employees.find({manager: ObjectId("4d85c7039ab0fd70a117d730")})

Массивы и вложенные документы

Удобно, когда требуется смоделировать отношения «один-ко-многим» или «многие-ко-многим». Например, если у сотрудника есть несколько менеджеров, мы просто можем сохранить их в виде массива:

db.employees.insert({_id: ObjectId("4d85c7039ab0fd70a117d733"), name: 'Siona', manager: [ObjectId("4d85c7039ab0fd70a117d730"), ObjectId("4d85c7039ab0fd70a117d732")] })

Кроме массивов MongoDB также поддерживает вложенные документы, например:

db.employees.insert({_id: ObjectId("4d85c7039ab0fd70a117d734"), name: ‘Chanima', family: {mother: 'Chani', father: 'Paul', brother: ObjectId("4d85c7039ab0fd70a117d730")}})

Вложенные документы можно запрашивать с помощью точечной нотации:

db.employees.find({'family.mother': 'Chanima'})

Литература

- Фаулер, Мартин, Садаладж, Прамодкумар Дж. NoSQL: новая методология разработки нереляционных баз данных. : Пер. с англ. - М.: ООО "И.Д. Вильямс", 2013. - 192 с.: ил.
- Эрик Редмонд, Джим. Р. Уилсон. Семь баз данных за семь недель. М.: 2013.