Технологии хранения данных в облачных инфраструктурах

1. Возникновение парадигмы NoSQL

Значение баз данных

- Самым главным преимуществом баз данных является возможность долговременного хранения больших объемов данных
- Чтобы сохранить данные надолго, необходимо записывать их в резервное хранилище, в роли которого обычно выступает жесткий диск
- Для большинства высокопроизводительных приложений (например, для текстовых процессоров) резервным хранилищем является файл, записанный в файловой системе
- Однако для большинства промышленных приложений резервным хранилищем является база данных

Значение баз данных

- База данных обеспечивает более высокую гибкость при хранении крупных объемов данных по сравнению с файловой системой в том смысле, что база данных позволяет прикладным программам быстро и легко получать небольшие порции информации
- Большую часть времени пользователи работают с разными частями этих данных, но иногда они могут оперировать одними и теми же данными.
- В результате возникает необходимость координации этих взаимодействий.

Обеспечение параллельности

- Как известно, параллельность трудно обеспечить в полной мере, предотвратив все возможные ошибки, которые могут произойти в процессе работы с данными
- Реляционные базы данных помогают предотвратить эти неприятности, управляя всем доступом к данным посредством транзакций
- Используя транзакции, можно внести изменение, и если возникнет ошибка при обработке этого изменения, выполнить откат, исправив ситуацию

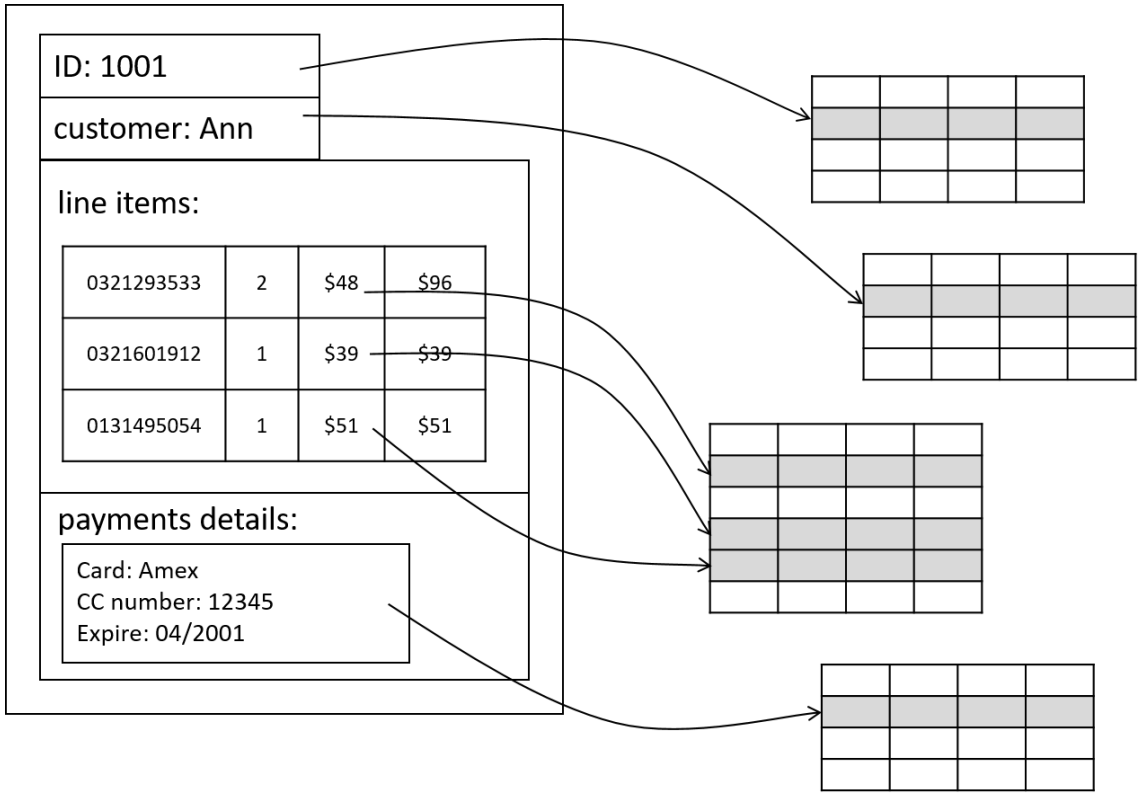
Значимость реляционных БД

- Реляционные базы данных получили признание благодаря тому, что они предоставляют важные преимущества почти стандартным способом. В результате разработчики баз данных могут изучить основную реляционную модель и применять ее во многих проектах
- Несмотря на различия между разными реляционными базами данных, их основной механизм остается одним и тем же: разные диалекты языка SQL похожи друг на друга и транзакции обрабатываются практически одинаково
- Реляционные базы данных имеют много преимуществ, но они не идеальны

Потеря соответствия

- Для разработчиков приложений основной проблемой была потеря соответствия: различие между реляционной моделью и структурами данных, находящимися в памяти.
- Реляционная модель данных представляет их в виде таблиц и строк (отношения и кортежи)
- Значения в реляционном кортеже должны быть простыми – они не должны содержать никаких структур, например, вложенных записей или списков. Это ограничение не относится к структурам данных, находящихся в памяти, которые могут быть намного сложнее отношений. В результате, если вы захотите использовать сложную структуру данных, находящихся в памяти, то должны будете преобразовать реляционное представление для хранения на диске. В итоге возникает потеря соответствия – два разных представления, требующих трансляции

Потеря соответствия



Интеграционные базы данных

- Структура, разработанная для интеграции многих приложений, рано или поздно становится сложнее
- Если приложение захочет изменить хранилище своих данных, оно должно будет координировать это со всеми приложениями, использующими базу данных
- В качестве альтернативы можно применить базу данных приложения (application database), которая открывает доступ к данным только для одного приложения, разработанного одним коллективом
- Поскольку команда, использующая приложение, управляет кодами как базы данных, так и приложения, ответственность за целостность базы данных можно возложить на код приложения.

Влияние веб-сервисов

- На протяжении 2000-х годов наблюдался явный сдвиг в сторону веб-сервисов, в которых приложения взаимодействуют посредством протокола HTTP
- Веб-сервисы создают новую форму широко используемого механизма взаимодействия – альтернативу использованию языка SQL с общими базами данных
- Веб-сервисы повысили гибкость обмениваемых структур данных. При работе с сервисами можно использовать более содержательные структуры данных с вложенными записями и списками. Обычно они представляются в виде документов на языке XML или JSON

«Атака кластеров»

- В начале 2000-х наблюдается укрупнение масштабов веб-сайтов и сервисов
- Появились крупные совокупности данных: связи, социальные сети, активность в блогах, картографические данные
- Рост объема данных сопровождался ростом количества пользователей – крупнейшие сайты достигли громадных размеров и обслуживали огромное количество клиентов

Вертикальное масштабирование

- Вертикальное масштабирование (scaling up) подразумевает укрупнение компьютеров, увеличение количества процессоров, а также дисковой и оперативной памяти
- Более крупные машины становятся все более и более дорогими, не говоря уже о том, что существует физический предел их укрупнения
- Альтернатива - использование большого количества небольших машин, объединенных в кластер

Миграция в кластеры

- Кластер маленьких машин может использовать обычное аппаратное обеспечение и в результате удешевить масштабирование. Кроме того, кластер более надежен – отдельные машины могут выйти из строя, но весь кластер продолжит функционирование

- **Но! Обычные реляционные базы данных не предназначены для работы на кластерах**

Проблемы реляционных БД при работе на кластерах

- Реляционные базы данных могут работать на разных серверах с разными наборами данных, эффективно выполняя фрагментацию базы данных (sharding)
- Хотя это позволяет разделить рабочую нагрузку, вся процедура фрагментации должна контролироваться приложением, которое должно следить за тем, какой сервер базы данных и за какой порцией данных обращается.
- Кроме того, утрачивается возможность управления запросами, целостностью данных, транзакциями и согласованностью между частями кластера.

Первый успешный опыт

BigTable

Dynamo

Что означает термин NoSQL?

- В июне 2009 в Сан-Франциско Йоханом Оскарссоном была организована встреча, на которой планировалось обсудить новые веяния на ИТ рынке хранения и обработки данных. Для яркой вывески для встречи требовалось найти емкий и лаконичный термин, который отлично укладывался бы в твиттеровский хэштег. Один из таких терминов предложил Эрик Эванс из RackSpace — **#NoSQL**.
- **Термин «NoSQL» имеет абсолютно стихийное происхождение и не имеет общепризнанного определения или научного обоснования**
- Это название скорее характеризует вектор развития ИТ в сторону от реляционных баз данных
- Расшифровывается как «Not Only SQL», хотя есть сторонники и прямого определения «No SQL»

Как работает база данных NoSQL (нереляционная БД)?

- В базах данных NoSQL для доступа к данным и управления ими применяются различные модели данных, в том числе документная, графовая, поисковая, с использованием пар «ключ-значение» и хранением данных в памяти.
- Базы данных таких типов оптимизированы для приложений, которые работают с большим объемом данных, нуждаются в низкой задержке и гибких моделях данных.
- Все это достигается путем смягчения жестких требований к непротиворечивости данных, характерных для других типов БД.

Для чего можно использовать базы данных NoSQL?

Базы данных NoSQL хорошо подходят для многих современных приложений, например мобильных, игровых, интернет-приложений, когда требуются гибкие масштабируемые базы данных с высокой производительностью и широкими функциональными возможностями, способные обеспечивать максимальное удобство использования.

Гибкость в NoSQL

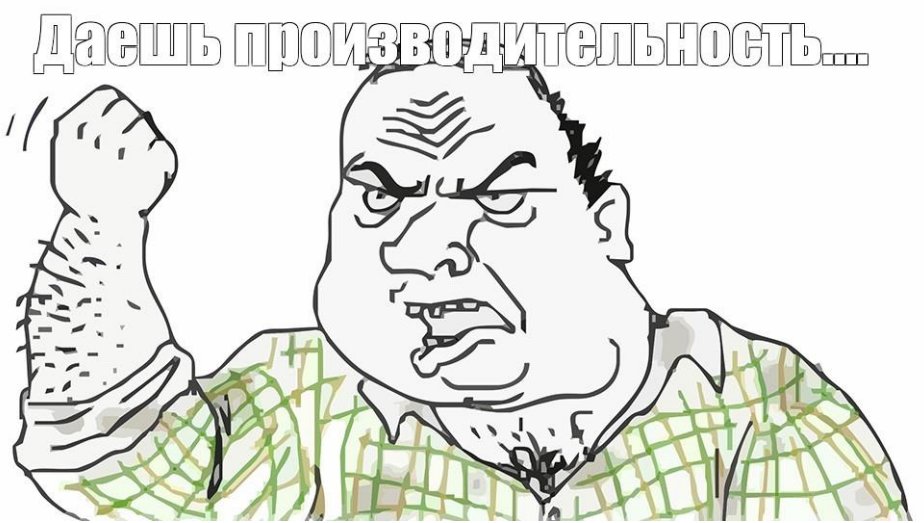
Как правило, базы данных NoSQL предлагают гибкие схемы, что позволяет осуществлять разработку быстрее и обеспечивает возможность поэтапной реализации. Благодаря использованию гибких моделей данных БД NoSQL хорошо подходят для частично структурированных и неструктурированных данных

Масштабируемость в NoSQL

Базы данных NoSQL рассчитаны на масштабирование с использованием распределенных кластеров аппаратного обеспечения, а не путем добавления дорогих надежных серверов. Некоторые поставщики облачных услуг проводят эти операции в фоновом режиме, обеспечивая полностью управляемый сервис

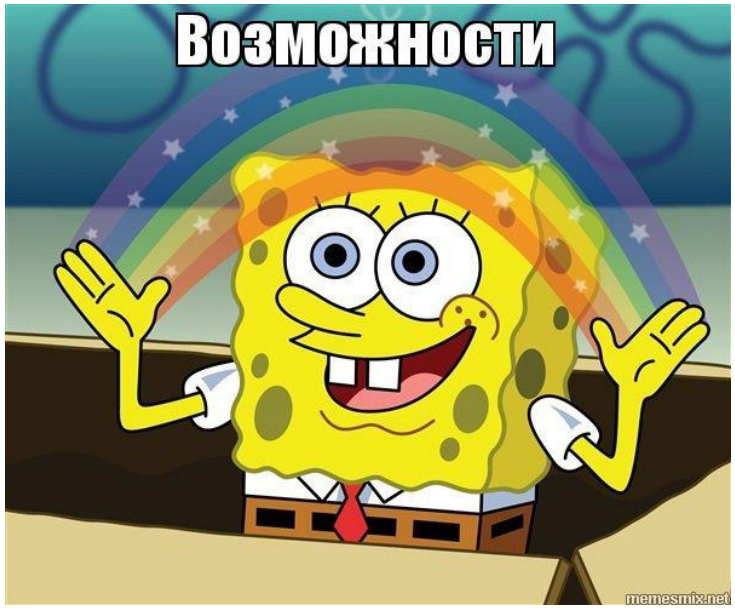
Высокая производительность в NoSQL

Базы данных NoSQL оптимизированы для конкретных моделей данных (например, документной, графовой или с использованием пар «ключ-значение») и шаблонов доступа, что позволяет достичь более высокой производительности по сравнению с реляционными базами данных



Широкие функциональные возможности в NoSQL

Базы данных NoSQL предоставляют API и типы данных с широкой функциональностью, которые специально разработаны для соответствующих моделей данных



Сравнение баз данных SQL (реляционных) и NoSQL (нереляционных)

Подходящие рабочие нагрузки

Реляционные базы данных

Реляционные БД предназначены для транзакционных и строго непротиворечивых приложений обработки транзакций в режиме реального времени (OLTP) и хорошо подходят для аналитической обработки в режиме реального времени (OLAP).

Базы данных NoSQL

Базы данных NoSQL (на основе пар «ключ-значение», документные, графовые и работающие в памяти) ориентированы на OLTP для целого ряда шаблонов доступа к данным, в том числе для приложений с низкой задержкой. Поисковые БД NoSQL предназначены для аналитики частично структурированных данных.

Модель данных

Реляционные базы данных

Реляционная модель нормализует данные и преобразует их в таблицы, состоящие из строк и столбцов. Схема жестко задает таблицы, строки, столбцы, индексы, отношения между таблицами и прочие элементы базы данных. Такая БД обеспечивает целостность ссылочных данных в отношениях между таблицами.

Базы данных NoSQL

В базах данных NoSQL применяются различные модели данных, в том числе документные, графовые, поисковые, с использованием пар «ключ-значение» и хранением данных в памяти.

Свойства ACID

Реляционные базы данных

Реляционные базы данных обеспечивают набор свойств ACID: атомарность, непротиворечивость, изолированность, надежность.

- **Атомарность** требует, чтобы транзакция выполнялась полностью или не выполнялась вообще.
- **Непротиворечивость** означает, что сразу по завершении транзакции данные должны соответствовать схеме базы данных.
- **Изолированность** требует, чтобы параллельные транзакции выполнялись отдельно друг от друга.
- **Надежность** подразумевает способность восстанавливаться до последнего сохраненного состояния после непредвиденного сбоя в системе или перебоя в подаче питания.

Базы данных NoSQL

Базы данных NoSQL зачастую предлагают компромисс, смягчая жесткие требования свойств ACID ради более гибкой модели данных, которая допускает горизонтальное масштабирование. Благодаря этому БД NoSQL – отличный выбор для примеров использования с высокой пропускной способностью и низкой задержкой, в которых требуется горизонтальное масштабирование, не ограниченное рамками одного инстанса.

Производительность

Реляционные базы данных

Производительность главным образом зависит от дисковой подсистемы. Для обеспечения максимальной производительности часто требуется оптимизация запросов, индексов и структуры таблицы.

Базы данных NoSQL

Производительность обычно зависит от размера кластера базового аппаратного обеспечения, задержки сети и вызывающего приложения.

Масштабирование

Реляционные базы данных

Реляционные базы данных обычно масштабируются путем увеличения вычислительных возможностей аппаратного обеспечения или добавления отдельных копий для рабочих нагрузок чтения.

Базы данных NoSQL

Базы данных NoSQL обычно поддерживают высокую разделяемость благодаря шаблонам доступа на основе пар «ключ-значение» с возможностью масштабирования на основе распределенной архитектуры. Это повышает пропускную способность и обеспечивает устойчивую производительность почти в неограниченных масштабах.

API

Реляционные базы данных

Запросы на запись и извлечение данных составляются на языке SQL. Эти запросы анализирует и выполняет реляционная база данных.

Базы данных NoSQL

Объектно-ориентированные API позволяют разработчикам приложений без труда осуществлять запись и извлечение структур данных, размещенных в памяти. Благодаря использованию ключей секций приложения могут вести поиск по парам «ключ-значение», наборам столбцов или частично структурированным документам, содержащим серийные объекты и атрибуты приложений.

Агрегированные модели данных

Модель данных

- Модель данных - это модель, с помощью которой мы воспринимаем и обрабатываем свои данные.
- Для людей, работающих с базами данных, модель данных описывает способ взаимодействия с данными, находящимися в базе.
- Этим она отличается от модели хранилища, описывающей, как база данных хранит данные и манипулирует ими.

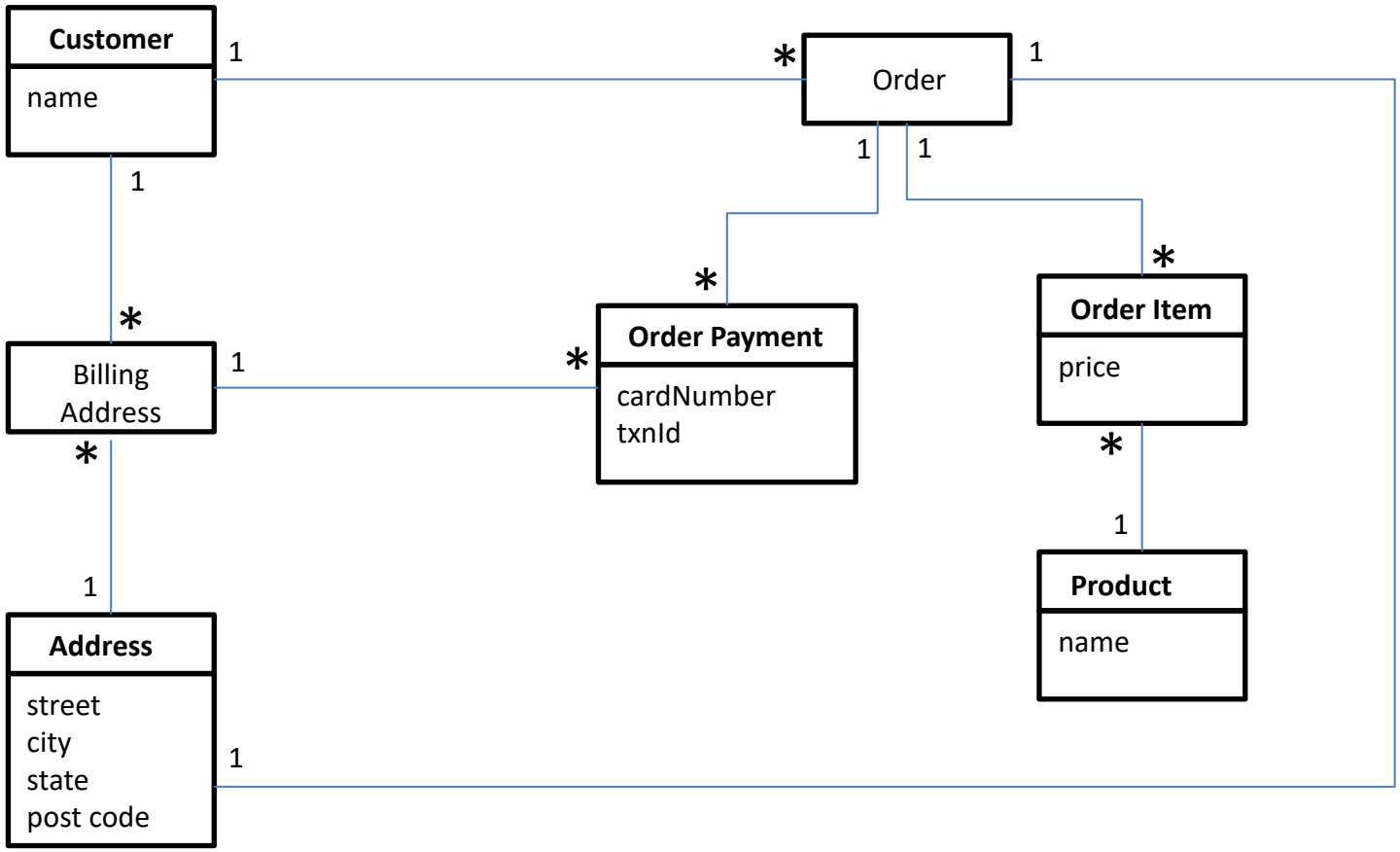
Модель данных в NoSQL

- Одной из основных особенностей технологии NoSQL является отказ от реляционной модели.
- Каждое решение в рамках технологии NoSQL использует свою собственную модель.
- Эти модели разделяются на четыре категории: ключ-значение, документ, семейство столбцов и граф.
- Первые три модели имеют общее свойство, которое называется агрегатной ориентацией (aggregate orientation)

Агрегаты

- Агрегат - это термин, пришедший из предметно-ориентированного проектирования.
- В предметно-ориентированном проектировании агрегатом называют коллекцию связанных объектов, которая интерпретируется как единое целое и представляет собой единицу для манипулирования данными и управления их согласованностью.
- Агрегаты облегчают работу баз данных на кластерах, поскольку агрегат представляет собой естественную единицу репликации и фрагментации. Кроме того, агрегаты упрощают разработку прикладных программ, которые часто манипулируют данными с помощью агрегированных структур.

Модель данных, ориентированная на реляционную базу данных



Данные для реляционной базы данных

Customer

Id	Name
1	Martin

Orders

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

Id	OrderId	ProductId	Price
100	99	27	32.45

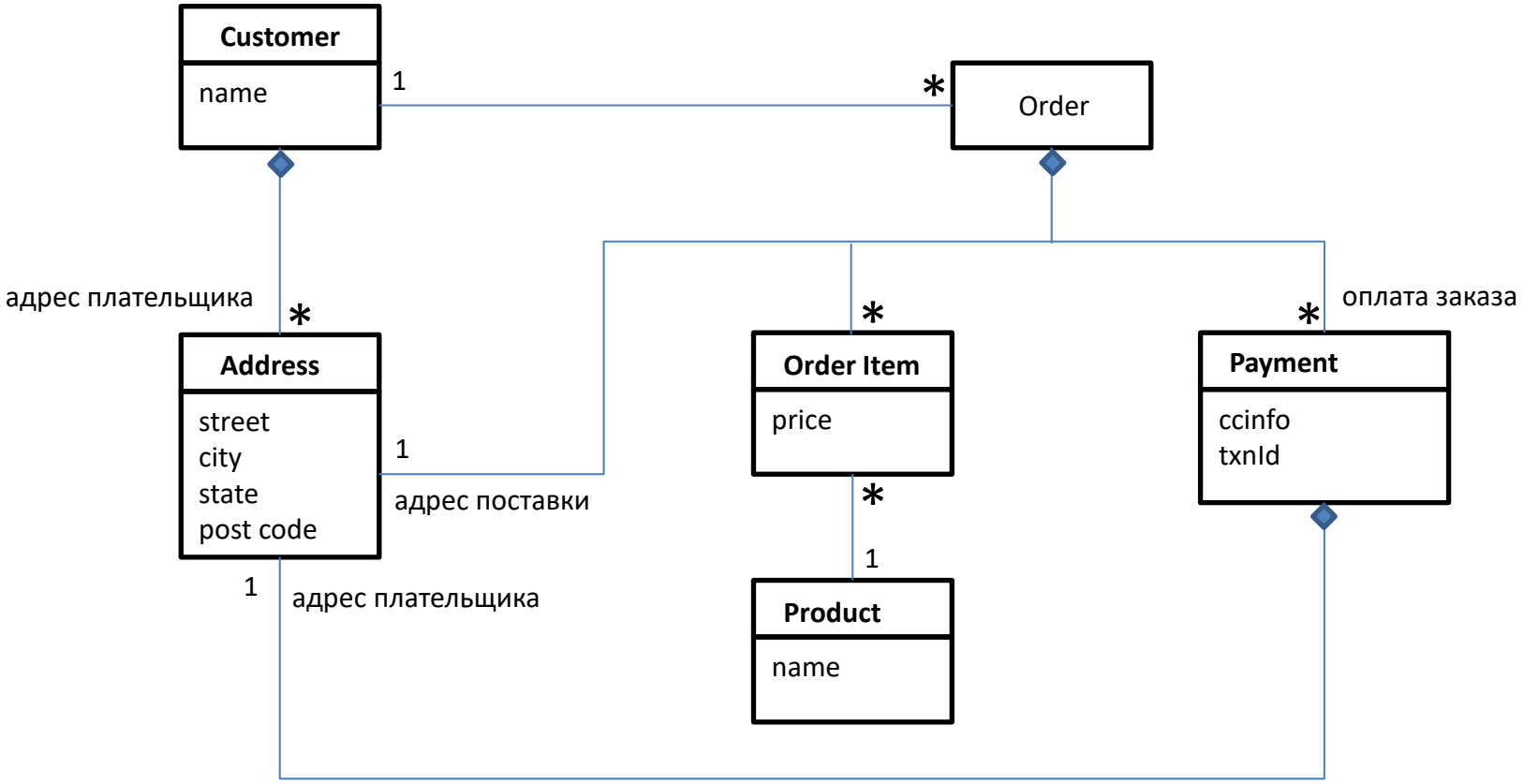
Address

Id	City
77	Chicago

OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rfl

Агрегатная модель данных

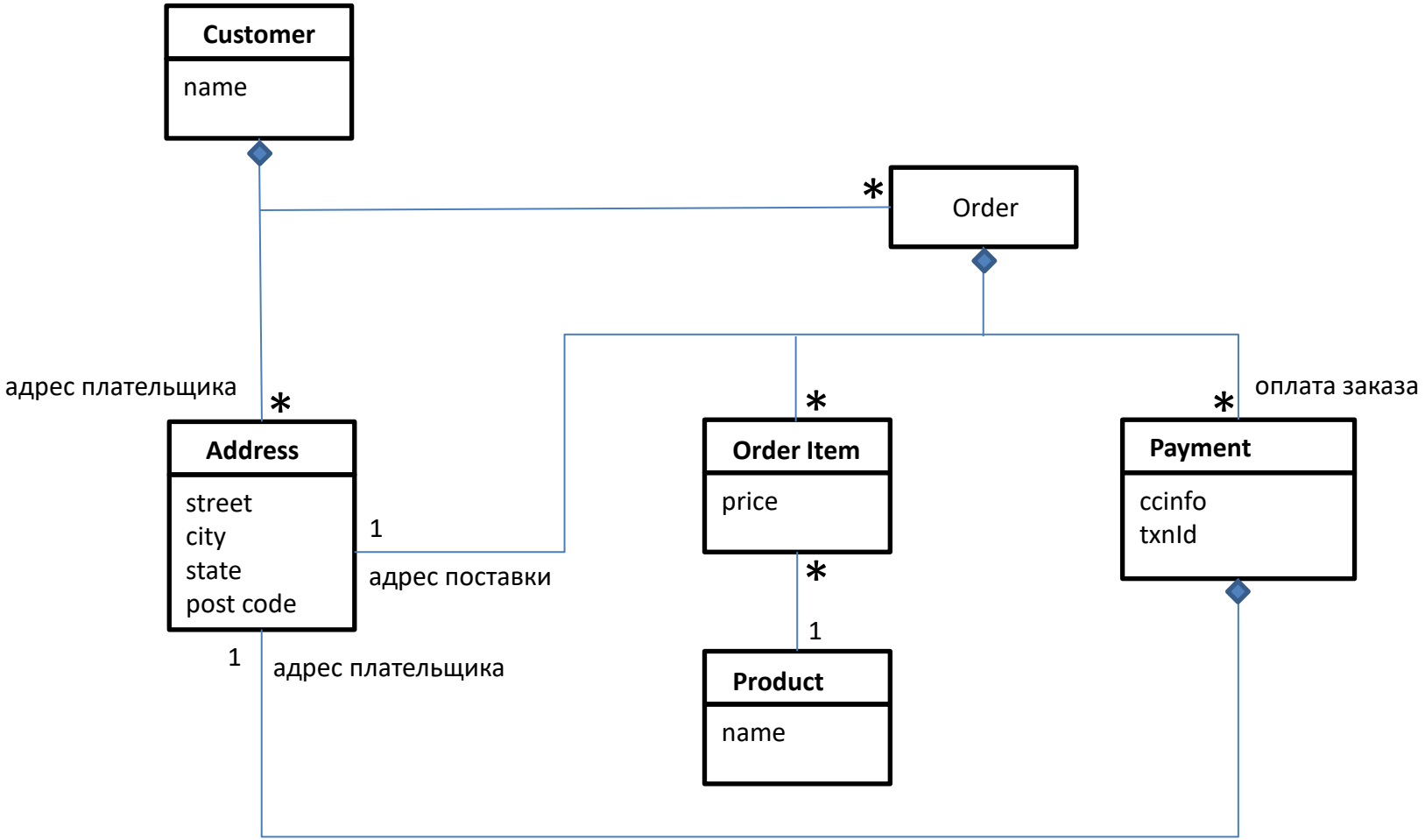


Данные в формате JSON для NoSQL

```
//Клиенты
{
    "id":1,
    "name":"Martin", “billingAddress": [ { "city": "Chicago"}]
}

//Заказы {
   "id":99,
   "customerId":1,
   "orderItems":[
        { "productid":27,
          "price": 32. 45,
          "productName": "NoSQL Distilled“
        }
        ] ,
    "shippingAddress": [ { "city": "Chicago"}]
    "orderPayment": [
         {
            "ccinfo":"1000-1000-1000-1000",
            "txnId":"abelif879rft",
            "billingAddress": {"city": "Chicago"}
         }
        ] ,
  }
```

Объединение объектов



Изменение данных в формате JSON

```
//Клиенты
{
   "customer": {
   "id": 1,
   "name": "Martin",
   "billingAddress": [{"city": "Chicago"}],
   "orders": [
       {
          "id":99,
          "customerId":1,
          "orderItems": [
          {
             "productid":27,
             "price": 32.45,
             "productName": "NoSQL Distilled"
           }
] ,
"shippingAddress":[{"city":"Chicago"}]
  "orderPayment":[
    {
       "ccinfo":"1000-1000-1000-1000",
       "txnid":"abelif879rft",
       "billingAddress": {"city": "Chicago"}
    }],
  }]
 }
}
```

Последствия ориентации на агрегаты

- Реляционная база данных не учитывает понятие агрегата
- Работая с агрегатно-ориентированными базами данных, мы имеем более четкую семантику, позволяющую сосредоточиться на единице взаимодействия в хранилище данных
- Агрегатная ориентация очень облегчает работу на кластерах - позволяет минимизировать количество узлов, которые необходимо опросить, чтобы собрать данные

ACID-транзакции

ACID:

- атомарные (atomic),
- согласованные (consistent),
- изолированные (isolated),
- долговечные (durable)

Часто говорят, что базы данных NoSQL не поддерживают транзакции ACID и тем самым не обеспечивают согласованность. Это упрощение. В целом это относится к тем агрегатно-ориентированным базам данных, у которых нет транзакций ACID, охватывающих несколько агрегатов. Вместо этого они поддерживают атомарные манипуляции с отдельными агрегатами по очереди (из кода приложения)

Отношения в агрегатах

- Связи между агрегатами осуществляется при помощи идентификаторов
- Некоторые NoSQL-базы открывают содержимое агрегатов для построения индексов и сложных запросов
- Атомарность поддерживается только в содержимом отдельного агрегата
- **При обращении к нескольким агрегатам одновременно NoSQL-базы становятся неудобными**

Неструктурированность

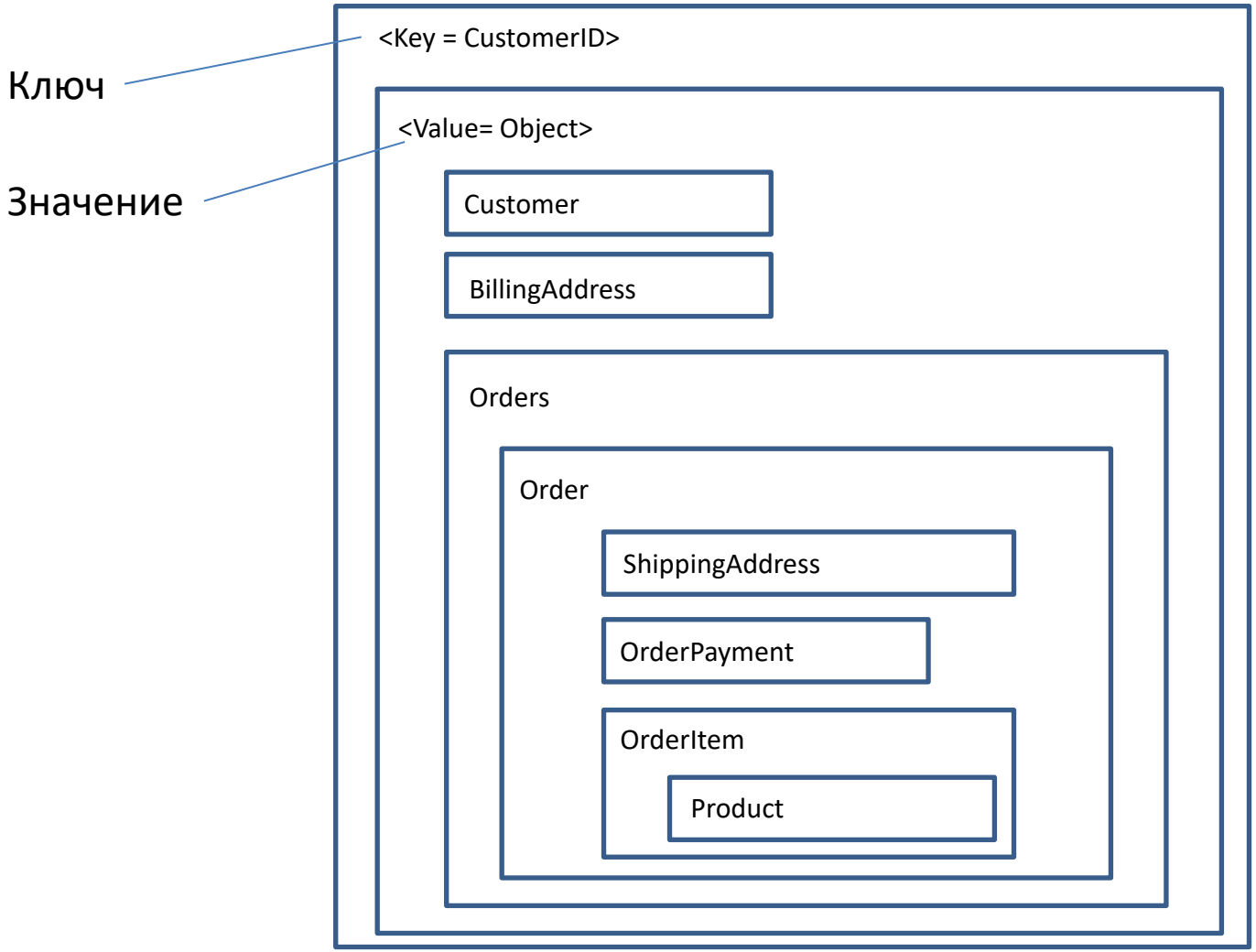
- Помимо обеспечения удобных изменений, неструктурированные базы данных облегчают обработку **неоднородных данных**, т. е. данных, в которых все записи имеют разные наборы полей
- Даже в неструктурированных базах данных обычно существует неявная схема - набор предположений о структуре данных в коде, манипулирующем этими данными
- По существу, неструктурированные базы данных переносят схему в код приложения, который к ней обращается
- При множестве обращений выход один - инкапсулировать все взаимодействия с базой данных в отдельном приложении и интегрировать его с другими приложениями через веб-сервисы.

Материализованные представления

- Представления, вычисленные заранее и записанные в кеш-память диска. Материализованные представления эффективны при работе с часто считываемыми данными, но могут оказаться устаревшими
- В базах данных NoSQL нет представлений, они могут иметь заранее вычисленные и кешированные запросы, к которым также применяется термин «материализованное представление»

Моделирование доступа к данным

- **«Ключ-значение»**



Моделирование доступа к данным

- **«Ключ-значение»**