

Лекция 10

Характеристики и методика моделирования базы данных

1. Характеристики процесса моделирования базы данных

Модель базы данных (database model, data model⁵) — способ описания базы данных с помощью формализованного (в т.ч. графического) языка на некотором уровне абстракции.

Упрощённо: описание базы данных, по которому она потом будет создана (по аналогии с проектом здания, по которому оно потом будет построено).

У одной и той же базы данных может быть несколько взаимосвязанных моделей, отличающихся уровнем детализации и целью.

Уровни моделирования базы данных

Уровень	Что описывает	Чем оперирует
---------	---------------	---------------

Бизнес-анализ, выявление требований заказчика и параметров предметной области

Увеличение степени детализации ↓	Логический уровень	Инфологический (концептуальный) уровень	Предметная область без привязки к виду баз данных	Сущности, атрибуты, некоторые связи
		Даталогический (логический) уровень	Предметная область с привязкой к виду базы данных или даже конкретной СУБД	Сущности, атрибуты, связи, ключи, некоторые индексы и представления
		Физический уровень	Технические аспекты реализации базы данных под управлением конкретной СУБД	Сущности, атрибуты, связи, ключи, индексы, представления, триггеры, хранимые подпрограммы, методы доступа, кодировки, права доступа и т.д. и т.п.

Системное администрирование, конфигурирование СУБД

1 - Инфологический уровень

Упрощённо: описание предметов и явлений реального мира, данные о которых потом будут помещены в базу данных.

Цель

Отразить бизнес-требования и логическую структуру данных

Результат

- Концептуальная модель данных в виде ER-диаграммы (в нотации Чена или Crow's Foot) или UML-диаграммы
- Глоссарий терминов
- Текстовое описание сущностей, атрибутов и связей

2 - Даталогический уровень

Упрощённо: описание предметов и явлений реального мира по правилам выбранной СУБД.

Цель

Преобразовать концептуальную модель в нормализованные таблицы, ключи, ограничения

Результат

Модель представляется в виде диаграммы в формате IDEF1X или UML.

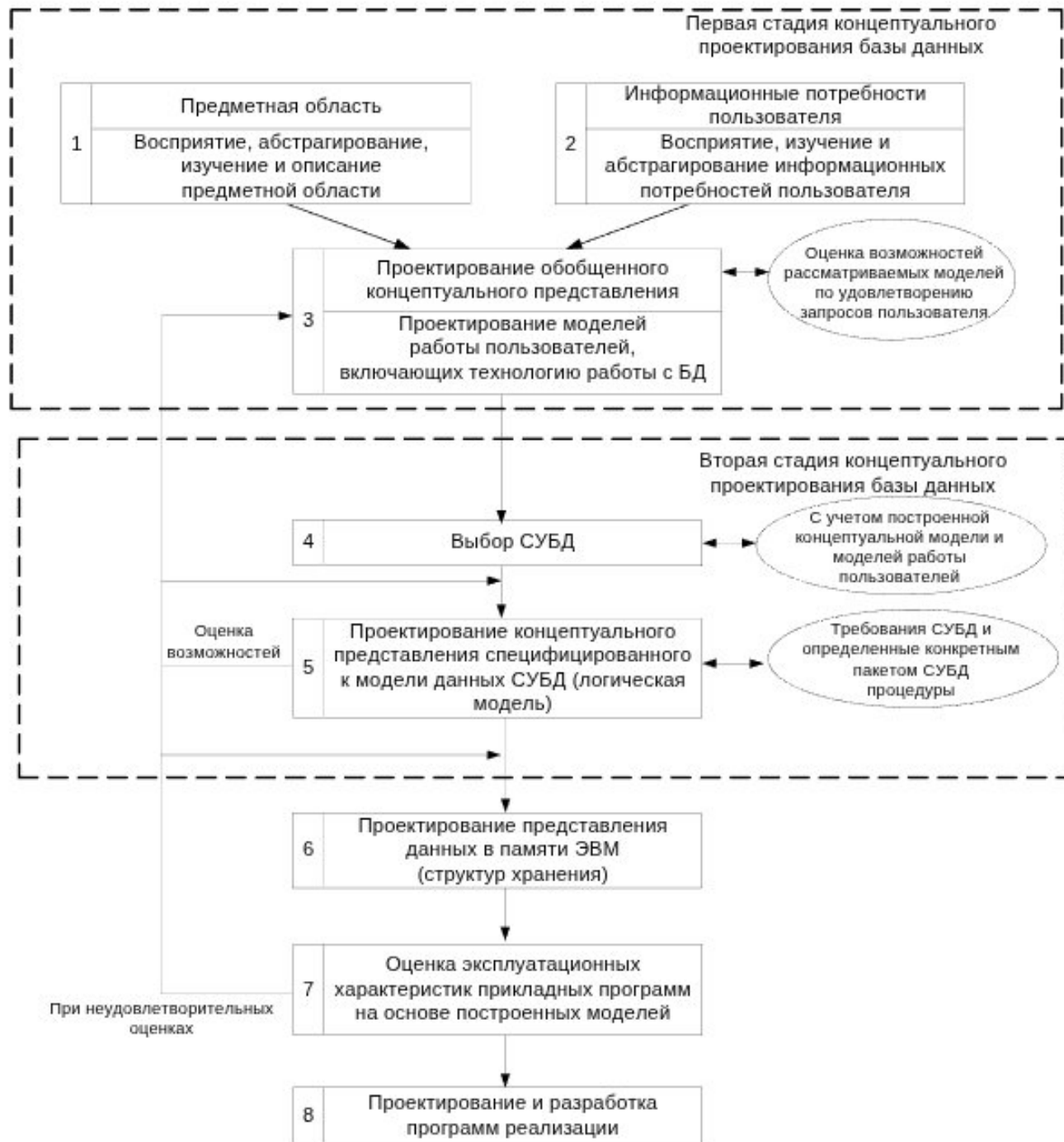
Упрощённо: описание составных частей базы данных таким образом, чтобы на его основе можно было автоматически сгенерировать SQL-код для создания базы данных.

Цель

Оптимизация производительности и хранения с учётом аппаратных и программных ограничений

Результат

- SQL-скрипты для создания таблиц, индексов, представлений, хранимых процедур
- Словесные описания необходимых изменений и настроек
- Фрагменты конфигурационных файлов
- Подготовленные cmd/bash-скрипты
- reg-файлы
- ...



Итерационность процесса моделирования

- Анализ требований может выявить ограничения на физическом уровне (например, необходимость высокой скорости отклика), которые повлияют на логическую или даже концептуальную модель.
- При проектировании логической структуры может обнаружиться, что некоторые связи в инфологической модели трудно реализуемы или неэффективны, что ведёт к её уточнению.
- Тестирование и использование системы может выявить новые требования или неудачные решения, требующие возврата к предыдущим уровням.

Таким образом, процесс цикличен: переход от одного уровня к другому сопровождается обратной связью, уточнениями и корректировками.

Нисходящий подход (top-down)

- Начинается с анализа предметной области → строится инфологическая модель → затем логическая → затем физическая.
- Преимущества:
 - Сильная ориентация на бизнес-требования.
 - Целостность и согласованность модели.
- Типичен при разработке новых систем «с нуля».

Восходящий подход (bottom-up)

- Начинается с анализа существующих данных, форм, отчётов, унаследованных таблиц → выявляются сущности и связи → обобщаются до концептуальной модели.
- Преимущества:
 - Учёт реальных данных и практик.
 - Эффективен при реинжиниринге (реорганизации) существующих БД.
- Часто используется при миграции, интеграции или рефакторинге.

Нисходящее и восходящее моделирование

Комбинированный подход

На практике чаще применяется гибрид:

- Основные сущности и процессы моделируются сверху вниз,
- Детали и ограничения уточняются снизу вверх на основе реальных данных или технических требований.

Вывод

Процесс моделирования базы данных не является строго линейным. Он:

- **Итерационен:** включает циклы уточнения и корректировки между уровнями;
- **Гибок:** допускает как нисходящее (от бизнеса к реализации), так и восходящее (от данных к модели) проектирование;
- **Адаптивен:** учитывает обратную связь от пользователей, аналитиков, разработчиков и администраторов БД.

Этот подход повышает качество модели, её соответствие реальным потребностям и техническим ограничениям, что особенно важно в сложных и меняющихся проектах.

2. Требования к модели базы данных

Требования к модели БД

- Адекватность предметной области
- Удобство использования
- Производительность
- Защищенность данных

Адекватность предметной области

База данных должна позволять выполнять все необходимые операции, которые объективно нужны в реальной жизни в контексте той работы, для которой предназначена база данных.

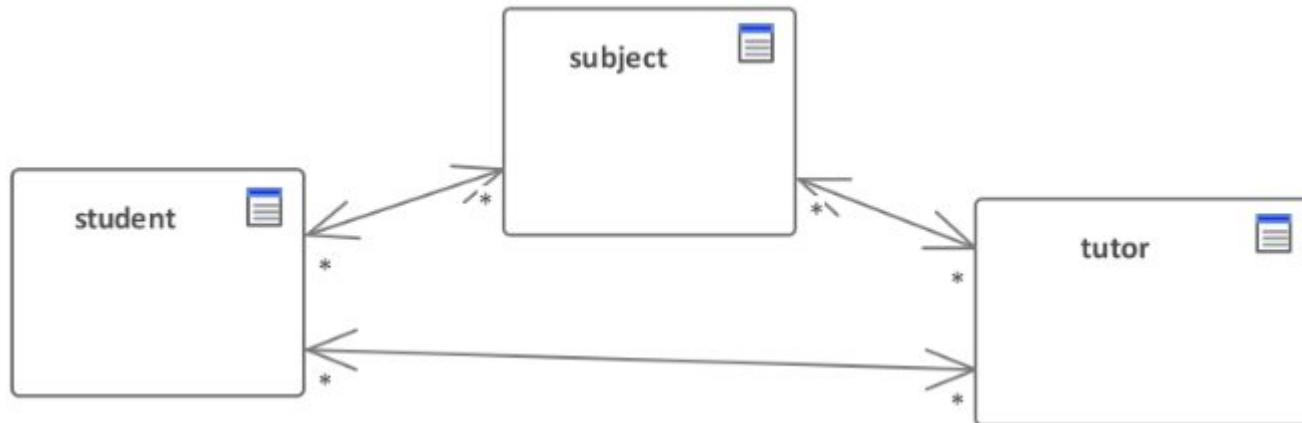
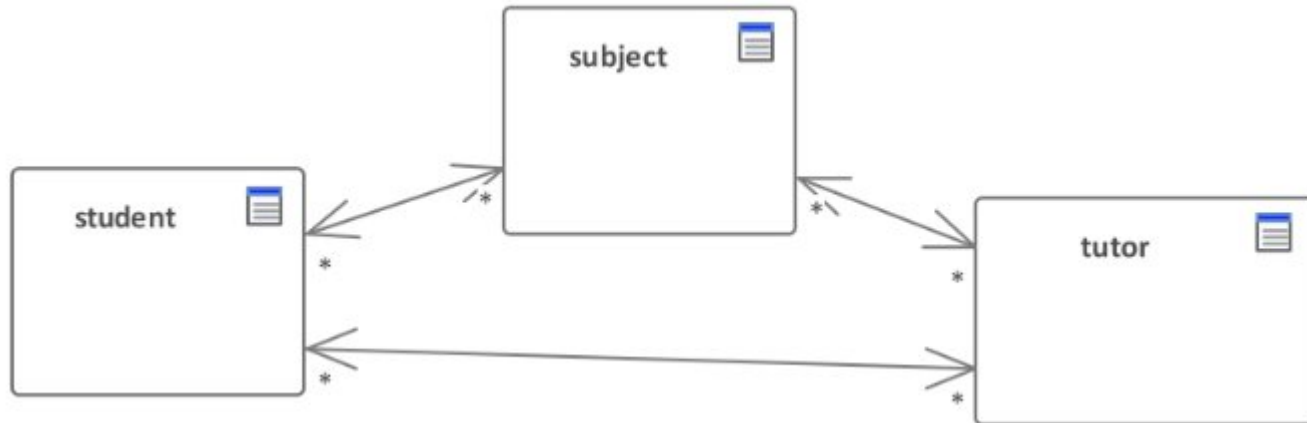


Схема БД, содержащая скрытую ошибку.

Много студентов (сущность student) изучает много предметов (сущность subject), которые ведут много преподавателей (сущность tutor).

Адекватность предметной области



Система эксплуатируется, через год для важного отчета понадобился список студентов, у которых профессор Иванов вел матанализ.

Это определить невозможно, в БД такая информация не сохранялась. Поэтому такая структура БД не является адекватной для данной предметной области.

Адекватность предметной области

Система эксплуатируется, через год для важного отчета понадобился список студентов, у которых профессор Иванов вел матанализ.

Это определить невозможно, в БД такая информация не сохранялась. Поэтому такая структура БД не является адекватной для данной предметной области.

Ещё одна форма нарушения адекватности предметной области — нарушения нормализации, которые приводят к возникновению **аномалий** операций с данными

Удобство использования

Речь идёт об «удобстве для программиста» при использовании базы данных:

- При взаимодействии с приложениями (упор делается на простоту и лёгкость написания безошибочных запросов)
- В процессе дальнейшего развития БД.

person

<u>p_id</u>	p_name	...
1	И.И. Иванов	...
2	ПП Петров	...
3	С Сидоров	...
4

Неоптимальное хранение данных - трудно упорядочить список по фамилии (в запросе из строки нужно выделить фамилию с помощью встроенных функций и рекурсивных запросов).

person

<u>p_id</u>	p_surname	p_initials	...
1	Иванов	И.И.	...
2	Петров	ПП	...
3	Сидоров	С	...
4

Оптимальное хранение данных - список по фамилии упорядочивается простым запросом.

3. Методика моделирования реляционной базы данных

Моделирование реляционной БД

Основные задачи:

- Определить нужные таблицы с учетом нормализации БД
- Указать первичные и внешние ключи
- Связать нужные таблицы друг с другом
- Определить ограничения для обеспечения целостности данных

Ожидаемый результат:

- Схема базы данных в нотации IDEF1x или UML
- SQL-скрипты для создания БД в конкретной СУБД

Нужна методика, алгоритм

Терминология

- **Предметная область** – ограниченная область реального мира. База данных - это модель (абстракция, упрощение) предметной области.
- **Сущность предметной области** – абстрактное понятие, которое объединяет группу реальных объектов, имеющих одинаковые свойства.
- **Экземпляры сущности** - реальные объекты, объединенные в одну сущность.
- **Атрибуты сущности** – свойства, которыми описывается сущность.

Пример

Предметная область – факультет математики и ИТ Мордовского госуниверситета.

Нужно спроектировать БД для хранения информации о студентах, преподавателях и администрации факультета, а также об учебном процессе и успеваемости студентов.

Шаг 1. Изучаем процессы предметной области

Кто и какие действия выполняет, какие документы используются в процессе выполнения.

Студенты:

- учатся на определённой специальности;
- изучают дисциплины, входящие в эту специальность;
- в течение семестра набирают баллы;
- в сессию сдают экзамены.

Преподаватели:

- работают на определённой кафедре;
- читают лекции по дисциплинам;
- проводят л/р;
- принимают экзамены.

Шаг 1. Изучаем процессы предметной области

На какие вопросы нужно ответить с помощью базы данных

- Статистика по студентам и преподавателям
- Анализ успеваемости
- Расписание занятий и экзаменов
- ...

Шаг 2. Определяем сущности

Сущность предметной области – абстрактное понятие, которое объединяет группу реальных объектов, имеющих одинаковые свойства.

Процесс	Кандидат на сущность
Студенты учатся в группах	СТУДЕНТ, ГРУППА
Студенты сдают экзамены	СТУДЕНТ, ЭКЗАМЕН
Лекции проводятся в аудиториях, лабораторные работы – в компьютерных лабораториях	ЛЕКЦИЯ, ЛАБОРАТОРНАЯ РАБОТА, АУДИТОРИЯ, ЛАБОРАТОРИЯ
Факультетом управляет декан	ДЕКАН, ФАКУЛЬТЕТ
На факультете организованы кафедры	КАФЕДРА
...	

Шаг 2. Определяем сущности

Сущность предметной области – абстрактное понятие, которое объединяет группу реальных объектов, имеющих одинаковые свойства.

СТУДЕНТ

ПРЕПОДАВАТЕЛЬ

ДЕКАН

ДЕКАНАТ

СЕКРЕТАРЬ ДЕКАНАТА

КАФЕДРА

ЛАБОРАТОРИЯ

АУДИТОРИЯ

ЭКЗАМЕН

ОЦЕНКА

КОМПЬЮТЕР В ЛАБОРАТОРИИ

КОНДИЦИОНЕР

Шаг 2. Определяем сущности

Сущность предметной области – абстрактное понятие, которое объединяет группу реальных объектов, имеющих одинаковые свойства.

СТУДЕНТ

ПРЕПОДАВАТЕЛЬ

ДЕКАН

ДЕКАНАТ

СЕКРЕТАРЬ ДЕКАНАТА

КАФЕДРА

ЛАБОРАТОРИЯ

АУДИТОРИЯ

ЭКЗАМЕН

ОЦЕНКА

КОМПЬЮТЕР В ЛАБОРАТОРИИ

КОНДИЦИОНЕР

Шаг 2. Выделяем атрибуты сущностей



Шаг 2. Выделяем атрибуты сущностей

СТУДЕНТ

Номер зачетной книжки
ФИО
Дата рождения
Место проживания
Дата поступления
Форма обучения

ПРЕПОДАВАТЕЛЬ

Табельный номер
ФИО
Дата рождения
Место проживания
Кафедра
Должность

КАФЕДРА

Название
Заведующий

Дисциплина

Код дисциплины
Название

Шаг 3. Устанавливаем связи между сущностями

Структура описания связи

Подлежащее сказуемое дополнение

Студент изучает дисциплину.

Преподаватель принимает экзамен.

Студент сдает экзамен.

В одном предложении участвуют ровно две сущности.

Шаг 4. Определяем типы связей между сущностями

- **Связь 1:1.** Одиночный экземпляр сущности одного класса связан с одиночным экземпляром сущности другого класса.
- **Связь 1:M.** Единый экземпляр сущности одного класса связан со многими экземплярами сущности другого класса.
- **Связь M:N.** Несколько экземпляров сущности одного класса связаны с несколькими экземплярами сущности другого класса.

Структура описания связи

Один студент изучает много дисциплин.

Одну дисциплину изучает много студентов.

Один преподаватель работает на одной кафедре.

Одна кафедра включает много преподавателей.

Шаг 4. Определяем типы связей между сущностями

Описываем каждую связь двумя предложениями, переставляя подлежащее и дополнение. Подлежащее в каждом предложении обязательно должно начинаться со слова «Один» («Одна», «Одно»).

Первое предложение описывает связь со стороны первой сущности, второе – со стороны второй сущности.

Структура описания связи

Один студент изучает много дисциплин.

Одну дисциплину изучает много студентов.

Один преподаватель работает на одной кафедре.

Одна кафедра включает много преподавателей.

Результат шагов 1-4: Визуализация модели

1. ER-диаграммы

2. UML-диаграммы классов

ER-диаграммы

Назначение: Специально разработаны для моделирования структуры баз данных.

Основные элементы:

- **Сущности (Entities)** — представляют объекты предметной области (например, «Клиент», «Заказ»).
- **Атрибуты (Attributes)** — свойства сущностей (например, «Имя», «Дата рождения»).
- **Связи (Relationships)** — ассоциации между сущностями (например, «Клиент делает Заказ»).
- Поддержка **кардинальности** (один-к-одному, один-ко-многим, многие-ко-многим).
- Иногда используются **слабые сущности, идентифицирующие связи, наследование** (в расширенных версиях, например, EER).

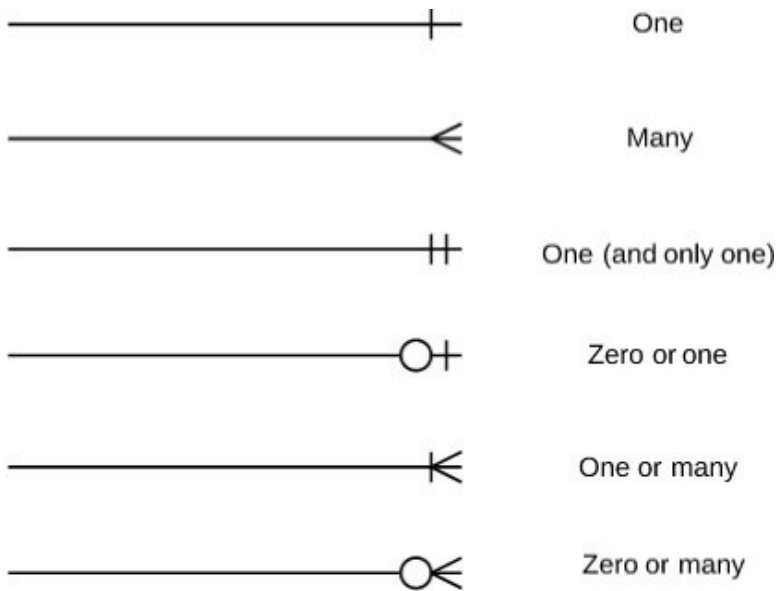
ER-диаграммы

- Ориентированы на структуру данных и связи.
- Легко преобразуются в реляционную схему БД.
- Стандартизированные нотации: Чен (Chen notation), Crow's Foot и др.

ER-диаграммы в нотации Чена



ER-диаграммы в нотации Crow's foot



Сущности

- атрибуты
- первичные ключи

Связи между сущностями

- ТИПЫ

UML-диаграммы классов

Назначение: Часть языка UML, первоначально предназначена для моделирования объектно-ориентированных программных систем, но применима и к моделированию БД.

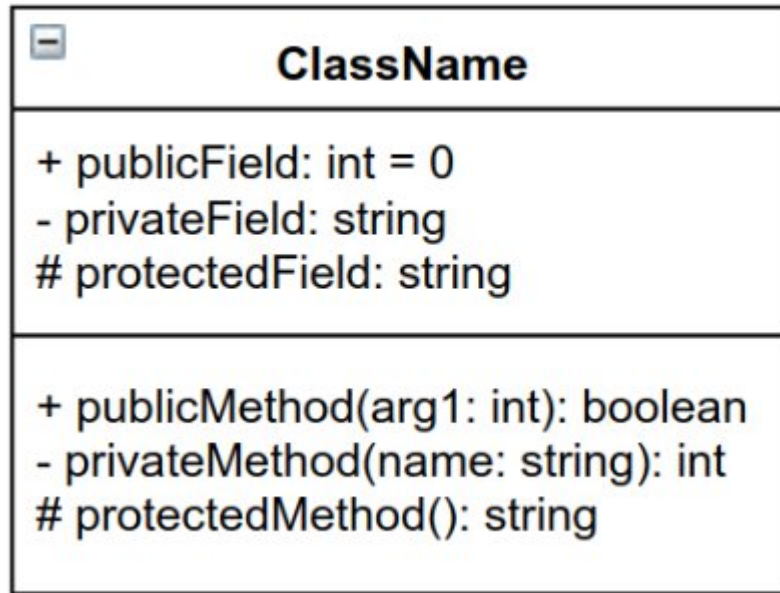
Основные элементы:

- **Классы** — аналоги сущностей (например, класс «Customer»).
- **Атрибуты** — поля класса (аналоги атрибутов сущностей).
- **Операции/методы** — поведение (часто игнорируются при моделировании БД).
- **Ассоциации** — связи между классами, с указанием мультипликации (аналог кардинальности).
- **Поддержка наследования, агрегации, композиции.**

UML-диаграммы классов

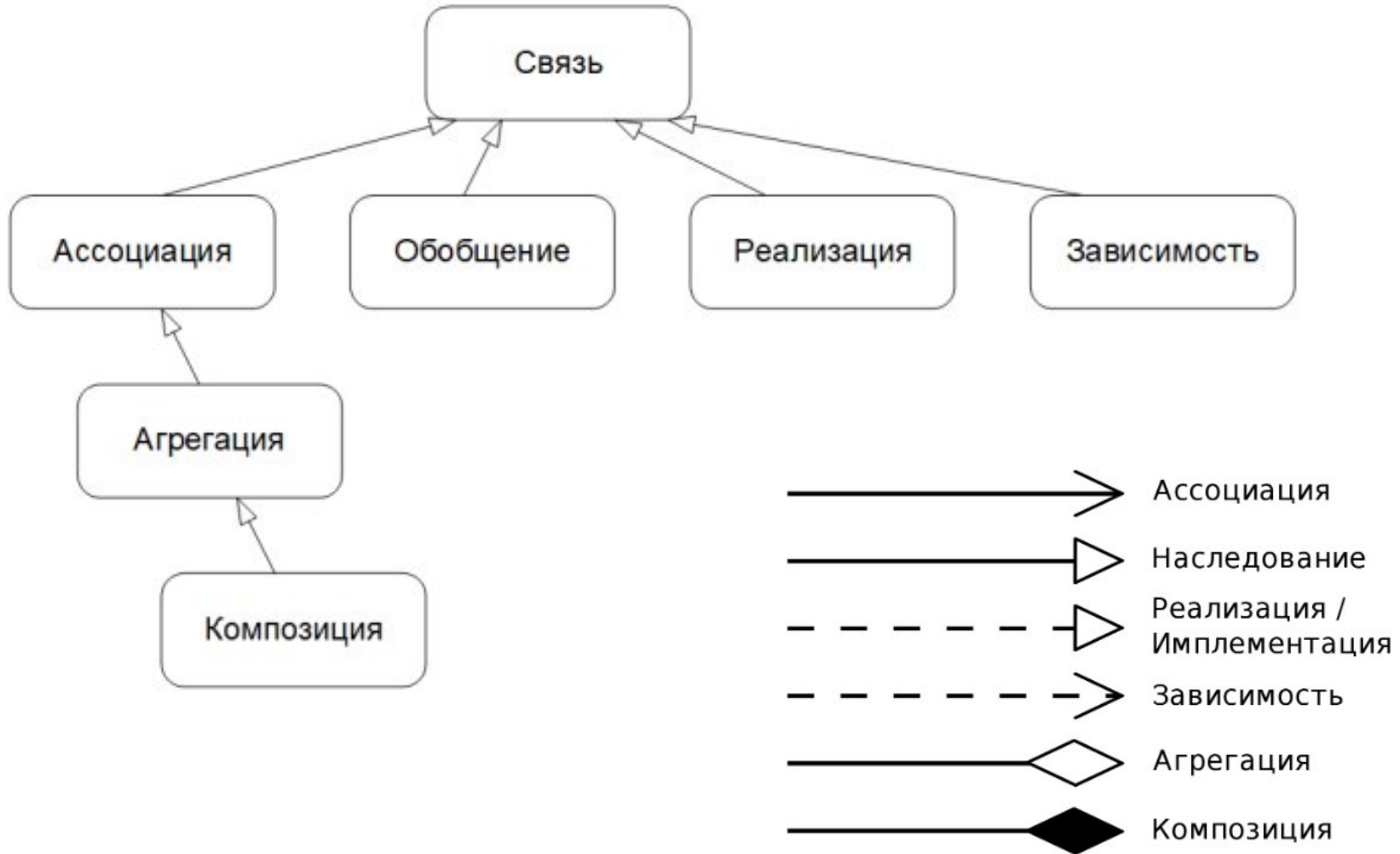
- Более универсальны — подходят и для проектирования ПО, и для описания БД.
- Могут включать поведенческую информацию (методы), которая не нужна при построении БД.
- Часто используются в средах, где разработка ПО и проектирование БД идут параллельно.

UML-диаграммы классов



Класс с атрибутами и операциями

UML-диаграммы классов



Отношения между классами

Связи между классами

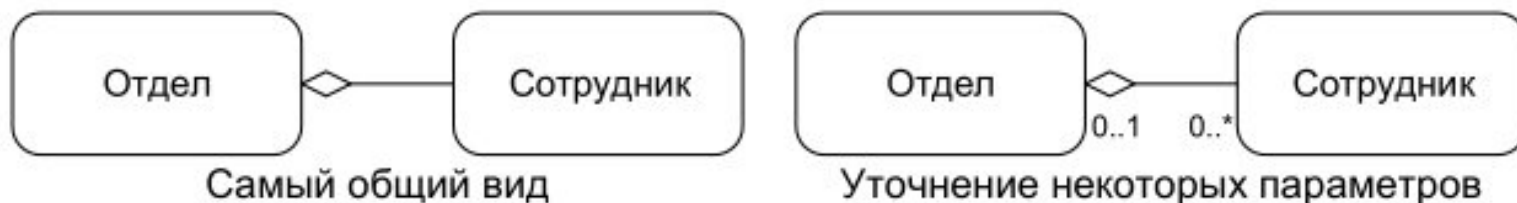
Ассоциация — самый общий, универсальный и «ни к чему не обязывающий вариант»: просто отражается тот факт, что некие сущности находятся во взаимосвязи



- электронный пропуск принадлежит сотруднику (а не наоборот);
- электронный пропуск обязан принадлежать ровно одному сотруднику;
- у сотрудника может не быть электронного пропуска;
- у сотрудника может быть не более одного электронного пропуска.

Связи между классами

Агрегация — частный случай ассоциации, показывающий, что дочерние элементы входят в состав родительского элемента (но могут и существовать отдельно).



- в отделе может быть от нуля до бесконечности сотрудников;
- сотрудник может не принадлежать никакому отделу;
- сотрудник может принадлежать не более, чем одному отделу.

Связи между классами

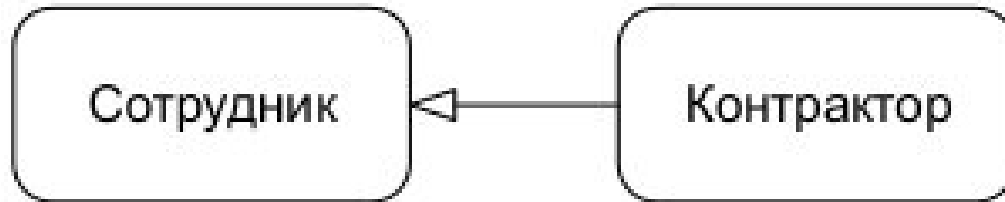
Композиция — ещё один частный случай ассоциации, показывающий, что дочерние элементы входят в состав родительского элемента, но, в отличие от агрегации, здесь дочерние элементы не могут существовать самостоятельно (без родительского элемента).



- каждый текст обязан относиться хотя бы к одному музыкальному произведению;
- текст может относиться к более, чем одному музыкальному произведению;
- к каждому музыкальному произведению может относиться от нуля до бесконечности текстов.

Связи между классами

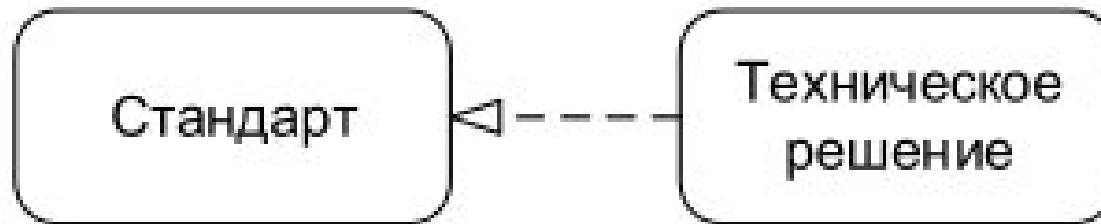
Обобщение — вид связи, показывающий, что её дочерний элемент является частным случаем родительского.



Контрактор является частным случаем сотрудника

Связи между классами

Реализация — вид связи, показывающий, что дочерний элемент реализует поведение, задаваемое родительским элементом.



Техническое решение подчинено стандарту, т.к. обязано «вести себя» так, как сказано в стандарте.

Связи между классами

Зависимость — вид связи, показывающий, что изменения в родительском элементе обязательно приводят к изменениям в дочернем элементе (но не наоборот).

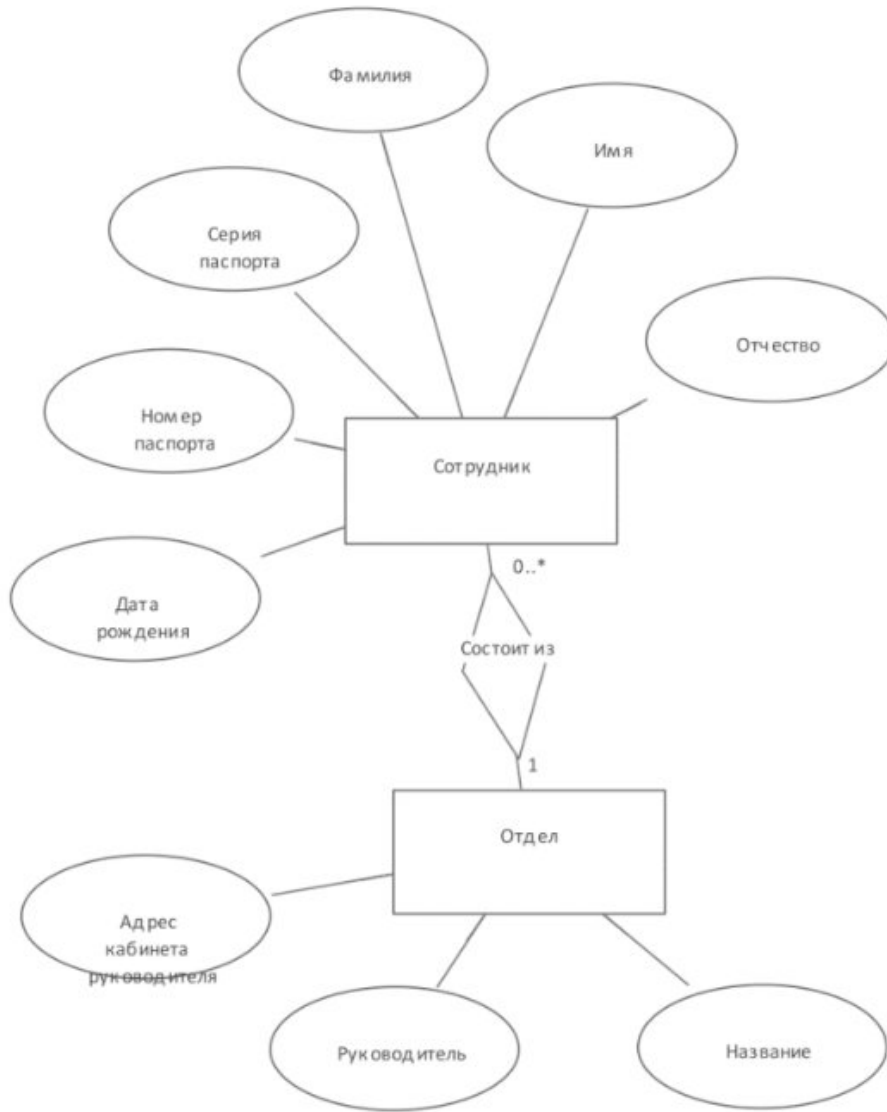


- у каждого сезона есть хотя бы один вид погоды (но может быть и больше, до бесконечности);
- каждый вид погоды обязан относиться хотя бы к одному сезону (но может относиться и к большему количеству сезонов, до бесконечности).

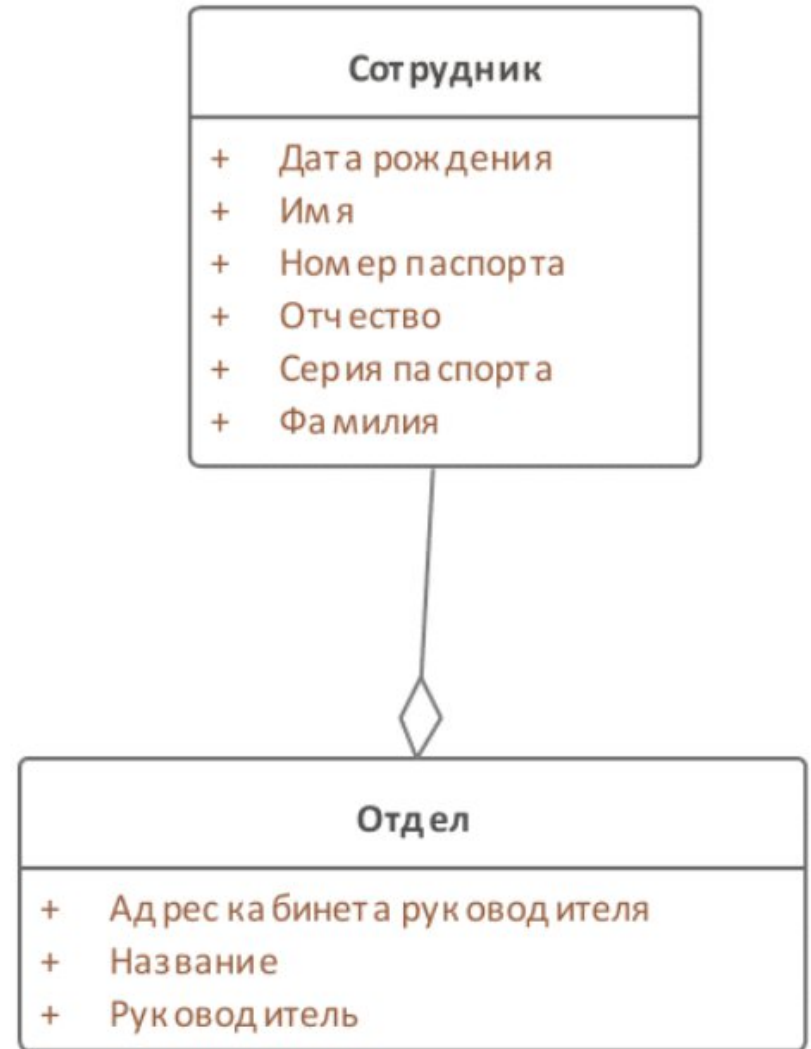
Соответствие между элементами UML и реляционной модели

ЭЛЕМЕНТ БД	АНАЛОГ В UML-ДИАГРАММЕ КЛАССОВ
Таблица	Класс
Столбец (поле)	Атрибут класса
Тип данных (INT, VARCHAR)	Тип атрибута (int, String и т.д.)
Первичный ключ	Атрибут с пометкой <code>{PK}</code> или стереотипом <code>«PK»</code>
Внешний ключ	Ассоциация между классами + атрибут с пометкой <code>{FK}</code>
Кардинальность связи	Мультипликаторы (1, 0..1, 0.., 1..)
NOT NULL	Ограничение <code>{non nullable}</code> или отсутствие <code>= null</code> по умолчанию
Уникальность (UNIQUE)	Ограничение <code>{unique}</code>

Сравнение ER-диаграммы и UML-диаграммы

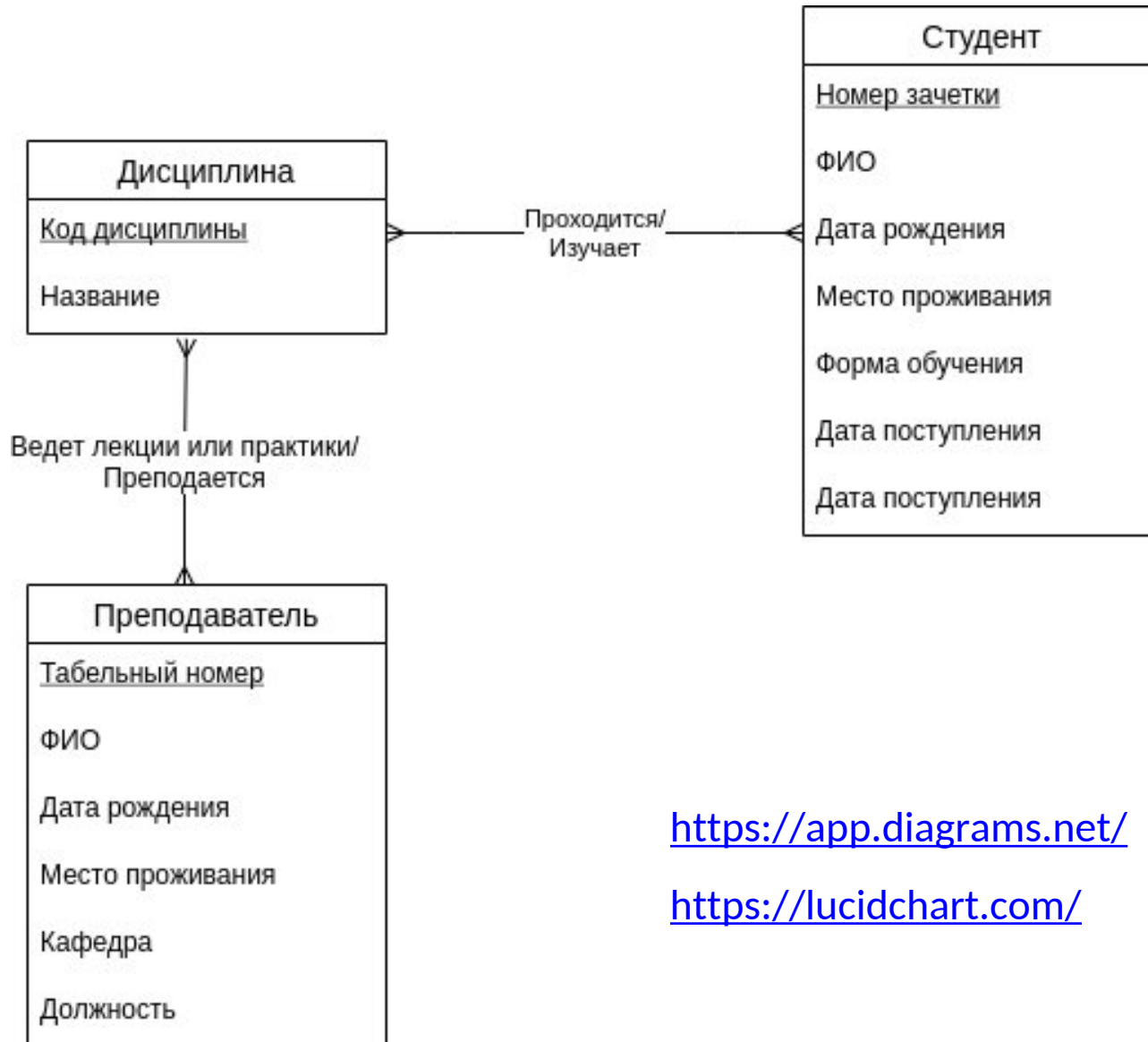


ER-диаграмма



UML-диаграмма

Результат шагов 1-4: Часть ER-диаграммы



<https://app.diagrams.net/>

<https://lucidchart.com/>

Шаг 5. Переходим от сущностей к таблицам

Одна сущность - одна таблица

Для каждой сущности выделяем **первичный ключ** (Primary Key, PK).

Естественный одиночный PK - это атрибут, значение которого:

- не может быть пустым
- уникально для каждого экземпляра сущности

Составной PK - состоит из комбинации атрибутов

Синтетический PK - искусственный атрибут с порядковой нумерацией

Шаг 6. Строим связи между таблицами

Правило 1. Для построения связи «**один ко многим**» нужно скопировать первичный ключ из одной главной таблицы в подчиненную.

Новое неуникальное поле в подчиненной таблице называется **внешним ключом** (Foreign Key, FK)

Правило 2. Для построения связи «**многие ко многим**» нужно создать дополнительную ассоциирующую таблицу и скопировать туда первичные ключи двух исходных таблиц в качестве внешних ключей.

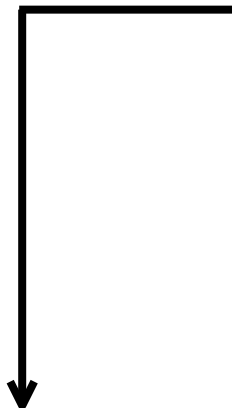
Правило 3. Для построения связи «**один к одному**» нужно скопировать первичный ключ из одной таблицы в другую, причем во второй таблице это поле должно стать первичным ключом.

Шаг 6. Строим связи между таблицами

Связь «Один ко многим»

Таблица «Кафедры»

Код кафедры (PK)	Название
100	Кафедра фундаментальной информатики
101	Кафедра систем автоматизированного проектирования
102	Кафедра математического анализа, алгебры и геометрии



Табельный номер (PK)	Код кафедры (FK)	Фамилия
1234	100	Иванов
1237	102	Петров
3456	100	Сидоров

Таблица «Сотрудники»

Шаг 6. Строим связи между таблицами

Таблица «Дисциплины»

Код дисциплины (PK)	Название
100	Базы данных
101	ООП
102	Языки программирования

Таблица «Студенты»

Код студента (PK)	Фамилия
200	Антонов
202	Белов
210	Волков

Связь «Многие ко многим»

Код дисциплины (FK)	Код студента (FK)
100	200
101	200
100	202

Шаг 6. Строим связи между таблицами

Таблица «Дисциплины»

Код дисциплины (PK)	Название
100	Базы данных
101	ООП
102	Языки программирования

Таблица «Студенты»

Код студента (PK)	Фамилия
200	Антонов
202	Белов
210	Волков

Связь «Многие ко многим»

Код дисциплины (FK)	Код студента (FK)
100	200
101	200
100	202

Если есть таблица, то должна быть и сущность!

Шаг 6. Строим связи между таблицами

Таблица «Дисциплины»

Код дисциплины (PK)	Название
100	Базы данных
101	ООП
102	Языки программирования

Таблица «Студенты»

Код студента (PK)	Фамилия
200	Антонов
202	Белов
210	Волков

Связь «Многие ко многим»

Таблица «Экзамены»

Код экзамена (PK)	Код дисциплины (FK)	Код студента (FK)	Дата	Баллы
100	100	200	25.05.2020	86
101	101	200	28.05.2020	58
210	100	202	30.05.2020	76

**Дополнительная
сущность**

Шаг 6. Строим связи между таблицами

Связь «один к одному» используется редко, т.к. две таблицы с одинаковым первичным ключом можно объединить.

Нужна для построения **отношения категоризации**.

Таблица «Оборудование»

Код единицы (PK)	Категория	Процессор	ОЗУ	ПЗУ	Формат	Тип
1	Компьютер	Intel Core i5	DDR3 8	SSD 500		
2	Компьютер	Intel Core i7	DDR4 8	HDD 1000		
3	Компьютер	AMD A10	DDR4 16	SSD 250		
4	Принтер				A4	Лазерный
5	Принтер				A4	Струйный
6	Принтер				A3	Лазерный

Шаг 6. Строим связи между таблицами

Таблица «Оборудование»

Код единицы (ПК)	Категория
1	Компьютер
2	Компьютер
3	Компьютер
4	Принтер
5	Принтер
6	Принтер

Связь «Один к одному»
для категоризации

Таблица «Компьютеры»

Код компьютера (ПК)	Процессор	ОЗУ	ПЗУ
1	Intel Core i5	DDR3 8	SSD 500
2	Intel Core i7	DDR4 8	HDD 1000
3	AMD A10	DDR4 16	SSD 250

Таблица «Принтеры»

Код принтера (ПК)	Формат	Тип
4	A4	Лазерный
5	A4	Струйный
6	A3	Лазерный

Шаг 7. Проверяем нормализацию

Первая НФ требует, чтобы каждое поле таблицы было неделимым (не должно делиться на более мелкие значения) и не содержало повторяющихся групп.

Таблица находится во **второй НФ**, если она находится в первой НФ и не содержит полей, которые зависят от части первичного ключа. Те поля, которые зависят только от части первичного ключа, должны быть выделены в отдельные таблицы.

Третья НФ требует, чтобы таблица была во второй НФ и в ней не было транзитивных зависимостей между неключевыми полями.

Шаг 7. Проверяем нормализацию

Первая НФ требует, чтобы каждое поле таблицы было неделимым (не должно делиться на более мелкие значения) и не содержало повторяющихся групп.

Если верно определены атрибуты сущностей и правильно построены связи, то первая нормальная форма не будет нарушаться.

Шаг 7. Проверяем нормализацию

Таблица находится во **второй НФ**, если она находится в первой НФ и не содержит полей, которые зависят от части первичного ключа. Те поля, которые зависят только от части первичного ключа, должны быть выделены в отдельные таблицы.

Если проектировать таблицу так, чтобы в ней был одиночный первичный ключ, то зависимость поля от части первичного ключа невозможна (у первичного ключа нет частей).

Поэтому если грамотно выделить сущности и их атрибуты, а потом выбрать одиночные первичные ключи для каждой сущности, то вторая нормальная форма выполняется автоматически.

Шаг 7. Проверяем нормализацию

Третья НФ требует, чтобы таблица была во второй НФ и в ней не было транзитивных зависимостей между неключевыми полями.

Если верно определены атрибуты сущностей и правильно построены связи, то третья нормальная форма не будет нарушаться.

Шаг 7. Проверяем нормализацию

Первая НФ требует, чтобы каждое поле таблицы было неделимым (не должно делиться на более мелкие значения) и не содержало повторяющихся групп.

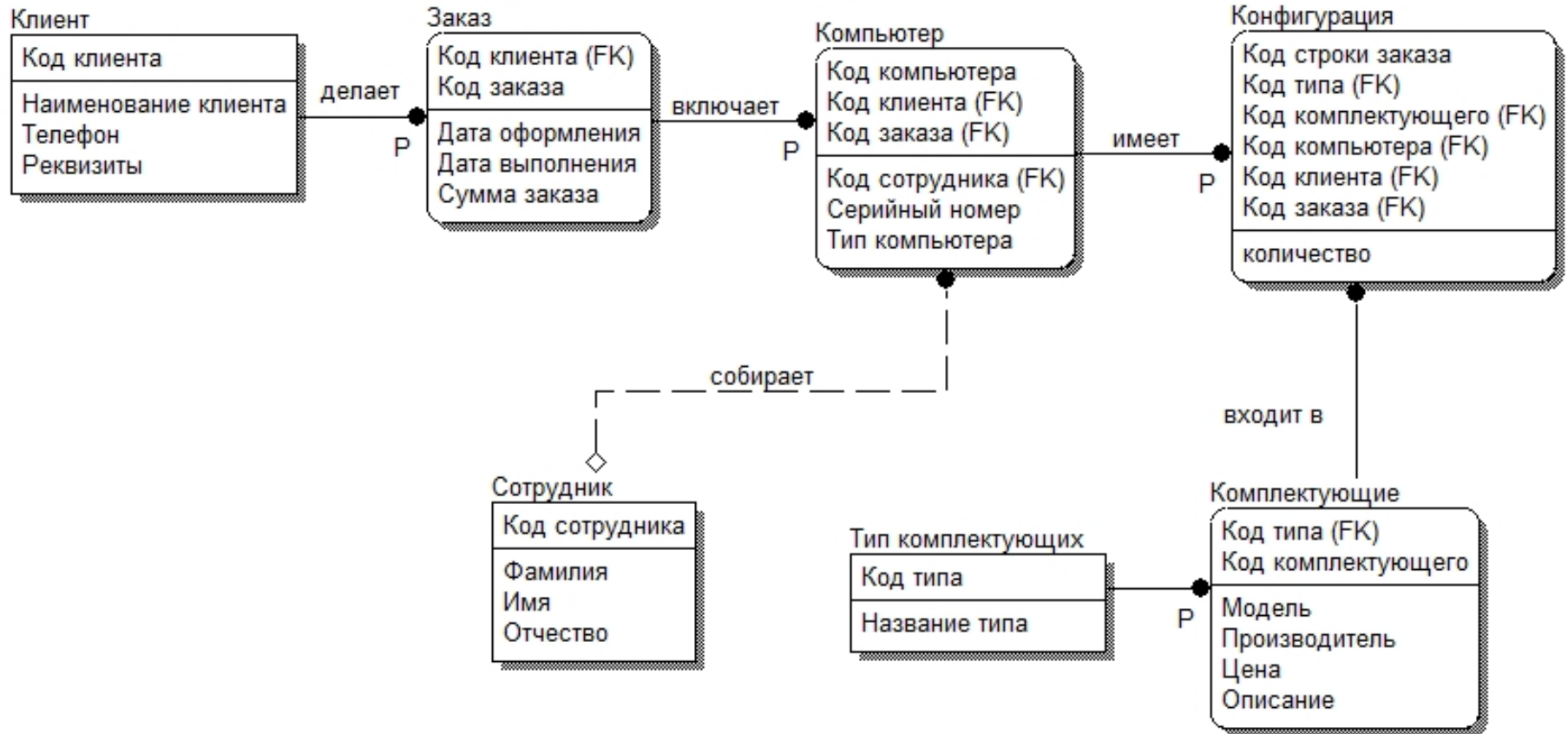
Таблица находится во **второй НФ**, если она находится в первой НФ и не содержит полей, которые зависят от части первичного ключа. Те поля, которые зависят только от части первичного ключа, должны быть выделены в отдельные таблицы.

Третья НФ требует, чтобы таблица была во второй НФ и в ней не было транзитивных зависимостей между неключевыми полями.

=====

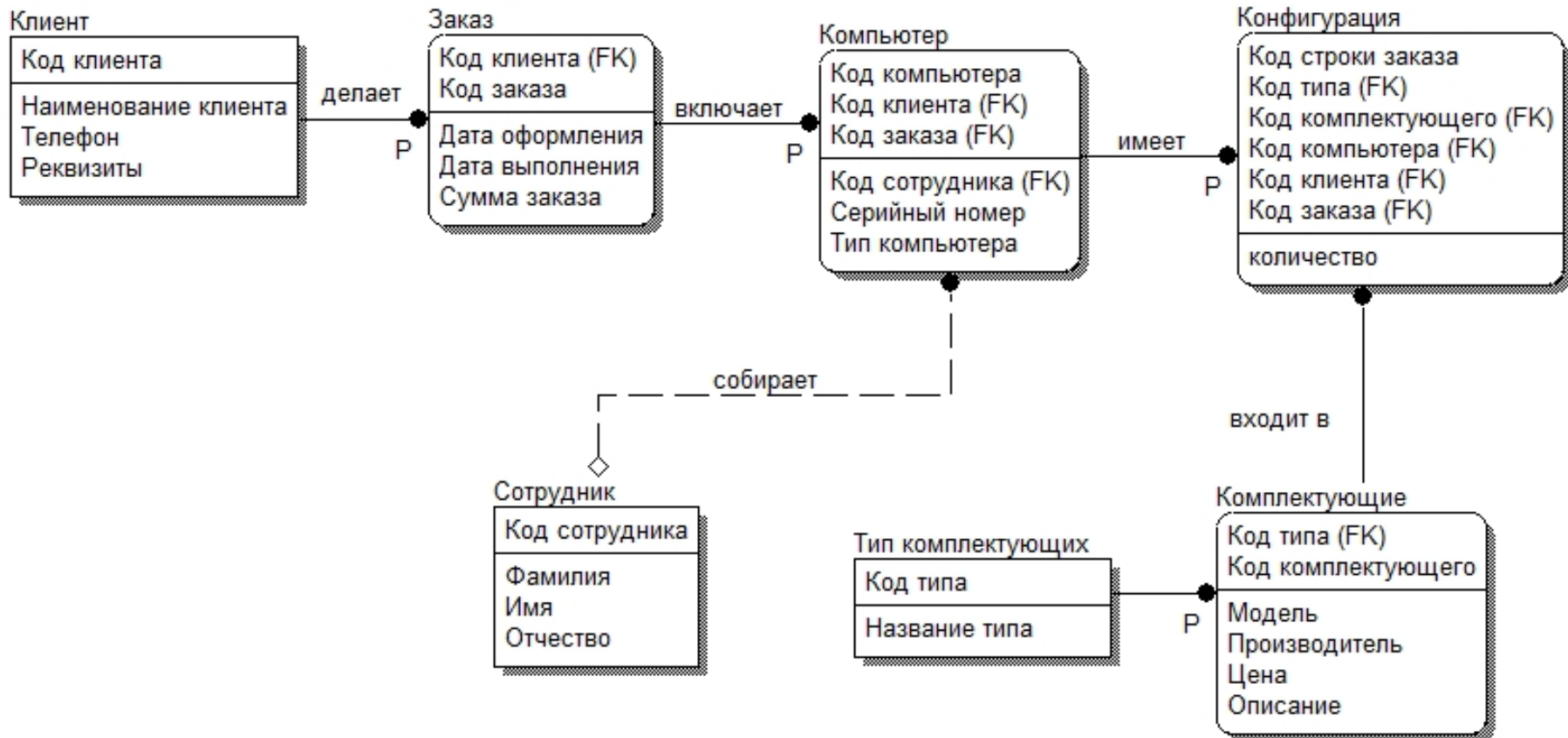
Нарушение второй или третьей НФ означает, что при проектировании была пропущена сущность или атрибуты были распределены по сущностям неправильно.

Результат шагов 5-7: диаграмма IDEF1X



Нотация IDEF1X

Зависимая (дочерняя) таблица изображается с закругленными углами.



Нотация IDEF1X

Виды связей:

- **Идентифицирующая.** Внешний ключ в дочерней таблице входит в состав первичного ключа. Дочерняя сущность не может существовать независимо от родительской.
- **Обязательная.** При вводе в новой строки в дочернюю таблицу заполнение полей внешнего ключа обязательно, причем введённые значения должны совпадать с полями первичного ключа какой-либо записи в родительской таблице. Экземпляр сущности не может существовать без связанного экземпляра другой сущности.
- **Необязательная.** Заполнение полей внешнего ключа в дочерней таблице необязательно или введённые значения могут не совпадать со значениями полей первичного ключа ни в одной записи родительской таблицы. Экземпляр сущности может существовать независимо, без связанного экземпляра другой сущности.

Нотация IDEF1X

	ИДЕНТИФИЦИРУЮЩАЯ	ОБЯЗАТЕЛЬНАЯ
О чём говорит?	"Я не могу существовать без родителя и его ID"	"Я должен быть связан с другим объектом"
Где видно?	В структуре первичного ключа	В кардинальности ($\text{min} \geq 1$)
Графика IDEF1X	Сплошная линия, квадрат	(1,N) или (1,1)
На практике	Для "слабых" сущностей (зависимых)	Для обеспечения целостности данных

💡 Простое правило:

- Идентифицирующая = «Я — часть другого объекта, мой ID включает его ID».
- Обязательная = «Я должен быть привязан к кому-то, чтобы существовать».

Нотация IDEF1X

1. Идентифицирующая и обязательная связь

Сущности:

- Заказ (OrderID PK)
- Позиция заказа (OrderID PK + LineNumber PK, Product , Qty)

Связь:

- Позиция заказа принадлежит ровно одному Заказу .
- Первичный ключ Позиции — составной: OrderID + LineNumber .

→ Это идентифицирующая связь:

- OrderID из Заказа входит в первичный ключ.
- Без Заказа позиция не существует.

→ Также обязательная: (1,N) — каждая позиция обязана относиться к заказу.

Нотация IDEF1X

2. Неидентифицирующая, но обязательная связь

Сущности:

- Клиент (CustomerID PK)
- Заказ (OrderID PK, CustomerID FK, Date)

Связь:

- Каждый Заказ обязан иметь Клиента .
- Но OrderID — собственный уникальный ключ, CustomerID — просто внешний ключ (не часть PK).

→ Связь неидентифицирующая:

- Заказ можно идентифицировать по OrderID независимо от клиента.

→ Но обязательная: (1,1) — не бывает заказа без клиента.

3. Неидентифицирующая и необязательная связь

Сущности:

- **Сотрудник** (**EmployeeID** PK)
- **Проект** (**ProjectID** PK)

Связь:

- Сотрудник может работать над проектом, но не обязан.
- Поле **CurrentProjectID** в таблице **Сотрудник** — может быть NULL.

→ Связь неидентифицирующая:

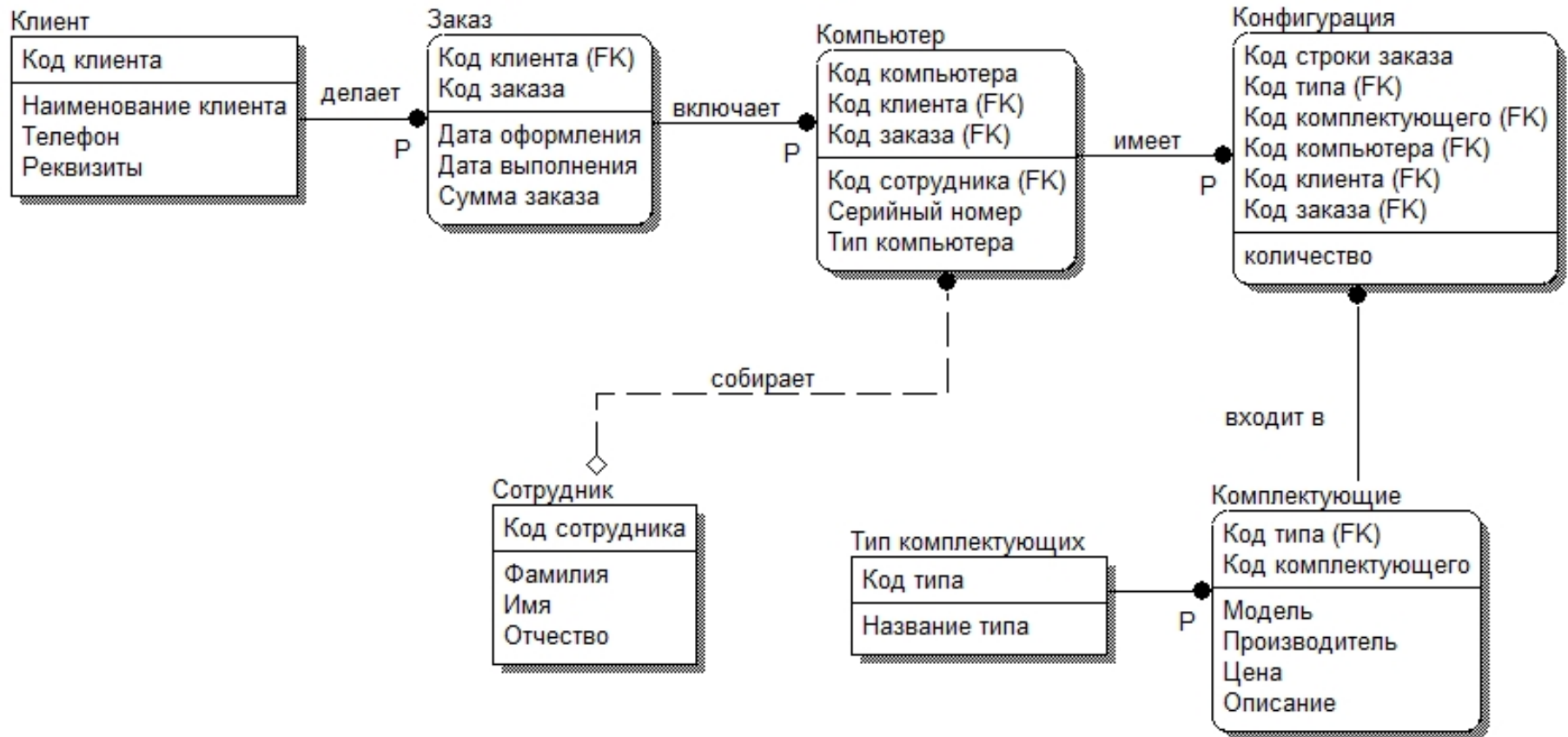
- **Сотрудник** имеет свой **EmployeeID** .

→ Необязательная: **(0,1)** — сотрудник может не участвовать ни в одном проекте.

Нотация IDEF1X

Внешний вид	Тип и обязательность связи	Мощность связи справа
—————	Обязательная, идентифицирующая	1
—————●	Обязательная, идентифицирующая	0 .. ∞
—————● Z	Обязательная, идентифицирующая	0 или 1
—————● P	Обязательная, идентифицирующая	1 .. ∞
—————● <число>	Обязательная, идентифицирующая	<число>
-----●	Обязательная, неидентифицирующая	0 .. ∞
◇-----●	Необязательная, неидентифицирующая	0 .. ∞

Результат шагов 5-7: диаграмма IDEF1X



Шаг 8. Определяемся с СУБД

- Выбираем СУБД, где будет создана база данных.
- Уточняем типы и размерности полей таблиц.
- Реализуем ограничения на значения полей и правила ссылочной целостности.
- Пишем SQL-скрипты создания таблиц для выбранной СУБД.

Методика проектирования реляционной БД

1. Изучаем процессы предметной области
 2. Определяем сущности и их атрибуты
 3. Устанавливаем связи между сущностями
 4. Определяем типы связей
 5. Переходим от сущностей к таблицам
 6. Строим связи между таблицами
 7. Проверяем нормализацию
 8. Определяемся с СУБД
-
- ER-диаграмма /
UML-диаграмма
- Диаграмма IDEF1x /
UML
- SQL-скрипты