Этапы проектирования баз данных:

1. Системный анализ предметной области
2. Инфологическое проектирование
3. Выбор СУБД
4. Датологическое проектирование
5. Физическое проектирование

1 Системный анализ предметной области

На первом этапе анализируются информационные потребности будущих пользователей баз данных. Рассматриваются формы входных и выходных потоков, которые будут составлять основу баз данных. Затем уточняются алгоритмы и процедуры обработки данных хранимой в базе данных. Формируются требования, которым должна удовлетворять проектируемая база данных и определяется примерный список объектов предметной области, свойства которых будут использоваться при разработке базы данных.

2 Инфологическое проектирование (концептуальное проектирование)

Инфологическое (концептуальное) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области» (например, «концептуальная модель базы данных» и «концептуальная модель предметной области»), поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

- описание информационных объектов, или понятий предметной области и связей между ними.
- описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

Существует два подхода к моделированию данных:
1. Модель «Сущность-связь»
2. Семантическая объектная модель

Эти модели представляют собой языки для описания структуры данных и их связей в представлениях пользователей. Моделирование данных, подобно блок-схемам, отражают логику программы.

Модель «Сущность-Связь».

Сущность – любой различимый объект (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет и т.д. Необходимо различать такие понятия, как тип сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое. Экземпляр сущности относится к

конкретной вещи в наборе. Например, типом сущности может быть ГОРОД, а экземпляром – Москва, Киев и т.д.

Атрибут – поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей (например, ЦВЕТ может быть определен для многих сущностей: СОБАКА, АВТОМОБИЛЬ, ДЫМ и т.д.). Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Примерами атрибутов для сущности АВТОМОБИЛЬ являются ТИП, МАРКА, НОМЕРНОЙ ЗНАК, ЦВЕТ и т.д. Здесь также существует различие между типом и экземпляром. Тип атрибута ЦВЕТ имеет много экземпляров или значений: Красный, Синий, Банановый, Белая ночь и т.д., однако каждому экземпляру сущности присваивается только одно значение атрибута. Абсолютное различие между типами сущностей и атрибутами отсутствует. Атрибут является таковым только в связи с типом сущности. В другом контексте атрибут может выступать как самостоятельная сущность. Например, для автомобильного завода цвет – это только атрибут продукта производства, а для лакокрасочной фабрики цвет – тип сущности.

Ключ – 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. Для сущности Расписание ключом является атрибут Номер_рейса или набор: Пункт_отправления, Время_вылета и Пункт_назначения (при условии, что из пункта в пункт вылетает в каждый момент времени один самолет).

Связь – ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Типы связей:
1). *Связь один к одному* (1:1) – одиночный экземпляр сущности одного типа связан с одиночным экземпляром сущности другого типа.
2). *Связь один ко многим* (1:M) – один экземпляр сущности связан со многими экземплярами другой сущности.
3). *Связь многие ко многим* (M:N) – несколько экземпляров одной сущности связаны с несколькими экземплярами другой сущности.

Модель «Сущность-Связь» или ER-диаграммы включают в себя изображения сущностей в виде прямоугольников, а связей в виде ромбиков. На ER-диаграммах атрибуты обозначаются эллипсами. Если атрибутов у сущности много, то чтобы не загружать ER-диаграмму, атрибуты помещают в прямоугольник, в котором идет перечисление всех атрибутов сущности.

Для объяснения терминов степень связи и класс принадлежности рассмотрим несколько примеров отношений между сущностями.

Сущность РАБОТНИК связана отношением ВОЗГЛАВЛЯЕТ с сущностью ОТДЕЛ. Будем считать, что работник может возглавлять только один отдел. С другой стороны, отдел может иметь только одного руководителя. Говорят, что такая связь имеет степень "один-к-одному"

Обозначение отношения "один-к-одному"

Сущность РАБОТНИК связана с сущностью ОТДЕЛ еще и отношением РАБОТАЕТ. В любом отделе может работать множество работников. Однако работник может работать только в одном конкретном отделе. Говорят, что такая связь имеет степень "один-ко-многим"

Обозначение отношения "один-ко-многим"

Сущность РАБОТНИК связана с сущностью ПРОФЕССИЯ отношением ИМЕЕТ. Любой работник может иметь несколько профессий. Разные работники могут иметь одну и ту же профессию. Отходя от норм русского языка, можно сказать, что "любая профессия может принадлежать разным работникам". Такое отношение называют "многие-ко-многим".

Обозначение отношения "многие-ко-многим"

Класс принадлежности связей:

Приведенные примеры объясняют значение характеристики "степень связи". Другой важной характеристикой связи является "класс принадлежности". Так, рассмотренному ранее отношению ВОЗГЛАВЛЯЕТ обязательно принадлежит сущность ОТДЕЛ, поскольку любой отдел должен иметь руководителя. Однако для сущности РАБОТНИК эта связь принадлежит необязательно. Так как не любой работник должен быть начальником отдела. С учетом этого уточнения графическое обозначение связи между ОТДЕЛОМ и РАБОТНИКОМ должно иметь вид.

Обозначение класса принадлежности связи

Все вышеприведенные примеры представляют бинарные связи. Иногда при анализе выявляются и более сложные типы связей.

Выявив все сущности, информация о которых должна быть представлена в БД, и описав связи между ними, мы имеем концептуальную модель БД.

3 Выбор СУБД

При выборе СУБД руководствуются следующими соображениями:
- аппаратное обеспечение, на котором в дальнейшем будет работать проектируемая база данных;
- системное программное обеспечение, с которым будет в последствии работать проектируемая база данных и соответствующее ей приложения;
- методология и подходы, к программированию реализованные в той или иной СУБД;
- модель данных, которая встроена в конкретную СУБД;

Выбор СУБД полностью определяется на II этапе построения базы данных, т. к. оно зависит от той модели данных, которая встроена в выбранную СУБД.

4 Датологическое проектирование (логическое)

Логическое проектирование базы данных - процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации.

Логическое проектирование является четвертым этапом проектирования базы данных, на котором происходит уточнение и преобразование концептуальной модели, созданной на предыдущем этапе, в логическую модель данных. Логическая модель данных создается на основе выбранной модели организации данных целевой СУБД (реляционная, сетевая, иерархическая или объектно-ориентированная), но этапе игнорирует все остальные характеристики выбранной СУБД, например, любые особенности физической организации ее структур хранения данных и построения индексов.

Переход от ER-модели к схеме реляционной базы данных

1. Каждая простая сущность превращается в таблицу (отношение). Имя сущности становится именем таблицы. Каждый простой атрибут становится столбцом таблицы с тем же именем.

2. Компоненты уникального идентификатора сущности превращаются в первичный ключ. Если имеется несколько возможных уникальных идентификаторов, выбирается наиболее используемый. Учитываются также следующие факторы:

- длина ключа – в качестве первичного ключа выбирается, как правило, самый короткий из вероятных ключей;
- стабильность – желательно выбирать в качестве первичного ключа атрибуты, которые не изменяются;
- мнемоничность – при прочих равных условиях следует отдавать предпочтение тем из вероятных ключей, которые легче запомнить.

Некоторые СУБД (Access, Paradox и др.) позволяют автоматически генерировать в качестве ключа таблицы поле типа «счетчик». Этот искусственный код можно использовать для простых объектов, если в предметной области не предполагается применение другой системы кодирования (ОКПО, ОКОНХ, ИНН).

Если в состав уникального идентификатора входят связи, то к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (процесс может продолжаться рекурсивно).

3. Строим отношения между сущностями по правилам, применяемым в выбранной СУБД.

4. Индексы создаются для первичного ключа (уникальный индекс), а также внешних ключей и тех атрибутов, которые будут часто использоваться в запросах.

Получение отношений из диаграммы ER-типа

1 Предварительные отношения бинарных связей 1:1

Правило 1. Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей является обязательным, то требуется только одно отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

Получаем отношение: **Преподаватель** (КП, Фамилия, Телефон, КД, Дисциплина, число часов), степень связи **1:1**, класс принадлежности **обязательный**

Таблица 1 Преподаватель

КП	Фамилия	Телефон	КД	Дисциплина	Число часов
П1	Иванов	23-45-66	Д1	ПОИС	48
П2	Андреев	23-33-67	Д2	Математика	56
П3	Суслов	22-67-85	Д3	Физика	34
П4	Репин	27-45-64	Д4	Информатика	68

Правило 2. Если степень бинарной связи 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение двух отношений. Под каждую сущность выделяется одно отношение, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которого класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности

Пример 1.

Класс принадлежности сущности **Преподаватель – *обязательный*,** а сущности **Дисциплина – *необязательный***

Получаем отношения: **Преподаватель**(КП, Фамилия, Телефон, КД), **Дисциплина**(КД, наименование, число часов)

Таблица 2. Преподаватель

КП	Фамилия	Телефон	КД
П1	Иванов	23-45-66	Д1
П2	Андреев	23-33-67	Д2
П3	Суслов	22-67-85	Д3
П4	Репин	27-45-64	Д4

Таблица 3. Дисциплина

КД	Дисциплина	Число часов
Д1	ПОИС	48
Д2	Математика	56
Д3	Физика	34
Д4	Информатика	68

Пример 2.

Класс принадлежности сущности **Преподаватель – не*обязательный*,** а сущности **Дисциплина – *обязательный***

Получаем отношения: **Преподаватель** (КП, Фамилия, Телефон), **Дисциплина**(КД, наименование, число часов, КП)

Таблица 4. Преподаватель

КП	Фамилия	Телефон
П1	Иванов	23-45-66
П2	Андреев	23-33-67
П3	Суслов	22-67-85
П4	Репин	27-45-64

Таблица 5. Дисциплина

КД	Дисциплина	Число часов	КП
Д1	ПОИС	48	П1
Д2	Математика	56	П2
Д3	Физика	34	П3
Д4	Информатика	68	П4

Правило 3. Если степень бинарной связи равна 1:1 и класс принадлежности ни одной из сущностей не является обязательным, то необходимо использовать три отношения: по одному для каждой сущности и одно отношение для связи. Причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

Класс принадлежности обеих сущностей **Преподаватель и Дисциплина – *необязательный***

Получаем отношения: **Преподаватель** (КП, Фамилия, Телефон), **Дисциплина**(КД, наименование, число часов, КП), **Читает** (КП, КД**)**

Таблица 6. Преподаватель

КП	Фамилия	Телефон
П1	Иванов	23-45-66
П2	Андреев	23-33-67
П3	Суслов	22-67-85
П4	Репин	27-45-64

Таблица 7. Дисциплина

КД	Дисциплина	Число часов
Д1	ПОИС	48
Д2	Математика	56
Д3	Физика	34
Д4	Информатика	68

Таблица 8. Читает

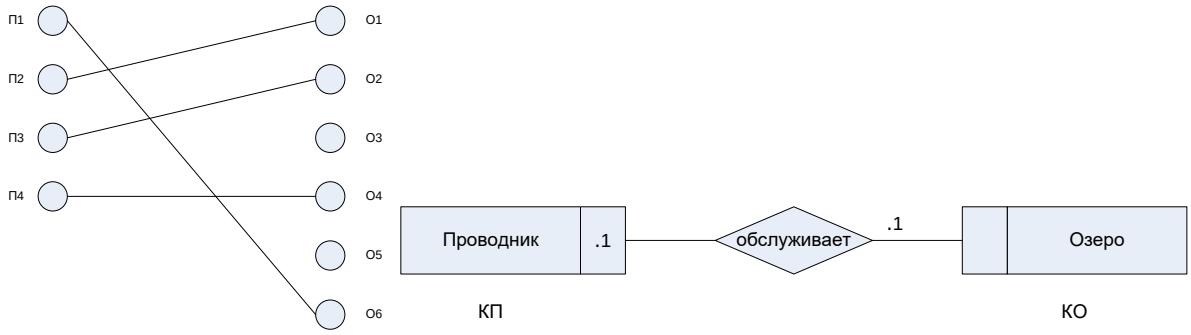
КП	КД
П1	Д1
П2	Д2
П3	Д3

Пример решения задачи.

Предметная область: профессиональные рыболовные проводники Мещерских озер и озера, которые они обслуживают. Разрешается закрепление не более одного проводника за одним озером, и по соглашению между проводниками каждый из них обслуживает только одно озеро. Таким образом, степень связи 1:1

Атрибуты: имя проводника, код проводника, номер телефона, ежедневная плата, максимально допустимое число людей в группе рыбаков (размер), название озера, код озера, рыболовный рейтинг и основной вид вылавливаемой в озере рыбы

Решение: Предположения перед построением ER-диаграммы: все проводники имеют работу, некоторые озера проводниками не обслуживаются. Следовательно, класс принадлежности сущности Проводник обязательный, а сущности Озеро необязательный.



Получает отношения: **Проводник** (<u>КП</u>, Фамилия, Телефон, Плата, Размер, КО),

Озеро (<u>КО,</u> Наименование, Рейтинг, Вид)

Таблица 9 Проводник

КП	Фамилия	Телефон	Плата	Размер	КО

Таблица 10 Озеро

КО	Наименование	Рейтинг	Вид

Заключение: из анализа диаграмм следует, что оба отношения находятся в НФБК. Детерминанты КП и КО являются ключевыми атрибутами.

2. Предварительные отношения бинарных связей 1:N

Правило 4. Если степень бинарной связи равна 1:N и класс принадлежности N-связной сущности является обязательным, то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ 1-связной сущности должен быть добавлен как атрибут в отношение, отводимое N-связной сущности.

Таблица 11 Преподаватель

КП	Фамилия	телефон
П1	Иванов	23-45-66
П2	Андреев	23-33-67
П3	Суслов	22-67-85
П4	Репин	27-45-64

Таблица 12 Дисциплина

КД	Дисциплина	Число часов	КП
Д1	ПОИС	48	П1
Д2	Математика	56	П2
Д3	Физика	34	П3
Д4	Информатика	68	П4

Правило 5: Если степень бинарной связи равна 1:N и класс принадлежности N-связной сущности является необязательным, то необходимо формирование трех отношений: по одному для каждой сущности и одно отношение для связи. Причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

Получаем отношения: **Преподаватель** (КП, Фамилия, Телефон), **Дисциплина**(КД, наименование, число часов), **Читает** (КП, КД**)**

Таблица 13 Преподаватель

КП	Фамилия	телефон
П1	Иванов	23-45-66
П2	Андреев	23-33-67
П3	Суслов	22-67-85
П4	Репин	27-45-64

Таблица 14 Дисциплина

КД	Дисциплина	Число часов
Д1	ПОИС	48
Д2	Математика	56
Д3	Физика	34
Д4	Информатика	68

Таблица 15 Читает

КП	КД
П1	Д1
П2	Д2
П3	Д3
П4	Д4

3. Предварительные отношения бинарных связей N:M

Правило 6. Если степень бинарной связи равна M:N, то для хранения данных необходимо три отношения: по одному для каждой сущности и одно отношение для связи. Причем ключ каждой сущности используется в качебстве первичного ключа соответствующего отношения. Отношение связи должно иметь в числе своих атрибутов ключи каждой сущности.

Получаем отношения: **Преподаватель** (КП, Фамилия, Телефон), **Дисциплина**(КД, наименование, число часов), **Читает** (КП, КД**)**

Таблица 16 Преподаватель

КП	Фамилия	телефон
П1	Иванов	23-45-66
П2	Андреев	23-33-67
П3	Суслов	22-67-85
П4	Репин	27-45-64

Таблица 17 Дисциплина

КД	Дисциплина	Число часов
Д1	ПОИС	48
Д2	Математика	56
Д3	Физика	34
Д4	Информатика	68

Таблица 18 Читает

КП	КД
П1	Д1
П2	Д2
П3	Д3
П4	Д4

Нормализация и нормальные формы

Нормализация - метод создания набора отношений с заданными свойствами на основе требований к данным, установленных в некоторой организации.

Процесс нормализации был впервые предложен Э. Ф. Коддом. Нормализация часто выполняется в виде последовательности тестов с целью проверки соответствия (или несоответствия) некоторого отношения требованиям заданной нормальной формы. Сначала были предложены только три вида нормальных форм: первая (1НФ), вторая (2НФ) и третья (3НФ). Затем Р. Бойсом и Э. Ф. Коддом было сформулировано более строгое определение третьей нормальной формы, которое получило название нормальной формы Бойса-Кодда (НФБК). Все эти нормальные формы основаны на функциональных зависимостях, существующих между атрибутами отношения, Нормальные формы более высокого порядка, которые превосходят НФБК, были введены позднее. К ним относятся четвертая (4НФ) и пятая (5НФ) нормальные формы. Но на практике эти нормальные формы более высоких порядков используются крайне редко.

Процесс нормализации является формальным методом, позволяющим определять отношения на основе их первичных или потенциальных ключей и функциональных зависимостей, существующих между их атрибутами. Проектировщики баз данных могут использовать нормализацию в виде наборов тестов, применяемых к отдельным отношениям с целью нормализации реляционной схемы до заданной конкретной формы, что позволит предотвратить возможное возникновение аномалий обновления.

Нормализация — это формальный метод анализа отношений на основе их первичного ключа (или потенциальных ключей) и существующих функциональных зависимостей. Он включает ряд правил, которые могут использоваться для проверки отдельных отношений таким образом, чтобы вся база данных могла быть нормализована до желаемой степени. Если некоторое требование не удовлетворяется, то противоречащее данному требованию отношение должно быть разделено на отношения, каждое из которых (в отдельности) удовлетворяет всем требованиям нормализации.

Чаще всего нормализация осуществляется в виде нескольких последовательно выполняемых этапов, каждый из которых соответствует определенной нормальной форме, обладающей известными свойствами. В ходе нормализации формат отношений становится все более ограниченным (строгим) и менее восприимчивым к аномалиям обновления.

При работе с реляционной моделью данных важно понимать, что для создания отношений приемлемого качества обязательно только выполнение требований первой нормальной формы (1НФ). Все остальные формы могут использоваться по желанию проектировщиков. Но для того чтобы избежать аномалий обновления, нормализацию рекомендуется выполнять как минимум до третьей нормальной формы (3НФ).

Первая нормальная форма (1НФ)

Ненормализованная форма (ННФ) - таблица, содержащая одну или несколько повторяющихся групп данных.

Первая нормальная форма (1НФ) - отношение, в котором на пересечении каждой строки и каждого столбца содержится одно и только одно значение.

Для преобразования ненормализованной таблицы в первую нормальную форму (1НФ) в исходной таблице следует найти и устранить все повторяющиеся группы данных. Повторяющейся группой называется группа, состоящая из одного или нескольких атрибутов таблицы, в которой возможно наличие нескольких значений для единственного значения ключевого атрибута (атрибутов) таблицы. Существуют два способа исключения повторяющихся групп из ненормализованных таблиц.

1. Повторяющиеся группы устраняются путем ввода соответствующих данных в пустые столбцы строк с повторяющимися данными, т.е. пустые места заполняются дубликатами неповторяющихся данных (этот способ частно называют "выравниванием" таблицы). Полученная таблица, теперь называемая отношением, содержит элементарные (или

единственные) значения на пересечении каждой строки с каждым столбцом и поэтому находится в 1НФ. Однако, полученное отношение имеет определенную избыточность данных, устраняемую в ходе дальнейшей нормализации.

2. Один атрибут или группа атрибутов назначаются ключом ненормализованной таблицы, а затем повторяющиеся группы изымаются и помещаются в отдельные отношения вместе с копиями ключа исходной таблицы, в новых отношениях устанавливаются свои первичные ключи. При наличии в ненормализованной таблице нескольких повторяющихся групп или повторяющихся групп, содержащихся в других повторяющихся группах, данный прием применяется до тех пор, пока повторяющихся групп совсем не останется. Полученный набор отношений будет находиться в первой нормальной форме только тогда, когда ни в одном из них не будет повторяющихся групп атрибутов. Данный способ находится в 1НФ и обладает меньшей избыточностью данных, чем первый способ.

При использовании первого подхода выровненное отношение 1НФ раскладывается в ходе дальнейшей нормализации на те же отношения, которые могли быть сразу же получены с помощью второго подхода.

Пример приведения таблицы к первой нормальной форме Исходная, ненормализованная, таблица:

Сотрудник	Номер телефона
Иванов И. И.	283-56-82 390-57-34
Петров П. Ю.	708-62-34

Таблица, приведённая к 1NF:

Сотрудник	Номер телефона
Иванов И. И.	283-56-82
Иванов И. И.	390-57-34
Петров П. Ю.	708-62-34

Вторая нормальная форма (2НФ)

Полная функциональная зависимость: если A и B - атрибуты отношения, то атрибут B находится в полной функциональной зависимости от атрибута A, если атрибут B является функционально зависимым от A, но не зависит ни от одного собственного подмножества атрибута A.

Функциональная зависимость A->B является полной функциональной зависимостью, если удаление какого-либо атрибута из A приводит к утрате этой зависимости. Функциональная зависимость A->B называется частичной, если в A есть некий атрибут, при удалении которого эта зависимость сохраняется.

Вторая нормальная форма применяется к отношениям с составными ключами, т.е. к таким отношениям, первичный ключ которых состоит из двух или нескольких атрибутов. Дело в том, что отношение с первичным ключом на основе единственного атрибута всегда находится, по крайней мере, в форме 2НФ. Отношение, которое не находится в форме 2НФ, может быть подвержено аномалиям обновления.

Вторая нормальная форма (2НФ) - отношение, которое находится в первой нормальной форме и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

Нормализация отношений 1НФ с приведением к форме 2НФ предусматривает устранение частичных зависимостей. Если в отношении между атрибутами существует частичная

зависимость, то функционально-зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

Пример приведения таблицы ко второй нормальной форме

Пусть Начальник и Должность вместе образуют первичный ключ в такой таблице:

Начальник	Должность	Зарплата	Наличие компьютера
Гришин	Кладовщик	20000	Нет
Васильев	Программист	40000	Есть
Васильев	Кладовщик	25000	Нет

Зарплату сотруднику каждый начальник устанавливает сам, но её границы зависят от должности. Наличие же компьютера у сотрудника зависит только от должности, то есть зависимость от первичного ключа неполная.

В результате приведения к 2NF получаются две таблицы:

Начальник	Должность	Зарплата
Гришин	Кладовщик	20000
Васильев	Программист	40000
Васильев	Кладовщик	25000

Здесь первичный ключ, как и в исходной таблице, составной, но единственный не входящий в него атрибут Зарплата зависит теперь от всего ключа, то есть полно.

Должность	Наличие компьютера
Кладовщик	Нет
Программист	Есть

Ясно, что отношение, находящееся в 1НФ, также может обладать избыточностью. Для её устранения предназначена вторая нормальная форма. Но прежде чем приступить к её описанию, сначала следует выявить недостатки первой.

Пусть исходное отношение содержит информацию о поставке некоторых товаров и их поставщиках.

Код поставщика	Город	Статус города	Код товара	Количество
ООО ЗАРЯ	Москва	20	Молоко	300
ООО ЗАКАТ	Ставрополь	20	Колбаса	400
ООО РАСВЕТ	Ярославль	20	Конфеты	100
ООО РАСВЕТ	Ярославль	10	Кефир	200
ООО ЗАРЯ	Москва	30	Сыр	300
ООО ЗАРЯ	Москва	30	Сыр	400
ООО ЗАКАТ	Ставрополь	15	Молоко	100

Заранее известно, что в этом отношении содержатся следующие функциональные зависимости:
{ {Код поставщика, Код товара} -> { Количество},
{Код поставщика} -> {Город},
{Код поставщика} -> {Статус},
{Город} -> {Статус} }

Первичный ключ в отношении: {Код поставщика, Код товара}.

Очевидно, что отношение обладает избыточностью: оно описывает две сущности — поставку и поставщика. В связи с этим возникают следующие аномалии:

- Аномалия вставки. В отношение нельзя добавить информацию о поставщике, который ещё не поставил ни одного товара.
- Аномалия удаления. Если от поставщика была только одна поставка, то при удалении информации о ней будет удалена и вся информация о поставщике.
- Аномалия обновления. Если необходимо изменить какую-либо информацию о поставщике (например, поставщик переехал в другой город), то придётся изменять значения атрибутов во всех записях о поставках от него.

Физический смысл избыточности исходного отношения заключается в том, что оно описывает *не одну* сущность, а *две — поставку* и *поставщика*.

Чтобы устранить эти аномалии, необходимо разбить исходное отношение на проекции:

1. В первую следует включить первичный ключ и все неключевые атрибуты *явно* зависимые от него.
2. В остальные проекции (в данном случае она одна) будут включены неключевые атрибуты, зависящие от первичного ключа *неявно*, вместе с той *частью* первичного ключа, от которой эти атрибуты зависят явно.

В итоге будут получены два отношения:

Код поставщика	**Код товара**	**Количество**
ООО ЗАРЯ	Молоко	300
ООО ЗАКАТ	Колбаса	400
ООО РАСВЕТ	Конфеты	100
ООО РАСВЕТ	Кефир	200
ООО ЗАРЯ	Сыр	300
ООО ЗАРЯ	Сыр	400
ООО ЗАКАТ	Молоко	100

Первому отношению теперь соответствуют следующие функциональные зависимости:
{Код поставщика, Код товара} -> {Количество}

Код поставщика	**Город**	**Статус города**
ООО ЗАРЯ	Москва	20
ООО ЗАКАТ	Ярославль	10
ООО РАСВЕТ	Ставрополь	30

Второму отношению соответствуют:
{ {Код поставщика} -> {Город},
{Код поставщика} -> {Статус},
{Город} -> {Статус} }

Такое разбиение устранило аномалии, описанные выше: можно добавить информацию о поставщике, который ещё не поставлял товар, удалить информацию о поставке без удаления информации о поставщике, а также легко обновить информацию в случае если поставщик переехал в другой город.

Теперь можно сформулировать определение второй нормальной формы, до которого, скорее всего, читатель уже смог догадаться самостоятельно: отношение находится **во второй нормальной**

форме (сокращённо 2НФ) тогда и только тогда, когда оно находится в первой нормальной форме и каждый его неключевой атрибут неприводимо зависим от первичного ключа.

Третья нормальная форма (3НФ)

Транзитивная зависимость: 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).

Транзитивная зависимость является одним из типов функциональной зависимости.

Третья нормальная форма (3НФ) - отношение, которое находится а первой и во второй нормальных формах и не имеет атрибутов, не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависит от этого первичного ключа.

Нормализация отношений 2НФ с образованием отношений 3НФ предусматривает устранение транзитивных зависимостей. Если в отношении существует транзитивная зависимость между атрибутами, то транзитивно зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

Определения второй (2НФ) и третьей (3НФ) нормальных форм, приведенные выше, не допускают наличия частичных или транзитивных зависимостей от первичного ключа отношения, поскольку только при этом условии можно избежать аномалий обновления. Но в этих определениях не рассматриваются другие потенциальные ключи отношения, даже если они существуют. Приведем общие определения форм 2НФ и 3НФ, в которых учитываются потенциальные ключи отношения.

Следует отметить, что реализация этого требования не влечет за собой корректировку определения формы 1НФ, поскольку эта нормальная форма не зависит от ключей и функциональных зависимостей. В качестве общих определений укажем, что атрибут первичного ключа входит в состав любого потенциального ключа и что частичные, полные и транзитивные зависимости рассматриваются с учетом всех потенциальных ключей отношения.

Вторая нормальная форма (2НФ) - отношение, находящееся в первой нормальной форме, в котором каждый атрибут, отличный от атрибута первичного ключа, является полностью функционально независимым от любого потенциального ключа.

Третья нормальная форма (3НФ) - отношение, находящееся в первой и второй нормальной форме, в котором ни один атрибут, отличный от атрибута первичного ключа, не является транзитивно зависимым ни от одного потенциального ключа.

При использовании этих общих определений форм 2НФ и 3НФ необходимо убедиться в отсутствии частичных и транзитивных зависимостей от всех потенциальных ключей, а не только от первичного ключа. Такое требование может повлечь за собой усложнение процесса нормализации, но эти общие определения налагают дополнительные ограничения на отношения и могут позволить выявить скрытую избыточность в отношениях, которая в ином случае могла остаться незамеченной.

Необходимо найти компромисс между стремлением к максимальному упрощению процесса нормализации путем исследования зависимостей только от первичных ключей, что позволяет выявить лишь наиболее обременительную и очевидную избыточность в отношениях, и тенденцией к использованию общих определений для повышения вероятности выявления скрытой избыточности. Но на практике чаще всего результаты декомпозиции являются одинаковыми,

независимо от того, используются ли определения форм 2НФ и 3НФ, основанные на первичных ключах, или общие определения, приведенные в настоящем разделе.

Пример приведения таблицы к третьей нормальной форме (Википедия)

Исходная таблица:

Фамилия	Отдел	Телефон
Гришин	1	11-22-33
Васильев	1	11-22-33
Петров	2	44-55-66

В результате приведения к 3NF получаются две таблицы:

Фамилия	Отдел
Гришин	1
Васильев	1
Петров	2

Отдел	Телефон
1	11-22-33
2	44-55-66

Ссылочная целостность

Ссылочной целостностью называют особый механизм, осуществляемый средствами СУБД или программистом, ответственный за поддержание непротиворечивых данных в связанных релятивными отношениями таблицах. Ссылочная целостность подразумевает, что в таблицах, имеющих релятивные связи, нет ссылок на несуществующие записи. Если мы удалим из списка студента Иванова И.И., и при этом не изменим таблицу со сданными экзаменами, ссылочная целостность будет нарушена, в таблице с экзаменами появится "мусор" - данные, на которые не ссылается ни одна запись из таблицы студентов. Ссылочная целостность будет нарушена.

Таким образом, если мы удаляем из списка студента Иванова И.И., следует позаботиться о том, чтобы из таблицы со сданными экзаменами также были удалены все записи, на которые ранее ссылалась удаленная запись главной таблицы. Существует несколько видов изменений данных, которые могут привести к нарушению ссылочной целостности:

1. Удаляется запись в родительской таблице, но не удаляются соответствующие связанные записи в дочерней таблице.
2. Изменяется запись в родительской таблице, но не изменяются соответствующие ключи в дочерней таблице.
3. Изменяется ключ в дочерней таблице, но не изменяется значение связанного поля родительской таблицы.

Многие СУБД блокируют действия пользователя, которые могут привести к нарушению связей. Нарушение хотя бы одной такой связи делает информацию в БД недостоверной. Если мы, например, удалили Иванова И.И., то теперь номер 1 принадлежит Петрову П.П.. Имеющиеся связи указывают, что он сдал экзамены по математике и физике, но не сдавал экзаменов по русскому языку и литературе. Достоверность данных нарушена. Конечно, в таких случаях в качестве ключа обычно используют счетчик - поле автоинкрементного типа. Если удалить запись со значением 1, то другие записи не изменят своего значения, значение 1 просто невозможно будет присвоить

какой-то другой записи, оно будет отсутствовать в таблице. Путаницы в связях не случится, однако все равно подчиненная таблица будет иметь "потерянные" записи, не связанные ни с какой записью главной таблицы. Механизм ссылочной целостности должен запрещать удаление записи в главной таблице до того, как будут удалены все связанные с ней записи в дочерней таблице.

Пример проектирования БД

Задача: спроектировать БД справочник абонентов.

1. Системный анализ предметной области

Область данных достаточно простая, поэтому подробный анализ проводить не будем, а выделим основные моменты:
Справочник содержит основную информацию об абоненте: ФИО, адрес проживания, дата рождения, возраст, пол, список телефонов абонента, тип телефона, фото.
Справочник может содержать дополнительную информацию об абоненте: адрес проживания.
Основные действия в телефонном справочнике тоже просты: нужно добавлять, редактировать, удалять записи, иметь возможность просмотра и поиска записей.

2. Инфологическое проектирование

Информационные потребности пользователя (анализ запросов).

При разработке данного примера была выбрана следующая предметная область: «Телефонный справочник».

В ней необходимо отразить:

- Список абонентов
- Список телефонов абонента
- Возможность ввода, редактирование, удаление информации
- Возможность поиска
- Возможность фильтрации информации
- Смена адреса абонента

Определение сущностей и связей

Сущность - это объект, информация о котором должна быть представлена в базе данных. Экземпляр сущности - это информация о конкретном представителе объекта.

Связь - соединение между двумя и более сущностями. Экземпляр связи - конкретная связь между конкретными представителями объектов.

Сущности, представленные в данном примере:

- **Абонент** (содержит информацию об абоненте)
- **Телефон** (список телефонов абонента)
- **Адрес** (адрес абонента: улица, дом, квартира)

Разработка инфологической модели предметной области.

Определение функций пользователя, атрибутов, ключей

Атрибут - свойство сущности или связи.

Ключ сущности - атрибут или набор атрибутов, используемый для однозначной идентификации экземпляра сущности.

Ключи и атрибуты, в данном курсовом проекте:

Сущность **Абонент**. Содержит следующие атрибуты: Фамилия Имя Отчество, Пол, Дата рождения, Возраст, фото. Фамилия Имя Отчество является ключом.

Сущность **Телефон**. Содержит следующие атрибуты: Телефон, тип телефона. Телефон является ключом.

Сущность **Адрес**. Содержит следующие атрибуты: Улица, дом, квартира. Улица является ключом.

Выявление и описание ограничений целостности

Под целостностью данных, как понимаются ссылочные ограничения, т.е. те ограничения, которые нужно соблюдать для сохранения целостности связи между таблицами, в случае если в них будут изменяться или удаляться записи.

Для обеспечения целостности данных в Access есть 4 варианта:

1. Если не указано каскадное обновление связей, то предотвращается изменение значений первичного ключа в главной таблице, если существуют связанные записи в подчиненной таблице.
2. Если указано каскадное обновление связей, то при изменении значений первичного ключа будут изменяться соответствующие значения в связанной таблице.
3. Если не указано каскадное удаление связанных записей, то предотвращается удаление связанных записей из главной таблицы, если имеются связанные с ней записи в подчиненной.
4. Если указано каскадное удаление, то связанные записи подчиненной таблицы удаляются автоматически.

В данном примере у связи Абонент-Телефон и Абонент-Адрес установлены такие параметры как каскадное обновление и каскадное удаление связей, у связи Адрес-Улица установлен только параметр как каскадное обновление, а каскадное удаление связей нет (Мы не можем удалить из справочника улицу, если там проживают абоненты). Свойства этих параметров описаны выше.

Также к ограничениям целостности можно отнести ограничения на столбец и на таблицу, а точнее на значения данных в них. К таким можно отнести следующие ограничения:

Запрещение null значения: данные, заносимые в столбец или таблицу, не должны равняться нулю.

Ограничения на допустимые значения полей: условие, которому должны удовлетворять данные, вносимые в таблицу.

Ограничение первичного ключа: на практике рекомендуется для каждой таблицы создавать первичный ключ, особенностью которого является не допуск null значения.

Ограничение уникальных ключей: необходимость ввода различных (уникальных) данных.

В данном курсовом проекте используются следующие ограничения данных в таблицах:

Таблица **Абонент**

В поле **Дата рождения** на данные накладывается ограничение не моложе 18 лет и старше 100 лет. Ограничение мягкое позволяющее при необходимости ввести нужные данные.

В поле **Пол** на данные накладывается ограничение на значение только М и Ж.

В поле **Дом** на данные накладывается ограничение на значение >0 и <1000.

В поле **Квартира** на данные накладывается ограничение на значение >0 и <1000.

Поле **Возраст** доступно только для чтения и рассчитывается.

Таблица **Телефон**

В поле **Телефон** данные вводить по формату правильного написания номера телефона.

В поле **Тип телефона** на данные накладывается ограничение на значение домашний, мобильный, служебный.

3. Выбор СУБД

4. Датологическое проектирование

5. Физическое проектирование