Лекция 1

Основные понятия

Модели данных

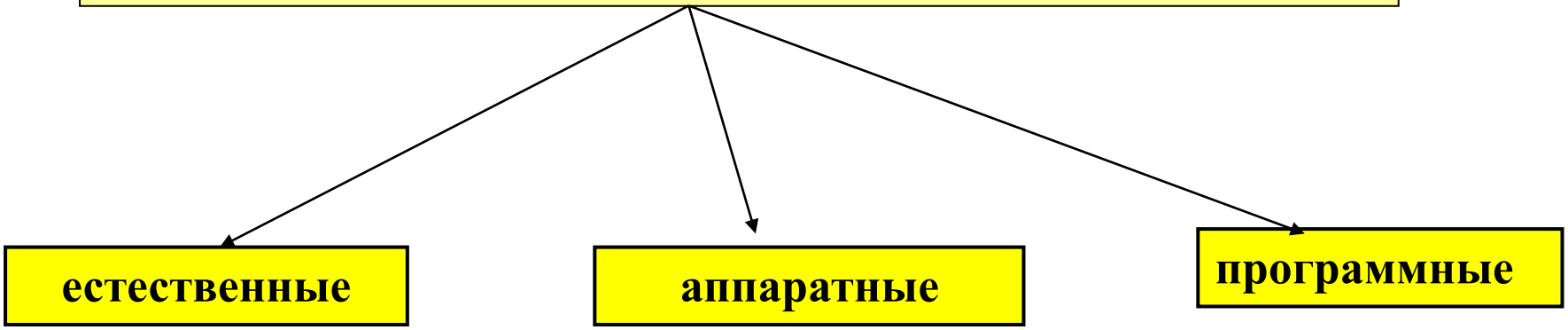
Реляционная модель: нормальные формы

Проектирование БД

Данные и методы

Данные **в информатике - это результат фиксации событий и явлений реального мира в знаковой форме (зарегистрированные сигналы)**

Методы **воспроизведения и обработки данных**



Понятие информации в информатике

Информация **- это продукт динамического взаимодействия объективных данных и субъективных методов (естественных, аппаратных, программных и других), рассмотренный в *контексте* этого взаимодействия**

Данные + Методы = Информация

Контекстным считается тот метод, который является общепринятым для работы с данными определенного типа

в законе РФ «Об информации, информатизации и защите информации» (от 20.02.95 № 24-ФЗ) дается следующее определение термина «информация» – ***сведения о лицах, предметах, фактах, событиях и процессах независимо от способа их представления***.

Знание являются результатом *восприятия* и *понимания* **информации** в контексте опыта при систематическом изучении.

Классификация информации

БД, БнД, СУБД - определения

Определения из общеотраслевых руководящих материалов Государственного комитета по науке и технике (ГКНТ), 1982 г.

База данных (БД) - именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Банк данных (БнД) - система специальным образом организованных данных - баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для централизованного накопления и коллективного многоцелевого использования данных.

Система управления базами данных (СУБД) - совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Программы, с помощью которых пользователи работают с базой данных, называются ***приложениями***.

СУБД обеспечивает одновременное корректное выполнение независящих друг от друга приложений при работе с единой БД. Каждое приложение должно учитывать изменения в БД, вносимые другими приложениями.

Что такое информационная система?

Автоматизированная информационная система (АИС) - программно-аппаратный комплекс для хранения и обработки информации с целью обеспечения запросов пользователей

Состав информационной системы

- Данные
- Лингвистическое обеспечение
- Программное обеспечение
- Кадровое обеспечение
- Техническое обеспечение
- Организационное обеспечение

Два класса АИС

- Документальные
- Фактографические

Используются разные алгоритмы обработки и способы хранения данных

Данные в фактографической АИС

Фактографические информационные системы ориентированы на полное и точное представление данных достаточно простой смысловой структуры.

Любая АИС оперирует данными в пределах своей ***предметной области***.

Предметная область представляется как некоторая совокупность выделенных ***объектов*** и ***явлений*** части реального мира и ***связей*** между ними.

Объект может быть:
реальным - человек, предмет, отдел;
абстрактным - счет покупателя, назначение врача.

Каждый объект характеризуется некоторым набором ***свойств*** и ***признаков***, имеющих определенные значения.

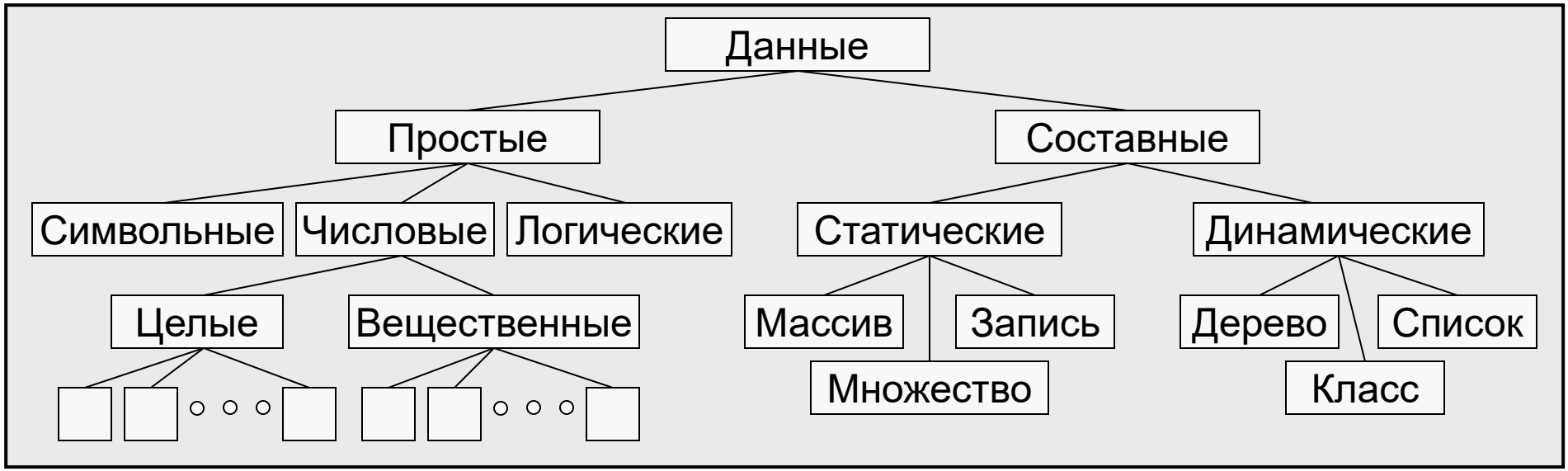
Объекты, обладающие одинаковым набором свойств, относят к одному ***классу объектов***.

Типы данных

Для представления в адресном пространстве памяти компьютера ***все данные***, характеризующие объекты реального мира, подразделяются ***на несколько типов***.

Тип данного определяет ***способы*** **:**

- кодирования данных при размещении в памяти;
- обработки данных при выборке из памяти;
- выполнения операций над данными.



- ***простые*** - форма представления определяется архитектурой ЭВМ,
- ***составные*** - конструируемые пользователем для решения конкретной задачи

Сущности, их атрибуты и связи

В фактографических информационных системах применяются два специальных понятия:
сущность - для обозначения объектов и явлений любого вида,
атрибут - для обозначения свойств сущности.

Сущности - пациент, направление врача, расписание работы отделения, врач, ...
Атрибуты - фамилия, год рождения, время приема врача, зарплата сотрудника, номер кабинета, диагноз пациента, наименование подразделения, ...

Связи между сущностями описываются с помощью дополнительных атрибутов, устанавливаемых для сущностей.

Связи между сущностями обеспечивают возможность отыскания одних сущностей по значениям других.

Подобное представление данных лежит в основе логической модели описания предметной области «***Сущность-связь***»

Сущность - определение

Сущность - любой различимый объект (объект, который можно отличить от другого), информацию о котором необходимо хранить в базе данных.

Тип сущности определяет набор однородных сущностей: личностей, предметов, событий, идей, выступающих как целое.

Экземпляр сущности - определяет конкретную вещь в наборе. Например, для типа сущности **Город** его экземпляры – **Москв**а, **Киев**.

Атрибут, ключ - определения

Атрибут **-** поименованная характеристика сущности.

Наименования атрибутов уникальны для конкретного типа сущности, но могут быть одинаковыми для сущностей разных типов. Например, атрибут **Цвет** может быть определен для многих сущностей: **Стул**, **Автомобиль**, **Здание**.

Атрибуты используются для характеристики сущностей. Например, атрибутами сущности **Автомобиль** могут быть **Тип**, **Марка**, **Номер**, **Цвет**

Каждому экземпляру сущности может быть присвоено ***только одно значение каждого атрибута***.

Наименования атрибутов и сущностей условны в пределах предметной области. Для автомобильного завода **Цвет** – атрибут продукта производства. Для лакокрасочной фабрики **Цвет** – тип сущности.

Ключ **-** это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.

Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся атрибутам.

Связь - определение

Связь - ассоциирование двух или более типов сущностей.

Связи между сущностями устанавливаются для выполнения одного из основных требований к организации данных – обеспечения возможности отыскания одних сущностей по значениям других.

Сложные базы данных могут содержать тысячи типов сущностей. Теоретически между ними может быть установлено более миллиона связей.

Для выполнения только функции хранения отдельных, не связанных между собой данных, структура данных могла бы быть очень простой.

Наличие множества связей между сущностями и определяет сложность представления информационной модели предметной области.

БД в фактографических АИС. СУБД

Отражение предметной области
в виде структурированной совокупности данных,
характеризующих состав объектов, их свойства и взаимосвязи,
принято называть ***базой данных*** (БД). Данные и их описание.

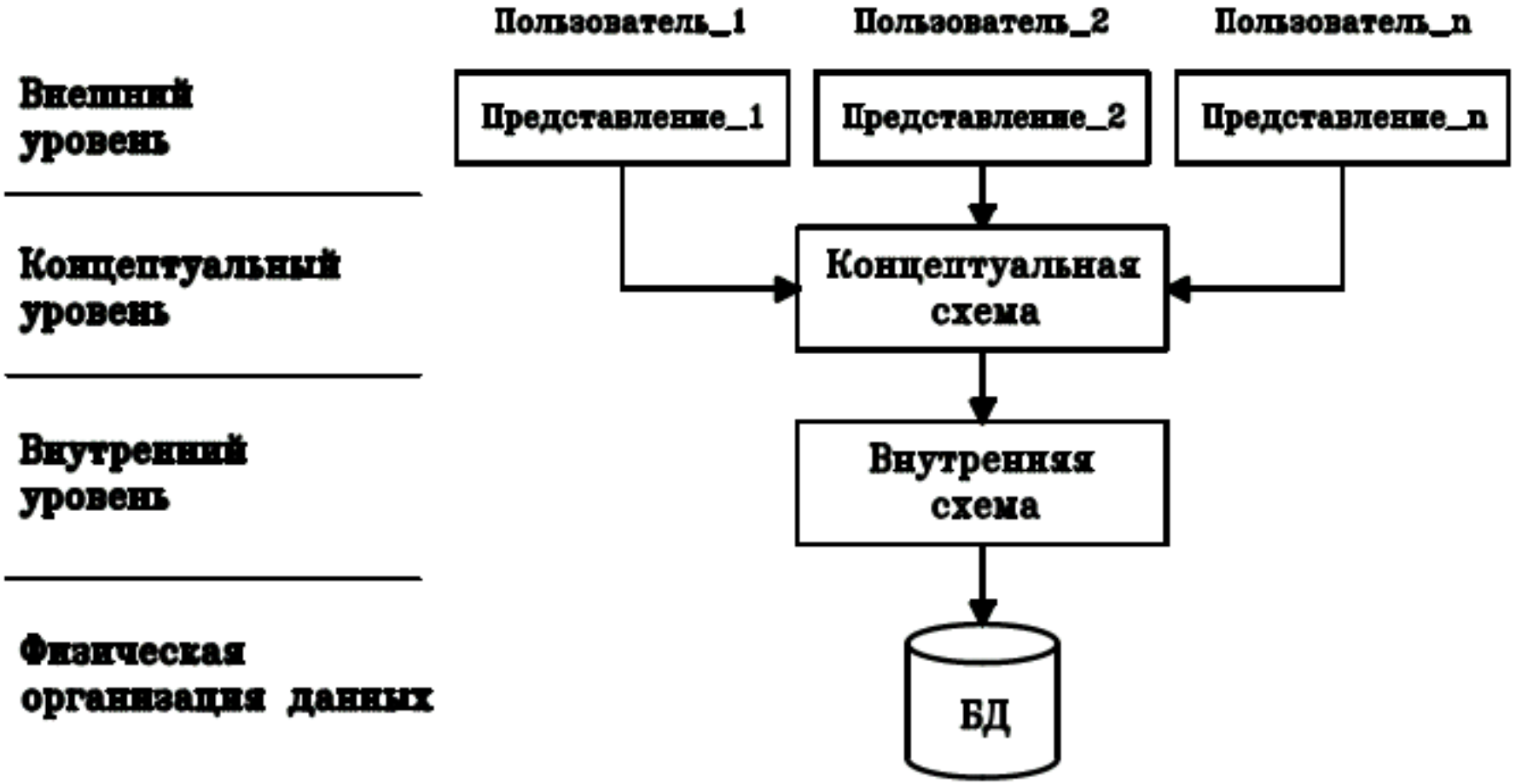
Для реализации баз данных в электронном виде и доступа к данным с помощью компьютеров существует специальный программный инструментарий - ***системы управления базами данных*** (СУБД).

СУБД обеспечивает:

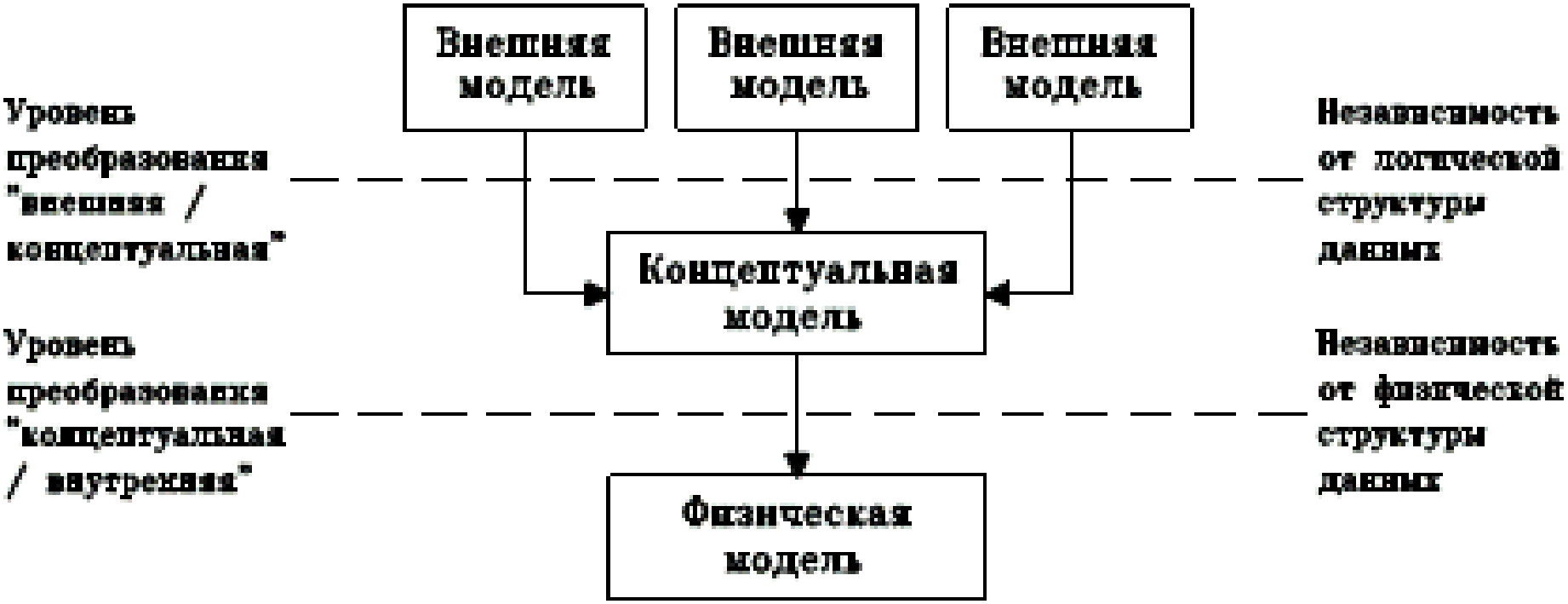
1. Создание базы данных (описание структуры, описание связей, данные)
2. Наполнение и обновление данных для поддержания БД в актуальном состоянии (анализ на входе)
3. [illegible]
4. [illegible]
5. Анализ данных (эффективный первичный анализ)
6. Защиту данных от несанкционированного доступа (пароли, шифрование)

Современные СУБД - достаточно совершенные инструменты, успешно применяющиеся в самых различных областях деятельности.

Трехуровневая архитектура ANSI-SPARC

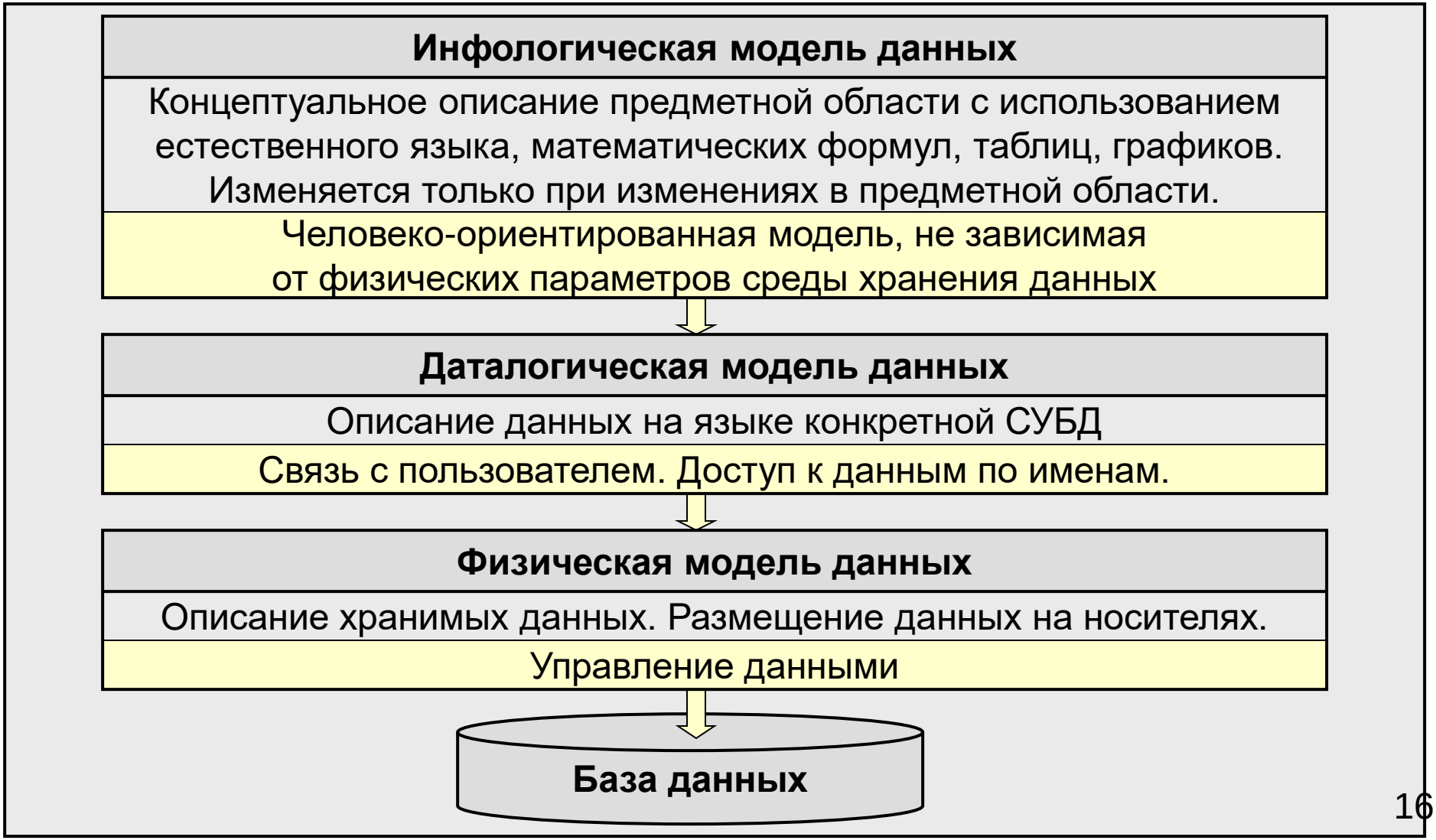


Реализация независимости от данных в трехуровневой архитектуре



Архитектура СУБД

В основе функционирования СУБД лежит трехуровневая архитектура ***описания*** данных, их структуры и связей.



Инфологическая модель «сущность-связь»

На инфологическом уровне описание данных наиболее удобно представляется моделью «***Сущность-связь***»**(ER-модель)**

Модель «Сущность-связь» с помощью логического представления данных определяет их возможные значения в контексте взаимосвязи с другими данными

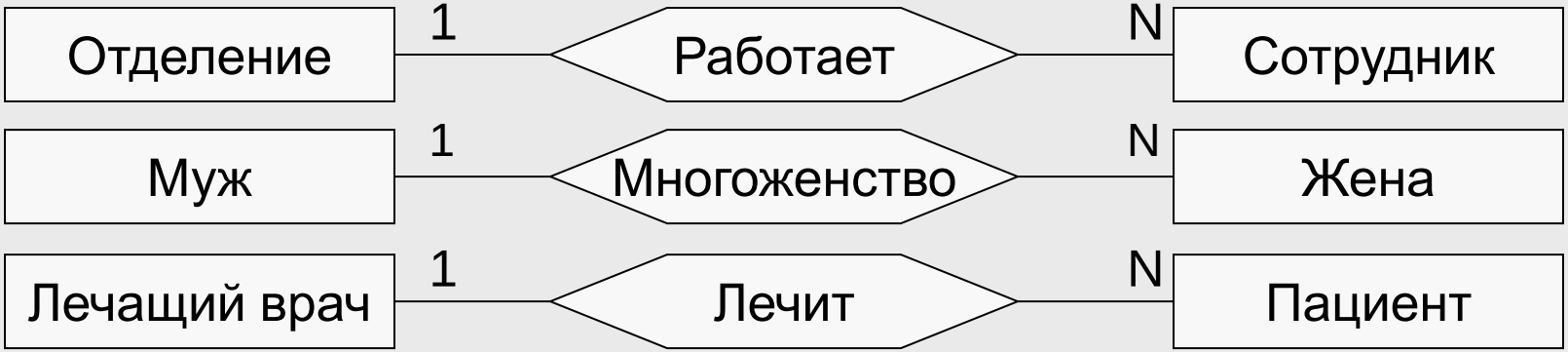
Между двумя сущностям, например A и B, возможны четыре вида связей

1. Связь «один к одному» (1:1) Каждому экземпляру сущности A соответствует 1 или 0 экземпляров сущности B

Типы связей

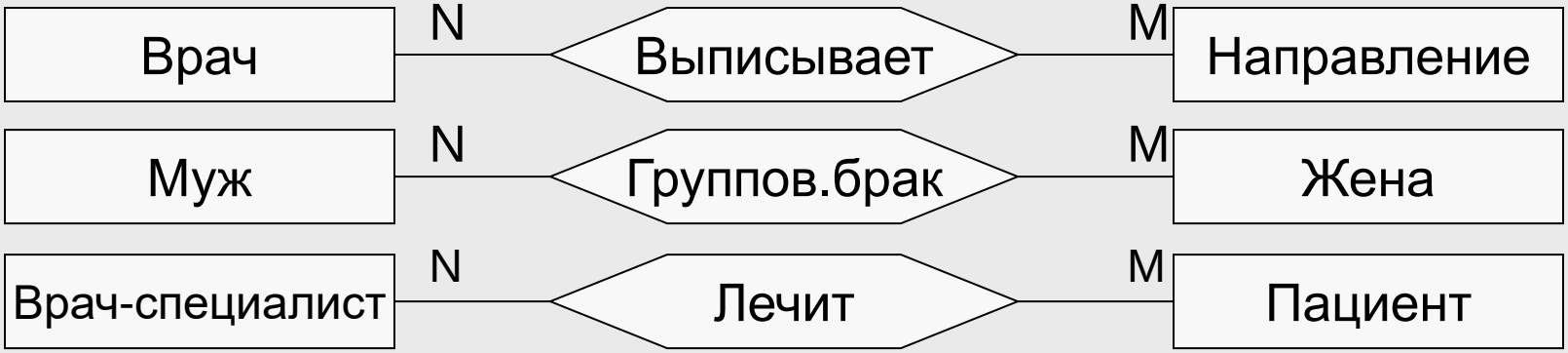
2. Связь «один ко многим» (1:N). 3. Связь «многие к одному» (N:1)

Одному экземпляру сущности A
соответствует 0, 1 или несколько экземпляров сущности B



4. Связь «многие ко многим» (N:M)

Каждому экземпляру сущности A
соответствует 0, 1 или несколько экземпляров сущности B и наоборот



Даталогические модели данных

Даталогические модели данных описывают связи между сущностями, задавая правила ведения и выборки данных.

Любая модель должна содержать три компоненты:

- ***структуру данных***, описывающую точку зрения пользователя на представление данных предметной области;

- ***набор допустимых операций***, выполняемых на структуре данных, язык определения данных (ЯОД), язык манипулирования данными (ЯМД).

- ***ограничения целостности*** - механизм защиты базы данных от неверных изменений или разрушений (обычно в связях).

Принято рассматривать три (четыре, пять) модели данных:

- ***иерархическую*** модель;

- ***сетевую*** модель;

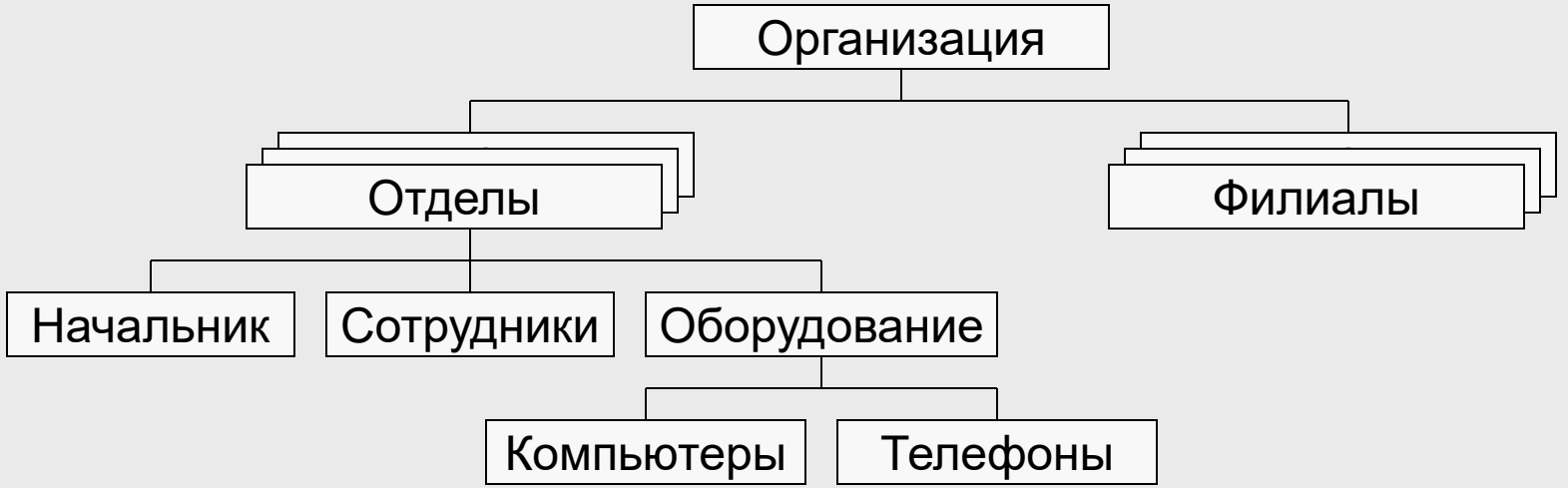
- ***реляционную*** модель;

- *SQL-модель;*

- **объектно-ориентированную(ОО)** модель**;**

- *NoSQL-модель;*

Иерархическая модель данных



Самая простая структура - дерево с общим корнем и попарными связями. Каждый узел, кроме верхнего, связан только с одним узлом - своим предком

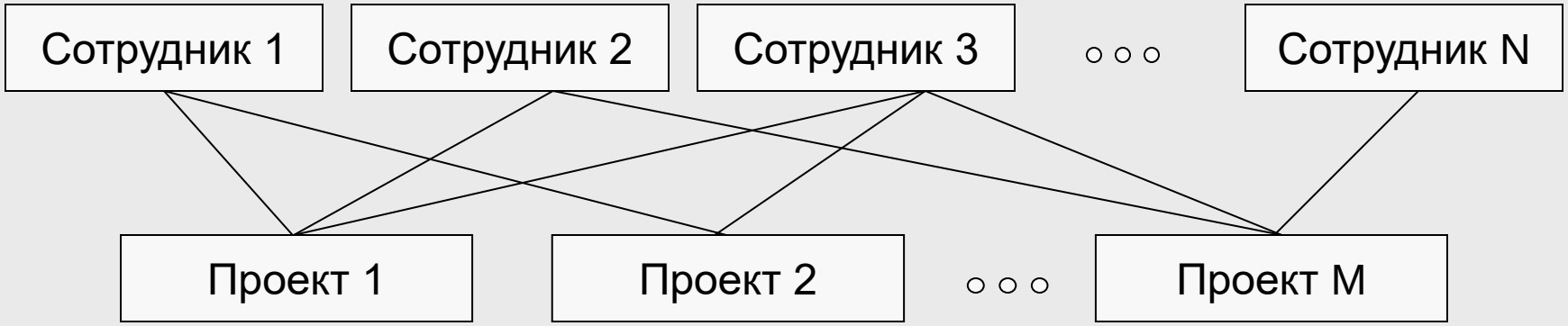
Допустимые операции

1. Найти указанное дерево.
2. Перейти от одного дерева к другому.
3. Перейти от одной записи к другой внутри дерева.
4. Перейти от одной записи к другой в порядке обхода иерархии.
5. Вставить новую запись в указанную позицию.
6. Удалить текущую запись.

Ограничение целостности

Никакой потомок не может существовать без своего родителя.

Сетевая модель данных



Каждый порождаемый узел может иметь более одного исходного узла.
Каждый узел может быть связан с любым другим узлом.

Допустимые операции

1. Найти конкретную запись в наборе однотипных записей.
2. Перейти от предка к первому потомку по некоторой связи.
3. Перейти к следующему потомку в некоторой связи.
4. Перейти от потомка к предку по некоторой связи.
5. Создать новую запись.
6. Уничтожить запись.
7. Модифицировать запись.
8. Включить в связь.
9. Исключить из связи.
10. Переставить в другую связь.

Ограничение целостности

Поддержание целостности по ссылкам.

Реляционная модель данных

Наиболее гибкая и доминирующая в настоящее время модель данных, в которой вся информация представляется в виде таблиц.

Номер	ФИО	Должность	Телефон
134	Иванов Иван Иванович	Начальник	2-36
135	Петров Петр Петрович	Ст.менеджер	2-43
136	Васильев Василий Васильевич	Гл.бухгалтер	2-27
138	Сидоров Сидор Сидорович	Менеджер	3-28
141	Сергеев Сергей Сергеевич	Менеджер	3-28

В основе этой модели данных лежит система понятий реляционной алгебры: таблица, отношение, строка, столбец, первичный ключ и т.д.

Механизмы выполнения операций

Слово ***relation - отношение*** - в реляционной алгебре означает ***«таблица»***.
В реляционных языках (языках манипулирования реляционными данными) используются два базовых механизма манипулирования реляционными данными:

1) реляционная алгебра, основанная на теории множеств;
2) реляционное исчисление, базирующееся на математической логике.

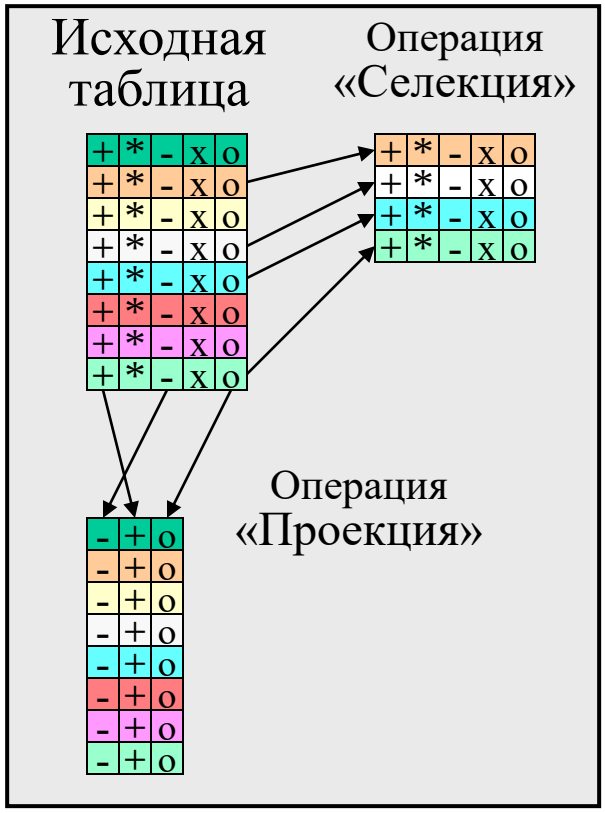
Любой запрос на обработку данных может быть выражен с помощью одного оператора реляционного языка, например языка **SQL (Structured Query Language)**.

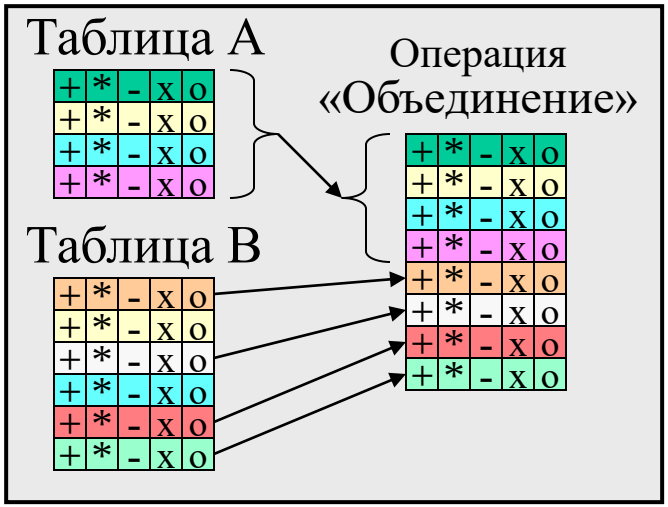
Ограничения целостности

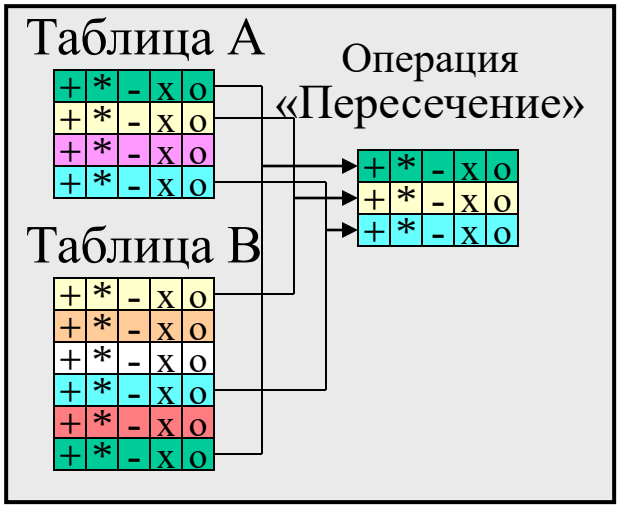
1. Требование целостности сущностей, или уникальности первичных ключей.
2. Требование целостности по ссылкам, или наличия внешнего ключа.

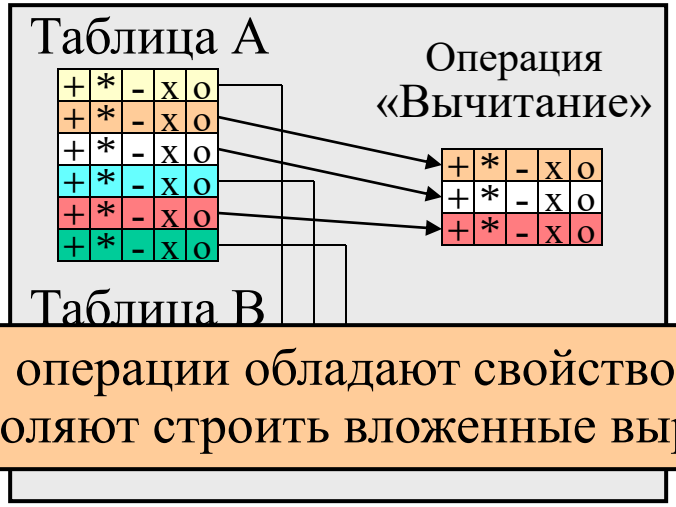
Операции над отношениями

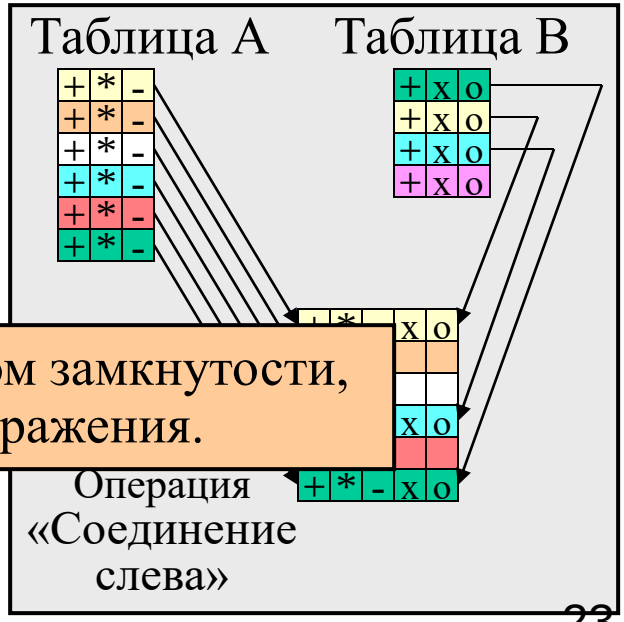
Результатом операций над отношениями является ***отношение***











Други

- «Соед
- «Переименование атрибута»;
- «Присваивания»,
- «Произведение»;
- «Деление».

Реляционные операции обладают свойством замкнутости, т.е. позволяют строить вложенные выражения.

Реляционные БД

В основе реляционной БД лежит реляционная модель данных, в которой все данные представляются в виде таблиц, и операции реляционной алгебры для работы с отношениями.

Таблица состоит из строк и столбцов и имеет имя, уникальное в пределах БД.

Строки таблиц содержат сведения о сущностях и называются ***записями***, а в реляционной алгебре - ***кортежами***.

Значения свойств объектов - атрибутов сущностей - составляют столбцы - ***поля*** таблицы.

Множество возможных значений атрибута сущностей составляют ***домен***.

Каждое поле имеет имя, уникальное в таблице. Строки имен не имеют.

Данные об объектах одного класса хранят в одной таблице.

Порядок следования строк не определен, их количество не ограничено.

Нормальные формы

Ненормализованная форма или нулевая нормальная форма (UNF) базы данных

По реляционной теории строки в таблицах не должны быть пронумерованы, т.е. порядок строк не имеет значения, так же как не имеет значения порядок столбцов.

Т.е. например, если мы поменяем порядок столбцов, или порядок строк, ничего измениться не должно, это не должно ни на что повлиять.

Таким образом по реляционной теории мы не можем обратиться к определённой строке или столбцу по ее номеру.

И если Ваши таблицы соблюдают эти принципы, то можно переходить к нормализации базы данных.

Рассмотрим небольшой пример. Достаточно часто в Excel можно встретить таблицы следующего вида

Порядковый номер строки	A	B
1	Иван	Иванов
2	Сергей	Сергеев
3	John	Smith
4	Иван	Иванов

first_name	last_name
Иван	Иванов
Сергей	Сергеев
John	Smith
Иван	Иванов

Нормализация отношений

Разбиение (декомпозиция) таблиц на несколько более мелких представляет собой процедуру нормализации отношений.

Окончательная цель нормализации - получение такой модели данных, в которой ***каждый факт появляется лишь в одном месте***, т.е. исключается избыточность информации и ее последствия.

Выделяют следующую последовательность нормальных форм: 1НФ, 2НФ, 3НФ, НФБК, 4НФ, 5НФ.

Каждой нормальной форме соответствует определенный набор ограничений на структуру данных. Каждая следующая форма отличается от предыдущей некоторым дополнительным ограничением.

Нормализация отношений - последовательный процесс приведения структуры данных к формам более высокого уровня. Цель нормализации - исключить:

- избыточность данных и как следствие:
- потенциальную противоречивость (аномалии обновления);
- аномалии включения (при дополнении неполными новыми данными);
- аномалии удаления.

Требования первой нормальной формы (1NF)

1. В таблице не должно быть дублирующих строк
2. В каждой ячейке таблицы хранится атомарное значение (одно не составное значение, отсутствуют массивы и списки в любом виде)
3. В столбце хранятся данные одного типа
4. Не должно быть полей, которые обозначают различные виды одного и того же, например, товаров.

Пример приведения таблицы к первой нормальной форме

Следующая таблица не находится даже в первой нормальной форме, так как у нас есть дублирующие строки (John Smith), а в некоторых ячейках хранятся списки значений (*каждый номер телефона — это одно значение*)

Таблица в ненормализованном виде.

Сотрудник	**Контакт**
Иванов И.И.	123-456-789
Сергеев С.С.	Рабочий тел Домашний 999
John Smith	123-456-789
John Smith	123-456-789

Таблица в нормализованном виде:

Сотрудник	**Телефон**	**Тип телефона**
Иванов И.И.	123-456-789	
Иванов И.И.	987-654-321	
Сергеев С.С.	555-666-777	Рабочий телефон
Сергеев С.С.	777-888-999	Домашний телефон
John Smith	123-456-789	

Нормализация базы данных

Нормализация (нормальные формы) – это разработка такой структуры БД, в которой нет избыточных данных и связей.

Основные принципы:

- Любое поле должно быть **неделимым – 1-я НФ.**

Фамилия и имя
Иван …
Петр …
…

→

Фамилия	Имя
Иванов	Петр
Петров	Иван
	…

- Не должно быть полей, которые обозначают различные виды одного и того же, например, товаров.

Год	Бананы	Киви
2006	…	1200
2007	…	1500
…		

→

Год	Код товара	Кол-во
2006	1	1200
2007	2	1500
…		

∞ — 1

Код	Товар
1	Бананы
2	Киви
…	

Требования второй нормальной формы (2NF)

Чтобы база данных находилась во второй нормальной форме (2NF), необходимо чтобы ее таблицы удовлетворяли следующим требованиям:

- Таблица должна находиться в первой нормальной форме
- Таблица должна иметь ключ
- Все неключевые столбцы таблицы должны зависеть от полного ключа (*в случае если он составной*)

Ключ – это столбец или набор столбцов, по которым гарантировано можно отличить строки друг от друга, т.е. ключ идентифицирует каждую строку таблицы.

По ключу мы можем обратиться к конкретной строке данных в таблице.

Если ключ составной, т.е. состоит из нескольких столбцов, то все остальные неключевые столбцы должны зависеть от всего ключа, т.е. от всех столбцов в этом ключе.

Если какой-то атрибут (столбец) зависит только от одного столбца в ключе, значит, база данных не находится во второй нормальной форме.

Приведение к второй нормальной форме

ФИО	Должность	Подразделение	Описание подразделения
Иванов И.И.	Программист	Отдел разработки	Разработка и сопровождение приложений и сайтов
Сергеев С.С.	Бухгалтер	Бухгалтерия	Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности
John Smith	Продавец	Отдел реализации	Организация сбыта продукции

Табельный номер	ФИО	Должность	Подразделение	Описание подразделения
1	Иванов И.И.	Программист	Отдел разработки	Разработка и сопровождение приложений и сайтов
2	Сергеев С.С.	Бухгалтер	Бухгалтерия	Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности
3	John Smith	Продавец	Отдел реализации	Организация сбыта продукции

Первая нормальная форма - 1NF

Данные о посещении поликлиники, сгруппированные по пациентам, представляют ненормализованное отношение:

Пациент	Г/р	Соц/полож	Отделение	Врач	Кат.	Оклад	Диагноз
Иванов И.И.	1940	Пенсионер	Терапевтическое	Петрова	2	2200	Радикулит
			Физиотерапевтич	Горячева	1	2100	Радикулит
			Хирургическое	Ножиков	3	2300	Ушиб колен
Васильев В.В.	1951	Служащий	Терапевтическое	Петрова	2	2200	ОРВИ
			Офтальмологиче	Зрачкова	4	2400	Соринка

Таблица находится ***в первой нормальной форме*** (1НФ), если значения полей во всех записях заданы однозначно, т.е. значения всех атрибутов каждой сущности ***атомарны*** :

Пациент	Г/р	Соц/полож	Отделение	Врач	Кат.	Оклад	Диагноз
Иванов И.И.	1940	Пенсионер	Терапевтическое	Петрова	2	2200	Радикулит
Иванов И.И.	1940	Пенсионер	Физиотерапевтич	Горячева	1	2100	Радикулит
Иванов И.И.	1940	Пенсионер	Хирургическое	Ножиков	3	2300	Ушиб колен
Васильев В.В.	1951	Служащий	Терапевтическое	Петрова	2	2200	ОРВИ
Васильев В.В.	1951	Служащий	Офтальмологиче	Зрачкова	4	2400	Соринка

Вторая нормальная форма - 2NF

Таблица находится ***во второй нормальной форме*** (2НФ), если она удовлетворяет определению 1НФ и не содержит неполных зависимостей неключевых полей от возможного первичного ключа.

Пациент	Г/р	Соц/полож	Отделение	Врач	Кат.	Оклад	Диагноз
Иванов И.И.	1940	Пенсионер	Терапевтическое	Петрова	2	2200	Радикулит
Иванов И.И.	1940	Пенсионер	Физиотерапевтич	Горячева	1	2100	Радикулит
Иванов И.И.	1940	Пенсионер	Хирургическое	Ножиков	3	2300	Ушиб колен
Васильев В.В.	1951	Служащий	Терапевтическое	Петрова	2	2200	ОРВИ
Васильев В.В.	1951	Служащий	Офтальмологиче				

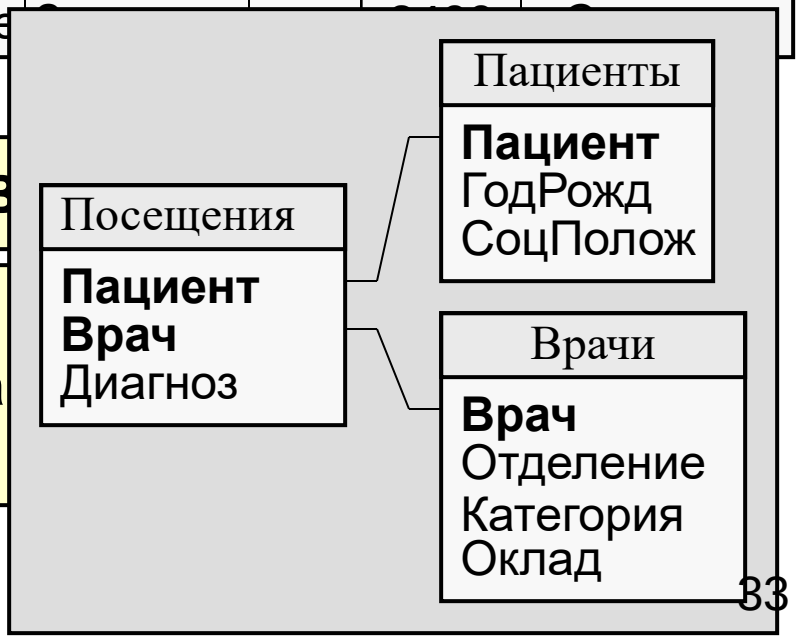
Посещения

Пациент
ГодРожд
СоцПолож
Отделение
Врач
Категория
Оклад
Диагноз

Возможный ключ: **Пациент-В**

Неполные зависимости:

ГодРожд, **СоцПолож** от **Па**
Отделен., **Катег.**, **Оклад** от



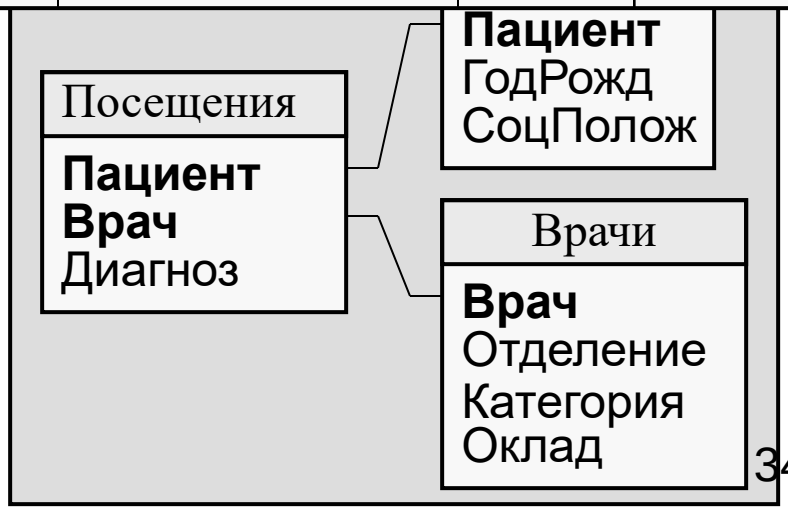
Приведение структуры данных к 2NF

Пациент	Г/р	Соц/полож	Отделение	Врач	Кат.	Оклад	Диагноз
Иванов И.И.	1940	Пенсионер	Терапевтическое	Петрова	2	2200	Радикулит
Иванов И.И.	1940	Пенсионер	Физиотерапевтич	Горячева	1	2100	Радикулит
Иванов И.И.	1940	Пенсионер	Хирургическое	Ножиков	3	2300	Ушиб колен
Васильев В.В.	1951	Служащий	Терапевтическое	Петрова	2	2200	ОРВИ
Васильев В.В.	1951	Служащий	Офтальмологиче	Зрачкова	4	2400	Соринка

Пациенты		
Пациент	ГодРожд	СоцПолож
Иванов И.И.	1940	Пенсионер
Васильев В.В.	1951	Служащий

Посещения		
Пациент	**Врач**	Диагноз
Иванов И.И.	Петрова	Радикулит
Иванов И.И.	Горячева	Радикулит
Иванов И.И.	Ножиков	Ушиб колен
Васильев В.В.	Петрова	ОРВИ
Васильев В.В.	Зрачкова	Соринка

Врачи			
Врач	Отделение	Катег.	Оклад
Петрова	Терапевтическое	2	2200
Горячева	Физиотерапевтич	1	2100
Ножиков	Хирургическое	3	2300
Зрачкова	Офтальмологиче	4	2400



Требования третьей нормальной формы (3NF)

Требование третьей нормальной формы (3NF) заключается в том, чтобы в таблицах отсутствовала транзитивная зависимость.

Транзитивная зависимость – это когда неключевые столбцы зависят от значений других неключевых столбцов.

Если в первой нормальной форме наше внимание было нацелено на соблюдение реляционных принципов, во второй нормальной форме в центре нашего внимания был первичный ключ, то в третьей нормальной форме все наше внимание уделено столбцам, которые не являются первичным ключом, т.е. неключевым столбцам.

Чтобы нормализовать базу данных до третьей нормальной формы, необходимо сделать так, чтобы в таблицах отсутствовали неключевые столбцы, которые зависят от других неключевых столбцов.

Иными словами, неключевые столбцы не должны пытаться играть роль ключа в таблице, т.е. они действительно должны быть неключевыми столбцами, такие столбцы не дают возможности получить данные из других столбцов, они дают возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение.

Главное правило третьей нормальной форме (3NF) звучит следующим образом: **таблица должна содержать правильные неключевые столбцы**

Номер	ФИО	Должность	Подразделение	Описание подразделения
1	Иванов И.И.	Программист	Отдел разработки	Разработка и сопровождение приложений и сайтов
2	Сергеев С.С.	Бухгалтер	Бухгалтерия	Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности
3	John Smith	Продавец	Отдел реализации	Организация сбыта продукции

Номер	ФИО	Должность	Подразделение
1	I		
2	C		
3	J		

3NF

Идентификатор подразделения	Подразделение	Описание подразделения
1	Отдел разработки	Разработка и сопровождение приложений и сайтов
2	Бухгалтерия	Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности
3	Отдел реализации	Организация сбыта продукции

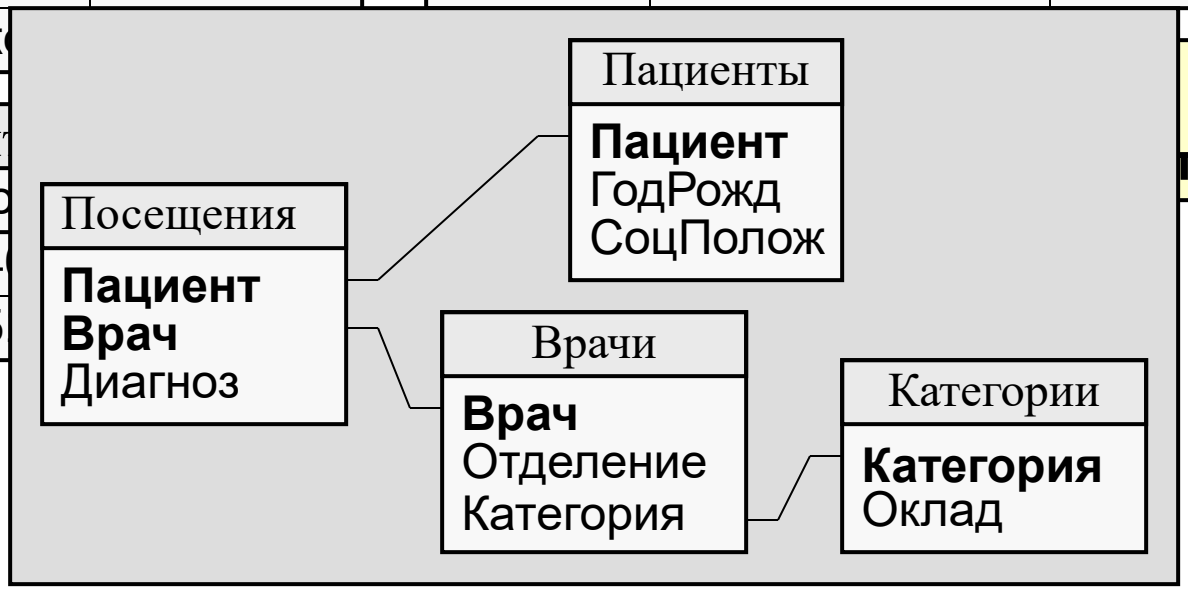
Третья нормальная форма - 3NF

Таблица находится ***в третьей нормальной форме*** (3НФ), если она удовлетворяет определению 2НФ и ни одно из ее неключевых полей не зависит функционально (однозначно) от любого другого неключевого поля.

Посещения		
Пациент	**Врач**	Диагноз
Иванов И.И.	Петрова	Радикулит
Иванов И.И.	Горячева	Радикулит
Иванов И.И.	Ножиков	Ушиб колен
Васильев В.В.	Петрова	ОРВИ
Васильев В.В.	Зрачк	

Врачи			
Врач	Отделение	Катег.	Оклад
Петрова	Терапевтическое	2	2200
Горячева	Физиотерапевтич	1	2100
Ножиков	Хирургическое	3	2300
Зрачкова	Офтальмологиче	4	2400

Пациент	
Пациент	ГодРо
Иванов И.И.	194
Васильев В.В.	195



тегория

Приведение структуры данных к 3NF

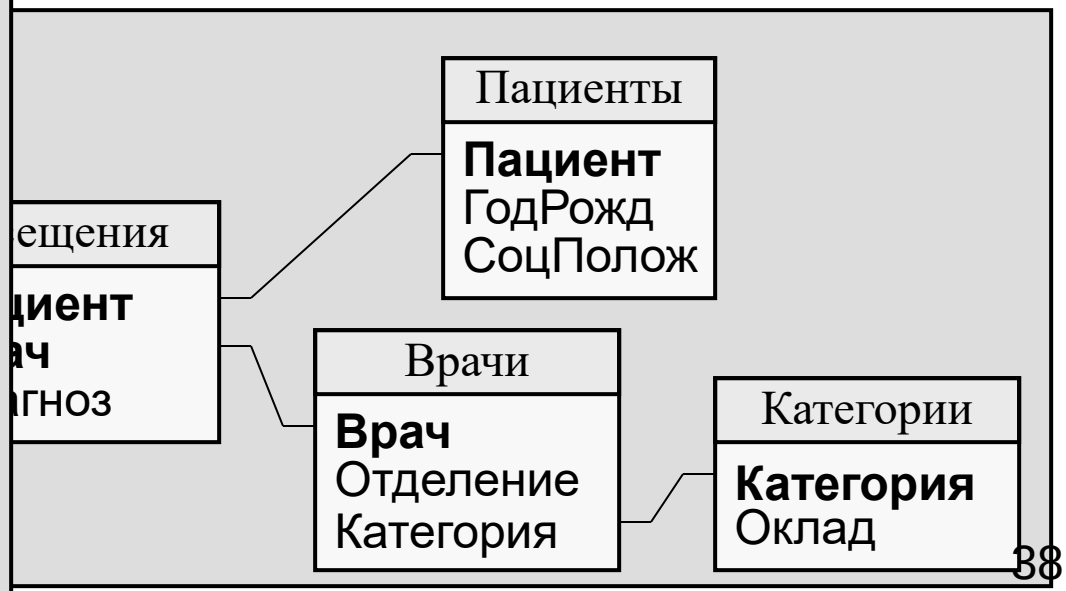
Посещения		
Пациент	**Врач**	Диагноз
Иванов И.И.	Петрова	Радикулит
Иванов И.И.	Горячева	Радикулит
Иванов И.И.	Ножиков	Ушиб колен
Васильев В.В.	Петрова	ОРВИ
Васильев В.В.	Зрачкова	Соринка

Пациенты		
Пациент	ГодРожд	СоцПолож
Иванов И.И.	1940	Пенсионер
Васильев В.В.	1951	Служащий

Врачи			
Врач	Отделение	Катег.	Оклад
Петрова	Терапевтическое	2	2200
Горячева	Физиотерапевтич	1	2100
Ножиков	Хирургическое	3	2300
Зрачкова	Офтальмологиче	4	2400

Врачи		
Врач	Отделение	Катег.
Петрова	Терапевтическое	2
Горячева	Физиотерапевтич	1
Ножиков	Хирургическое	3
Зрачкова	Офтальмологиче	4

Категории	
Категория	Оклад
1	2100
2	2200
3	2300
4	2400



Создание таблиц в СУБД

На практике процесс нормализации заканчивается приведением структуры данных к 3НФ.

3НФ является компромиссом между полной нормализацией структуры данных,

Сотрудники							
N	ФИО	Должность	Оклад	Ставка	Кабинет	Вн.тел.	Отделен.
14	Сидорова С.С.	Ст.мед.сестра	1800	1,5	23	3-25	Терапевт.
16	Ножиков Н.Н.	Хирург	2300	1	16	1-24	Хирургич.
17	Петрова П.П.	Терапевт	2200	1,5	26	2-67	Терапевт.

нормализации структуры данных.

Создание таблиц состоит в задании имени таблицы, имен полей, указании типов данных в полях и выделении ключевых полей. В некоторых СУБД автоматически создается поле для неизменяемого порядкового номера записи, которое может быть использовано в качестве ключа.

Сотрудники		
Поле	Тип	Ключ
КодСотрудника	Счетчик	*
ФИО	Текст	
Должность	Текст	
Оклад	Денежный	
Ставка	Веществ	
Кабинет	Целый	
ВнутрТелефон	Текст	
Отделение	Текст	

Ключевые поля и связи

Таблицы,создаваемые в СУБД, обычно соответствуют 2НФ, т.е. значения атрибутов атомарны и все поля определяются первичным ключом.

Сотрудники

N	ФИО	Должность	Оклад	Ставка	Кабинет	Вн.тел.	Отделен.
14	Сидорова С.С.	Ст.мед.сестра	1800	1,5	23	[illegible]	2
16	Ножиков Н.Н.	Хирург	2300	1	[illegible]	1-[illegible]4	3
17	Петрова П.П.	Терапевт	2200	1,5	26	2-67	2

Для однозначного соответствия данных в таблицах устанавливаются связи между ними путем задания ссылок на значения ключевых полей.

Поле, с помощью значений которого однозначно определяется подстановка данных из записей другой таблицы, называется ***внешним ключом***.

Подразделения

N	Наименование
1	Дирекция
2	Терапевтическое отд.
3	Хирургическое отд.
4	Физиотерапевтич.каб.

Тип данных внешнего ключа должен совпадать с типом данных первичного ключа.

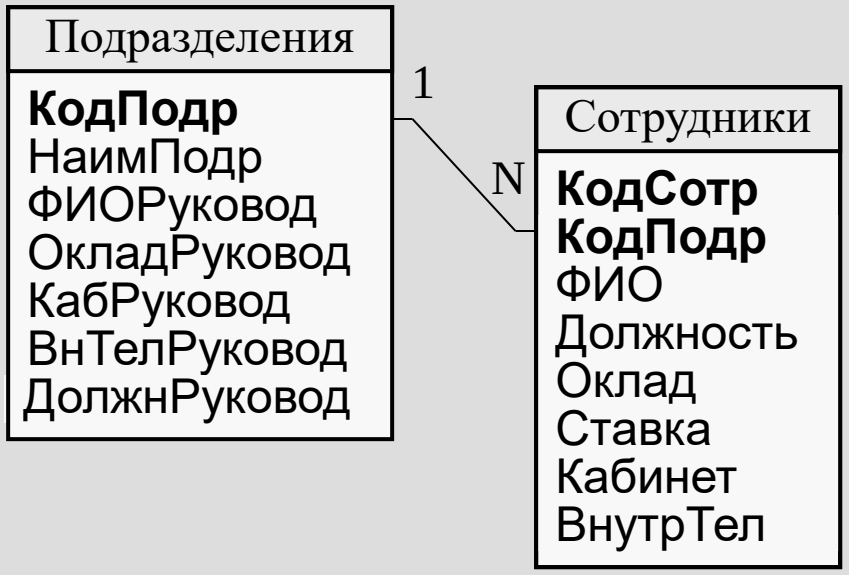
Установление связей между таблицами

Для установления связей между таблицами и задания их типов в разных СУБД используются разные приемы.

Сотрудники

N	ФИО	Должность	Оклад	Ставка	Кабинет	Вн.тел.	Отделен.
14	Сидорова С.С.	Ст.мед.сестра	1800	1,5	23	3-25	2
16	Ножиков Н.Н.	Хирург	2300	1	16	1-24	3
17	Петрова П.П.	Терапевт	2200	1,5	26	2-67	2

Структура данных



Подразделения

N	Наименование
1	Дирекция
2	Терапевтическое отд.
3	Хирургическое отд.
4	Физиотерапевтич.каб.

Обычно поля внешних ключей располагают сразу после полей первичного ключа.Часто их включают в состав первичного ключа

Этапы проектирования базы данных

Проект реляционной базы данных -
это набор взаимосвязанных отношений, для которых определены все атрибуты, первичные ключи и ограничения целостности.

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания предметной области к формализованному описанию объектов предметной области в терминах некоторой модели.

Выделяют четыре этапа проектирования реляционной базы данных:

- системный анализ предметной области;
- построение инфологической модели предметной области;
- даталогическое проектирование базы данных;
- физическое проектирование базы данных.

Между вторым и третьим этапами принимается решение об использовании конкретной СУБД для реализации проекта.

Системный анализ предметной области

На этапе системного анализа предметной области выполняется подробное словесное описание ее объектов и связей между ними.

Существуют два подхода к выбору состава и структуры предметной области:

- функциональный подход;
- предметный подход.

Функциональный подход - принцип движения «от задач». Он применяется, когда известны функции группы лиц и комплексов проблем, для обслуживания информационных потребностей которых создается БД или ИС.

Предметный подход - когда невозможно выделить минимальный набор объектов предметной области, и информационные потребности будущих пользователей БД или ИС жестко не фиксируются. В описание включаются только наиболее существенные для предметной области объекты и взаимосвязи. В этом случае БД наращивается по мере ее эксплуатации.

На практике часто применяется компромиссный вариант.

Итог: описание информации об объектах, которая должна храниться; формулировки задач, которые будут решаться, описание алгоритмов их решения, описание генерируемых выходных документов, описание входных документов.

Автоматизация этапа системного анализа

На этапе системного анализа выполняется семантическое (смысловое, концептуальное) моделирование БД с помощью CASE-технологий (Computer Aided Software Engineering).

Цель - организация интерфейса проектировщика и конечного пользователя с ИС на уровне представлений предметной области, а не на уровне структур данных.

Пакет **ERwin 7.2** - имеет поддержку 25 СУБД. Использует два уровня представления модели: логический и физический. Процесс построения БД состоит из трех этапов.

1. Определение сущностей, связей между ними, задание атрибутов и ключей.
2. Назначение соответствий имя сущности - имя таблицы, атрибут сущности - поле таблицы, задание ограничений.
3. Генерация базы данных.

ERwin имеет интерфейс обычной программы.

Построение инфологической модели

Инфологическая модель должна включать такое формализованное описание предметной области, которое легко читается неспециалистами в области информационных технологий.

Это описание должно быть емким, представлять глубину и корректность проработки проекта и не быть привязанным к конкретной СУБД.

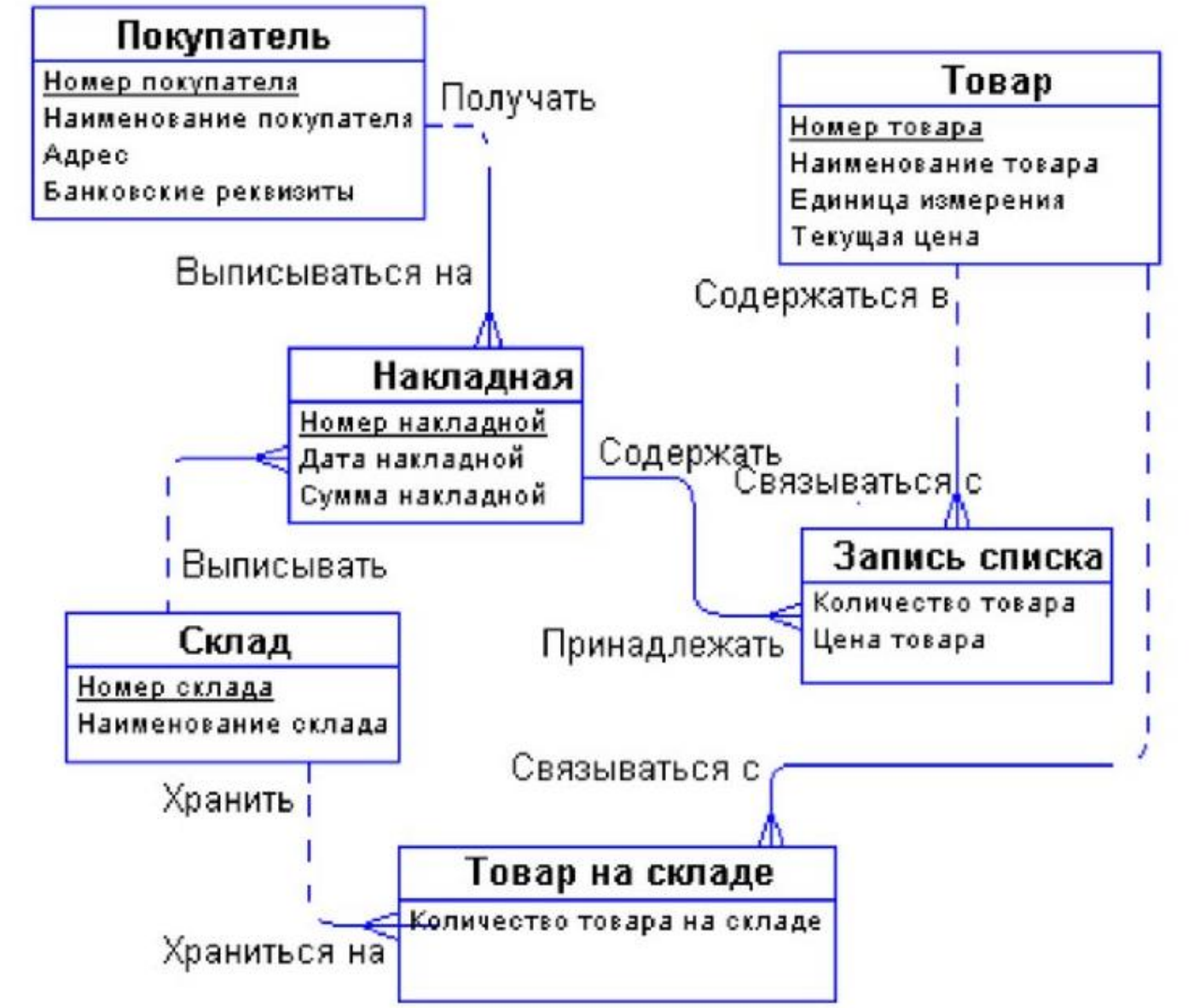
Для реализации этапа построения инфологической модели стандартом является модель «сущность-связь» (ER-модель).

Существуют программные CASE-средства преобразования проекта БД из ER-модели в реляционную даталогическую модель, соответствующую конкретной СУБД. Однако, не существует единой общепринятой системы обозначений для ER-модели, и разные CASE-системы используют разные графические нотации.

При создании относительно несложных проектов (до 30 таблиц) и заранее очевидном выборе конкретной СУБД для реализации проекта этапы построения инфологической модели и даталогического проектирования могут быть совмещены. В этом случае при построении информационной модели предметной области используется терминология и графические примитивы выбранной СУБД.

Модель «сущность-связь» (ER-модель)

20.09.2022



Даталогическое проектирование

На этапе даталогического проектирования разрабатывается схема БД в терминах конкретной СУБД.

Состав таблиц, их наименования.

Перечень, порядок следования и типы полей.

Ключи и индексы, их состав и свойства.

Связи, их типы и свойства.

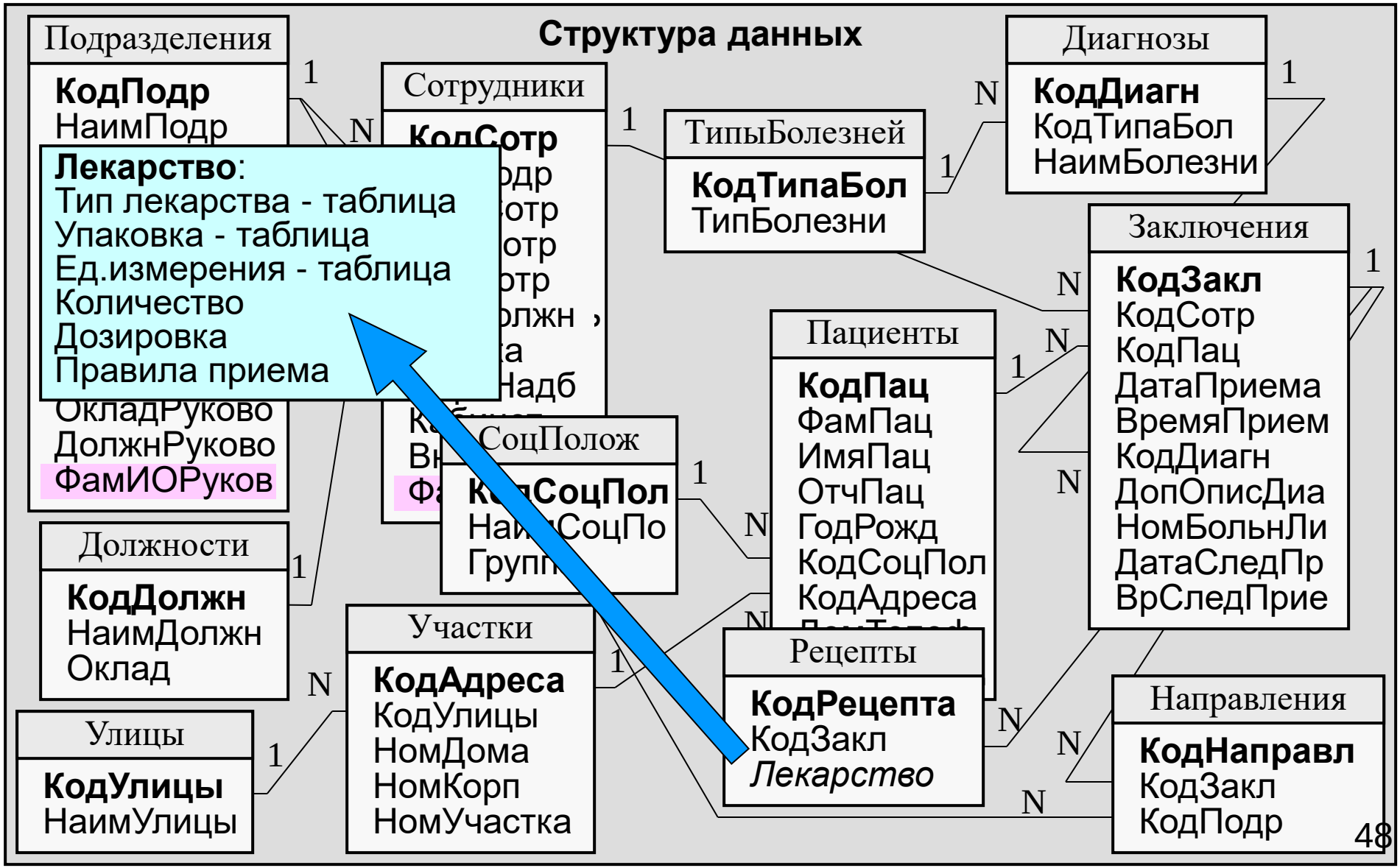
Свойства, параметры и форматы данных для полей.

Два пути проектирования схемы БД:

- ***декомпозиция*** отношений на основе анализа зависимостей между атрибутами - замена отношений их проекциями методом нормализации - классический вариант;
- ***синтез*** схемы из исходных элементарных зависимостей между объектами предметной области.

Пример даталогического проектирования

Схема базы данных «***Поликлиника***»



Понятие транзакции

Под ***транзакцией*** понимается воздействие на БД, переводящее ее из одного целостного состояния в другое.

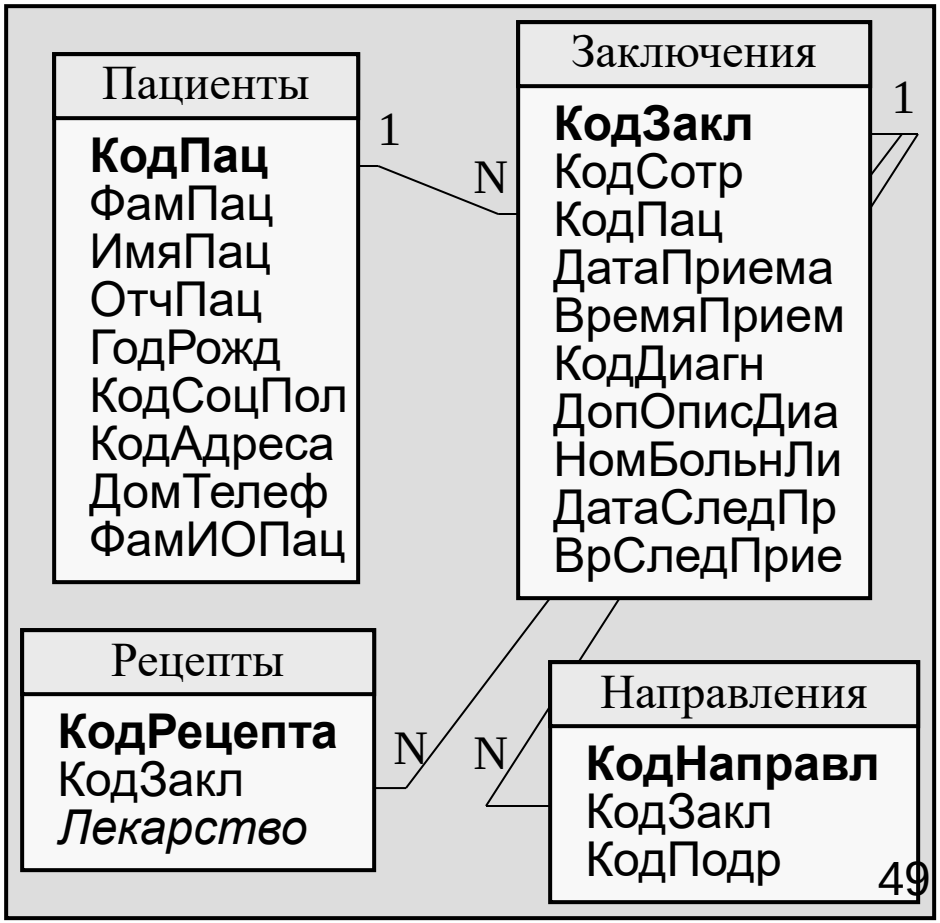
Воздействие на БД выражается в изменении данных в таблицах: изменение района проживания пациента - удаление всех его данных или списание в архив.

При успешном завершении транзакции выполняется одновременное подтверждение всех изменений.

Если одно из изменений завершается неуспешно, то не подтверждается ни одно из изменений.

В результате должен быть произведен ***откат*** к прежнему состоянию БД.

Корректное выполнение транзакций обеспечивает ***семантическую (смысловую) целостность*** БД.



Язык запросов SQL

Для получения сведений из БД пользователь дает СУБД ***запрос***. СУБД обрабатывает его и отправляет пользователю результат.

Запросы к СУБД формулируются на специальных ***языках запросов***. С их помощью указывается какие данные надо получить, но не задается способ выполнения в отличие от языков программирования.

Фактическим стандартом языка запросов для современных реляционных СУБД является ***SQL*** (***Structured Query Language***) версий 2 и 3 - ***SQL2, SQL3***.

Запросы на языке SQL строятся с использованием операторов

SELECT	Выбрать строки
INSERT	Вставить строку
UPDATE	Обновить строку
DELETE	Удалить строки
COMMIT	Завершить транзакцию
ROLLBACK	Откатить транзакцию
CREATE TABLE	Создать таблицу
CREATE INDEX	Создать индекс

Результат выполнения запроса формируется в виде таблицы.

CREATE

```
CREATE TABLE student
(St_id int NOT NULL PRIMARY KEY,
St_name varchar(45) NOT NULL,
Speciality int(10) not null);
```

INSERT

```
insert into student (St_id, St_name, Speciality) values
        (1, "Иванов", 2), (2, "Петров", 1);
```

UPDATE

UPDATE имя_таблицы SET
имя_столбца_1=значение, ..., имя_столбца_n=значение
[WHERE УСЛОВИЕ];

DELETE

DELETE FROM имя_таблицы
[**WHERE** условие_удаления]

Создание таблицы (Пример)

```
CREATE TABLE attainment(
    At_id    int(3) NOT NULL PRIMARY KEY,
    St_id    int(3) NOT NULL,
    Sub_id   int(3) NOT NULL,
    T_id     int(3) NOT NULL,
    Date     date,
    Rating   int(4) NOT NULL,
    FOREIGN KEY(St_id) REFERENCES student(St_id),
    FOREIGN KEY(Sub_id) REFERENCES subject(Sub_id),
    FOREIGN KEY(T_id) REFERENCES teacher(T_id));
```

Обобщенный формат оператора ALTER TABLE

ALTER TABLE имя_таблицы

[ADD [COLUMN] имя_столбца тип_данных [NOT NULL][UNIQUE]

[DEFAULT <значение>][CHECK (<условие_выбора>)]]

[DROP [COLUMN] имя_столбца [RESTRICT | CASCADE]]

[ADD [CONSTRAINT [имя_ограничения]]

[{PRIMARY KEY (имя_столбца [,...n]) |[UNIQUE (имя_столбца [,...n])}

|[FOREIGN KEY (имя_столбца_внешнего_ключа [,...n]) REFERENCES имя_род_таблицы [(имя_столбца_род_таблицы [,...n])],

[MATCH {PARTIAL | FULL} [ON UPDATE {CASCADE| SET NULL | SET DEFAULT | NO ACTION}] [ON DELETE {CASCADE| SET NULL | SET DEFAULT | NO ACTION}] |[CHECK(<условие_выбора>)][,...n]}]

[DROP CONSTRAINT имя_ограничения [RESTRICT | CASCADE]]

[ALTER [COLUMN] SET DEFAULT <значение>]

[ALTER [COLUMN] DROP DEFAULT]

Оператор SELECT

Оператор выбора строк **SELECT** реализует ***все операции*** реляционной алгебры и имеет достаточно сложный синтаксис.

Самый простой запрос содержит только два обязательных ***ключевых слова***, которые включает в себя запрос любой сложности: **SELECT** и **FROM:**

```
SELECT *
FROM Улицы, ТипыБолезней
```

Общий синтаксис запроса:

```
SELECT <имена полей (что)>
FROM <имена таблиц (откуда)>
```

Символ * (звезда) используется для выбора всех полей исходных таблиц.

Таким способом выполняется реляционная операция «Декартово произведение отношений».

Улицы

КодУли	НаимУл
1	Вешняк
2	Юности

ТипыБолезней

КодТипаБ	ТипБоле
1	Сердечно
2	Опорнодв

Результат

КодУли	НаимУл	КодТипаБ	ТипБоле
1	Вешняк	1	Сердечно
1	Вешняк	2	Опорнодв
2	Юности	1	Сердечно
2	Юности	2	Опорнодв

Обобщенный оператор SELECT

```
SELECT столбцы
FROM таблица
 [WHERE условие_фильтрации_строк]
 [GROUP BY столбцы_для_группировки]
 [HAVING условие_фильтрации_групп]
 [ORDER BY столбцы_для_сортировки]
```

Получение проекций

Для получения проекций результата выполнения реляционных операций достаточно в операторе выбора строк **SELECT** указать имена полей, включаемых в таблицу-результат.

```
SELECT НаимДолжн, Оклад
FROM Должности
```

Должности
КодДолжн
НаимДолжн
Оклад

Результат
НаимДолжн
Оклад

```
SELECT Улицы.НаимУлицы,
    ТипыБолезней.ТипБолезни
FROM Улицы, ТипыБолезней
```

```
SELECT НаимУлицы,
    ТипБолезни
FROM Улицы, ТипыБолезней
```

Улицы
КодУлицы
НаимУлиц

ТипыБолезней
КодТипаБол
ТипБолезни

Результат
НаимУлицы
ТипБолезни

Улицы	
КодУли	**НаимУл**
1	Вешняк
2	Юности

ТипыБолезней	
КодТипаБ	**ТипБоле**
1	Сердечно
2	Опорнодв

Результат	
НаимУл	**ТипБоле**
Вешняк	Сердечно
Вешняк	Опорнодв
Юности	Сердечно
Юности	Опорнодв

Исключение повторений

Для исключения строк-дубликатов из таблицы-результата надо после ключевого слова **SELECT** указать ключевое слово **DISTINCT**.

Участки
КодАдреса
КодУлицы
НомДома
НомКорп
НомУчастка

Участки

КодАдреса	КодУлицы	НомДома	НомКорп	НомУчастка
1	1	4	1	2
2	2	2	2	2
3	6	9	2	1
4	2	2	1	2
5	3	7		3
6	6	9А		1
7	4	3	1	3

Запрос. Выдать перечень обслуживаемых поликлиникой участков.

```
SELECT НомУчастка
FROM Участки
```

Результат

НомУчастка
2
2
1
2
3
1
3

```
SELECT DISTINCT НомУчастка
FROM Участки
```

Результат

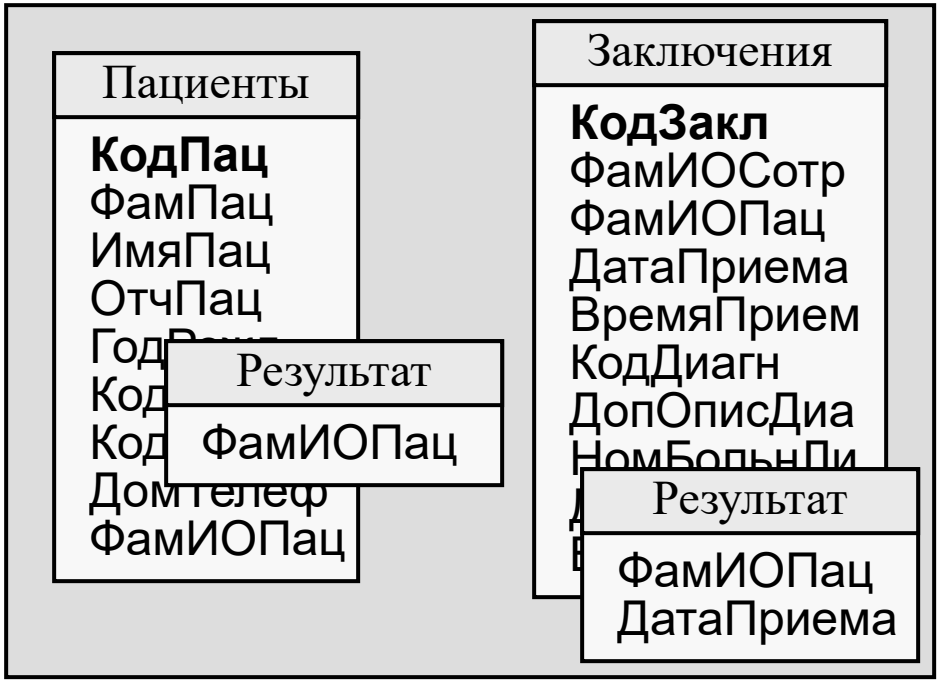
НомУчастка
2
1
3

Отбор записей из таблиц

Условия отбора записей из таблиц задаются в операторе **SELECT** с помощью ***предикатов*** после ключевого слова **WHERE**

Общий синтаксис запроса:
SELECT <имена полей>
FROM <имена таблиц>
WHERE <предикат (условие)>

Запрос. Выдать список пациентов с годом рождения до 1942.
SELECT DISTINCT ФамИОПац
FROM Пациенты
WHERE ГодРожд<=1941



Запрос. Выдать список пациентов с датой приема, посетивших поликлинику в августе 2001 года в первой половине рабочего дня.
SELECT DISTINCT ФамИОПац, ДатаПриема
FROM Заключения
WHERE ДатаПриема BETWEEN 01.08.01 AND 31.08.01
AND ВремяПриема < 13:30

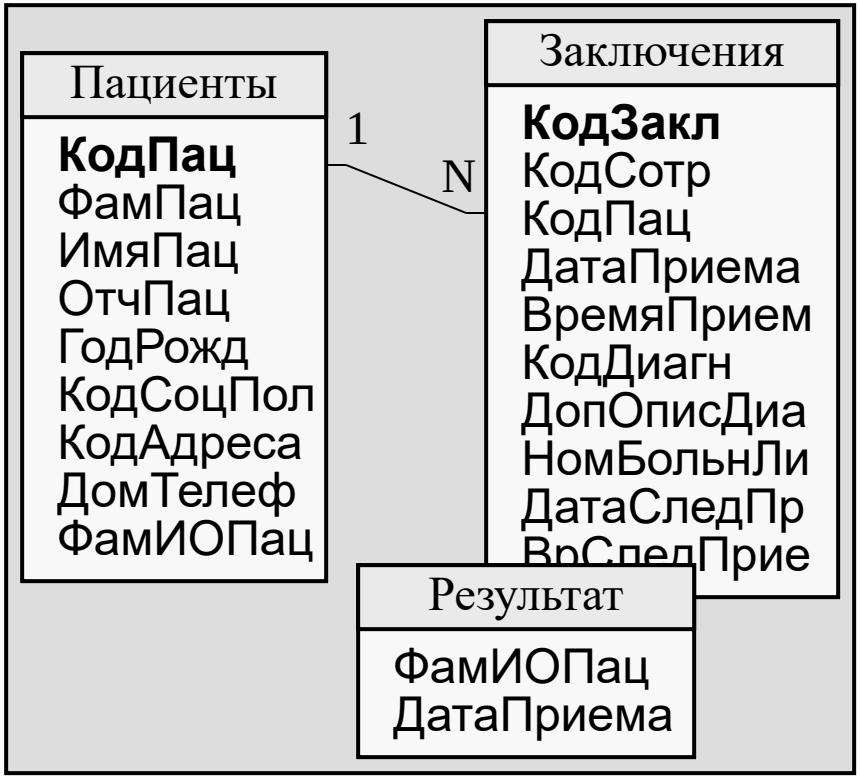
Отбор записей из нескольких таблиц

Отбор записей из нескольких таблиц реализует реляционную операцию «Соединение».

После ключевого слова **WHERE** в качестве условия соединения таблиц размещают предикаты сравнения значений их ключевых полей.

Запрос. Выдать список пациентов с датами посещения поликлиники, назначенных на повторный прием.

```
SELECT DISTINCT ФамИОПац,
    ДатаПриема
FROM Пациенты, Заключения
WHERE Пациенты.КодПац=
    Заключения.КодПац
    AND ДатаСледПосещ IS NOT NULL
```



Если значение поля в некоторой записи не задано, оно считается ***неопределенным*** и обозначается **NULL**.

Предикаты для проверки на неопределенность: **IS NULL** и **IS NOT NULL**.

Операции с неопределенным значением

Неопределенное значение в теории БД интерпретируется как значение, неизвестное на данный момент времени.

Для неопределенных значений, обозначаемых **NULL**, не действуют стандартные правила сравнения: два неопределенных значения всегда не равны друг другу.

Введение неопределенного значения потребовало расширить в рамках теории БД классическую двузначную логику до трехзначной.

Таблица истинности логических операций

A	B	NOT A	A AND B	A OR B
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
TRUE	NULL	FALSE	NULL	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	NULL	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL	TRUE
NULL	FALSE	NULL	FALSE	NULL
NULL	NULL	NULL	NULL	NULL

Группировка данных в таблицах

Для анализа данных с одинаковыми значениями атрибутов записи таблицы разбиваются на группы. Интересующие значения указываются после ключевого слова **GROUP BY**.

В результате для полученных групп записей можно определить некоторые характерные значения с помощью ***агрегатных функций***, вычисляющих одно значение для каждой группы:

COUNT	Количество записей
SUM	Сумма значений
AVG	Среднее значение
MIN	Минимальное значение
MAX	Максимальное значение

Запрос. Сколько участков обслуживает поликлиника (ск.разных номеров участков).

```
SELECT COUNT (DISTINCT НомУчастка)
FROM Участки
GROUP BY НомерУчастка
```

Участки

КодАдреса
КодУлицы
НомДома
НомКорп
НомУчастка

Результат

НомУчастка	COUNT (*)
2	3
1	2
3	

Результат

COUNT (DI
3

Пример сложного запроса

Запрос. Какие отделения поликлиники и какое количество пенсионеров 2-го участка с диагнозом «Радикулит» посетило за осенне-зимний период 2000-01гг?


```
SELECT DISTINCT Подразделения.НаимПодр,
    COUNT(DISTINCT Участки. НомУчастка)
FROM Подразделения
W
    WHERE Подразделения.КодПодр = Сотрудники.КодПодр
        AND Сотрудники.КодСотр = Заключения.КодСотр
        AND Заключения. КодПац = Пациенты.КодПац
        AND Пациенты.КодСоцПол = СоцПол.КодСоцПол
        AND Пациенты.КодАдреса = Участки.КодАдреса
GROUP BY Подразделения. НаимПодр
```