В. Б. Пикулев

БАЗЫ ДАННЫХ

1. Основы теории баз данных

scilink.ru, 2021

Основные понятия

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемая для хранения, обработки и представления информации.

Файловая система – пример информационной системы для структурирования данных и управления данными. Однако для файловой системы характерно слабая структуризация: имя и расширение файла могут не соответствовать типу и виду данных, хранящихся в нём, одни и те же данные могут дублироваться в различных каталогах, а ожидаемые копии могут отличаться друг от друга, обычно отсутствует схема организации данных, поиск работает медленно и не всегда гибко организован.

База данных призвана обеспечить согласованное хранение информации в любых системах, которые могут поддерживать механизмы обеспечения целостности и внутренней непротиворечивости данных. Если некая информационная система поддерживает согласованное хранение информации в нескольких файлах, можно говорить о том, что она поддерживает *базу данных*.

В широком смысле база данных – это совокупность сведений, связанных с какой-либо предметной областью.

Основные понятия

База данных (БД, database) – структурированный набор логически связанных между собой данных, содержащий сведения о конкретной предметной области.

Система управления базами данных (СУБД, DBMS) – программное обеспечение, с помощью которого возможно определять, создавать, поддерживать в актуальном состоянии базу данных и осуществлять к ней контролируемый доступ.

Транзакция – последовательность операций над базой данных, рассматриваемых с точки зрения СУБД как единое целое.

Функции СУБД:

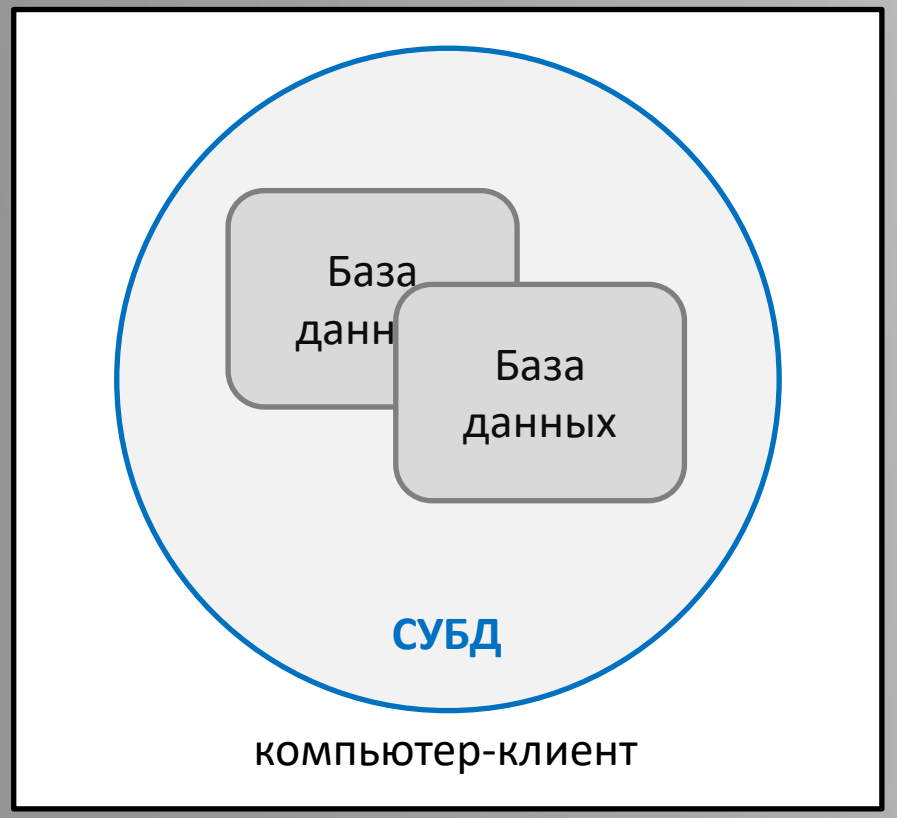
- Обеспечение целостности и непротиворечивости данных в любых условиях функционирования информационной системы
- Предотвращение несанкционированного доступа к данным
- Организация многопользовательского доступа к данным
- Обеспечение максимальной эффективности при работе с данными

Литература

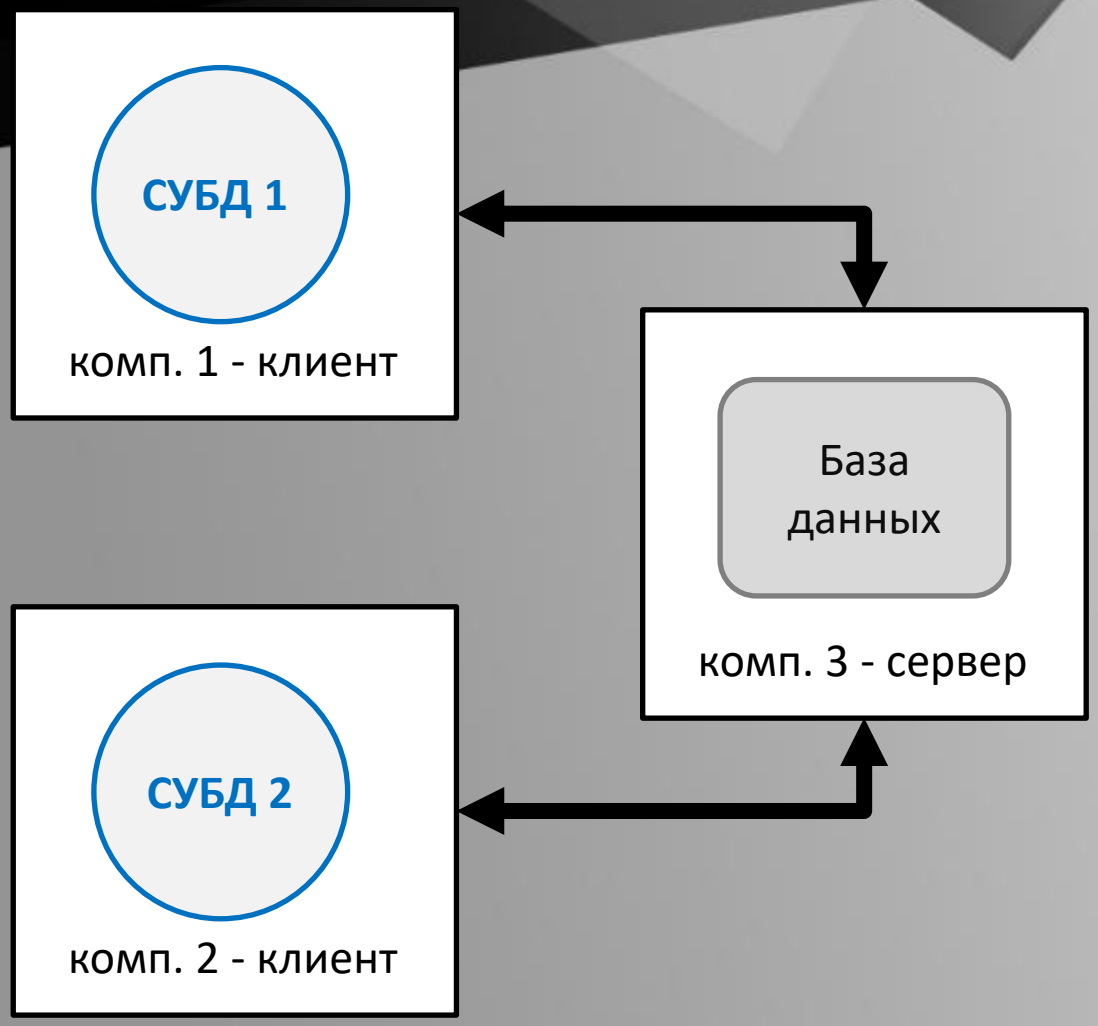
1. **Конноли, Т.** Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Конноли, К. Бегг – М: Издательский дом "Вильямс", 2003. – 1440 с.
2. **Грофф, Д.** SQL: полное руководство, 3-е изд. / Д. Р. Грофф, П. Н. Вайнберг, Э. Дж. Оппель - М.: ООО "И.Д. Вильямс", 2015. - 960 с.
3. **Новиков Б.** Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. — М.: ДМК Пресс, 2020. - 582 с.
4. **Лузанов П.** Postgres. Первое знакомство / П. В. Лузанов, Е. В. Рогов, И. В. Лёвшин. – М.: "Постгрес Профессиональный", 2019. – 157 с.
5. **Моргунов Е.** PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. – СПб.: БХВ-Петербург, 2018. – 336 с.
6. **Бондарь А.** MS SQL Server 2012. – СПб.: БХВ-Петербург, 2013. – 608 с.
7. **Колисниченко Д.** PHP и MySQL. Разработка Web-приложений. – СПб.: БХВ-Петербург, 2013. – 560 с.

Схемы доступа к данным

1. Централизованная БД хранится на одном выделенном компьютере, доступ к которому имеет ограниченное количество пользователей, обычно в режиме прямого либо терминального доступа. При отсутствии сетевого подключения защищённость данных достаточно высока, но эффективность работы с такой БД минимальная. Схема реализуется с использованием однопользовательских СУБД.

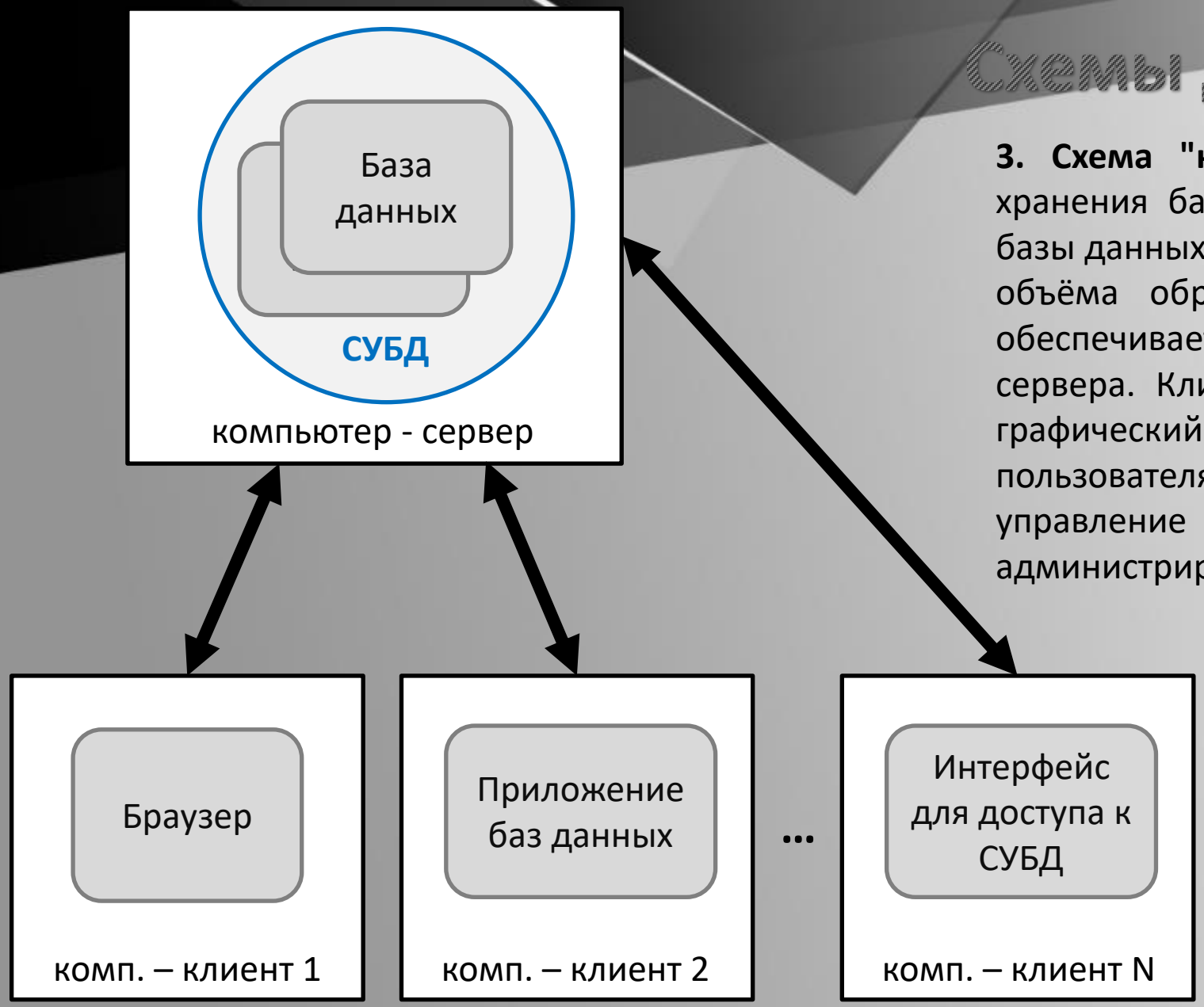


Схемы доступа к данным



2. Файловый сервер – на одном из компьютеров располагается банк данных – файлы, которые могут совместно обрабатывать несколько СУБД пользователей. Схему возможно применять для очень небольшого числа клиентов, при этом защищённость данных практически отсутствует, а скорость работы системы низкая. На клиентских компьютерах обычно используются однопользовательские СУБД.

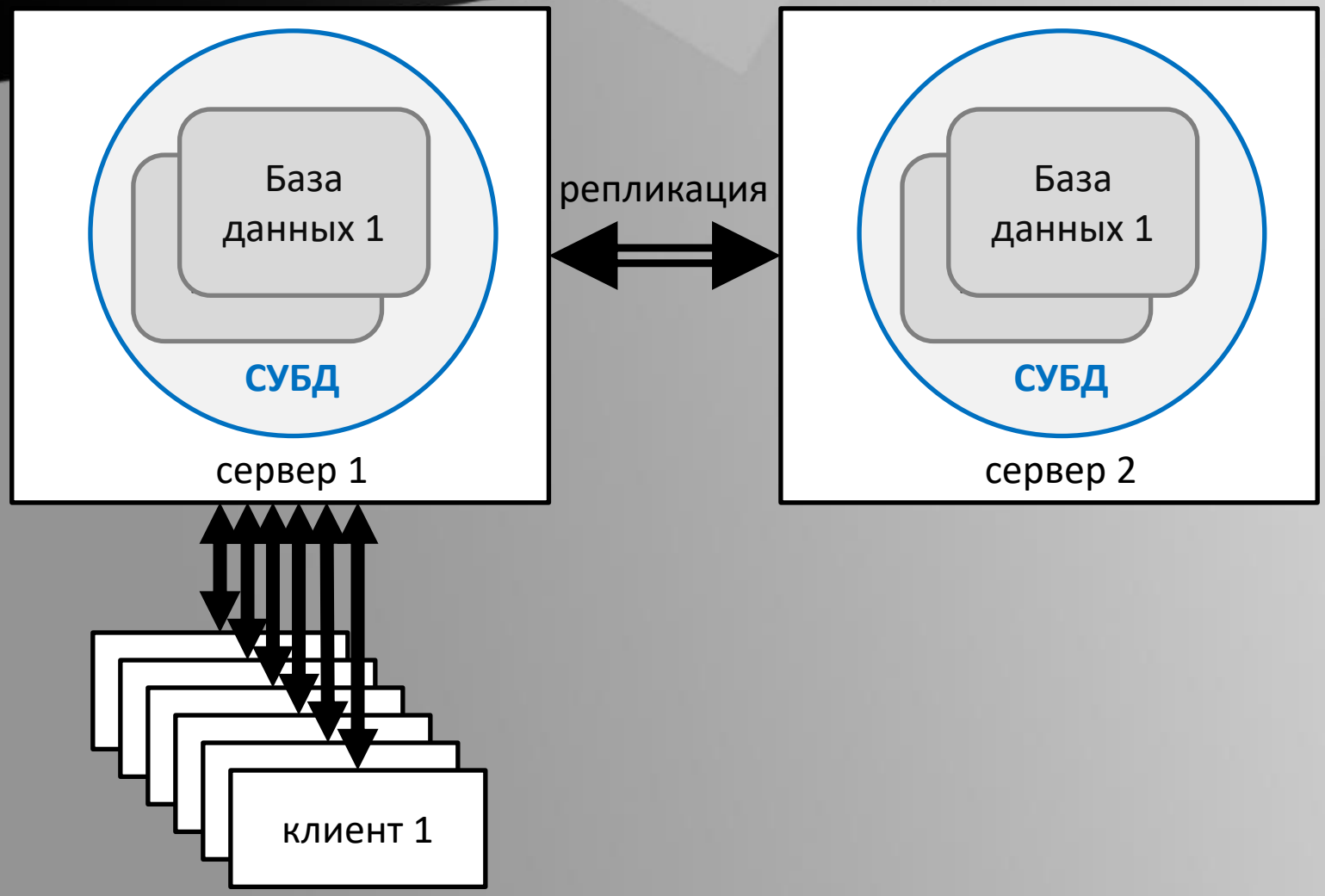
Схемы доступа к данным



3. Схема "клиент-сервер" подразумевает, что помимо хранения базы данных центральный компьютер (сервер базы данных) должен обеспечивать выполнение основного объёма обработки этих данных. Целостность данных обеспечивается программно-аппаратной реализацией сервера. Клиентская часть (Front-End) обычно реализует графический интерфейс и находится на компьютере пользователя; серверная часть (Back-End) обеспечивает управление данными, разделение информации, администрирование и безопасность.

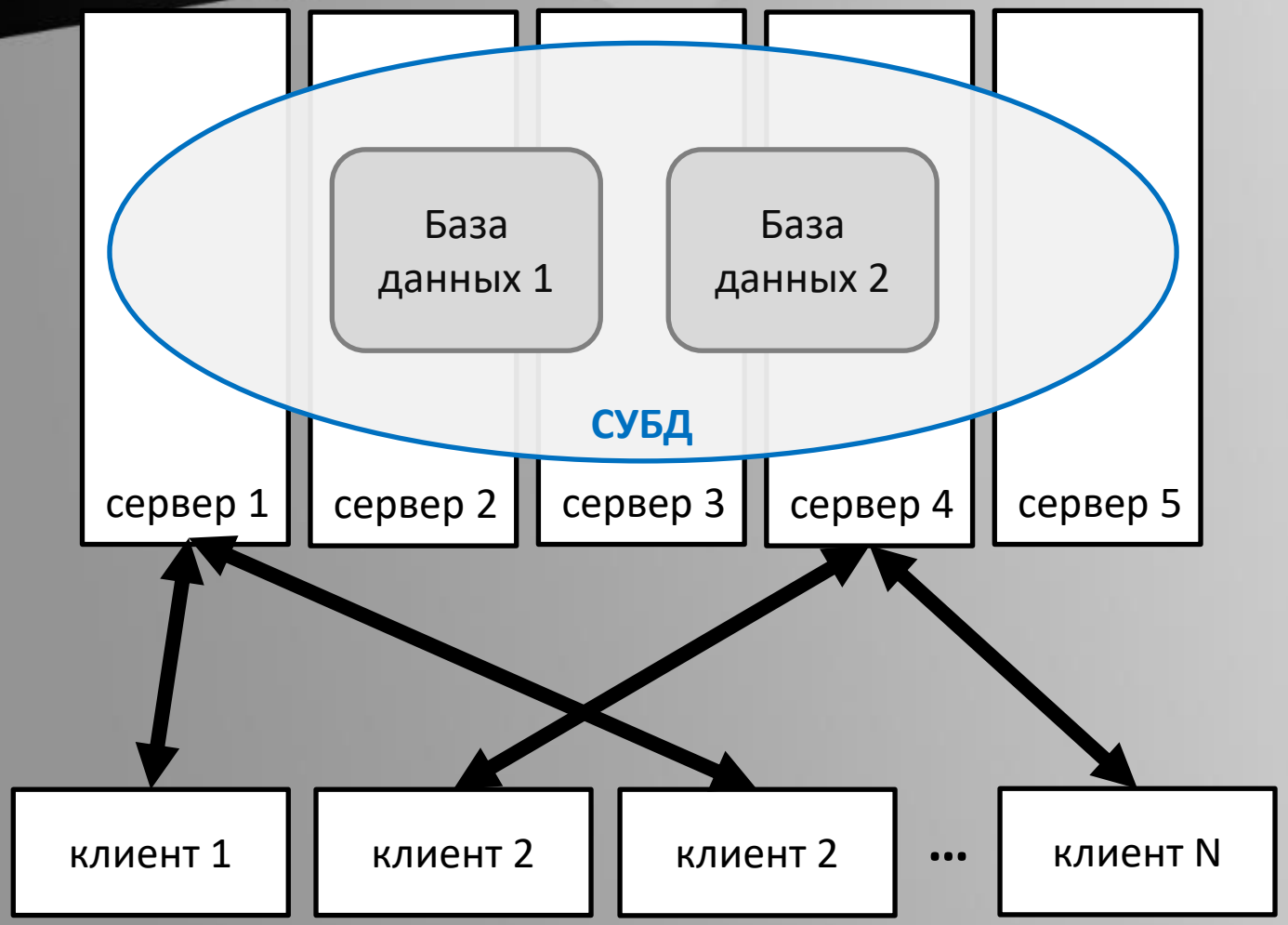
Недостатком схемы является ограниченная надёжность сервера и невозможность работы с большим числом клиентов.

Схемы доступа к данным



4. Распределённая база данных является развитием клиент-серверной схемы и предполагает различные способы репликации и распределения информации между несколькими серверами, начиная от дублирования базы данных на втором удалённом сервере с возможностью аварийного подключения к нему части пользователей до "облачной" базы данных, физически локализованной одновременно на нескольких серверах с распределением аппаратных ресурсов между пользователями.

Схемы доступа к данным



4. Распределённая база данных является развитием клиент-серверной схемы и предполагает различные способы репликации и распределения информации между несколькими серверами, начиная от дублирования базы данных на втором удалённом сервере с возможностью аварийного подключения к нему части пользователей до "облачной" базы данных, физически локализованной одновременно на нескольких серверах с распределением аппаратных ресурсов между пользователями.

Вариант 2

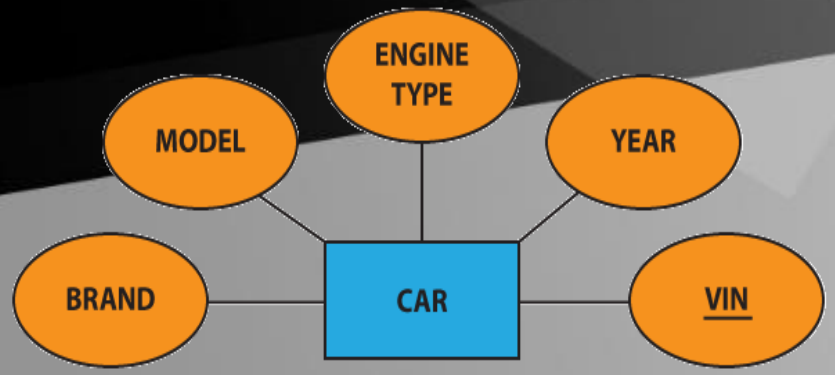
СУБД формирует абстрактное представление данных, скрывая конкретные особенности хранения и управления данными.

Модели данных

Модель данных – это способ представления данных, который позволяет пользователям и разработчикам трактовать совокупность данных как информацию о конкретной предметной области.

- **Инфологическая (внешняя) модель** – обобщённое, не привязанное к какой-либо вычислительной системе или СУБД описание предметной области. Представляет собой легко воспринимаемый человеком способ описания объектов предметной области, их свойств и взаимодействий. *Примеры:* ER-диаграммы, язык UML (Uniform Modelling Language) и др.
- **Концептуальная модель** – способ определения архитектуры базы данных, определяющий структуру объектов и способы установления связей между ними. *Примеры:* иерархическая, сетевая, реляционная, объектно-ориентированная модели данных.
- **Даталогическая (внутренняя) модель** – описание данных на языке конкретной СУБД. *Примеры:* документальные модели, языки Transact SQL, PL/SQL, C# и др.
- **Физическая модель** – низкоуровневые программно-аппаратные средства для хранения данных и обеспечения контролируемого доступа к ним. *Примеры:* логическая структура файлов баз данных, откомпилированные тексты запросов, журналы транзакций и др.

Инфологическая модель данных

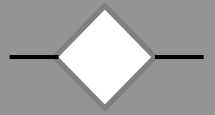


Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления информации, которую предполагается хранить в создаваемой базе данных. Типичным инструментом инфологического моделирования являются диаграммы "сущность-связь" (Entity-Relationship, ER).

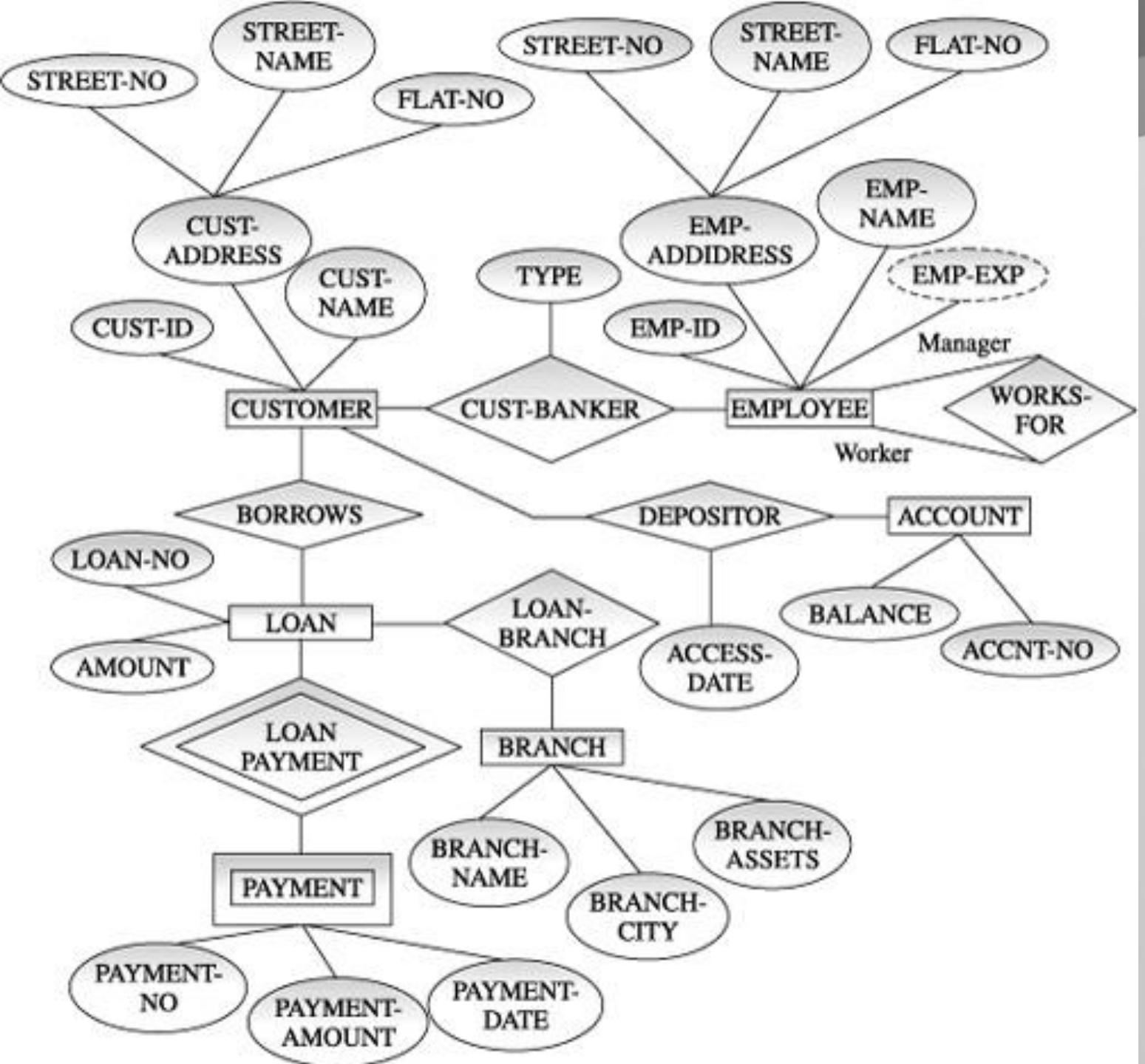
Сущность – уникальный обобщённый объект (класс) предметной области, имеющий существенное значение для рассматриваемой предметной области. Сущность имеет уникальное имя и, как правило, описывает набор однотипных экземпляров, отличающихся друг от друга по значениям своих свойств.

Атрибут сущности – характеристика экземпляра сущности, определяющая свойства конкретного представителя класса.

Ключ сущности – неизбыточный набор атрибутов, однозначно идентифицирующий конкретный экземпляр сущности. Ключ может быть первичным, возможным (потенциальным).

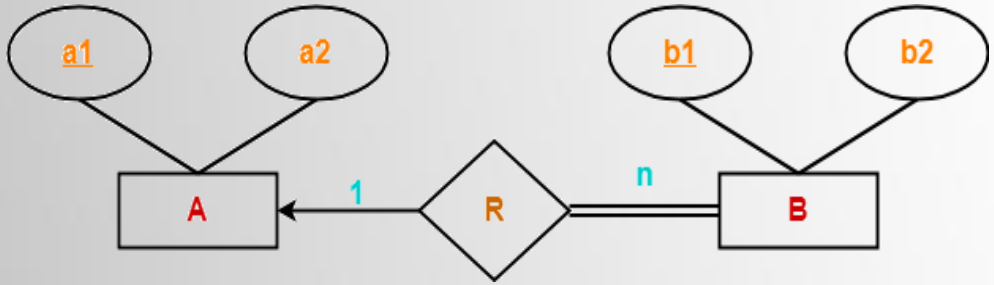


Связь – ассоциация между сущностями, показывающая, каким образом сущности соотносятся или взаимодействуют между собой. Связь может существовать между несколькими сущностями или между сущностью и ею же самой. Связь всегда имеет имя и может иметь атрибуты.



Примеры ER-диаграмм

Общепринятой является нотация **П. Чена**: множества сущностей изображаются в виде прямоугольников, множества связей – в виде ромбов. Атрибуты изображаются в виде овалов и соединяются линией с одной связью или с одной сущностью.



Типы связей между сущностями

Связи делятся на три типа по множественности:

- **один-к-одному** (1:1) – экземпляр одной сущности связан только с одним экземпляром другой сущности. *Пример:* студент и его зачётная книжка.

- **один-ко-многим** (1:N) – один экземпляр сущности может быть связан с несколькими экземплярами другой сущности. *Пример:* писатель и книги, которые он написал.

- **многие-ко-многим** (M:N) – один экземпляр первой сущности связан с несколькими экземплярами второй сущности, и наоборот, один экземпляр второй сущности связан с несколькими экземплярами первой сущности. *Пример*: студенты слушают лекции многих преподавателей, и преподаватели читают лекции многим студентам. Тип «многие ко многим» является временным типом связи, допустимым на ранних этапах разработки инфологической модели. При переходе к даталогической модели она заменяется на две связи (1:N и 1:M) путём создания промежуточной сущности. В нашем случае это может быть сущность с именем «расписание занятий».

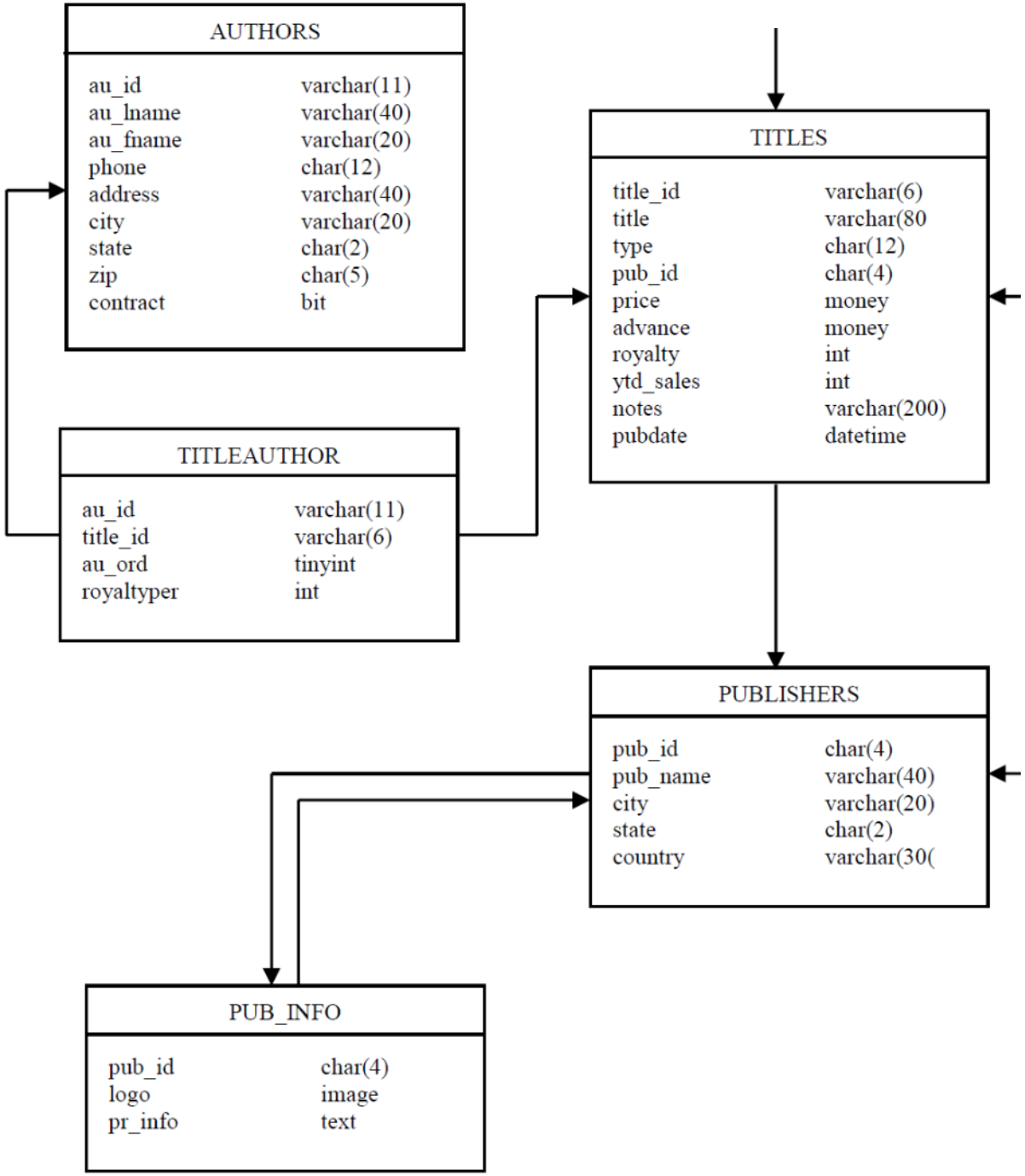
При выборе типа связи следует обращать внимание на то, как была поименована связь (насколько конкретизирована она для данной предметной области).

Типы связей между сущностями

Связи делятся на два типа <u>по модальности</u>:

- **обязательная** – в ней должен участвовать каждый экземпляр сущности. *Пример:*

- **возможная** – допустима ситуация, когда несколько экземпляров (или даже ни один экземпляр) сущности не участвует в связи. *Пример:*



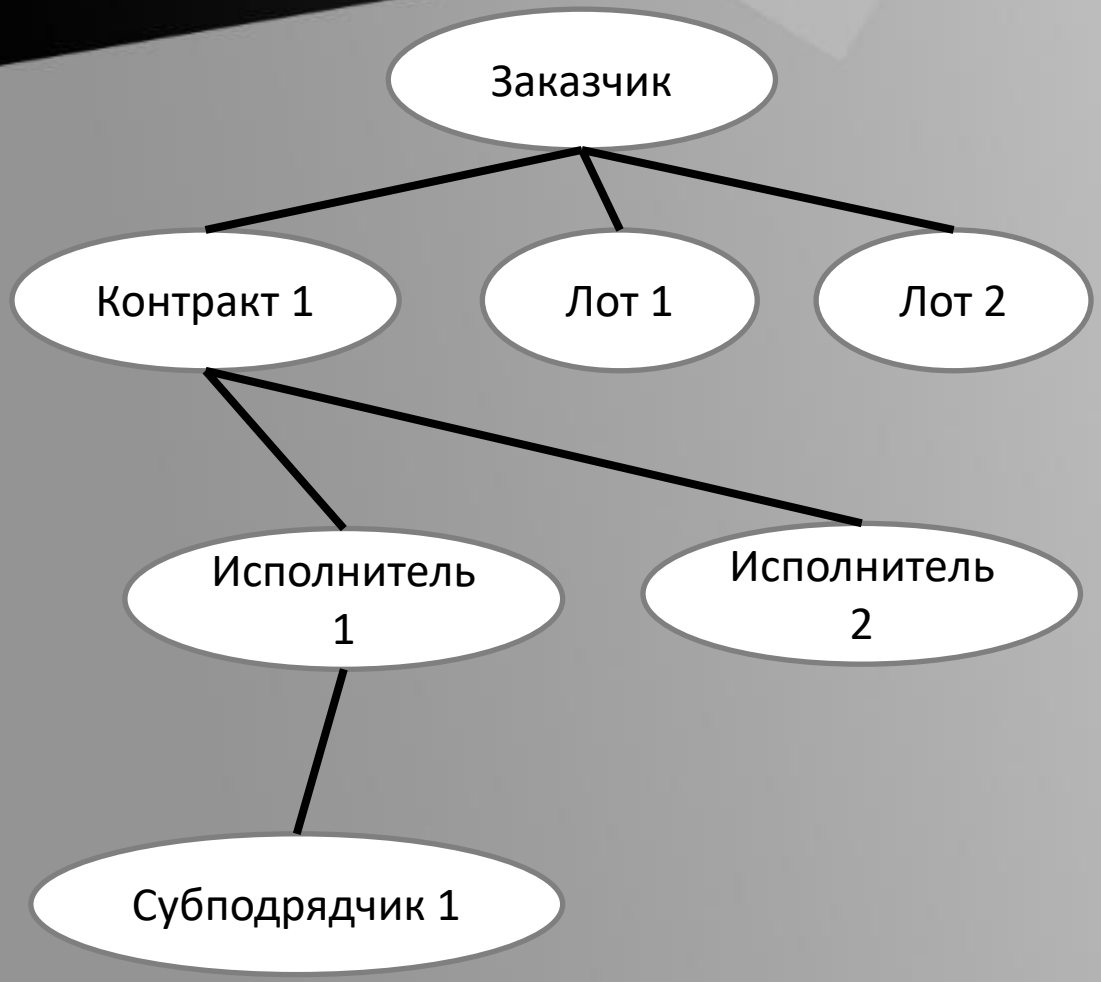
ER-диаграмма БД pubs (фрагмент)

В результате построения модели предметной области в виде набора сущностей и связей получается связный граф. В полученном графе не должно быть **циклических связей** – они выявляют некорректность модели.

В ER-диаграмме допускается принцип **категоризации сущностей**. Это значит, что сущность может быть представлена в виде нескольких подтипов, каждый из которых имеет некоторое подмножество атрибутов и связей, наследуемых из основной сущности.

Концептуальные модели данных

1. Иерархическая модель



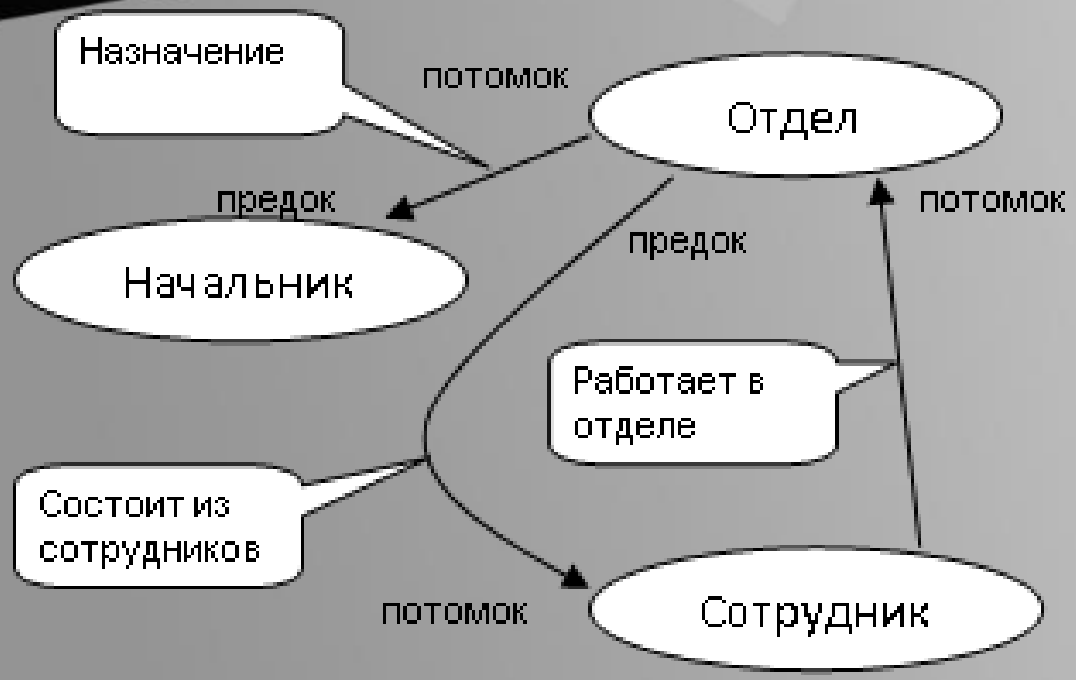
В иерархической модели база данных состоит из набора записей (полей данных), упорядоченных в древовидную (иерархическую) структуру. Основное правило: никакой потомок не может существовать без своего родителя, причём потомок имеет единственного родителя. СУБД автоматически поддерживает целостность ссылок между предками и потомками.

Иерархическая модель очень удобна для восприятия человеком и используется во многих предметных областях (управление, контроль и учёт, систематизация результатов, планирование и т. п.).

Недостатком модели является сложность реорганизации данных и невозможность выполнения «горизонтальных» запросов к данным, не связанных с иерархической структурой.

Концептуальные модели данных

2. Сетевая модель



Сетевая модель является расширением модели иерархической. Минимальной информационной единицей здесь является элемент. Более сложной единицей является агрегат. Агрегат типа "вектор" - линейный набор элементов данных (например "Адрес: город, улица, дом, квартира"). Записью называется совокупность агрегатов или элементов данных. Для любых двух типов записей может быть задано любое количество связей. Связи именуются.

Каждый порождённый элемент (потомок) может иметь более одного порождающего элемента (предка). Сетевая БД может представлять непосредственно все виды связей, присущих данным. По этим данным можно перемещаться, исследовать и запрашивать их всевозможными способами. Однако любой запрос к сетевой БД предполагает выработку собственного механизма навигации по этой базе. При этом автоматически целостность данных СУБД не поддерживается.

Стандарт сетевой модели впервые был определен в 1975 году организацией CODASYL. В качестве языка программирования использовался Cobol.

Концептуальные модели данных

3. Объектно-ориентированная модель

В объектно-ориентированной парадигме предметная область моделируется как множество классов взаимодействующих объектов.

Работа с данными ведется с помощью одного из объектно-ориентированных языков программирования общего назначения, таких как SmallTalk, C++ или Java.

Целостность данных поддерживается на уровне объектов: поля и методы можно объявить как "скрытые", отношения наследования поддерживаются автоматически, однако большая часть алгоритмов управления данными реализуется программным путём.

Структура объектно-ориентированной модели соответствует концепции объектно-ориентированного программирования и базируется на понятиях "класс" и "экземпляр класса". Взаимодействие классов описывается с помощью трёх ключевых понятий:

- **Инкапсуляция** – каждый объект хранит в себе структурированный набор данных (т.е. информацию о предметной области) и набор методов, с помощью которых можно получить доступ к данным этого объекта.
- **Наследование** – подразумевает возможность создавать из классов объектов новые классы объектов, которые наследуют структуру и методы своих предков, добавляя к ним (или исключая) структуру данных и методы, отражающие их собственную индивидуальность.
- **Полиморфизм** – различные объекты в зависимости от внешних событий могут вызывать одинаково названные методы, но по-разному реализованные.

4. Реляционная модель

В прикладном смысле реляционной считается такая база данных, в которой все данные представлены для пользователя в виде "плоских" (двумерных) таблиц, называемых отношениями (relation), и все операции над базой данных сводятся к манипуляциям с этими таблицами. В простейшем случае реляционная модель описывает единственную двумерную таблицу, но чаще всего эта модель описывает структуру и взаимосвязи (relationship) между несколькими таблицами.

В строгом изложении реляционная модель определяется тремя частями, описывающими разные аспекты реляционного подхода: **структурной, манипуляционной и целостной**. В структурной части модели фиксируется, что единственной структурой данных, используемой в реляционных БД, является *нормализованное отношение*. В манипуляционной части модели утверждаются два фундаментальных механизма манипулирования реляционными БД: *реляционная алгебра* и *реляционное исчисление*. В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД: требование *целостности сущностей* и требование *целостности по ссылкам*.

Рейтинг современных СУБД

358 systems in ranking, September 2020

Rank Sep 2020	Rank Aug 2020	Rank Sep 2019	DBMS	Database Model	Score Sep 2020	Score Aug 2020	Score Sep 2019
1.	1.	1.	Oracle	Relational, Multi-model	1369.36	+14.21	+22.71
2.	2.	2.	MySQL	Relational, Multi-model	1264.25	+2.67	-14.83
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1062.76	-13.12	-22.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	542.29	+5.52	+60.04
5.	5.	5.	MongoDB	Document, Multi-model	446.48	+2.92	+36.42
6.	6.	6.	IBM Db2	Relational, Multi-model	161.24	-1.21	-10.32
7.	7.	↑ 8.	Redis	Key-value, Multi-model	151.86	-1.02	+9.95
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	150.50	-1.82	+1.23
9.	9.	↑ 11.	SQLite	Relational	126.68	-0.14	+3.31
10.	↑ 11.	10.	Cassandra	Wide column	119.18	-0.66	-4.22
11.	↓ 10.	↓ 9.	Microsoft Access	Relational	118.45	-1.41	-14.26
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	91.61	+0.69	+5.54
13.	13.	↓ 12.	Splunk	Search engine	87.90	-2.01	+0.89
14.	14.	↑ 15.	Teradata	Relational, Multi-model	76.39	-0.39	-0.57
15.	15.	↓ 14.	Hive	Relational	71.17	-4.12	-11.93

Основы теории реляционных баз данных

Основные определения

- **Домен** (**Domain**) – некоторое конечное множество. Обозначение: D_i , где i – номер домена. Отдельный элемент домена обозначим d_i.
- **Полное декартово произведение** множеств – набор всевозможных сочетаний из n элементов каждое, где каждый элемент берётся из своего домена. Описание: $D_1 \times D_2 \times \ldots \times D_n$.
- **Отношение** (**relation**) R – подмножество декартова произведения множеств $R \subseteq D_1 \times D_2 \times \ldots \times D_n$ ($n \geq 1$), необязательно различных. Число n называется **степенью отношения**.
- **Атрибутом** (**attribute**) называют домен, входящий в отношение. **Степень отношения** определяет количество атрибутов в отношении.
- **Кортежем** (**tuple**) называют декартово произведение элементов множеств $d_1 \times d_2 \times \ldots \times d_n$.
- Атрибуты, значения которых однозначно идентифицируют кортежи, называются **ключевыми.** Отношение может содержать несколько ключей. Всегда один из ключей объявляется **первичным (primary key),** его значения не могут обновляться. Все остальные ключи отношения называются **возможными (потенциальными) ключами**.
- **Схемой отношения** S называется перечень имён атрибутов данного отношения с указанием домена, к которому они относятся. Описание: $S_R = (A_1, A_2, \ldots, A_n)$, $A_i \subseteq D_i$.
- Набор именованных схем отношений представляет собой **схему базы данных**.

Основы теории реляционных баз данных

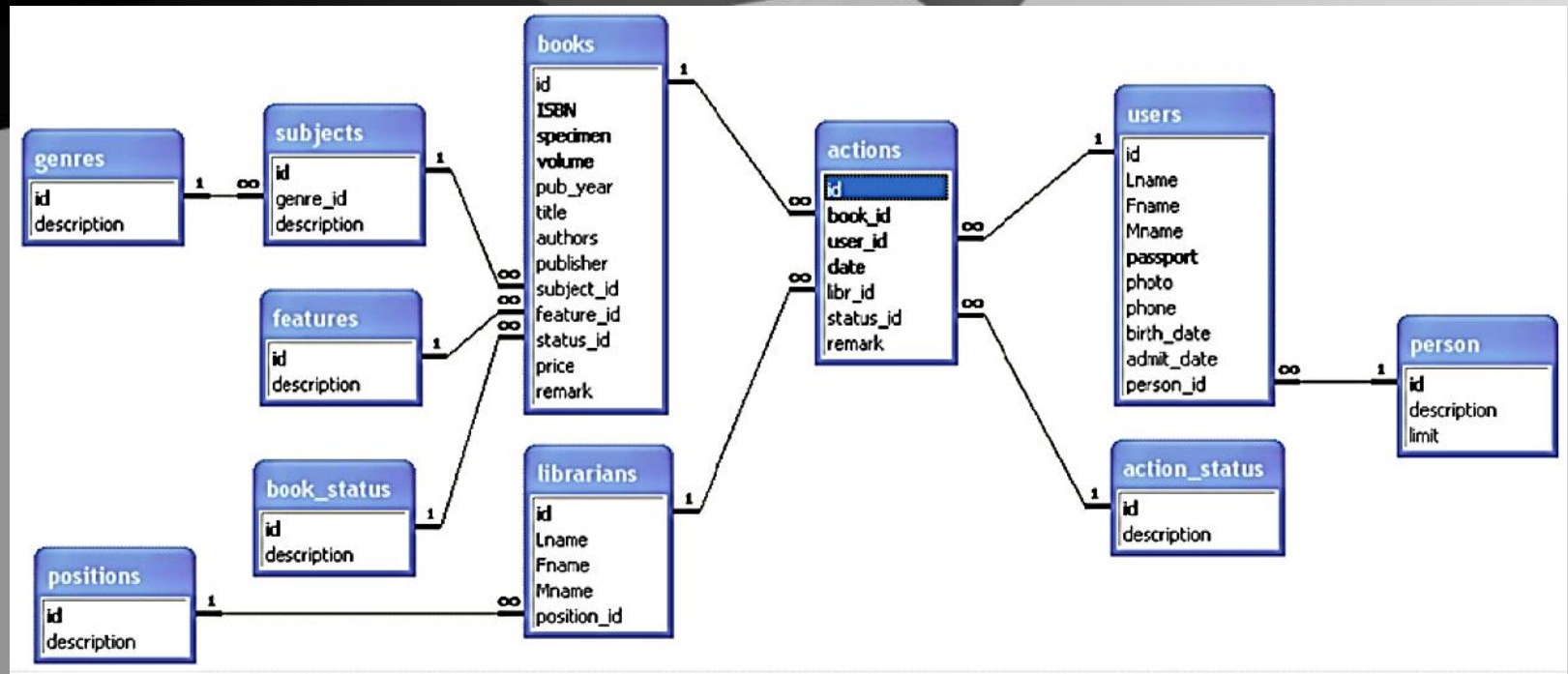


Таблица упрощённых определений:

Инфологическая модель	Реляционная модель	Терминология СУБД
Сущность	Отношение	Таблица
Атрибут	Атрибут	Поле (имя столбца)
Экземпляр сущности	Кортеж	Запись (строка таблицы)

Основные свойства реляционных баз данных

- **В отношении нет одинаковых кортежей** (все строки таблицы различаются по своему содержанию). Следствие: наличие у каждого кортежа первичного ключа.
- **Отсутствует упорядоченность кортежей** (таблица с переставленными строками остаётся той же самой таблицей).
- **Отсутствует упорядоченность атрибутов** (таблица с переставленными столбцами остаётся той же самой таблицей).
- **Все значения атрибутов отношения атомарные** (это свойство в настоящее время можно трактовать как строгую типизацию данных в ячейках таблицы, так что способ обработки данных в ячейках определяется типом этих данных).

Реляционная алгебра

Алгеброй называется множество объектов с заданной на нём совокупностью операций, замкнутых относительного этого множества.

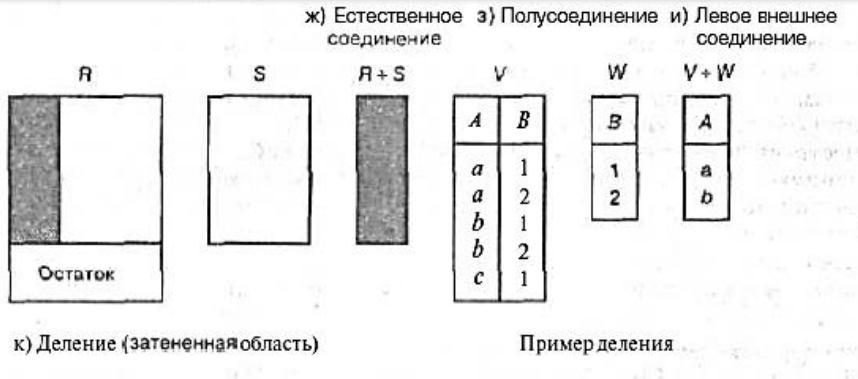
Теоретико-множественные операции:

1. **Объединение отношений.** $R_3 = R_1 \cup R_2$.
2. **Пересечение отношений**. $R_3 = R_1 \cap R_2$.
3. **Разность отношений**. $R_3 = R_1 - R_2$.
4. **Прямое (расширенное декартово) произведение отношений**. $R_3 = R_1 \otimes R_2$.

Специальные реляционные операции:

5. **Ограничение отношения (горизонтальная фильтрация или выборка)**. $R[\alpha(r)]$.
6. **Проекция отношения (вертикальная фильтрация)**. $R[B]$.
7. **Соединение отношений (соединение по условию)**. $R_3 = R_1 [\beta] R_2$.
8. **Деление отношений**. $R_3 = R_1[A:B]R_2$

Результатом операции деления отношений является набор кортежей отношения R_1, у которых имеется полный набор значений атрибутов, перечисленных в отношении R_2. Для этого нужно, чтобы в отношении R_2 была часть атрибутов (можно и один), которые есть в отношении R_1. В результирующем отношении R_3 показываются только те атрибуты из отношения R_1, которых нет в отношении R_2.



Реляционная алгебра

Примеры операций

Предметная область: экзамен по курсу «Базы данных», проводился два раза во время сессии (второй раз – для должников). Пусть имеются три отношения, имеющие следующие эквивалентные схемы:
R1 = (Номер зачетки, ФИО, Группа) – список студентов, пришедших на экзамен №1;
R2 = (Номер зачетки, ФИО, Группа) – список студентов, пришедших на экзамен №2;
R3 = (Номер зачетки, ФИО, Группа) – список студентов, успешно сдавших экзамен в сессию;

Ударим реляционной алгеброй по следующим вопросам:

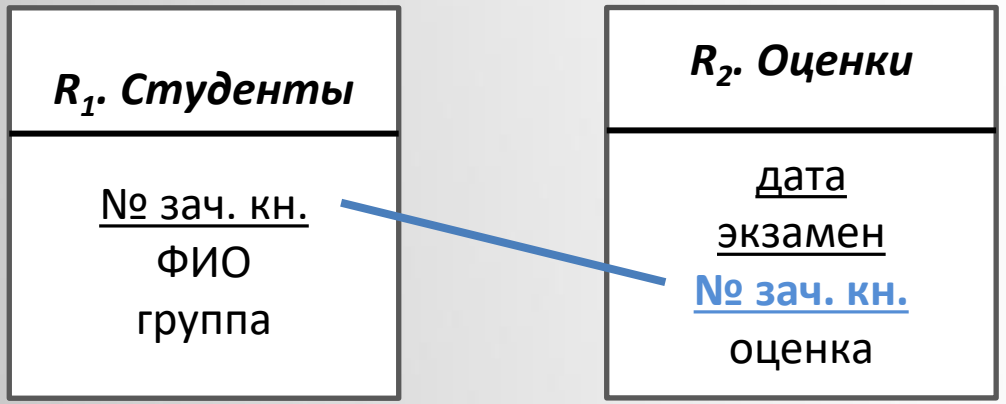
а) Какие студенты сдавали два раза, но так и не сдали экзамен?
Ответ: **R = R1 U R2 – R3**
б) Какие студенты сдавали экзамен только один раз, и сдали его?
Ответ: R = (R1 – R2 ∩ R3) U (R2 – R1 ∩ R3)
в) Какие студенты смогли сдать экзамен только со второго раза?
Ответ: R = R1 ∩ R2 ∩ R3
г) Какие студенты приходили только один раз, не сдали, и больше не появлялись?
Ответ: R = (R1 – R2) U (R2 – R1) – R3

Ключи отношений

Возможный (потенциальный) ключ отношения – набор атрибутов, однозначно определяющий кортеж отношения, причём при удалении любого атрибута из этого набора свойство однозначной идентификации кортежа теряется.

В общем случае в отношении может быть несколько возможных ключей. Среди них выбирают один (обычно самый удобный), который считается главным, и который называют **первичным ключом отношения**.

Подмножество атрибутов **A** отношения R_2 называется **внешним ключом**, если:
1. Существует отношение R_1 (R_1 и R_2 не обязательно различны) с потенциальным ключом **B**.
2. Каждое значение **A** в отношении R_2 всегда соответствует значению **B** для R_2, либо является **NULL**-значением.



Родительское отношение, *referenced*

Дочернее отношение, *referencing*

- Количество атрибутов внешнего ключа должно соответствовать количеству атрибутов потенциального ключа.
- Для внешнего ключа **НЕ** требуется, чтобы он был уникальным или был компонентом какого-либо потенциального ключа в своём отношении.
- Для атрибутов внешнего ключа разрешается иметь значение **NULL**.

Целостность данных
в реляционных базах данных

Под **целостностью** реляционной БД понимается соответствие информационной модели предметной области, хранимой в БД, самой предметной области для любого пользователя БД при любом запросе к данным.

Правила обеспечения целостности данных:

Структурная целостность: допускается работа только с однородными структурами данных типа «реляционное отношение».

Целостность объектов: первичный ключ отношения не должен содержать значений NULL.

Ссылочная целостность: база данных не должна содержать значений внешних ключей, для которых не существует соответствующих значений потенциальных ключей. Исключение – NULL.

Поддерживается следующие принципы взаимосвязи между экземплярами кортежей взаимосвязанных отношений:

- кортежи дочернего отношения *уничтожаются* при удалении кортежа родительского отношения, связанного с ними внешним ключом.
- кортежи дочернего отношения *модифицируются* при удалении кортежа родительского отношения, связанного с ними (ставится значение NULL).
- кортежи дочернего отношения *модифицируются* при изменении значений атрибутов родительской таблицы, на которые ссылается внешний ключ.

Все реляционные СУБД обязаны автоматически поддерживать целостность данных. Ссылочная целостность поддерживается реализацией механизма **каскадирования** (автоматическое изменение или удаление значений внешнего ключа).

Классическая технология проектирования реляционных БД связана с теорией нормализации, основанной на анализе функциональных зависимостей между атрибутами отношений.

Нормализация отношений

в реляционных базах данных

Функциональной зависимостью (functional dependency) набора атрибутов **B** отношения **R** от набора атрибутов **A** того же отношения называется такое соотношения проекций* **R[A]** и **R[B]**, при котором в каждый момент времени любому элементу проекции **R[A]** соответствует **только один** элемент проекции **R[B]**, входящий вместе с ним в какой-либо кортеж отношения **R**. Обозначение: **R.A** $\rightarrow$ **R.B** (детерминант).

* Проекция – это копия отношения, в которую не включены один или несколько атрибутов исходного отношения.

Взаимно-независимые – атрибуты, которые не зависят функционально друг от друга.

R

B	A	C
осень	сентябрь	1
осень	октябрь	31
осень	сентябрь	12
зима	декабрь	25
зима	декабрь	3
зима	январь	7

Нормализация – это процесс проектирования схемы базы данных с использованием декомпозиции.

Нормальные формы

1 NF **Отношение находится в первой нормальной форме тогда и только тогда, когда значения всех его атрибутов атомарны.**

2 NF **Отношение находится во второй нормальной форме тогда и только тогда, когда оно находится в первой нормальной форме и не содержит неполных* функциональных зависимостей непервичных атрибутов от атрибутов первичного ключа.**

Пример отношения, которое НЕ находится во 2 NF:

Сессия
№ зач. кн. ФИО группа дисциплина оценка

* Функциональная зависимость **R.A** → **R.B** называется **полной**, если набор атрибутов **B** отношения **R** функционально зависит от **A**, но не зависит функционально от любого подмножества **A**. В противном случае функциональная зависимость называется неполной.

Аномалии обновления:
- в результате ошибки ввода данных студенту по результатам одного или нескольких экзаменов приписали не ту группу.
- если студент не сдал ни одного экзамена, то он не существует

Нормальные формы

3 NF **Отношение находится в третьей нормальной форме тогда и только тогда, когда оно находится во второй нормальной форме и не содержит транзитивных* зависимостей.**

* Если для атрибутов **A**, **B** и **C** некоторого отношения существуют зависимости вида **A** → **B** и **B** → **C**, это означает, что атрибут **C** ***транзитивно*** зависит от атрибута **A** через атрибут **B**, при условии, что атрибут **A** функционально не зависит ни от атрибута **B**, ни от атрибута **C**.

BCNF **Отношение находится в нормальной форме Бойса-Кодда, если оно находится в третьей нормальной форме и каждый детерминант отношения является возможным ключом этого отношения.**

Нормальная форма Бойса-Кодда является более строгой версией формы 3NF, поскольку каждое отношение BCNF является также отношением 3NF, но не всякое отношение 3NF является отношением BCNF.

Пример отношения, которое НЕ находится в 3 NF:

Отдел
ID сотрудника № отдела № телефона

Между телефоном сотрудника и номером отдела есть неучтённая функциональная зависимость.

Нормальные формы

Пример отношения, которое находится в 3NF, но НЕ находится в BCNF:

Модель предметной области

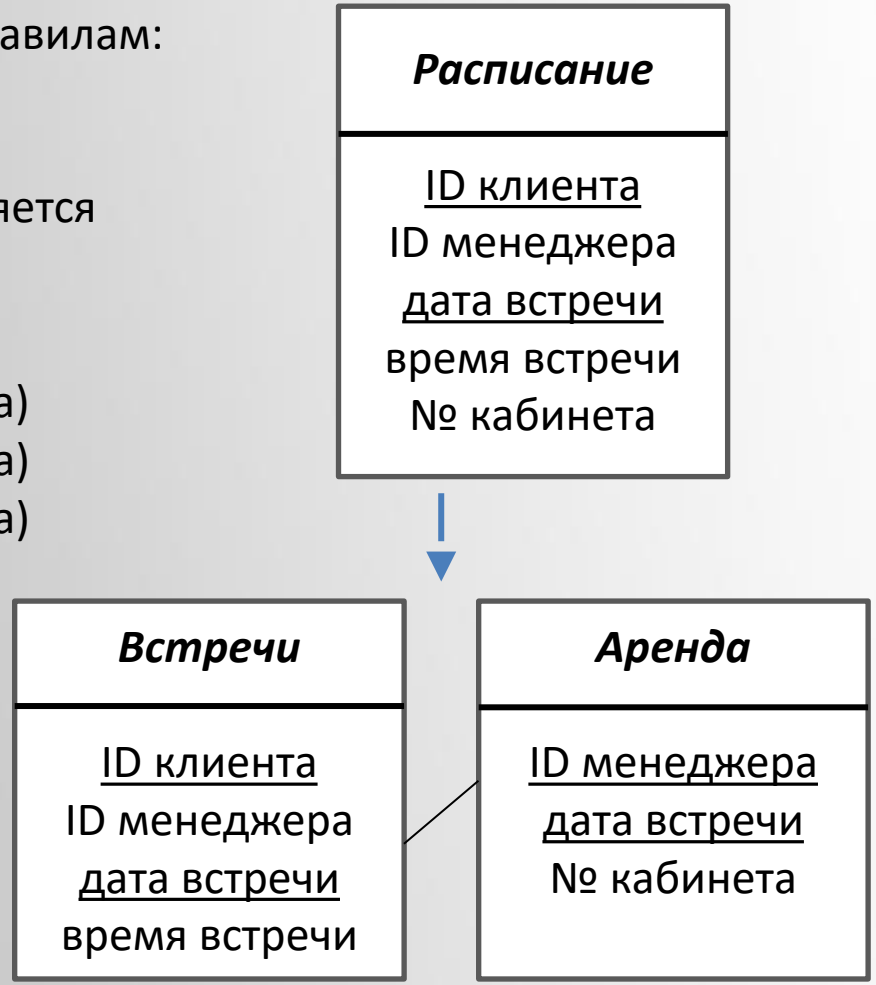
индивидуальное собеседование менеджера с клиентом по следующим правилам:

- у одного клиента не более одного собеседования в один день
- одновременно с одним менеджером общается только один клиент
- клиент общается с менеджером в отдельном кабинете, который выделяется менеджеру на весь рабочий день

Примеры функциональных зависимостей

- (ID клиента, дата встречи) → (ID менеджера, время встречи, № кабинета)
- (ID менеджера, дата встречи, время встречи) → (ID клиента, № кабинета)
- (№ кабинета, дата встречи, время встречи) → (ID клиента, ID менеджера)
- **(ID менеджера, дата встречи) → (№ кабинета)**

последняя функциональная зависимость нарушает требование BCNF, следовательно, нужна декомпозиция отношения на два других:



В большинстве практических случаев достижение 3NF либо BCNF является достаточным условием реализации проектов баз данных

Нормальные формы

4 NF Отношение находится в четвёртой нормальной форме в том и только в том случае, если существует многозначная* функциональная зависимость A –>> B и все остальные атрибуты отношения R функционально зависят от A.

В отношении **R(A,B,C)** существует многозначная зависимость **R.A –>> R.B** в том и только в том случае, если множество значений **B**, соответствующее паре значений атрибутов **A** и **C**, зависит только от **A** и не зависит от **C**.

Результат приведения исходного отношения в 4 NF:

Лаборатория: сотрудники
эксп. установка сотрудник

Лаборатория: студенты
эксп. установка студент

Пример отношения, которое НЕ находится в 4 NF:

Лаборатория
эксп. установка сотрудник студент

При этом между сотрудником и студентом нет какой-либо функциональной зависимости.

Аномалии обновления данных:

- невозможно приписать сотрудника к экспериментальной установке без указания студента
- удаление студента либо сотрудника невозможно

Нормальные формы

5 NF **Отношение находится в пятой нормальной форме (нормальной форме *проекции-соединения*) в том и только в том случае, когда любая нетривиальная зависимость соединения* в R определяется потенциальным ключом этого отношения.**

* Отношение **R (A, B, … Z)** удовлетворяет **зависимости соединения (A, B, … Z)** в том и только в том случае, когда **R** восстанавливается без потерь путём соединения своих проекций на **A, B, … Z** (наборы атрибутов отношения **R**).

Пример отношения, которое НЕ находится в 5 NF:

Продажи
магазин фирма товар

Результат приведения исходного отношения в 5 NF:

R_1
магазин товар

R_2
магазин фирма

R_3
фирма товар

Особенности предметной области:

1. Магазин имеет право торговать только определёнными товарами
2. Магазин торгует только товарами определённых фирм
3. Фирма производит ограниченную номенклатуру товаров

Недостатки реляционной модели данных

- Реляционная модель данных не допускает естественного представления данных со сложной (иерархической) структурой, поскольку в ее рамках возможно моделирование лишь с помощью плоских отношений (таблиц). Все отношения принадлежат одному уровню, многие значимые связи между данными либо теряются, либо их поддержку приходится осуществлять в рамках конкретной прикладной программы.
- По определению в реляционной модели поля кортежа могут содержать лишь атомарные значения. Однако, в приложениях САПР (ГИС) и системах искусственного интеллекта должны проводиться операции со сложно-структурированными объектами (например, одинакового типа, но переменного размера).
- Слишком высокий уровень деструктуризации (раздробления) данных по многочисленным таблицам уменьшает скорость выполнения запросов и, одновременно, увеличивает их сложность, что приводит к совершению дополнительных ошибок при написании запросов.
- Нельзя гарантированно избежать такой ситуации, когда пользователь вводит данные, формально удовлетворяющие ограничениям целостности, но не соответствующие реальному состоянию предметной области. В этом случае фактически непротиворечивость данных будет нарушена.

Спасибо за внимание!