Лекция 1-2019_2020

- Содержание курса
- Примеры БД
- Основные понятия и определения
- История развития БД
- Модели представления данных

Содержимое курса

1. 1 модуль (2019) – Проектирование. 2 самостоятельные работы, тестирование, контрольная робота. MS Access 2010.
2. 2 модуль (2019 – Структурированный язык запросов (SQL). 2 самостоятельные работы, тестирование, контрольная робота. MS Access 2010.
3. Зачет. – Решение задачи, тестирование.
4. 3 модуль (2020) – Серверы баз данных. 4 самостоятельные работы, тестирование, к/р. MySQL, SQLyog, MySQL Workbench, PHPMyAdmin, PHP, HTML, сборки веб-серверов (Денвер,OpenServer, XAMPP), C# + VS, SQLite, PostgreSQL, Java + Intellij IDEA, PHP + MongoDB.
5. Курсовая работа (май 2020).
6. Экзамен (июнь)

Примеры баз данных.

1.Покупка в супермаркете.

2.Заказ путевки в туристическом агентстве.

3.Письма, полученные или отправленные по e-mail.

4.Мобильный телефон, звонки разделены на входящие, исходящие, пропущенные.

5. Некомпьютерная БД – телефонная книжка.

6. Некомпьютерная БД – карточки с дырочками в библиотеке.

История развития БД

60-е годы XX столетия – разработка проекта полета корабля Apollo на Луну. Подрядчик NAA разработал программное обеспечение под названием GUAM (Generalized Update Access Method). Идея GUAM – малые компоненты объединяются вместе как части более крупных компонентов до тех пор, пока не будет собран воедино весь проект (иерархическая структура).

NAA + IBM – создание системы IMS (Information Management System) для хранения иерархий записей с последовательным доступом (магнитные ленты). Используется до сих пор на большинстве крупных мейнфреймов.

Основные понятия и определения

Файл – это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные.

Некоторая часть реального мира, функционирующая как самостоятельная единица, представляет собой предметную область.

Всякая профессиональная деятельность связана с информацией, с ее организацией, хранением, сбором и выборкой. Для обработки информации и создаются информационные системы (ИС).

Основные понятия и определения

Совместно используемый набор логически связанных данных и их описаний, предназначенный для удовлетворения информационных потребностей организации, называется базой данных (БД).

Система управления базой данных (СУБД) – программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать БД, а также получать контролируемый доступ к БД.

Типы информации в базе данных

1. Данные пользователей.

2. Метаданные. Это “данные о данных”, описание структуры БД.

3. Индексы. Это специальные, избыточные данные. Используются для сортировки и для быстрого доступа к данным.

4. Метаданные приложений. Описание структуры и формата пользовательских форм, отчетов, запросов и других компонентов приложений. (Access, FoxPro)

5. Схема БД (Access)

Классификация СУБД

1. Полнофункциональные СУБД. Они имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД. (Clarion, DataFlex, dBase 4, MS Access, MS FoxPro, Paradox).

2. Серверы БД – предназначены для организации центров обработки данных в сетях. (MS SQL Server, MySQL, PostgreSQL, InterBase, FireBird, SyBase, Oracle)

Функции СУБД

1. Непосредственное управление данными во внешней памяти.
2. Управление буферами оперативной памяти.
3. Управление транзакциями. Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое.
4. Журнализация. Журнал – это особая часть БД, недоступная пользователям СУБД, в которую поступают записи обо всех изменениях основной части БД. «Упреждающая" запись в журнал (протокол Write Ahead Log - WAL).
5. Поддержка языков БД. Для РСУБД – SQL.

Для пользователя СУБД предоставляет :

1. Возможность создать базу данных.
2. Возможность вставлять, обновлять, удалять и извлекать информацию из базы данных.
3. Контролируемый доступ к базе данных.

Определения, связанные с БД

1. Модель представления данных – логическая структура хранимых в базе данных.

2. Приложение – программа или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи..

3. Администратор базы данных (АБД) – лицо или группа лиц, отвечающих за выработку требований к БД, ее проектирование, создание, эффективное использование и сопровождение.

Модели представления данных

СУБД основаны на моделях баз данных — определённых структурах для обработки данных. Каждая СУБД реализует одну из моделей баз данных для логической структуризации используемых данных. Эти модели являются главным критерием того, как будет работать и управлять информацией приложение.

Решений, реализующих различные модели баз данных, очень много. Периодически некоторые из них становятся очень популярными и используются на протяжении многих лет.

Модели представления данных

Основные модели представления данных:

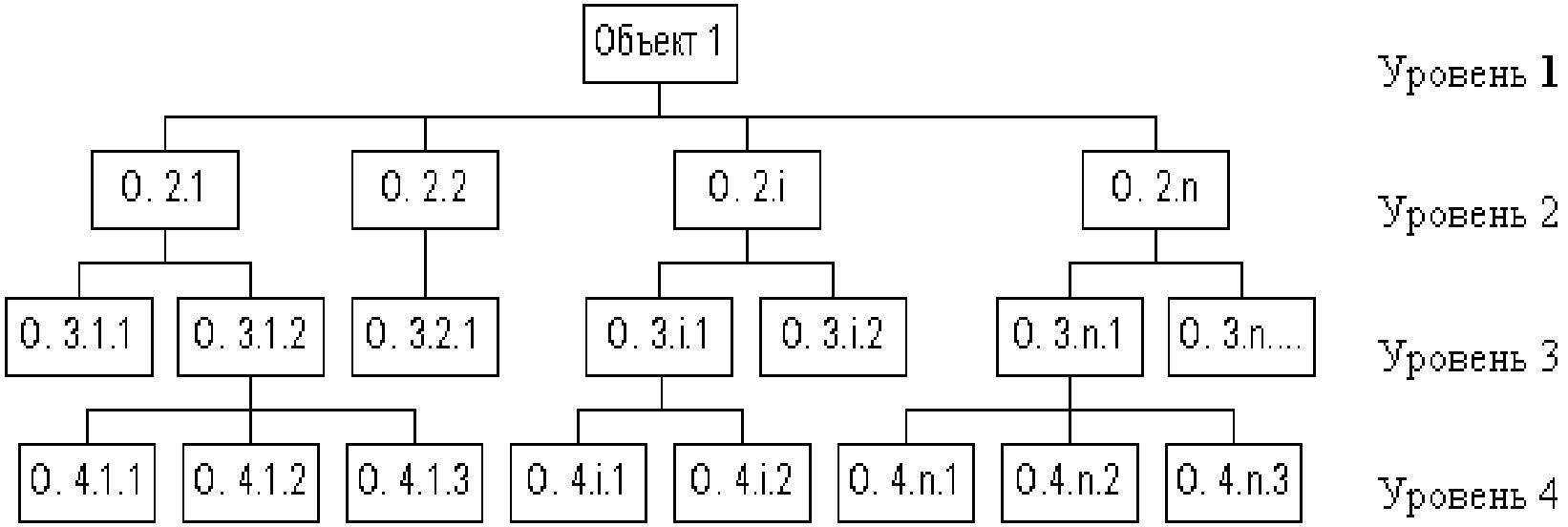
- иерархическая,
- сетевая,
- реляционная,
- постреляционная,
- многомерная,
- объектно-реляционная,
- объектно-ориентированная,
- NoSQL,
- NewSQL.

Разрабатываются модели – расширения известных.

Иерархическая модель

Иерархическая БД состоит из упорядоченного набора нескольких экземпляров одного типа дерева.

Основное правило: запись-потомок должна иметь в точности одного предка.



Иерархические СУБД: IMS, PC/Focus, Team-Up, Data Edge, ИНЭС, МИРИС.

Задача: предприятие-отделы-контракты-заказчики

Предприятие состоит из отделов, в которых работают сотрудники. В каждом отделе может работать несколько сотрудников, но сотрудник не может работать более чем в одном отделе. Предприятие в лице своих сотрудников выполняет работы по контракту с юридическими или физическими лицами (заказчиками).

Пример иерархической модели данных

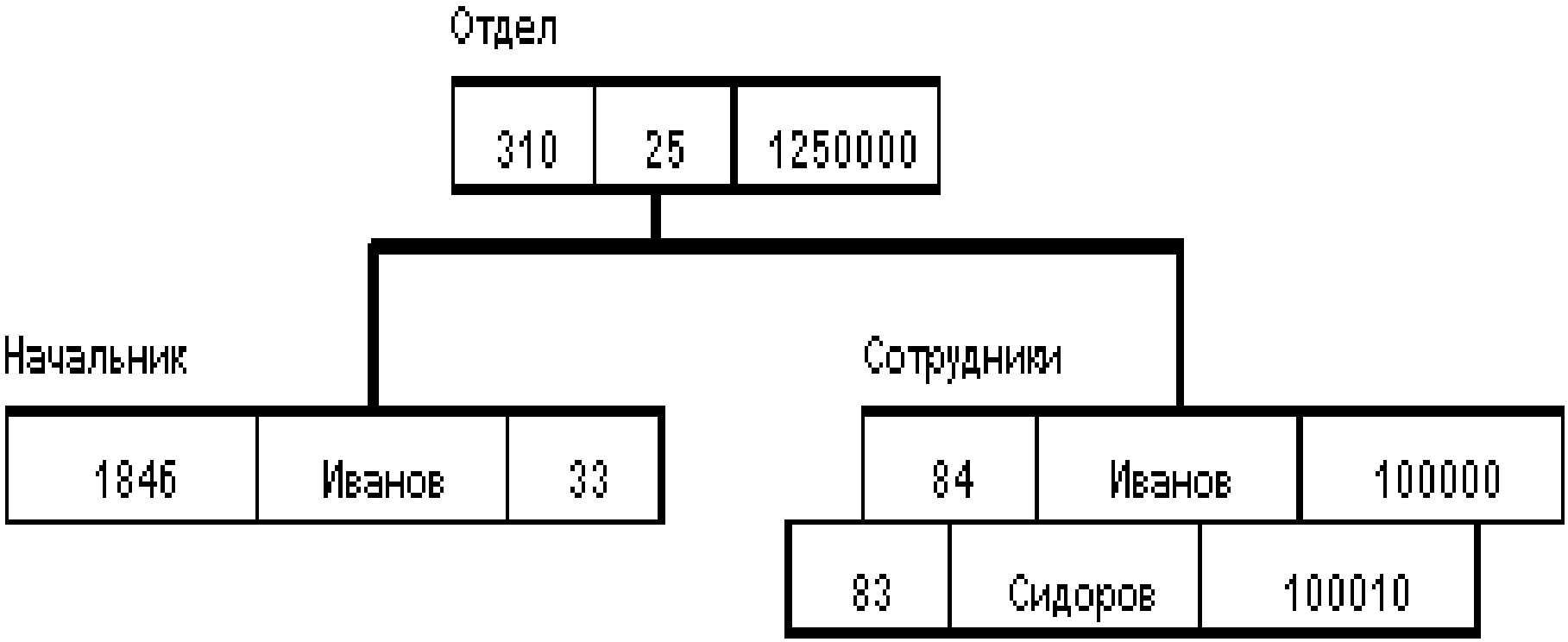
Отношение (а) отдел - сотрудники, состоящее из родительской записи ОТДЕЛ (*НАИМЕНОВАНИЕ_ОТДЕЛА, ЧИСЛО_РАБОТНИКОВ*) и дочерней записи СОТРУДНИК (*ФАМИЛИЯ, ДОЛЖНОСТЬ, ОКЛАД*).

Структура экземпляра дерева:

(а)

Для простоты полагаем, что имеются только две дочерние записи.

Пример иерархической модели данных



Пример экземпляра дерева с видоизмененной структурой (в отдел добавлено поле ФОНД, для сотрудника добавлен табельный номер). Деревьев столько, сколько отделов.

Пример иерархической модели данных

Операции:

1. Найти указанное дерево БД (например, отдел 310);
2. Перейти от одного дерева к другому;
3. Перейти от одной записи к другой внутри дерева (например, от отдела - к первому сотруднику);
4. Перейти от одной записи к другой в порядке обхода иерархии;
5. Вставить новую запись в указанную позицию;
6. Удалить текущую запись.

Пример иерархической модели данных

Отношение (б) заказчик - контракты – сотрудники:

ЗАКАЗЧИК (НАИМЕНОВАНИЕ_ЗАКАЗЧИКА, АДРЕС),

КОНТРАКТ (НОМЕР, ДАТА,СУММА),

ИСПОЛНИТЕЛЬ (ФАМИЛИЯ, ДОЛЖНОСТЬ, НАИМЕНОВАНИЕ_ОТДЕЛА)

(b)

Пример иерархической модели данных

Если исполнитель принимает участие более чем в одном контракте (связь типа M:N).

Отношение (c)

исполнитель - контракт:

(c)

ИСПОЛНИТЕЛЬ (ФАМИЛИЯ, ДОЛЖНОСТЬ, НАИМЕНОВАНИЕ_ОТДЕЛА)

КОНТРАКТ (НОМЕР, ДАТА,СУММА)

Пример иерархической модели данных

(a)

(b)

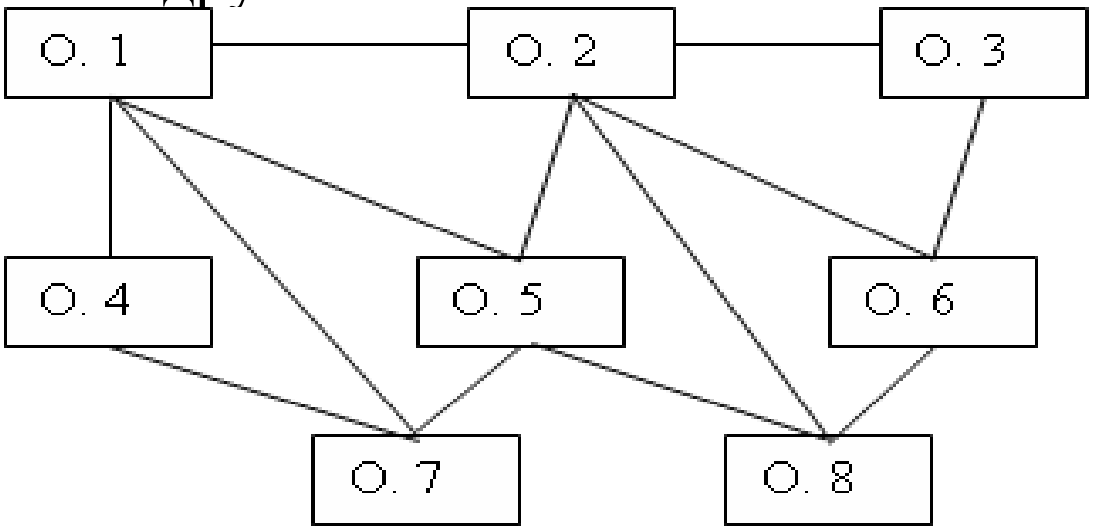
(c)

Недостатки иерархической модели для этой БД

- Частично дублируется информация между записями СОТРУДНИК и ИСПОЛНИТЕЛЬ (парные записи). В иерархической модели данных не поддерживается соответствие между парными записями.
- Иерархическая модель реализует отношение между исходной и дочерней записью по схеме 1:N, то есть одной родительской записи может соответствовать любое число дочерних. Если присутствует отношение (с) – опять дублирование информации.
- Затруднения при выполнении операций включения и удаления.

Сетевая модель

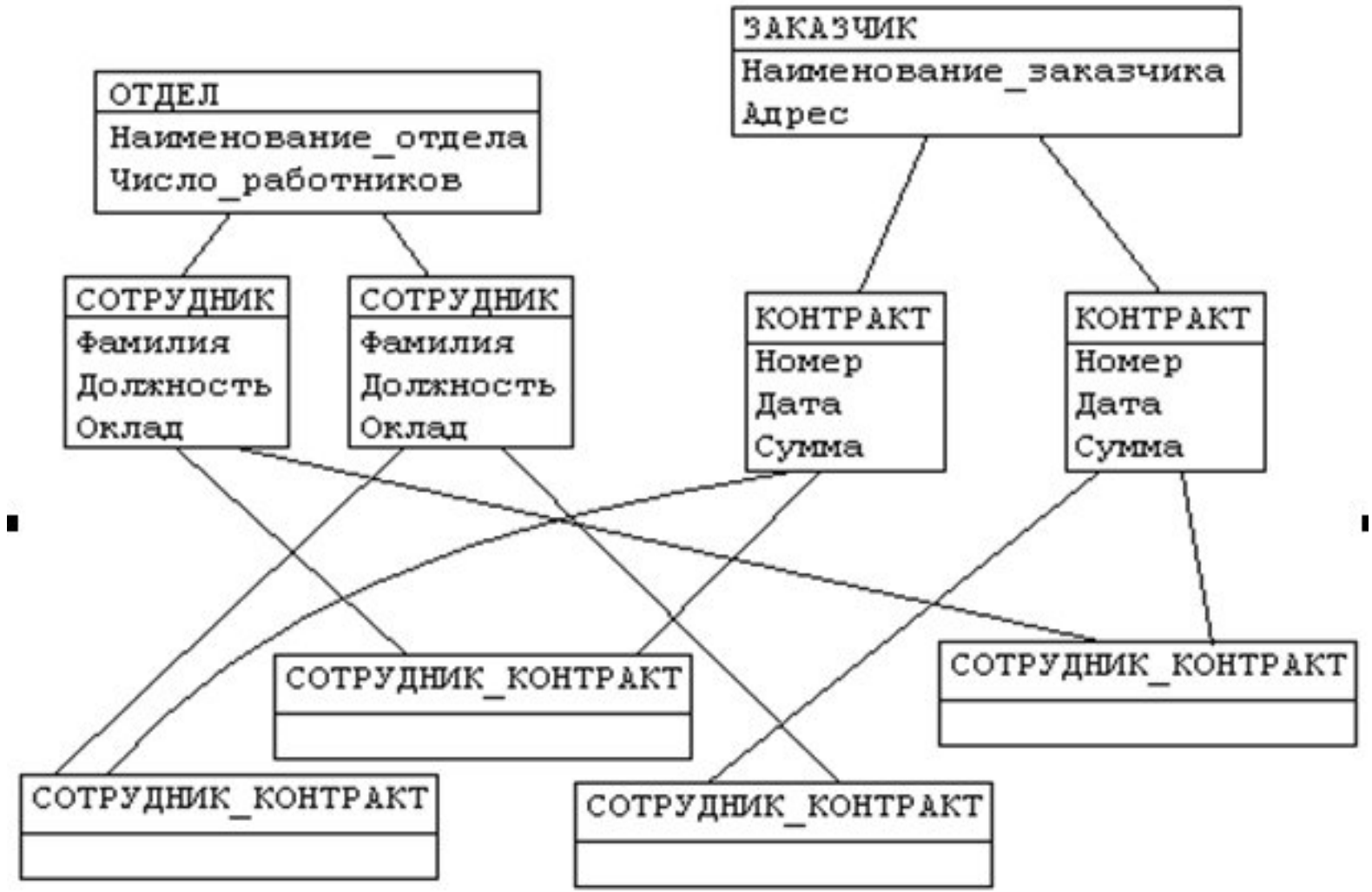
Середина 60-х — система IDS (Integrated Data Store), Чарльз Бачман, премия Тьюринга а работу «Программист как навигатор» . Развитие IDS — создание сетевых СУБД. Основное правило: любой объект может быть одновременно и главным, и подчиненным, и может участвовать в образовании любого числа взаимосвязей с другими объектами.



Сетевые СУБД: IDS, db_VistaIII, СЕТЬ, СЕТОР, КОМПАС.

Преобразование иерархической структуры примера в сетевую

Запись *СОТРУДНИК_КОНТРАКТ* не имеет полей, служит для связи записей *КОНТРАКТ* и *СОТРУДНИК*.



Недостатки иерархической и сетевой моделей

Основными недостатками сетевой модели данных являются: сложная структура памяти и необходимость понижать сложность сетевой модели (исключать имеющиеся циклы).

- Даже для выполнения простых запросов с использованием переходов и доступом к определенным записям необходимо создавать достаточно сложные программы.
- Независимость от данных существует лишь в минимальной степени.
- Отсутствие общепризнанных теоретических основ.

Реляционная модель

В 1970 году Эдгар Кодд (IBM) предложил реляционную модель данных – данные представляются в виде двумерных таблиц (отношений). Основа – теория множеств и логика предикатов. Достоинство – простота, понятность и удобство физической реализации на ЭВМ.

Проект **System R (**IBM). Цель проекта – доказать практичность реляционной модели. Результат проекта – **разработан структурированный язык запросов SQL**, который стал стандартным языком реляционных СУБД. Реляционные СУБД относятся к СУБД **второго поколения.**

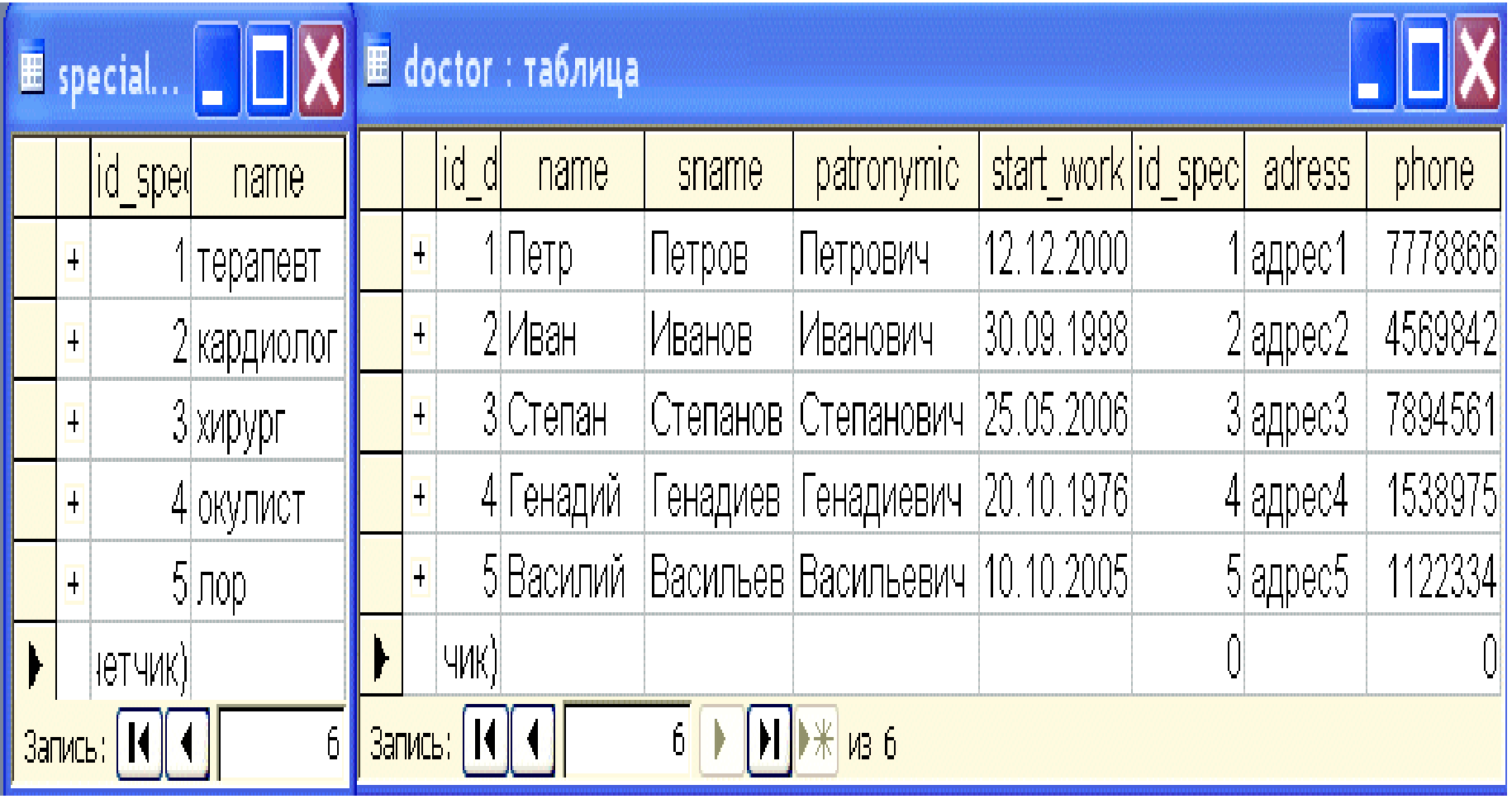
Реляционные СУБД предназначаются для **оперативной** обработки информации.

РСУБД: dBaseIII Plus и dBase IY, DB2, R:BASE, FoxPro, Visual FoxProe, Paradox, dBase for Windows, Access, Clarion, Ingres, Oracle (до 8 версии), ПАЛЬМА, HyTech.

Для того, чтобы таблица была отношением

- Все строки таблицы должны быть уникальны. Уникальность достигается за счет наличия ключей.
- Все строки таблицы должны иметь одну и ту же структуру, т.е. одно и то же количество столбцов с соответственно совпадающими именами.
- Имена столбцов таблицы должны быть различны, а данные в одном столбце должны быть однотипными.
- Значения атрибутов должны быть атомарными (скалярными), т.е. отношения не могут иметь в качестве компонент другие отношения или массивы.
- Порядок следования строк в таблице несущественен, так как влияет лишь на скорость доступа к строке.
- Порядок следования столбцов в таблице несущественен.

Реляционная модель



special...

	id_spec	name
+	1	терапевт
+	2	кардиолог
+	3	хирург
+	4	окулист
+	5	лор
▶	(етчик)	

Запись: 6

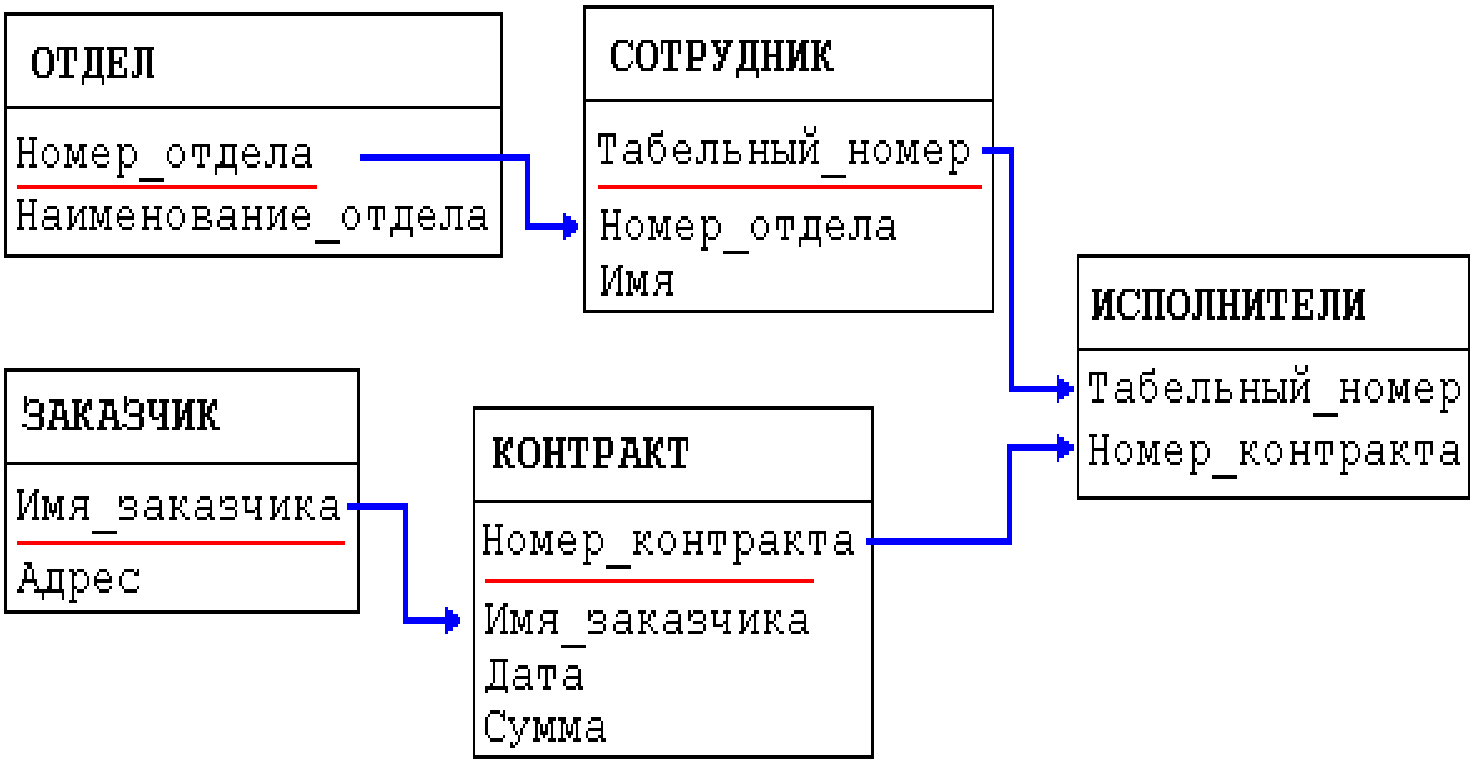
doctor : таблица

	id_d	name	sname	patronymic	start_work	id_spec	adress	phone
+	1	Петр	Петров	Петрович	12.12.2000	1	адрес1	7778866
+	2	Иван	Иванов	Иванович	30.09.1998	2	адрес2	4569842
+	3	Степан	Степанов	Степанович	25.05.2006	3	адрес3	7894561
+	4	Генадий	Генадиев	Генадиевич	20.10.1976	4	адрес4	1538975
+	5	Василий	Васильев	Васильевич	10.10.2005	5	адрес5	1122334
▶	чик)					0		0

Запись: 6 из 6

Реляционная модель данных к задаче о предприятии, сотрудниках и контрактах

Для связей между строками разных отношений используется дублирование их ключей. Например, связь между ОТДЕЛ и СОТРУДНИК создается путем копирования первичного ключа "Номер_отдела" из первого отношения во второе.



Постреляционная модель данных

Постреляционная модель данных – расширенная реляционная модель. Постреляционная модель допускает **многозначные** поля, значения которых состоят из **подзначений**. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Примеры ПРСУБД: uniVers, Bubba, Dasdb.

Задача о накладных и товарах. Реляционная модель данных.

Накладные

N накладной	Покупатель
03	Сельпо
83	Дигма
73	Таргет

Накладные-товары

N накладной	Товар	Количество
03	Сыр	3
03	Рыба	2
83	Лимонад	1
83	Сок	6
83	Печенье	2
73	Йогурт	1

Задача о накладных и товарах. Постреляционная модель данных.

Накладные

N накладной	Покупатель	Товар	Количество
03	Таргет	Сыр	3
		Рыба	2
83	Дигма	Лимонад	1
		Сок	6
		Печенье	2
73	Сельпо	Йогурт	1

Достоинства и недостатки постреляционной модели

Достоинство – возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостаток – сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

Реляционные и постреляционные БД служат для оперативной (транзакционной) обработки.

Многомерная модель

1993 год – статья Э. Кодда с 12 основных требований к системам класса OLAP (OnLine Analytical Processing).

Многомерные СУБД предназначены для интерактивной аналитической обработки информации в системах поддержки принятия решений.

Основные понятия: измерение и ячейка.

Измерение – множество однотипных данных, образующих одну из граней гиперкуба.

Ячейка – поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой (переменная либо формула).

Примеры систем, поддерживающих многомерное представление данных: Essbase, Media Multi-matrix, Oracle Express Server, Cache.

Задача о вычислении объема продажи автомобилей.

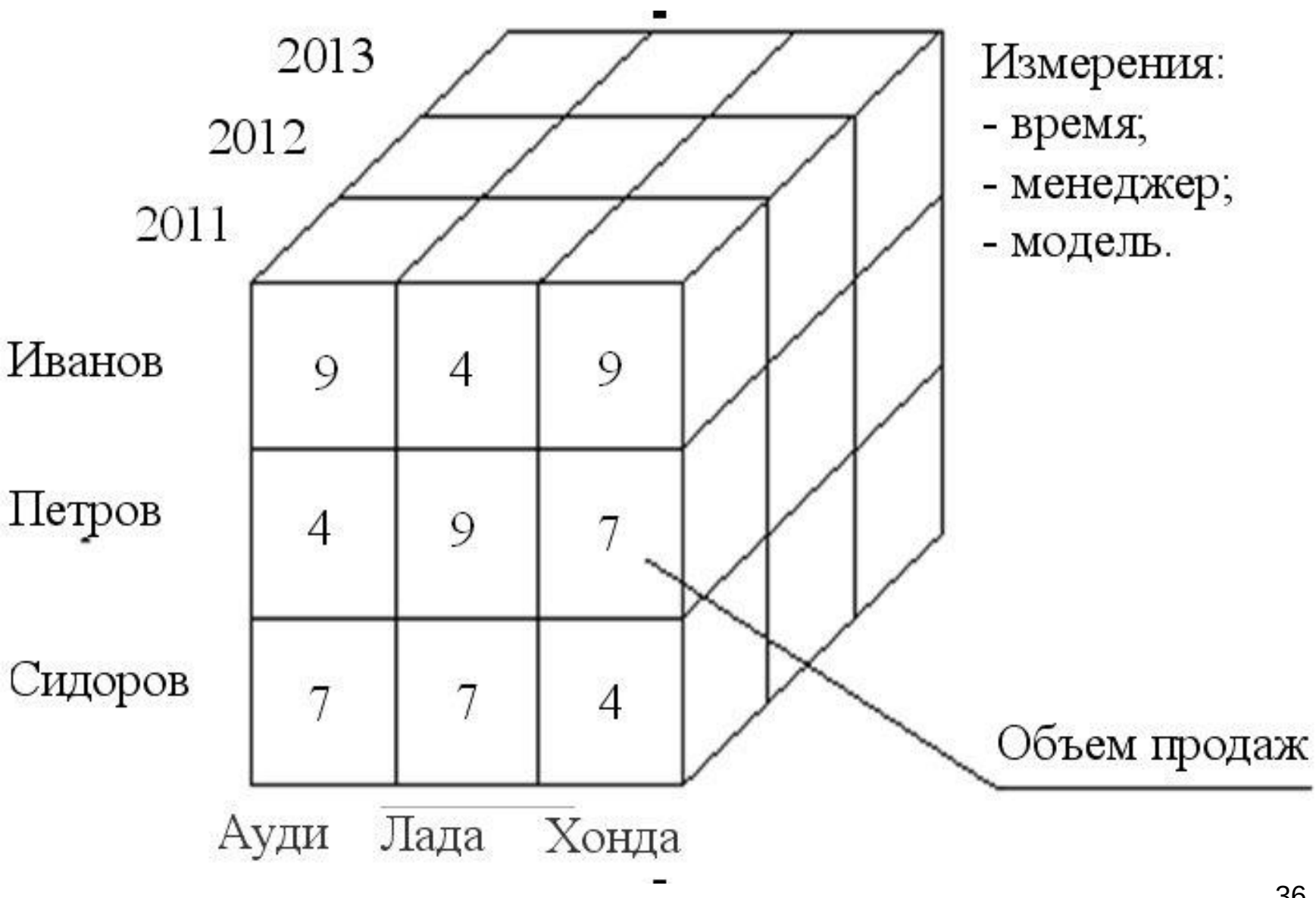
Реляционная модель

Модель	Месяц	Объем
Ауди	июнь	12
Ауди	июль	24
Ауди	август	5
Лада	июнь	2
Лада	июль	18
Хонда	июль	19

Многомерная модель (двухмерное представление)

Модель	Июнь	Июль	Август
Ауди	12	24	5
Лада	2	18	No
Хонда	No	19	No

Пример трехмерной модели .



Объектно-ориентированная модель

ООБД оперируют объектами , между данными и функциями их обработки устанавливаются взаимосвязи подобные соответствующим средствам в объектно-ориентированных языках программирования.

Поиск в ООБД состоит в выяснении сходства между объектом, задаваемым пользователем (объект-цель), и объектом, хранящимся в БД. *Примеры: G-Base, GemStone, Statice, ObjectStore, Versant, O2, ODB-Jupiter, Iris, Orion, Postgres.*

Объектные расширения СУБД Oracle

СУБД Oracle 11gR2 – объектно-реляционная БД, устойчивый, обеспеченный документацией релиз, поддерживает SQL 2008.

Oracle позволяет работать с объектами:

классами, массивами и вложенными таблицами.

Объектный тип определяется в CREATE TYPE.

Для класса: имя, атрибуты, свойства и методы.

Объект класс в СУБД Oracle

CREATE TYPE – описание класса:

```
CREATE TYPE TYPE1 AS OBJECT (
   ID_TYPE1 NUMBER(19),
   PARAM1 VARCHAR2(250),
   MEMBER FUNCTION FUNC1 RETURN NUMBER );
```

CREATE TYPE BODY – код методов класса.

Например:

```
CREATE TYPE BODY TYPE1 AS
MEMBER FUNCTION FUNC1 RETURN NUMBER IS
   BEGIN
   RETURN 1;
   END;
END;
```

Объект массив в СУБД Oracle

Массивы представляют собой упорядоченные списки однотипных объектов. При создании массива требуется указать тип значений и его максимальный размер.

Например:

```
CREATE TYPE VARRAY1_TYPE IS VARRAY (10000) OF VARCHAR2(20);
```

Oracle хранит массив как LOB (Large Object) в упакованном виде, при этом упаковкой и распаковкой значений из LOB занимается сама СУБД.

Объект вложенные таблицы в СУБД Oracle

Вложенные таблицы используются для хранения неупорядоченного списка объектов. Значения элементов списков всегда хранятся в отдельной таблице, а индексы элементов списка в родительской таблице.

Например:

```
CREATE TYPE TAB1_TYPE AS TABLE OF VARCHAR2(32);
CREATE TABLE TAB2( PARAM1 TAB1_TYPE )
NESTED TABLE PARAM1 STORE AS PARAM1_TAB;
```

Способы хранения объектов в СУБД Oracle

Объекты можно хранить в столбце таблицы. В этом случае вначале создают объектный тип данных (класс), а потом определяют атрибут родительской таблицы этим типом.

Например:

```
CREATE TYPE address_type AS OBJECT (
  zip       CHAR      ( 6 ),
  location  VARCHAR2  ( 200 ))

CREATE TABLE person (
  dname     VARCHAR2  ( 20 ),
  addr      address_type );
```

При вставке новых значений объектного типа осуществляется обращение к конструктору объекта, который автоматически создается СУБД при заведении нового типа.

Способы хранения объектов в СУБД Oracle

Другой способ хранения: создание таблицы объектов.

Таблицы объектов в Oracle представляют собой списки объектов, это таблицы из одного столбца объектного типа. На такие объекты можно ссылаться, в таблице можно разместить как объекты класса, так и объекты - наследники класса.

Например:

```
Create type player_type as object (
 id_player number(19),
 first_name varchar2(250),
 last_name varchar2(250),
 birthday date,  address varchar2(500));

create table players of player_type
(id_player primary key);
```

Достоинства и недостатки ООБД

Достоинства:

Удобно использовать в сочетании с объектно-ориентированными языками программирования для приложений.

Недостатки:

высокая понятийная сложность и **низкая** скорость выполнения запросов.

NoSQL

NoSQL – “безмодельный” подход.

Эти системы не используют строгую структуризацию данных. Способ структуризации данных заключается в избавлении от ограничений при хранении и использовании информации.

Сторонники NoSQL баз утверждают, что их можно использовать для создания более производительных, легче масштабируемых и проще программируемых систем. Тем не менее, реляционные базы данных удерживают стабильно доминирующие позиции.

SQL и NoSQL системы

Несколько ключевых различий между этими типами СУБД.

1. В SQL БД есть четкие схемы таблиц, в NoSQL нет заранее заданных схем документов.
2. В SQL существуют сложные связи между различными таблицами, в NoSQL, как правило, каждый документ является изолированной единицей и хранит в себе все имеющиеся данные.
3. Механизмы поддержки целостности данных есть в SQL и отсутствуют в NoSQL.
4. В SQL есть механизм транзакций, в NoSQL − только в пределах одного документа.
5. В идеальном случае NoSQL работает быстрее, но практика показывает, что объемы до 20 миллионов записей отлично перерабатываются SQL базами.

Модели для NoSQL БД

Модели и функциональные системы для NoSQL БД:

• Хранилище «ключ-значение». Данные хранятся в простой хеш-таблице. Доступ к данным осуществляется по ключам. По ключу можно получить значение или записать значение в базу. (Redis, MemcacheDB, Riak, Berkeley DB и т.п.).

• Хранилище колонок. Основная идея – группировка подобных значений в семейство столбцов. Такое семейство хранится в отдельном файле (Cassandra, Hbase, Google Big Table и т.п.).

Модели для NoSQL БД

• Документоориентированные СУБД. Это хранилище для документов типа XML, JSON, BSON и др. Документоориентированные хранилища отлично хранят несвязанную информацию больших объемов, даже если она очень разнится от сущности к сущности (MongoDB, CouchDB, MarkLogic, eXist и т.п.).

• Графовые СУБД хранят сущности и отношения между ними. Сущности представляются в виде узлов, которые имеют свойства. Отношения представляются в виде ребер, которые тоже могут иметь свойства. (OrientDB, Neo4J, ArangoDB, FlockDB, HyperGraphDB и т.п.).

NewSQL БД

NewSQL– класс современных реляционных СУБД, стремящихся совместить в себе преимущества NoSQL и транзакционные требования классических БД. Потребность в данных системах возникла у компаний, работающих с критическими данными (например, финансового сектора), которым требовались масштабируемые решения, в то время как решения NoSQL не могли предоставить транзакций и не отвечали требованиям надёжности данных.

Представители VoltDb, Google Cloud Spanner.

Spanner

1. Наличие NoSQL возможностей;
2. Поддержка распределенных транзакций;
3. Глобальная согласованность операций чтения между географически распределенными ДЦ, т.о. данные, которые возвращают операции чтения из разных ДЦ, всегда согласованны и непротиворечивы.
4. Автоматическая обработка отказов как вычислительных узлов, так и ДЦ;
5. Автоматическая миграция данных как между вычислительными узлами, так и между ДЦ.
6. И т.д.