

# История развития и причины появления СУБД

# Общие понятия

- **База данных** — совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.
- **База данных** — Совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

# Файловые системы

**Файловые системы**— Набор прикладных программ, которые выполняют для пользователей некоторые операции, например создание отчетов. Каждая программа хранит свои собственные данные и управляет ими.

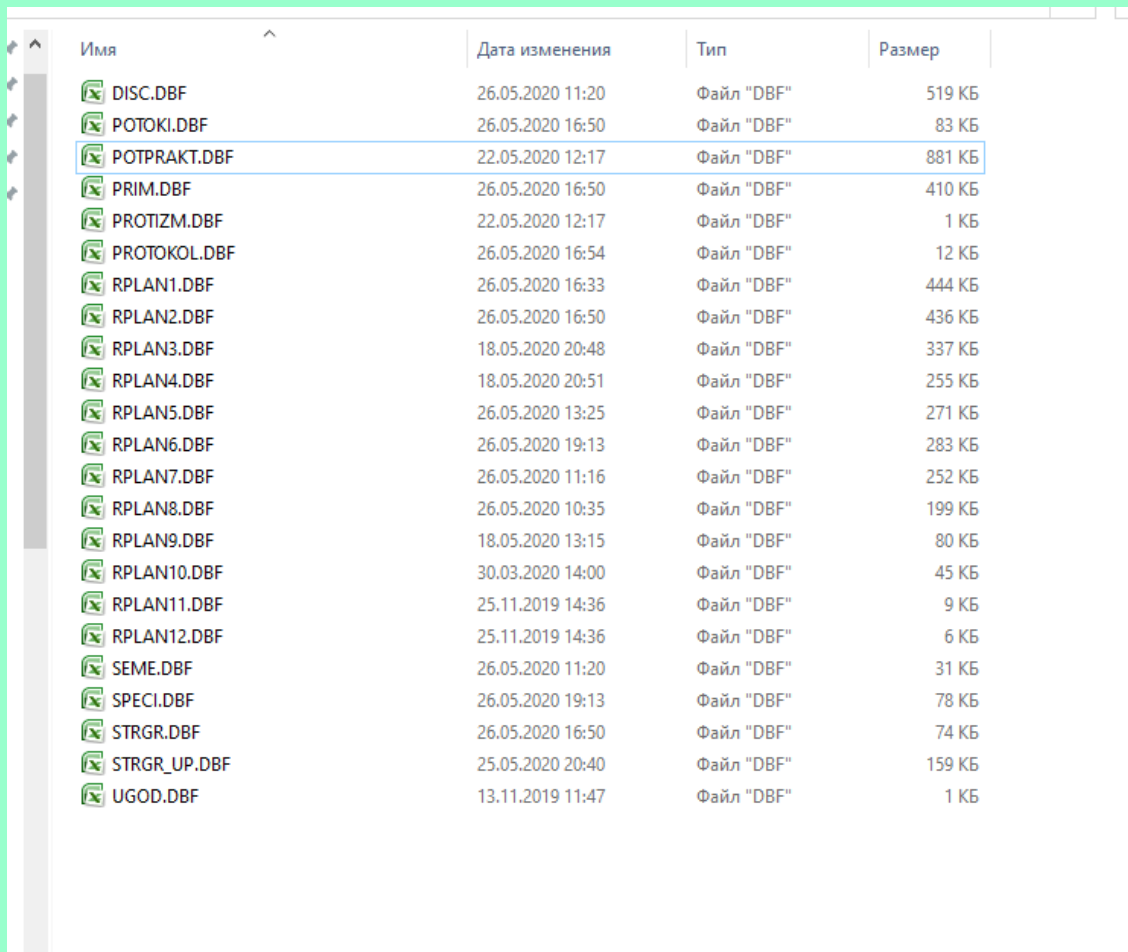
## Достоинства
























- Возможность автоматизированной обработки информации

## Недостатки

- Разделение и изоляция данных
- Дублирование данных
- Зависимость от данных
- Несовместимость файлов
- Фиксированные запросы/быстрое увеличение количества приложений

# Файловые системы



Имя	Дата изменения	Тип	Размер
 DISC.DBF	26.05.2020 11:20	Файл "DBF"	519 КБ
 POTOKI.DBF	26.05.2020 16:50	Файл "DBF"	83 КБ
 <b>POTPRAKT.DBF</b>	22.05.2020 12:17	Файл "DBF"	881 КБ
 PRIM.DBF	26.05.2020 16:50	Файл "DBF"	410 КБ
 PROTIZM.DBF	22.05.2020 12:17	Файл "DBF"	1 КБ
 PROTOKOL.DBF	26.05.2020 16:54	Файл "DBF"	12 КБ
 RPLAN1.DBF	26.05.2020 16:33	Файл "DBF"	444 КБ
 RPLAN2.DBF	26.05.2020 16:50	Файл "DBF"	436 КБ
 RPLAN3.DBF	18.05.2020 20:48	Файл "DBF"	337 КБ
 RPLAN4.DBF	18.05.2020 20:51	Файл "DBF"	255 КБ
 RPLAN5.DBF	26.05.2020 13:25	Файл "DBF"	271 КБ
 RPLAN6.DBF	26.05.2020 19:13	Файл "DBF"	283 КБ
 RPLAN7.DBF	26.05.2020 11:16	Файл "DBF"	252 КБ
 RPLAN8.DBF	26.05.2020 10:35	Файл "DBF"	199 КБ
 RPLAN9.DBF	18.05.2020 13:15	Файл "DBF"	80 КБ
 RPLAN10.DBF	30.03.2020 14:00	Файл "DBF"	45 КБ
 RPLAN11.DBF	25.11.2019 14:36	Файл "DBF"	9 КБ
 RPLAN12.DBF	25.11.2019 14:36	Файл "DBF"	6 КБ
 SEME.DBF	26.05.2020 11:20	Файл "DBF"	31 КБ
 SPECI.DBF	26.05.2020 19:13	Файл "DBF"	78 КБ
 STRGR.DBF	26.05.2020 16:50	Файл "DBF"	74 КБ
 STRGR_UP.DBF	25.05.2020 20:40	Файл "DBF"	159 КБ
 UGOD.DBF	13.11.2019 11:47	Файл "DBF"	1 КБ

# СУБД

- ***Система управления базой данных*** (СУБД) представляет собой совокупность программ, с помощью которых осуществляются управление структурой базы данных и контроль доступа к данным, хранящимся в ней. СУБД позволяет нескольким приложениям или пользователям осуществлять совместный доступ к данным.
- **СУБД.** Программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

# Предпосылки возникновения БД

- Предшественником СУБД была файловая система, т.е. набор приложений, которые выполняли отдельные необходимые для пользователя операции, такие как создание отчетов.
- Каждая программа определяла и управляла своими собственными данными.
- Хотя файловая система была значительным достижением по сравнению с ручной картотекой, ее использование все еще было сопряжено с большими проблемами, которые в основном были связаны с избыточностью данных и зависимостью программ от данных.
- Появление СУБД было вызвано необходимостью разрешить проблемы, характерные для файловых систем.

# Функции СУБД

- *Определение данных (создание)*
- *Манипулирование данными*
- *Оптимизация и выполнение запросов*
- *Защита и поддержка целостности данных*
- *Восстановление данных и поддержка параллельности*
- *Словарь данных (метаданные / хранение)*

# Преимущества и недостатки СУБД

## Преимущества

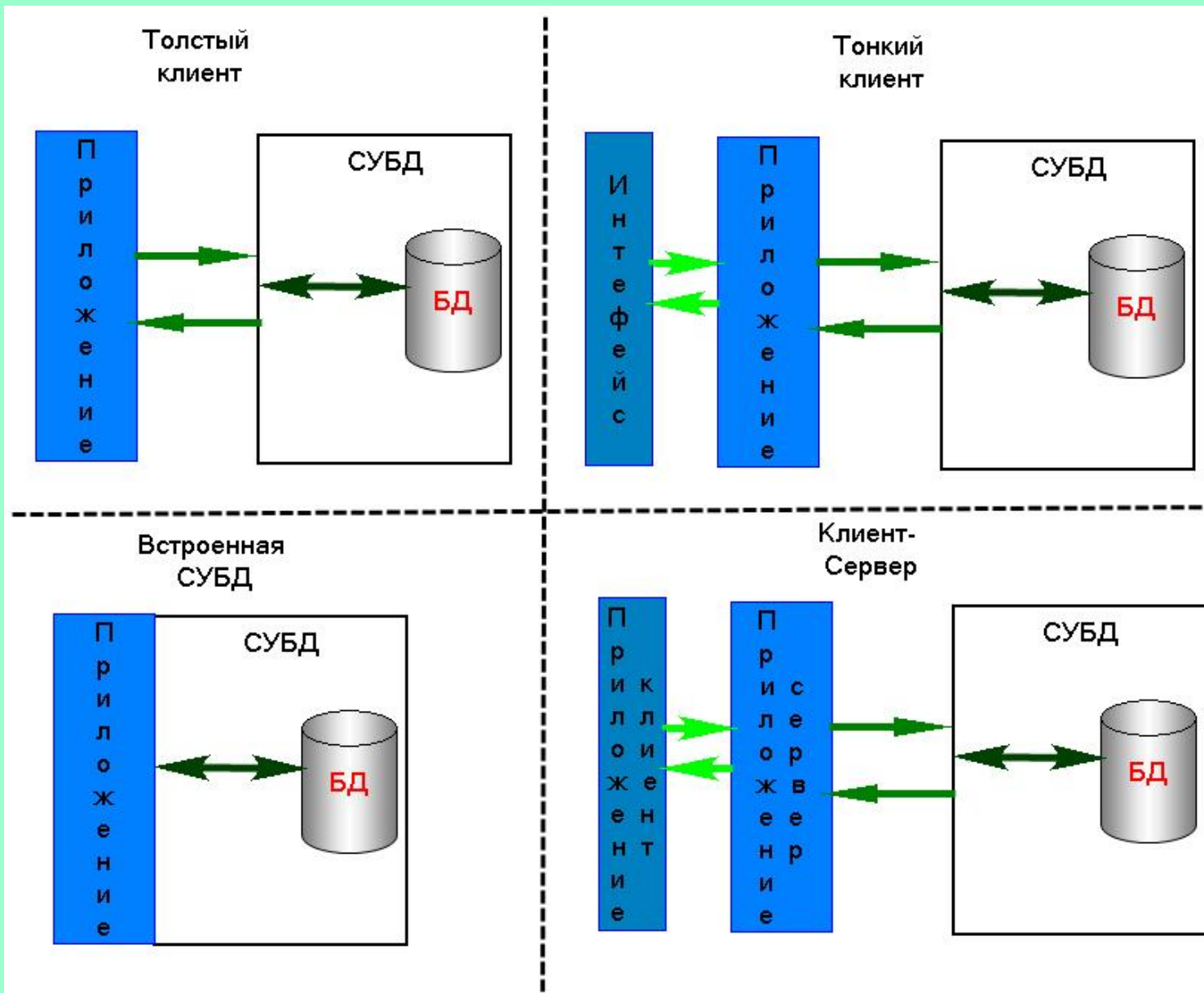
- Контроль за избыточностью данных
- Непротиворечивость данных
- Больше полезной информации при том же объеме хранимых данных
- Совместное использование данных
- Поддержка целостности данных
- Повышенная безопасность
- Применение стандартов
- Повышение эффективности с ростом масштабов системы
- Возможность нахождения компромисса при противоречивых требованиях
- Повышение доступности данных и их готовности к работе
- Улучшение показателей производительности
- Упрощение сопровождения системы за счет независимости от данных
- Улучшенное управление параллельной работой
- Развитые службы резервного копирования и восстановления

## Недостатки

- Сложность
- Размер
- Стоимость СУБД
- Дополнительные затраты на аппаратное обеспечение
- Затраты на преобразование (существующей системы к СУБД)
- Производительность
- Более серьезные последствия при выходе системы из строя



# БД,СУБД и приложение



# Типы СУБД по назначению



- **транзакционная обработка**

OLTP (On-Line Transaction Processing)  
— интерактивная транзакционная обработка

*Транзакцией в СУБД называется совокупность операций над данными, являющаяся неделимой*

Назначение:

- Обработка идёт **в режиме реального или приближенного к реальному времени.**
- Запросы представляют собой **интенсивный поток коротких операций** по вставке, изменению и удалению **небольшого числа записей** в БД.

- **аналитическая обработка**

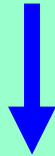
OLAP (On-Line Analytical Processing) —  
интерактивная аналитическая обработка

Назначение:

- ☐ **Время отклика системы не регламентировано.**
- Выборки представляют собой **одиночные тяжёлые запросы:** поиски и расчёты по множеству произвольных критериев могут охватывать значительную часть данных в базе.
- Данные находятся **в режиме чтения**, за исключением моментов их обновления.

Почему возникло разделение на транзакционную  
и аналитическую обработку?

Архитектуру оптимизируют под  
вставку/изменение или под выборку



СУБД, ориентированные на  
транзакционную обработку, менее  
эффективны при работе с  
аналитическими запросами и наоборот

# Модели данных

- *Модель данных*– это абстрактное, независимое, логическое определение структур данных, операторов над данными и прочего, что в совокупности составляет *абстрактную систему*, с которой взаимодействует пользователь.



# Исторический обзор моделей данных

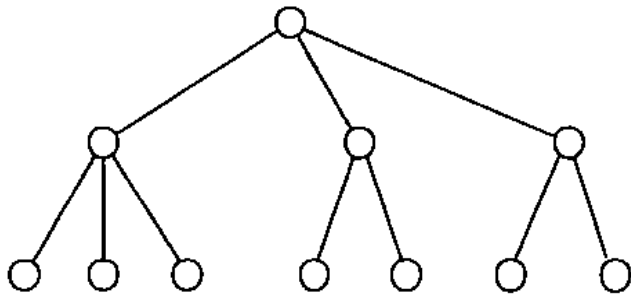
- Файловые структуры
- Иерархические СУБД
- Сетевые СУБД
- Реляционные СУБД
- Объектные СУБД (Объектно-ориентированные)
- Объектно-реляционные СУБД
- NOSQL СУБД
  - ключ-значение,
  - документные,
  - семейство столбцов,
  - графовые

# Вехи истории развития СУБД

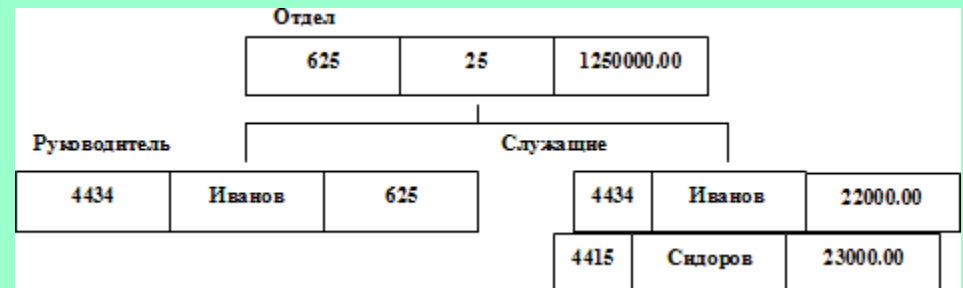
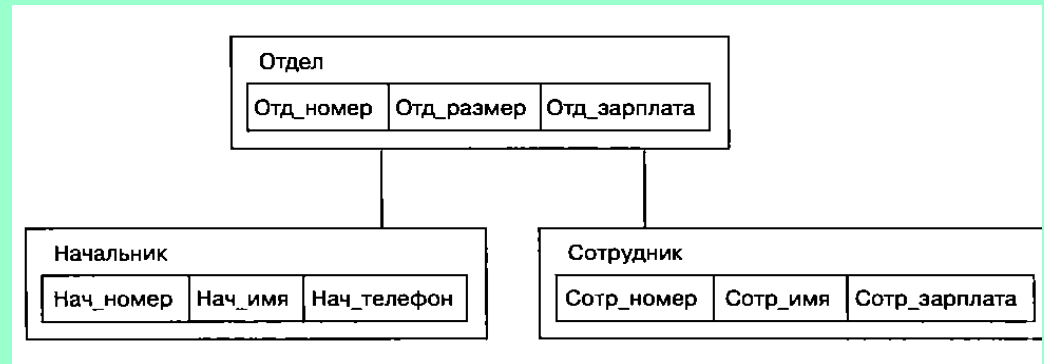
Год	Событие
1950-х и 1960-х	Появление и развитие иерархических СУБД
1969	Разработка Conference on Data Systems Languages (CODASYL) — постоянно действующая конференция по языкам обработки данных) стандарта сетевой модели данных
1969 конец/начало 1970	Опубликование Э.Ф. Коддом статьи «A Relational Model of Data for Large Shared Data Banks» , которая считается первой работой по реляционной модели данных. Но теория пока не начала использоваться: Внешняя концептуальная простота реляционной модели достигается за счет ресурсов компьютера, а компьютеры в то время не обладали достаточной мощностью для реализации реляционной модели.
1971	Для разработки стандартов для баз данных, создана конференция Conference on Data Systems Languages (CODASYL), уже существовавшая к тому времени.
Конец 1980-х и начале 1990-х гг	были выпущены первые объектные системы
1992	M. Atkinson, et al., «Object-Oriented Database System Manifesto». Building an Object-Oriented Database System: The Story of O2. Morgan Kaufman,.
Конец 1990-х годов	Создание объектно-реляционных систем
11 июня 2009 года.	Рождение термина "NoSQL" в современном смысле/ Он появился на конференции в Сан-Франциско, организованной Йоханом Оскарссоном (Johan Oskarsson), разработчиком программного обеспечения, живущим в Лондоне.

# Иерархические СУБД

Модель данных, реализующая структуру данных «дерево» с узлом, содержащим изначально жестко заданную табличную структуру



Типичным представителем (наиболее известным и распространенным) является СУБД IMS (Information Management System) компании IBM



# Иерархические СУБД

## *Достоинства*

- относятся к эффективному использованию памяти ЭВМ
- неплохие показатели времени выполнения основных операций над данными.
- удобство для работы с иерархически упорядоченной информацией.

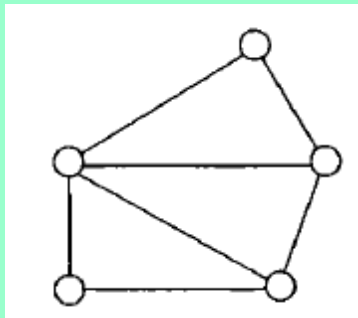
## *Недостатки*

- громоздкость для обработки информации с достаточно сложными логическими связями
- сложность понимания для обычного пользователя.



# Сетевые СУБД

Модель данных, основанная на графах, где для реализации связей в предметной области жестко задана связь между узлами графа(ребра), и в узле графа может стоять табличная структура изначально жестко заданной структуры.



Типичным представителем систем, основанных на сетевой модели данных, является СУБД IDMS (Integrated Database Management System),

# Сетевые СУБД

## *Достоинства*

- возможность эффективной реализации по показателям затрат памяти и оперативности.
- Большие в сравнении с иерархической моделью возможности в смысле допустимости образования произвольных связей.

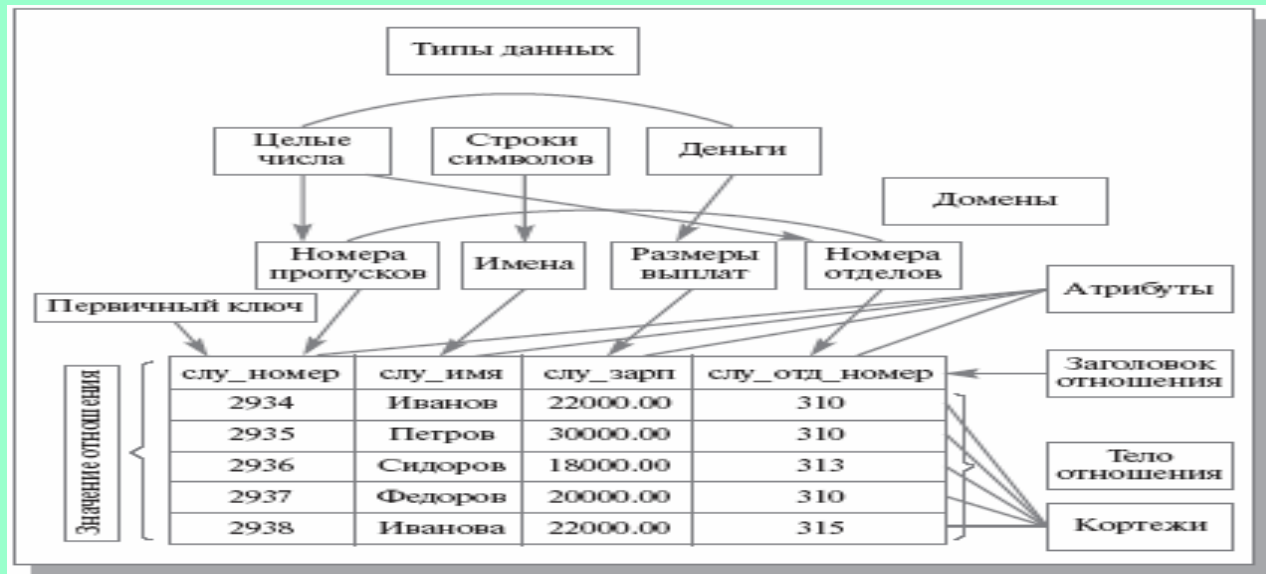
## *Недостатки*

- высокая сложность и жесткость схемы БД
- сложность для понимания и выполнения обработки информации в БД
- ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

# Реляционные СУБД

Relational (от Relation — отношение= таблица,  
НЕ Relationship = связь в концептуальной модели)

Примеры: Oracle до 8 версии, MS SQL Server, MySQL



# Реляционные СУБД

- *Достоинство*
- Простота
- понятность
- удобство физической реализации на ЭВМ.
- *Нерегламентированные запросы.*

## *недостатки*

- Неадекватное представление сущностей реального мира
- Семантическая перегрузка
- Слабая поддержка ограничений целостности и корпоративных ограничений
- Однородная структура данных
- Ограниченный набор операций
- Сложности при обработке рекурсивных запросов
- Проблема рассогласования типов данных
- Другие проблемы реляционных СУБД, связанные с параллельным выполнением, изменениями схемы и неразвитыми средствами доступа
- Сложность объектно-реляционного отображения

# ООП и Реляционная БД

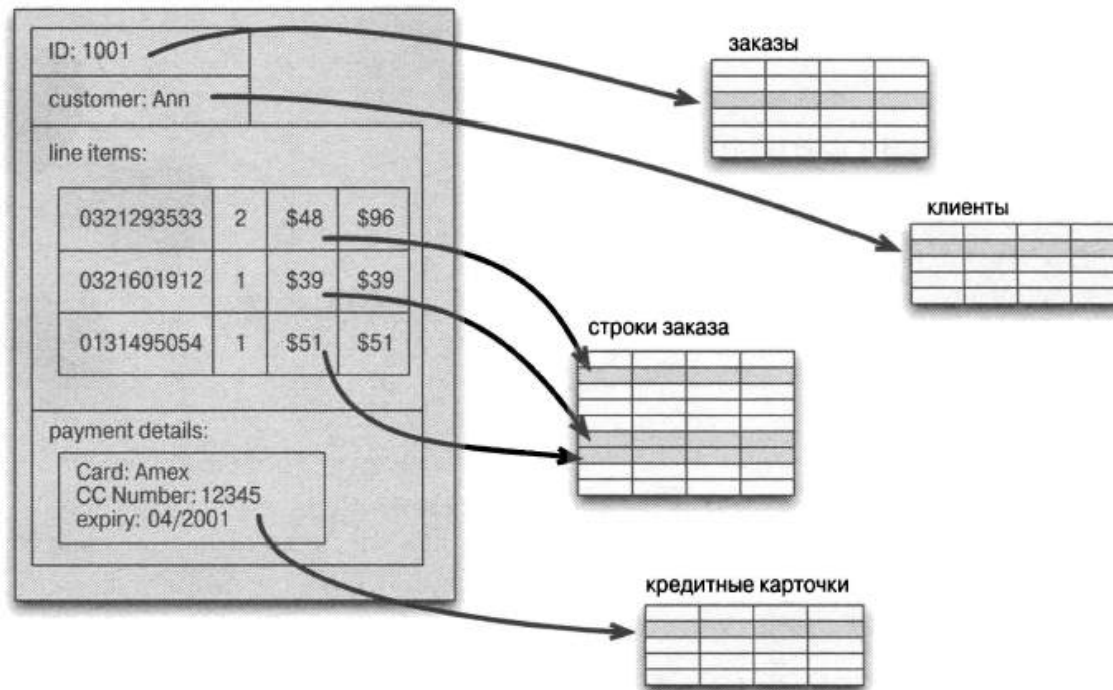
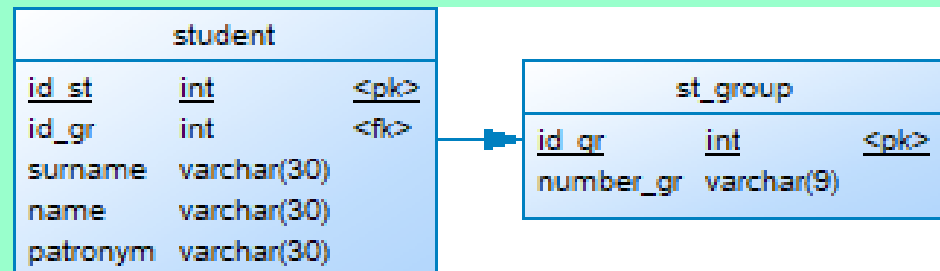
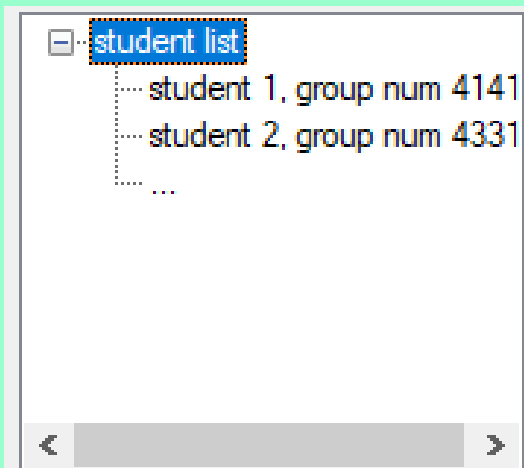
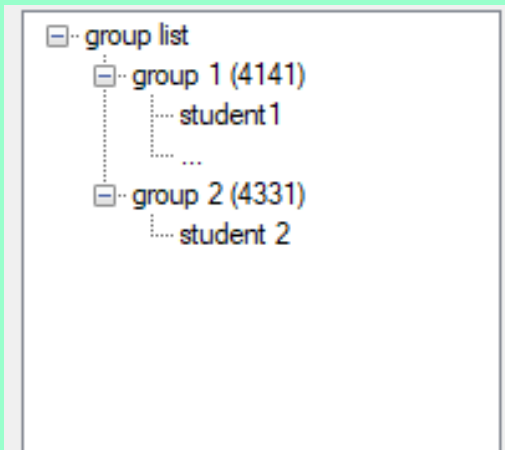


Рис. 1.1. Заказ, который в пользовательском интерфейсе выглядит как единая агрегированная структура, в реляционной базе данных разделяется на множество строк из многих таблиц

# ООП и Реляционная БД



# Реляционные СУБД непригодны/ неудобны

- Автоматизированное проектирование.
- Автоматизированное производство.
- Автоматизированная разработка программного обеспечения.
- Офисные информационные системы и мультимедийные системы.
- Цифровое издательское дело.
- Геоинформационные системы.
- Интерактивные и динамические Web-узлы

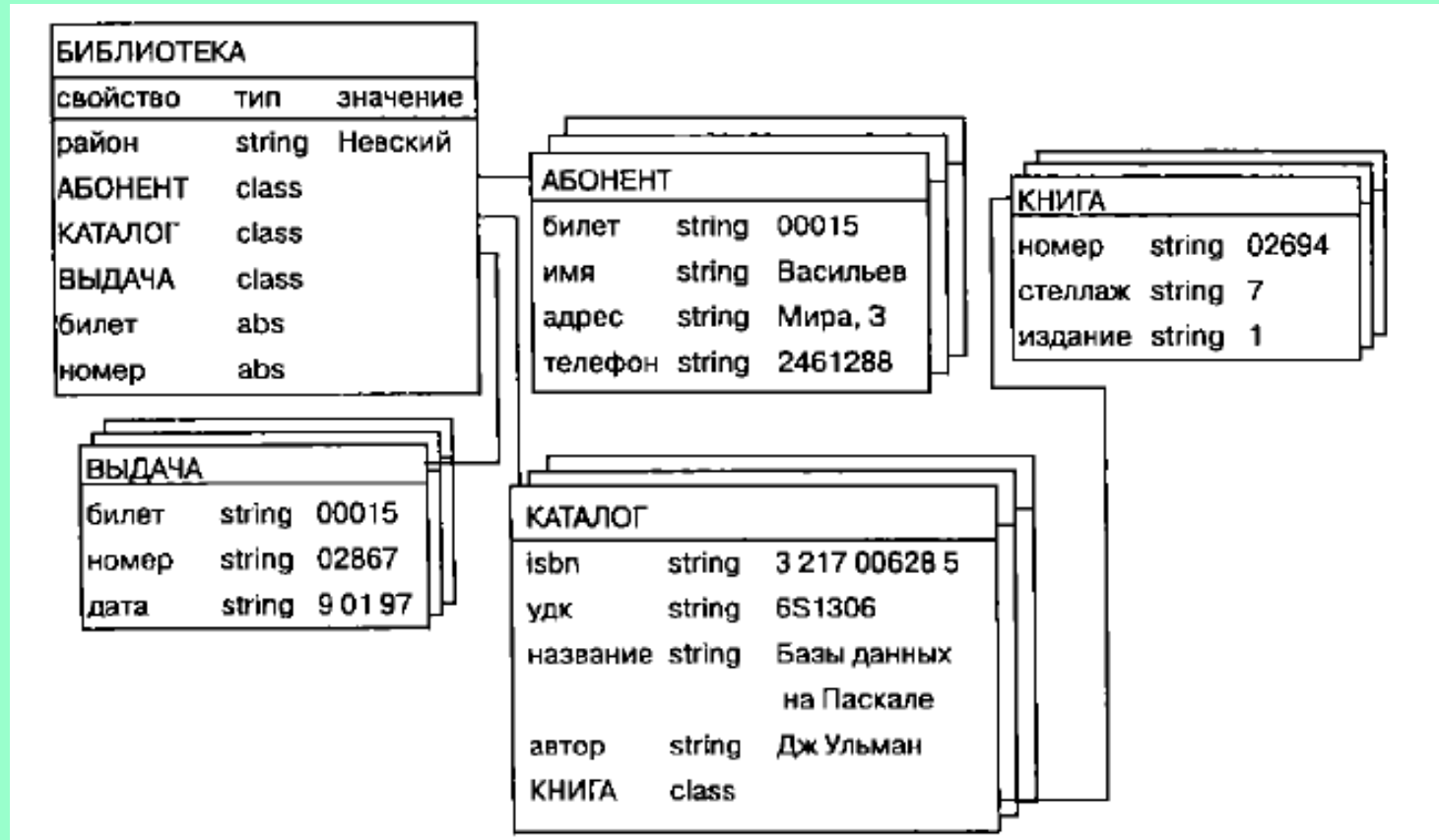
# Объектно-ориентированные СУБД

- Объектно-ориентированной модели данных (ООМД)—(Логическая) модель данных, которая учитывает семантику объектов, применяемую в *объектно-ориентированном программировании*.
- Наследование
- Инкапсуляция
- Полиморфизм

Пример ObjectStore, Ontos, Versant...



# Объектно-ориентированные СУБД



# **Манифест разработчиков объектно-ориентированных систем баз данных**

- **Правило 1. Поддержка сложных объектов**
- **Правило 2. Поддержка идентификации объектов**
- **Правило 3. Поддержка инкапсуляции**
- **Правило 4. Поддержка типов или классов**
- **Правило 5. Поддержка наследования типов или классов от их предков**
- **Правило 6. Поддержка динамического связывания**
- **Правило 7. Язык DML должен обладать вычислительной полнотой**
- **Правило 8. Набор типов данных должен быть расширяемым**
- **Правило 9. Поддержка перманентности данных**
- **Правило 10. Поддержка очень больших баз данных**
- **Правило 11. Поддержка параллельной работы многих пользователей**
- **Правило 12. Способность восстановления после сбоев аппаратных и программных средств**
- **Правило 13. Предоставление простых способов создания запросов к данным**

# Объектно-ориентированные СУБД

## Достоинства

- Улучшенные возможности моделирования
- Расширяемость
- Устранение проблемы несоответствия типов
- Более выразительный язык запросов
- Поддержка эволюции схемы
- Поддержка долговременных транзакций
- Применимость для сложных специализированных приложений баз данных
- Повышенная производительность

- Недостатки
- **Отсутствие универсальной модели данных**
- Недостаточность опыта эксплуатации
- Отсутствие стандартов
- Конкуренция со стороны СУБД других типов
- Влияние оптимизации запросов на инкапсуляцию
- Влияние **блокировки на уровне объекта на производительность**
- Сложность
- Отсутствие поддержки **представлений**
- Недостаточность средств обеспечения защиты

# Объектно-реляционные СУБД

Реляционная СУБД+ Элементы ООП=  
Объектно-реляционные СУБД

- Примеры: PostgreSQL, Postgres, Oracle ,начиная с 8 версии

# Объектно-реляционные СУБД

## Достоинства

- *повторное и совместное* использование компонентов
- расширенный реляционный подход позволяет воспользоваться обширным объемом накопленных знаний и опыта, связанных с разработкой реляционных приложений

## Недостатки

- сложность и связанные с ней дополнительные расходы.
- объектно-реляционных системах искажается объектная терминология
- Объекты фактически не являются очередным расширением понятия данных, поскольку представляют совершенно другую концепцию, с большим потенциалом выражения связей и правил поведения объектов реального мира

# Манифесты баз данных третьего поколения ( 1990)

- 1. Наличие развитой системы типов.
- 2. Поддержка механизма наследования.
- 3. Поддержка функций, включая процедуры и методы базы данных, а также механизма инкапсуляции.
- 4. Уникальные идентификаторы для записей должны присваиваться средствами СУБД только в том случае, когда нельзя использовать определяемые пользователем первичные ключи.
- 5. Правила (триггеры и ограничения) станут важнейшим компонентом будущих систем. Они не должны быть связаны с какой-то конкретной функцией или коллекцией.
- 6. Очень важно, чтобы все программные способы доступа к базе данных осуществлялись с помощью непроцедурного языка высокого уровня.
- 7. Должны существовать по меньшей мере два способа определения коллекций: один на основе перечисления элементов коллекции, а другой — с использованием языка запросов для определения принадлежности элемента к коллекции.
- 8. Важным является наличие обновляемых представлений.
- 9. Средства измерения производительности не должны иметь никакого отношения к моделям данных и не должны в них присутствовать.
- 10. СУБД третьего поколения должны быть доступны из многих языков высокого уровня.
- 11. Желательно, чтобы во многих языках программирования высокого уровня поддерживались конструкции, обеспечивающие доступ к перманентным данным. Эта поддержка должна обеспечиваться на основе применения каждой отдельной СУБД с помощью дополнительных средств компилятора и сложной системы поддержки выполнения программ.
- 12. Плохо ли это или хорошо, но язык SQL должен оставаться "межгалактическим языком работы с данными".
- 13. Запросы и ответы на них должны составлять самый нижний уровень взаимодействия клиента и сервера.

# Объектно-ориентированные возможности PL/SQL (Oracle )

Средства и возможности	8.0	8.1	9.1	9.2 и выше	11g и выше
Абстрактные типы данных как равноправные сущности базы данных	*	*	*	*	*
Абстрактные типы данных как параметры PL/SQL	*	*	*	*	*
Атрибуты с типами коллекций	*	*	*	*	*
Атрибуты типа REF для навигации по объектам внутри базы данных	*	*	*	*	*
Реализация логики методов на PL/SQL и C	*	*	*	*	*
Определяемая программистом семантика сравнения объектов	*	*	*	*	*
Представление реляционных данных как данных объектных типов	*	*	*	*	*
Статический полиморфизм (перегрузка методов)	*	*	*	*	*
Возможность «эволюции» типа путем модификации логики существующих методов или добавления новых методов	*	*	*	*	*
Реализация логики методов на языке Java		*	*	*	*
«Статические» методы (уровня класса, а не уровня экземпляра)		*	*	*	*
Возможность использования первичного ключа как идентификатора хранимого объекта, обеспечивающая декларативную целостность REF-ссылок		*	*	*	*
Наследование атрибутов и методов от пользовательских типов			*	*	*
Динамическая диспетчеризация методов			*	*	*
Супертипы, для которых не могут создаваться экземпляры (аналоги абстрактных классов языка Java)			*	*	*
Возможность эволюции типа путем удаления методов (и добавления для изменения сигнатуры)			*	*	*

# Объектно-ориентированные возможности PL/SQL

Средства и возможности	8.0	8.1	9.1	9.2 и выше	11g и выше
Возможность эволюции типа путем добавления и удаления атрибутов, автоматическое распространение изменений на связанные структуры физической базы данных			*	*	*
«Анонимные» типы: ANYTYPE, ANYDATA, ANYDATASET			*	*	*
Операторы понижающего преобразования (TREAT) и определения типа (IS OF), доступные в SQL			*	*	*
Операторы TREAT и IS OF в PL/SQL				*	*
Определяемые пользователем конструкторы				*	*
Вызовы методов супертипа в производном типе					*
Приватные атрибуты, переменные, константы и методы					
Наследование от нескольких супертипов					
Совместное использование объектных типов или экземпляров в распределенных базах данных без обращения к объектным представлениям					

<https://oracle-patches.com/db/sql/3940-объектно-ориентированные-возможности-pl-sql>



# Пример наследования Posgres

- `CREATE TABLE cities`  
( name text,  
population real,  
altitude int -- (высота в футах )  
);
- `CREATE TABLE capitals`  
( state char(2) )  
INHERITS (cities);
- `SELECT name, altitude FROM cities WHERE`  
`altitude > 500;`
- `SELECT name, altitude FROM ONLY cities`  
`WHERE altitude > 500;`

# NOSQL

- ключ-значение (кластеры в распр. системе)
- документные (документы, иерархия)
- семейство столбцов (большие данные, слабоструктурированные данные)
- графовые

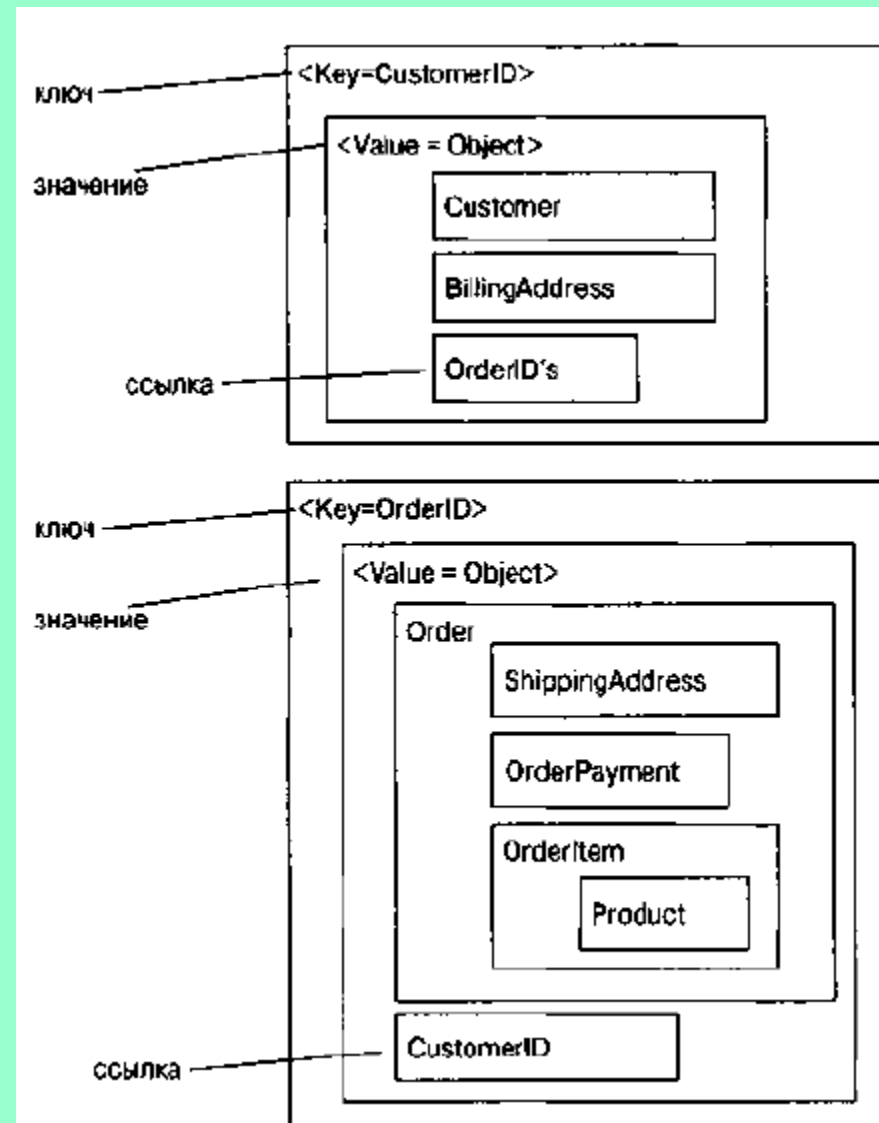
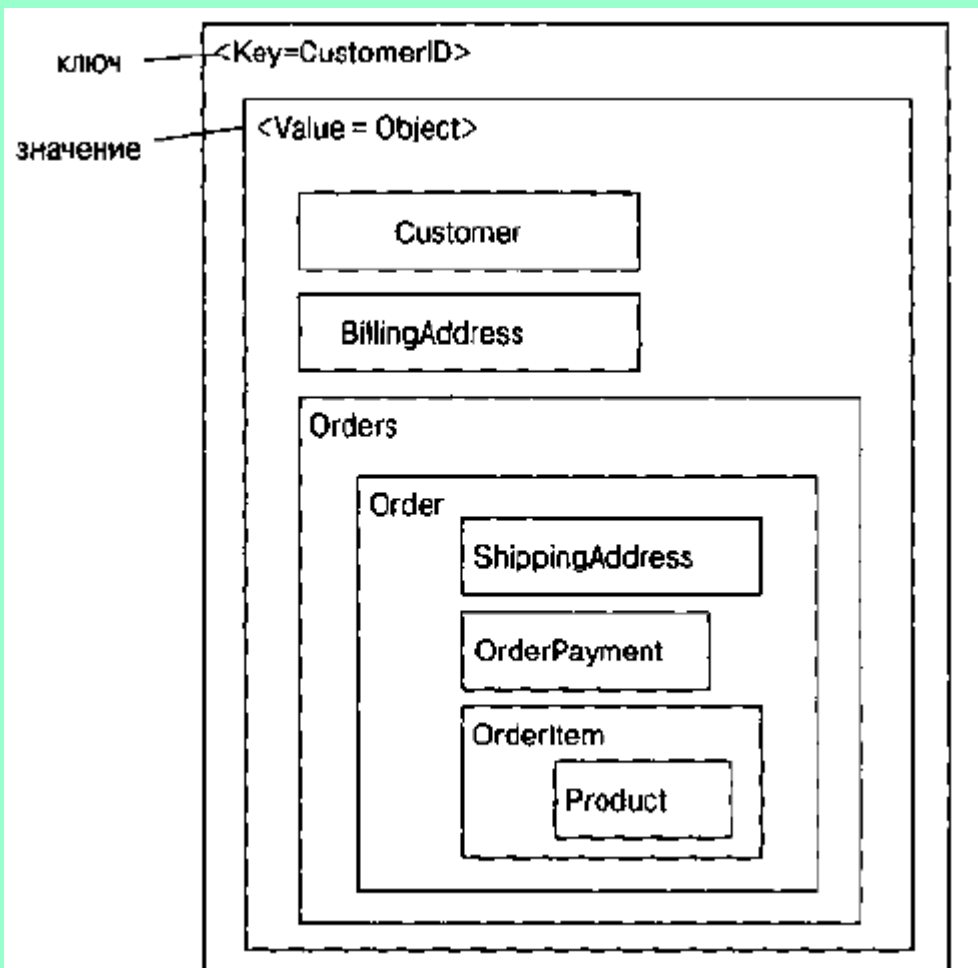
# Свойства NoSQL

- **Не используется SQL**
- **Неструктурированные (schemaless)**
- **Представление данных в виде агрегатов (aggregates)**
- **Слабые ACID свойства (транзакции)**
- **Распределенные системы, без совместно используемых ресурсов (share nothing)**

# ACID свойства

- **Atomicity** *Свойство атомарности* выражается в том, что транзакция должна быть выполнена в целом или не выполнена вовсе.
- **Consistency** *Свойство согласованности* гарантирует, что по мере выполнения транзакций данные переходят из одного согласованного состояния в другое — транзакция не разрушает взаимной согласованности данных.
- **Isolation** *Свойство изолированности* означает, что конкурирующие за доступ к базе данных транзакции физически обрабатываются последовательно, изолированно друг от друга, но для пользователей это выглядит так, как будто они выполняются параллельно.
- **Durability** *Свойство долговечности* трактуется следующим образом: если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах (даже в случае последующих ошибок).

# Агрегаты



# СУБД «ключ-значение»

- Хранилище типа "ключ-значение" - это простая хеш-таблица,.
- Клиент может либо получить значение по ключу, либо записать значение по ключу, либо удалить ключ из хранилища данных. Значение - это двоичный объект данных, который записан в хранилище без детализации его внутренней структуры. что именно хранится в этом объекте, определяет приложение.
- Примеры: Riak ,Redis (которую часто называют сервером Data Structure) [Redis], Memcached DB и ее версии ,[Memcached], Berkeley DB [Berkeley DB], HamsterDB

# СУБД «ключ-значение»

- Достоинства
- Хранение произвольных структур данных, например множеств, хеш-таблиц, строк
- Быстрый поиск по ключу
- Недостатки
- Согласованность данных относится только к операциям над отдельным ключом
- Не возможны запросы по атрибутам

# Документные СУБД

- Основной концепцией в документных базах данных является документ. База данных хранит и извлекает документы в форматах XML, JSON, BSON и т.д ..
- Эти документы представляют собой самоописываемые иерархические древовидные структуры данных, которые могут состоять из ассоциативных массивов, коллекций и скалярных значений.
- хранилища типа "ключ-значение", в которых значение допускает проверку
- Примеры: MongoDB



# СУБД «семейство столбцов»

- позволяют хранить данные с ключами, отображаемыми в значения, и группировать значения в многочисленных семействах столбцов, каждое из которых является ассоциативным массивом данных.

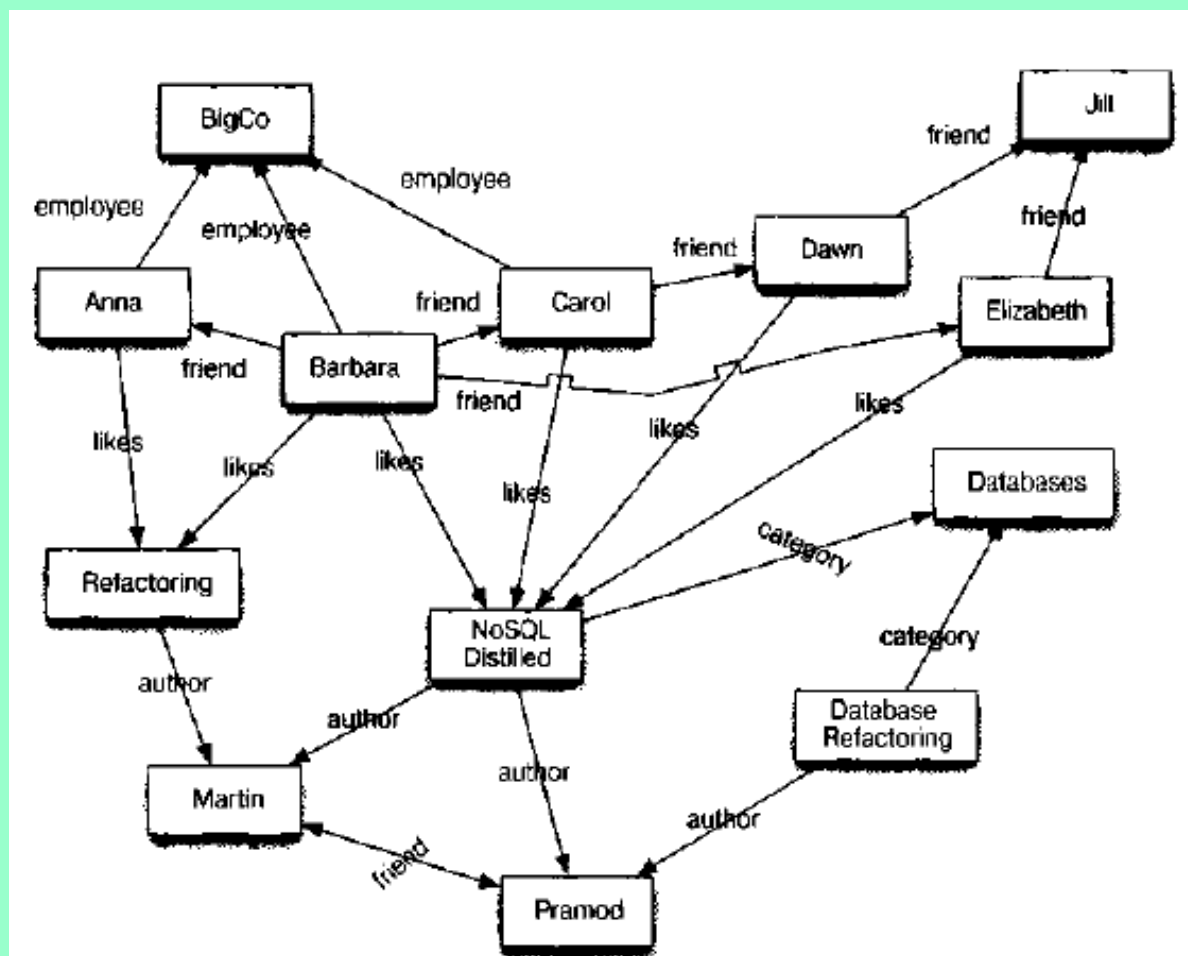
таблица 1						
ключ 1	столбец 1 (имя 1: значение)	столбец 2 (имя 2: значение)	столбец 3 (имя 3: значение)	столбец 4 (имя 4: значение)	столбец 5 (имя 5: значение)	столбец 6 (имя 6: значение)
ключ 2	столбец 1 (имя 1: значение)	столбец 3 (имя 3: значение)	столбец 4 (имя 4: значение)	столбец 6 (имя 6: значение)		
ключ 3	столбец 4 (имя 4: значение)	столбец 7 (имя 7: значение)	столбец 8 (имя 8: значение)			
ключ 4	столбец 1 (имя 1: значение)	столбец 2 (имя 2: значение)	столбец 3 (имя 3: значение)	столбец 4 (имя 4: значение)	столбец 5 (имя 5: значение)	

- Пример Cassandra [Cassandra], HBase [Hbase], Hypertable [Hypertable] и Amazon SimpleDB

# Графовые СУБД

- Графовые базы данных позволяют хранить сущности и отношения между ними.
- Сущности моделируются узлами, которые имеют свойства. Узел интерпретируется как экземпляр объекта в приложении. Отношения моделируются ребрами, которые могут иметь свойства. Ребра имеют направление; узлы организованы в соответствии с отношениями.

# Графовые СУБД



Пример:  
AllegroGraph,  
ArangoDB,  
FlockDB,  
Giraph,  
HyperGraphDB  
(использует модель  
мультиграфа),  
InfiniteGraph,  
InfoGrid,  
Neo4j (использует  
модель  
ориентированного  
графа)

# Свойства NoSQL

- **Не используется SQL**
- **Неструктурированные (schemaless)**
- **Представление данных в виде агрегатов (aggregates)**
- **Слабые ACID свойства (транзакции)**
- **Распределенные системы, без совместно используемых ресурсов (share nothing)**