

Севастопольский государственный университет
Кафедра «Информационные системы»

Управление данными

курс лекций

лектор:
ст. преподаватель кафедры ИС Абрамович А.Ю.



Лекция 4

ЖИЗНЕННЫЙ ЦИКЛ БАЗЫ ДАННЫХ.

Логическое проектирование.
Нормализация

Проектирование базы данных

Полный цикл разработки базы данных включает **концептуальное, логическое и физическое проектирование.**

Цель второй фазы проектирования базы данных состоит в создании **логической модели данных** для исследуемой предметной области.

На этом этапе уже должно быть известно, какая СУБД будет использоваться в качестве целевой - реляционная, сетевая, иерархическая или объектно-ориентированная, **но игнорируются все остальные характеристики выбранной СУБД**, например, любые особенности физической организации ее структур хранения данных и построения индексов.

Фаза логического проектирования предполагает следующие действия:

- преобразование концептуальной модели данных в логическую модель, в результате которого будет определена схема реляционной модели данных;
- проверка модели с помощью концепций последовательной нормализации;
- проверка поддержки целостности данных.

Проектирование базы данных

Логическая модель **позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями.**

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity Relationship Diagram, ERD);
- модель данных, основанная на ключах (Key Based model, KB);
- полная атрибутивная модель (Fully Attributed model, FA).

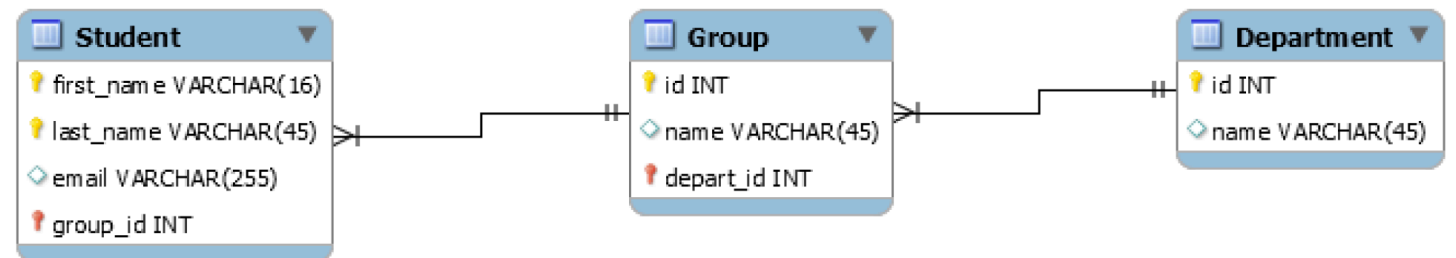
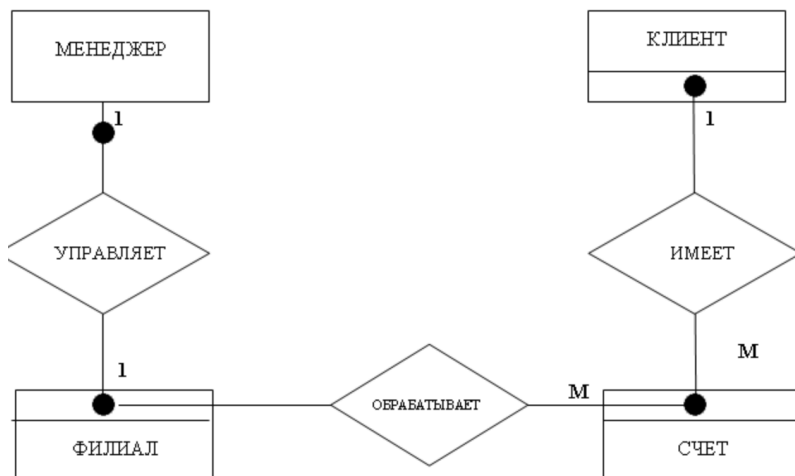


Диаграмма сущность-связь (ERD)

представляет собой модель данных верхнего уровня. Она **включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области**. Диаграмма сущность-связь **может включать связи «многие-ко-многим» и не включать описание ключей**. Основными понятиями ER-модели являются **сущность, связь и атрибут**.

Для построения ER-диаграммы **необходимо выделить основные сущности и составить сложно-сетевую структуру**.



Для дальнейшего построения ER-диаграммы необходимо **избавиться от связи М:М, перейдя к простой сетевой структуре**.

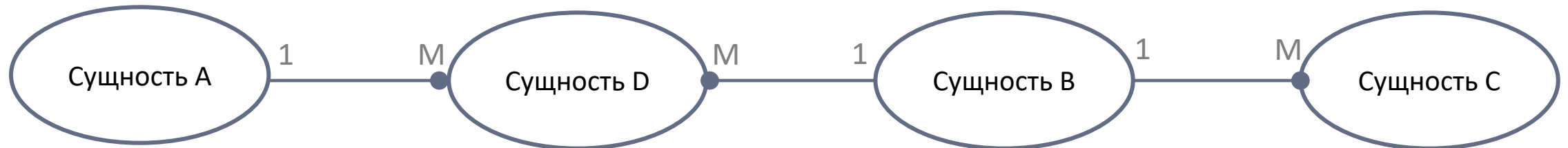
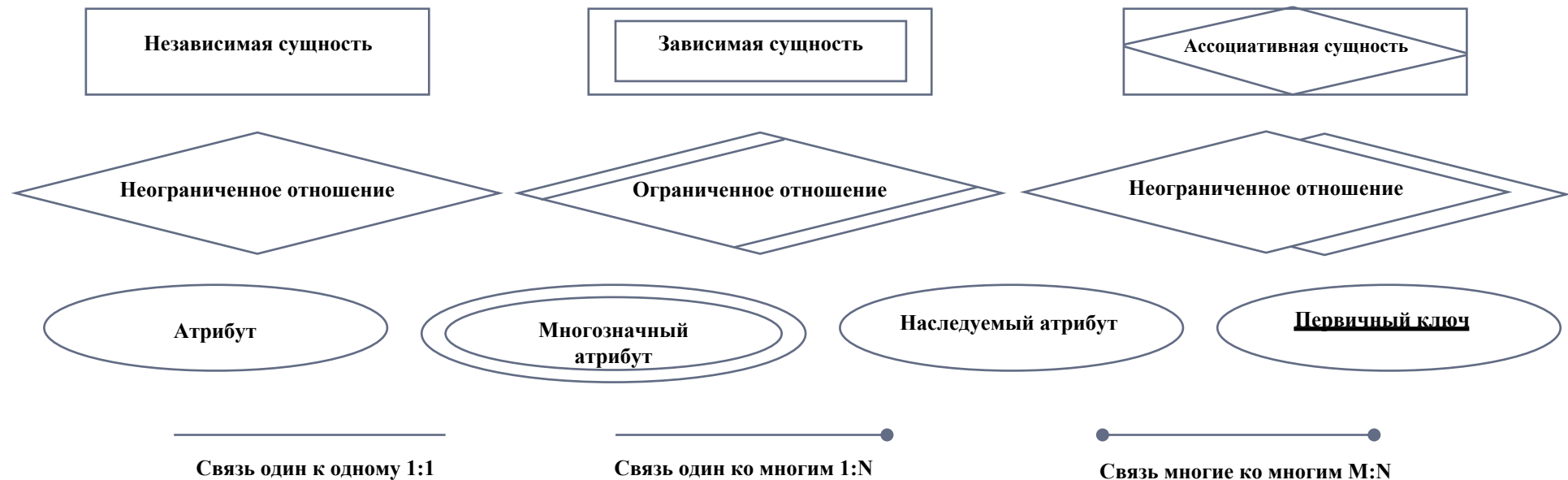


Диаграмма сущность-связь (ERD)

Нотация ERD была впервые введена Питером Ченом (Chen) и получила дальнейшее развитие в работах Ричарда Баркера. Диаграммы «сущность-связь» – ERD (Entity-Relationship Diagrams) (case-метод Баркера) являются наиболее распространенной методологией моделирования данных. С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.



Этапы построения ERD-диаграммы

На первом шаге производится определение сущностей. Для построения ER-диаграммы «с нуля» производится анализ предметной области и выделяются информационные объекты проектируемой системы, другими словами составляется список (пул) потенциальных сущностей (как правило, выделяются все существительные в текстовом описании предметной области).

На втором шаге необходимо описать каждый информационный объект набором характеристик (атрибутов), которые представляют важность с точки зрения выполняемых системой функций, то есть из списка потенциальных сущностей выделяются сущности, а остальное преобразуется в атрибуты сущностей.

На третьем шаге устанавливаются отношения и связи между сущностями по описанию предметной области на естественном языке. Определяются виды отношений и типы связей.

На четвертом шаге из списка атрибутов выделить атрибуты, способные однозначно идентифицировать экземпляры сущности, то есть определить первичные ключи.

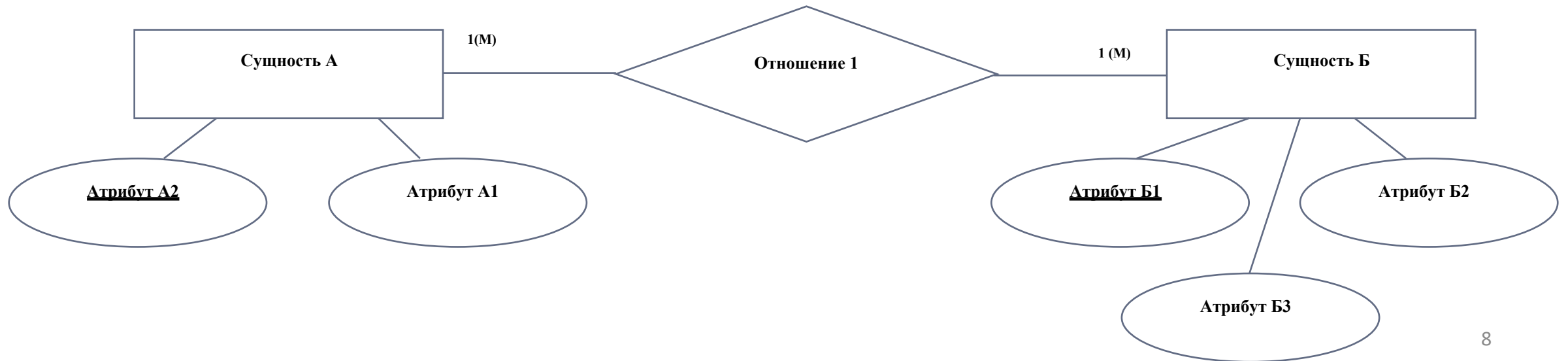
На пятом шаге строится ER-диаграмма.

Правила и рекомендации для построения ERD-диаграммы

Каждая сущность должна обладать **уникальным идентификатором**. Каждый экземпляр сущности должен **однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности**. Каждая сущность должна обладать некоторыми свойствами:

- иметь **уникальное имя**; к одному и тому же имени должна всегда применяться **одна и та же интерпретация**; одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- иметь **один или несколько атрибутов**, которые либо принадлежат сущности, либо наследуются через связь;
- иметь один или несколько атрибутов, которые **однозначно идентифицируют каждый экземпляр сущности** первичный ключ (PrimaryKey).

Каждая сущность может обладать **любым количеством связей** с другими сущностями модели.

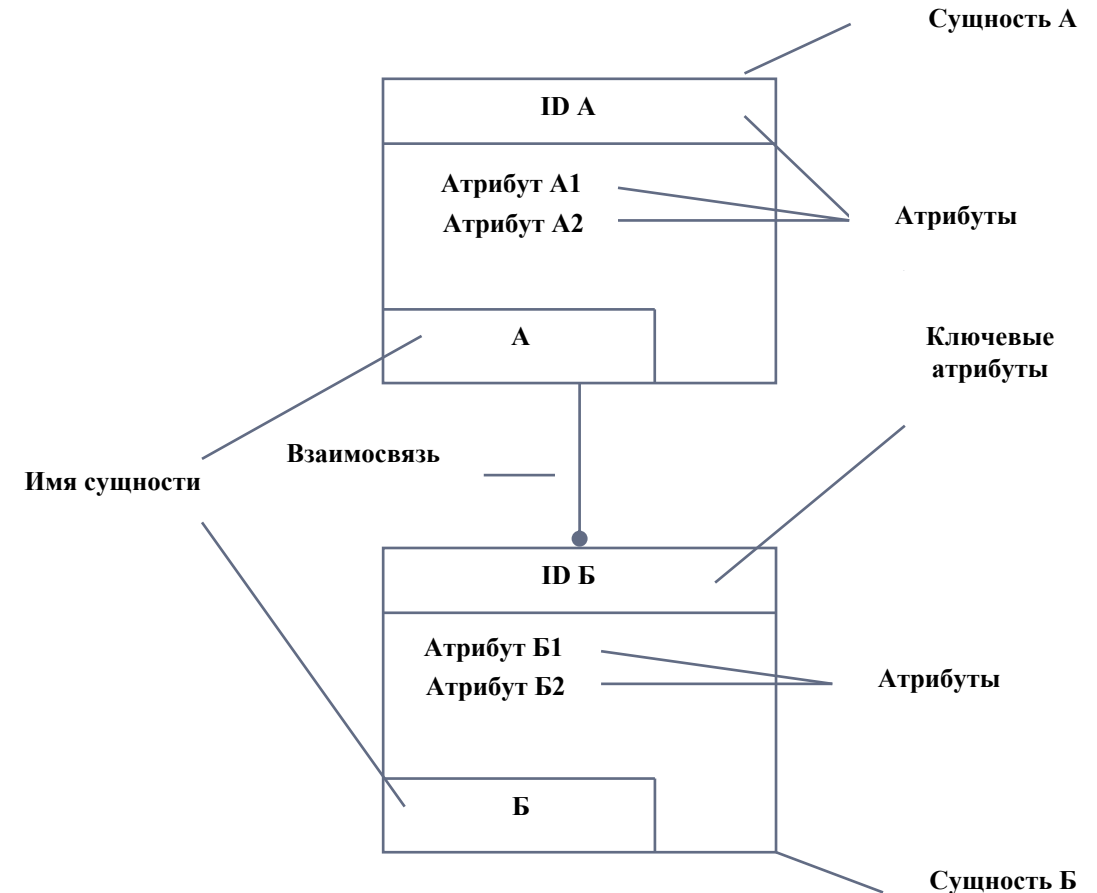


Модель основанная на ключах (Key Based model, KB)

это логическая модель, включающая описание всех сущностей и ключевых атрибутов, которые соответствуют предметной области. Она даёт более подробное представление о данных, включает описание всех сущностей и первичных ключей. **Главная задача модели данных**, основанной на ключах, является представление структуры данных и ключей, которые соответствуют предметной области.

Модель показывает ту же область что и область ERD, но, вместе с тем, отображает больше деталей.

Методология IDEF1 разработанная Рэмеем, основана на подходе Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

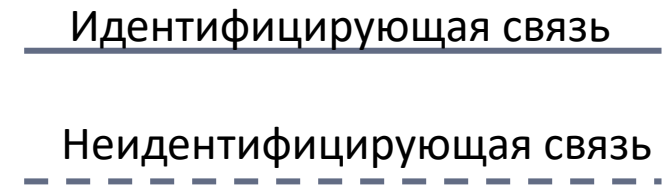
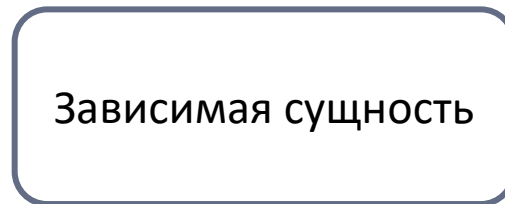
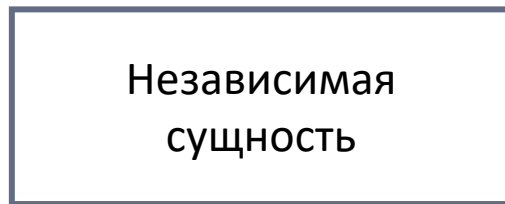


Полная атрибутивная модель в нотации IDEF1X

IDEF1X – предназначена для построения концептуальной схемы логической структуры реляционной базы данных, которая была бы независимой от программной платформы её конечной реализации. По сравнению с IDEF1 она дополнительно оперирует рядом понятий, правил и ограничений, такими как домены, представления, первичные, внешние и суррогатные ключи и другими, пришедшими из реляционной алгебры.

Использование **метода IDEF1X** наиболее целесообразно для построения логической структуры базы данных после того как все информационные ресурсы исследованы и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

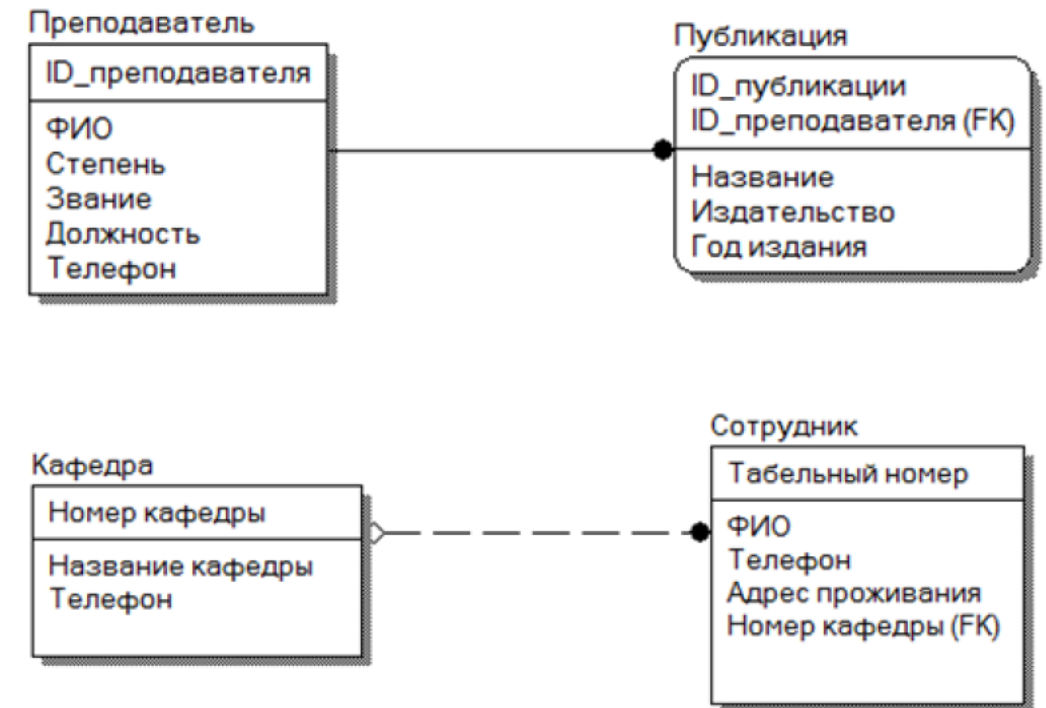
Данная методология описывает способы изображения двух типов сущностей – **независимой и зависимой**, и связей – **идентифицирующих и неидентифицирующих**.



Полная атрибутивная модель в нотации IDEF1X

В IDEF1X концепция зависимых и независимых сущностей усиливается **типом взаимосвязей между двумя сущностями**. Если требуется, чтобы **внешний ключ передавался в дочернюю сущность** (и, в результате, создавал зависимую сущность), то необходимо создать **идентифицирующую связь** между родительской и дочерней сущностью.

Неидентифицирующие связи, являющиеся уникальными для IDEF1X, также связывают родительскую сущность с дочерней. Неидентифицирующие связи используются для отображения другого типа передачи атрибутов внешних ключей – **передача в область данных дочерней сущности (под линией)**.



Полная атрибутивная модель в нотации IDEF1X

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER, является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая, несомненно, является значительным недостатком ERD.

Полная атрибутивная модель достигается нормализацией отношений до третьей или четвертой нормальной формы.

Для описания нормальных форм требуются следующие определения:

- **функциональная зависимость в отношении R:** атрибут Y функционально зависит от атрибута X (X и Y могут быть составными) в том и только в том случае, если каждому значению X соответствует в точности одно значение Y: $R.X \rightarrow R.Y$.
- **полная (неприводимая) функциональная зависимость:** функциональная зависимость $R.X \rightarrow R.Y$ называется полной, если атрибут Y не зависит функционально от любого точного подмножества X.
- **транзитивная функциональная зависимость:** функциональная зависимость $R.X \rightarrow R.Y$ называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости $R.X \rightarrow R.Z$ и $R.Z \rightarrow R.Y$ и отсутствует функциональная зависимость $R.Z \rightarrow R.X$.
- **многозначные зависимости в отношении R(A, B, C)** существует многозначная зависимость $R.A \twoheadrightarrow R.B$ в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

Нормализация

Целью нормализации базы данных является сокращение избыточности. Нормализация предполагает последовательное приведение схемы базы данных к так называемым нормальным формам, каждая последующая из которых предъявляет более строгие требования по сравнению с предыдущей.

Условия нахождения отношений в нормальных формах:

1НФ. Отношение R находится в первой нормальной форме, если выполняется условие атомарности значений атрибутов отношения.

2НФ. Отношение R находится во второй нормальной форме в том и только в том случае, когда находится в 1НФ, и каждый неключевой атрибут полностью зависит от первичного ключа.

3НФ. Отношение R находится в третьей нормальной форме в том и только в том случае, если находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Такое определение 3НФ оказывается несостоятельным при условии, что:

1. Отношение имеет два или более возможных ключа.

2. Эти потенциальные ключи являются составными.

3. Два или более потенциальных ключа перекрываются, т.е. имеют общие атрибуты.

Поэтому вводят понятие нормальной **формы Бойса-Кодда, или усиленной 3НФ.**

БКНФ. Отношение R находится в нормальной форме Бойса-Кодда в том и только в том случае, если каждый детерминант является возможным ключом.

Нормализация

Пример. Дана реляционная схема, включающая следующие отношения:

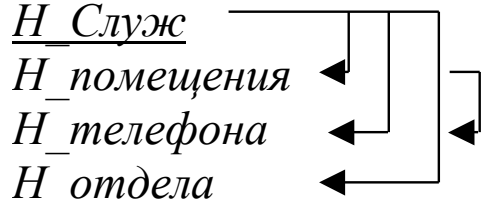
Отдел(Н_Отд, Размер бюджета, Н_руководителя)

Служащий(Н_Служ, Н_помещения, Н_телефона, Н_отдела)

Помещение(Н_помещения, Площадь, Н_телефона)

1НФ. Все выделенные атрибуты атомарны, следовательно, схема находится в 1НФ.

2НФ. Все неключевые атрибуты зависят полностью от ключей, следовательно, схема находится в 2НФ.

3НФ.  Из зависимостей между атрибутами отношения *Служащий* видно, что имеется транзитивная зависимость:
 $N_Служ \rightarrow N_помещения \rightarrow N_телефона.$

Для устранения транзитивных зависимостей выполняется проекция, в результате получаются 2 отношения:

(Н_служащего, Н_помещения, Н_отдела)

(Н_помещения, Н_телефона)

Второе из них входит в состав отношения *Помещение*, поэтому его можно удалить.

В результате получается схема:

Отдел(Н_Отд, Размер бюджета, Н_руководителя),

Служащий(Н_Служ, Н_помещения, Н_отдела),

Помещение(Н_помещения, Площадь, Н_телефона),

Нормализация

Пример БКНФ. Пусть есть отношение вида:

Служащие-Проекты(Н_служащего, Имя_служащего, Н_проекта, Работа_Служащего).

Возможными ключами отношения являются следующие пары атрибутов:

Н_служащего, Н_проекта;

Имя_служащего, Н_проекта.

В отношении есть следующие функциональные зависимости:

Н_служащего \rightarrow Имя_служащего;

Н_служащего \rightarrow Н_проекта;

Имя_служащего \rightarrow Н_служащего;

Имя_служащего \rightarrow Н_проекта;

Н_служащего, Н_проекта \rightarrow Работа_Служащего;

Имя_служащего, Н_проекта \rightarrow Работа_Служащего.

Предполагается, что Имя_служащего так же уникально, как и Н_служащего.

Видно, что отношение Служащие-Проекты находится в 3НФ.

Тот факт, что имеются функциональные зависимости атрибутов отношения от атрибута, являющегося частью первичного ключа, приводит к аномалиям.

Нормализация

Если вынести связь $\text{Н_служащего} \rightarrow \text{Имя_Служащего}$ в отдельное отношение, то получится два отношения:

1. Служащие(Н_Служащего , Имя_служащего).

Возможные ключи:

Н_Служащего ,

Имя_служащего .

Зависимости:

$\text{Н_Служащего} \rightarrow \text{Имя_служащего}$;

$\text{Имя_служащего} \rightarrow \text{Н_Служащего}$.

2. Служащие_Проекты(Н_служащего , Н_проекта , Работа_Служащего).

Возможный ключ: Н_служащего , Н_проекта .

Зависимости: $(\text{Н_служащего}, \text{Н_проекта}) \rightarrow \text{Работа_Служащего}$.

Такая схема находится в БКНФ.