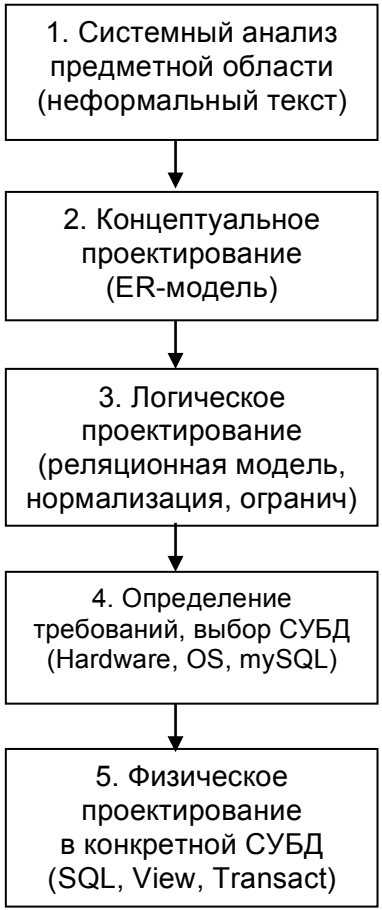
Концептуальное и логическое проектирование реляционных БД.



Процесс проектирования баз данных - анализ того, какого рода информация должна в ней представлена и каковы взаимосвязи между элементами информации. Структура или схема базы определяется средствами языков или систем обозначений пригодных для описания проектов. По завершению уточнений и согласований проект преобразовывается в форму, которая может быть воспринята конкретной выбранной СУБД – инструкции языка DDL (Data Definition Language).

В настоящее время существует несколько подходов к проектированию БД. Самый распространенный и традиционный подход ER-моделирование БД. ER-модель Модель (Entity-Relationship или Сущность-Связь) – это графическая модель, которая была предложена в 1976г. Питером Пин-Шэн Ченом.

Большинство современных коммерческих БД являются реляционными, т.е. данные представлены в виде специальных таблиц (отношений), связанных между собой.

Исторически ER-модель появилась позже других моделей представления данных, поэтому часть определений понятий в ER-модели заимствована. Для лучшего понимания материала методически более эффективно будет сохранить исторический порядок изложения теории построения БД.

Сначала рассмотрим математические основы реляционной модели, затем рассмотрим ER-моделирование, далее преобразование ERD в реляционную модель, затем нормализацию и только потом поговорим об этапе анализа предметной области.

1. Математические основы реляционной модели данных.

Реляционная модель данных основывается на математических принципах, которые вытекают из теории множеств и логики предикатов.

Эти принципы впервые были применены в области моделирования данных в конце 1960-х гг. доктором Е.Ф. Коддом, а впервые опубликованы - в 1970 г. Техническая статья "Реляционная модель данных для больших разделяемых банков данных" доктора Е.Ф. Кодда, опубликованная в 1970 г., является родоначальницей современной теории реляционных БД.

Тогда же появились первые прототипы реляционных систем управления базами данных (СУБД), но долгое время считалось невозможным добиться эффективной реализации таких систем.

Постепенное накопление методов и алгоритмов организации реляционных баз данных и управления ими привели к тому, что уже в середине 80-х годов реляционные системы практически вытеснили с мирового рынка ранние СУБД.

1.1. Правила Кодда.

https://ru.wikipedia.org/wiki/12_%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB_%D0%9A%D0%BE%D0%B4%D0%B4%D0%B0

Доктор Кодд определил 13 правил реляционной модели, но, счёт начал с 0.

0. Реляционная СУБД должна быть способна полностью управлять базой данных через ее реляционные возможности.
1. Информационное правило - вся информация в реляционной БД (включая имена таблиц и столбцов) должна определяться строго как значения в таблицах.
2. Гарантированный доступ - любое значение в реляционной БД должно быть гарантированно доступно для использования через комбинацию имени таблицы, значения первичного ключа и имени столбца
3. Поддержка пустых значений (null value) - СУБД должна уметь работать с пустыми значениями (неизвестными или неиспользованными значениями), в отличие от значений по умолчанию и независимо для любых доменов.
4. Онлайновый реляционный каталог - описание БД и ее содержания должны быть представлены на логическом уровне как таблицы, к которым можно применять запросы, используя язык базы данных.
5. Исчерпывающий язык управления данными - по крайней мере, один из поддерживаемых языков должен иметь четко определенный синтаксис и быть всеобъемлющим. Он должен поддерживать описание структуры данных и манипулирование ими, правила целостности, авторизацию и транзакции.
6. Правило обновления представлений (views) - все представления, теоретически обновляемые, могут быть обновлены через систему.

7. Вставка, обновление и удаление - СУБД поддерживает не только запрос на отбор данных, но и вставку, обновление и удаление
8. Физическая независимость данных - на программы-приложения и специальные программы логически не влияют изменения физических методов доступа к данным и структур хранилищ данных.
9. Логическая независимость данных - изменения структур таблиц, в пределах разумного, на программы-приложения и специальные программы логически не влияют.
10. Независимость целостности - язык БД должен быть способен определять правила целостности. Они должны сохраняться в онлайновом справочнике, и не должно существовать способа их обойти.
11. Независимость распределения - на программы-приложения и специальные программы логически не влияет, первый раз используются данные или повторно.
12. Неподрывность - невозможность обойти правила целостности, определенные через язык базы данных, использованием языков низкого уровня.

Кодд предложил применение реляционной алгебры в СУРБД, для разбиения данных в связанные наборы. Он организовал свою систему БД вокруг теории, основанной на наборах данных.

Формулируя принципы реляционной модели, доктор Кодд выбрал термин "отношение" (relation), т.к. он считал, что этот термин однозначен (в то время как, например, термин "таблица" имеет множество различных видов - таблица в тексте, электронная таблица и т.д.). Весьма распространено следующее заблуждение: реляционная модель названа так потому, что она определяет связи между таблицами. На самом деле, название этой модели происходит от отношений (таблиц базы данных), лежащих в ее основе.

1.2. Операции реляционной алгебры.

1.2.1. Терминология.

В основе РМД лежит понятие **отношения**, представляющего собой подмножество декартова произведения доменов. Элементы отношения называют **кортежами,** элементы кортежа – **атрибутами** (полями). Длина кортежа (количество атрибутов) определяет *арность отношения*, количество кортежей – *мощность отношения*.

Обращение к таблице происходит по имени. Каждый атрибут также имеет имя, принадлежит к определённому типу данных и характеризуется размером памяти, выделяемой под его хранение. Перечень атрибутов отношения с их типами и размерами называется **схемой отношения**. Отношения, построенные по одинаковой схеме, называют **односхемными**; по разным схемам – **разносхемными**.

На атрибут (группу атрибутов) могут накладываться **ограничения целостности**, т.е. правила, которым должно соответствовать значение атрибута (или соотношение значений атрибутов).

В стандартах SQL используются другие термины: отношение принято называть ***таблицей***, кортеж – ***строкой***, а атрибут – ***столбцом*** таблицы.

Каждая таблица хранит данные об одном типе объекта (сущности) предметной области, причём одна строка таблицы содержит данные об одном экземпляре объекта данного типа. Например, таблица *СТУДЕНТЫ* может хранить данные обо всех студентах института, а отдельная строка этой таблицы – данные о конкретном студенте.

1.2.2. Основные операции реляционной алгебры.

Данные в отношениях обрабатываются с помощью операций реляционной алгебры.

Операции реляционной алгебры (РА) применимы к реляционным отношениям (таблицам). Результатом выполнения операции реляционной алгебры также является отношение, построенное на основе одного или более исходных отношений. Существует пять основных операций РА и три вспомогательных, которые могут быть выражены через основные.

1. Проекция (project).

Это унарная операция (выполняемая над одним отношением), служащая для выбора подмножества атрибутов (столбцов) из отношения R. Она уменьшает арность результирующего отношения (количество столбцов) и может уменьшить мощность результирующего отношения (количество строк) за счёт исключения одинаковых кортежей.

Пример. Для отношение R(A,B,C), проекция πA,C(R) (по атрибутам A и C) будет как на рис.:

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

Проекция πA,C(R)

A	C
a	c
c	d

2. Селекция (select).

Это унарная операция, результатом которой является подмножество кортежей исходного отношения, соответствующих условиям, которые накладываются на значения одного или нескольких атрибутов. Арность результирующего отношения (количество столбцов) в итоге селекции не меняется.

Пример. Для отношения R(A,B,C), селекция σC=d(R) (при условии "значение атрибута C равно величине d") будет такой как на рис.:

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

Селекция σC=d(R)

A	B	C
c	a	d
c	b	d

3. Декартово произведение (cartesian product).

Это бинарная операция над разносхемными отношениями, соответствующая определению декартова произведения для РМД. Кортежи результирующего отношения состоят из всех атрибутов исходных отношений.

Пример. Для отношений R(A,B,C) и S(D,E), декартово произведение R×S будет как на рис.:

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

Отношение S

D	E
1	3
2	4

Декартово произведение R×S

A	B	C	D	E
a	b	c	1	3
a	b	c	2	4
c	a	d	1	3
c	a	d	2	4
c	b	d	1	3
c	b	d	2	4

4. Объединение (union).

Объединение – бинарная операция над односхемными отношениями результатом является отношение, которое включает в себя все кортежи обоих исходных отношений без повторов; причём, имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей и типы данных соответствующих полей.

Пример. Для отношений R(A,B,C) и S(A,D,E), объединение R U S будет таким как на рис.:

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

Отношение S

A	D	E
c	a	e
c	b	d

Объединение R U S

A	D	F
a	b	c
c	a	d
c	b	d
c	a	e

5. Разность (except).

Разность – бинарная операция над односхемными отношениями, результатом является отношение включающее множество кортежей R, не входящих в S; причём, имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей и типы данных соответствующих полей.

Пример. Для отношений R(A,B,C) и S(A,D,E), разность R–S будет такой как на рис.:

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

Отношение S

A	D	E
g	h	a
a	b	c
c	a	e

Разность R - S

A	D	F
c	a	d
c	b	d

1.3.2. Вспомогательные операции реляционной алгебры.

6. Пересечение (intersect).

Пересечение – бинарная операция над односхемными отношениями. Пересечение односхемных отношений R и S есть подмножество кортежей, принадлежащих обоим отношениям; причём, имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей и типы данных соответствующих полей.

Пересечение можно выразить через разность так: $R \cap S = R - (R - S)$.

Пример. Для отношений R(A,B,C) и S(A,D,E), разность R–S будет такой как на рис.:

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

Отношение S

A	D	E
g	h	a
a	b	c
c	a	e

Пересечение R ∩ S

A	D	F
a	b	c

7. Соединение (join).

Соединения – бинарная операция над разносхемными отношениями. Эта операция определяет подмножество декартова произведения двух разносхемных отношений. Кортеж декартова произведения входит в результирующее отношение, если выполняется условие соединения F, которое задаёт соотношение значений атрибутов разных таблиц.

Соединение можно выразить через селекцию так: $R>_F<S=\sigma_F (R\times S)$.

Если условием является равенство значений двух атрибутов исходных отношений, такая операция называется **эквисоединением**.

Естественным называется эквисоединение по одинаковым атрибутам исходных отношений.

Пример. Для отношений R(A,B,C) и S(A,D,E), естественное соединение R >< S будет как на рис., условие F не пишется, подразумевается равенство атрибутов:

Отношение R

A	B	C
a	b	c
c	a	d
c	h	c
g	b	d

Отношение S

A	D	E
g	h	a
c	b	c
h	d	d

Естествен. Соединение R >< S

A	B	C	D	E
c	a	d	b	c
c	h	c	b	c
g	b	d	h	a

8. Деление (division).

Деление – бинарная операция над разносхемными отношениями. Пусть отношение R содержит атрибуты $\{r_1,r_2,...,r_k, r_{k+1},...,r_n\}$, а отношение S – атрибуты $\{r_{k+1},...,r_n\}$. Тогда результирующее отношение R/S содержит атрибуты $\{r_1,r_2,...,r_k\}$. Кортеж отношения R включается в результирующее отношение R/S, если его декартово произведение с отношением S входит в R.

Деление можно выразить через проекцию так: $R / S = \pi_{r1,...,rk}(R) - \pi_{r1,...,rk}((\pi_{r1,...,rk}(R) \times S) - R)$.

Пример. Для отношений R(A,B,C) и S(A,B), частное R/S будет таким как показано на рис.

Отношение R

A	B	C	D
a	b	c	b
a	b	g	h
c	f	g	h
c	f	c	b
a	v	c	b
c	v	g	h

Отношение S

C	D
c	b
g	h

Частное R / S

A	B
a	b
c	f

2. Концептуальная модель данных.

Концептуальная модель данных, или концептуальное описание предметной области - начальный уровень проектирования баз данных.

С точки зрения теории реляционных БД, основные принципы реляционной модели на концептуальном уровне можно сформулировать следующим образом:

- все данные представляются в виде упорядоченной структуры, определенной в виде строк и столбцов и называемой отношением;
- все значения являются скалярами. Это означает, что для любой строки и столбца любого отношения существует одно и только одно значение;
- все операции выполняются над целым отношением, и результатом их выполнения также является целое отношение. Этот принцип называется замыканием.

Каждая строка отношения, содержащая данные, называется кортежем, каждый столбец отношения называется атрибутом (на уровне практической работы с современными реляционными БД используются термины "таблица", "запись" и "поле").

2.1. Элементы ER-модели.

ER-модель отображается графически, в виде диаграммы сущностей и связей (entity-relationship diagram - ERD) состоящих из множеств:

- Сущности
- Атрибуты
- Связи

Любой фрагмент предметной области может быть представлен как **множество сущностей**, между которыми существует некоторое **множество связей.**

Элементами описания реляционной модели данных на концептуальном уровне являются сущности**,** атрибуты**,** домены **и связи**.

Сущность - некоторый обособленный объект или событие, информацию о котором необходимо сохранять в базе данных, имеющий определенный набор свойств - атрибутов. Сущности могут быть как физические (реально существующие объекты: например, СТУДЕНТ с атрибутами - № зачетной книжки, фамилия, его факультет, специальность, № группы и т.д.), так и абстрактные (например, ЭКЗАМЕН с атрибутами - дисциплина, дата, преподаватель, аудитория и пр.).

Для сущностей различают ее тип и экземпляр. Тип сущности характеризуется именем и списком свойств, а экземпляр - конкретными значениями свойств.

Атрибуты сущности бывают:

- **Идентифицирующие и описательные**. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными.
- **Простые и составные**. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам и доменам данных (например, адрес). Решение об использовании составного атрибута или разбиении его на компоненты может быть связано с обеспечением высокой скорости работы с большими БД.
- **Однозначные и многозначные** - могут иметь соответственно одно или много значений для каждого экземпляра сущности.
- **Основные и производные**. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст человека вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности - множества значений (или домена), которые может принимать данный атрибут.

Домен - это набор всех допустимых значений, которые может содержать атрибут. Понятие "домен" часто путают с понятием "тип данных". Необходимо различать эти два понятия.

Тип данных - это физическая концепция, а домен - логическая. Например, "целое число" - это тип данных, а "возраст" - это домен.

Связи - на концептуальном уровне представляют собой простые ассоциации между сущностями. Например, утверждение "Покупатели приобретают продукты" указывает, что между сущностями "ПОКУПАТЕЛЬ" и "ПРОДУКТ" существует связь.

Спецификация связи. Каждая связь в реляционной модели характеризуется именем, типом, обязательностью, степенью и, возможно, атрибутами.

- **Типы связей (кардинальность).** Существует несколько типов связей между двумя сущностями: это связи "один к одному", "один ко многим" и "многие ко многим".
- **Клас связи (обязательность).** Если сущность одного типа оказывается по необходимости связанной с сущностью другого типа, то между этими типами объектов существует обязательная связь. Иначе связь является факультативной (необязательной).
- **Степень связи** определяется количеством сущностей, которые охвачены данной связью.
- Пример бинарной связи - связь работают между сущностями ОТДЕЛ и СОТРУДНИК.
- Пример тернарной связи – связь *экзаменуют(ся)* между сущностями *ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ*.
- **Атрибуты связи.** Из последнего примера видно, что связь также может иметь атрибуты (в данном случае для связи экзаменуются - это *Дата экзамена* и *Оценка*).

2.2. Построение ER-диаграммы для схемы базы данных.

Диаграмма "сущности-связи" (Entity-Relationship Diagrams, или ERD) служит для описания схемы базы на концептуальном уровне проектирования. Метод был предложен в 1976 г. Питером Пин Шань Ченом (Peter Pin Shan Chen).

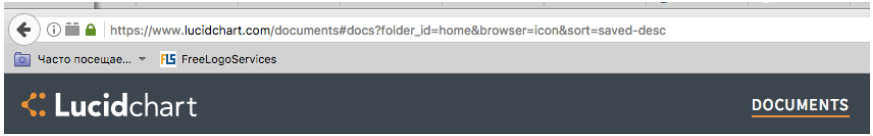
В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.).

При разработке ER-модели можно выбрать любую нотацию и даже использовать некоторую **собственную смешанную нотацию** удобную разработчику.

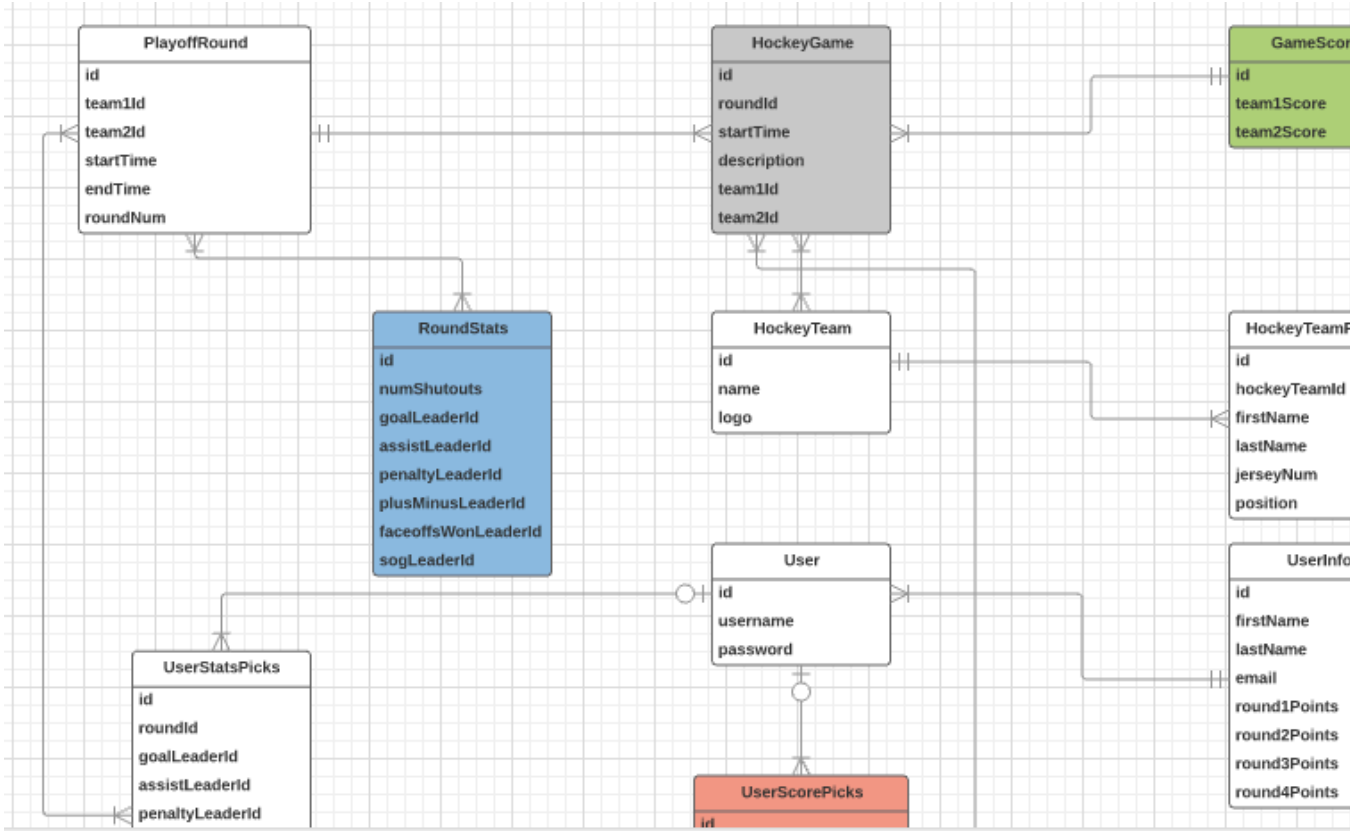
По сути, все варианты диаграмм "сущность-связь" исходят из одной идеи - рисунок всегда нагляднее текстового описания. Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Различные программные средства, реализующие даже одну и ту же нотацию, могут отличаться своими возможностями. Рекомендуется использовать один из инструментов создания ERD, например:

- Draw.io (draw.io),
- LucidChart (lucidchart.com).

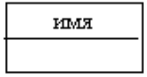
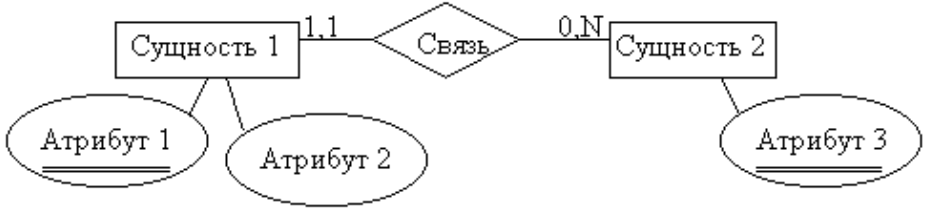


Пример ERD из LucidChart в нотации близкой к нотации Мартина.

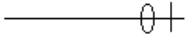
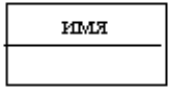
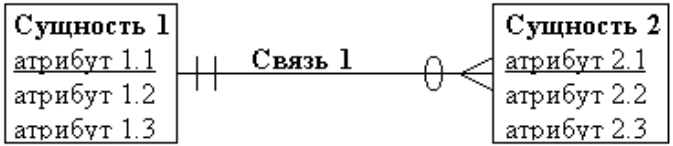


2.2.1. Обзор нотаций, используемых при построении диаграмм "сущность-связь".

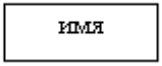
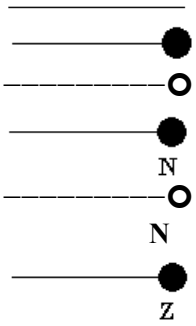
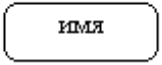
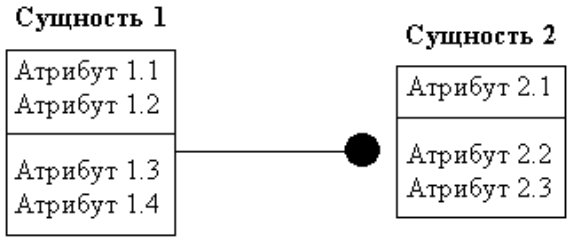
2.2.1.1. Нотация Чена.

Элемент диаграммы	Обозначает	Элемент диаграммы	Обозначает
	Независимая сущность		Атрибут
	Зависимая сущность		Первичный ключ
	Родительская сущность в иерархической связи		Внешний ключ (понятие внешнего ключа вводится в реляционной модели данных)
	Связь		Многозначный атрибут
	Идентифицирующая связь		Получаемый (наследуемый) атрибут в иерархических связях
			Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается её обязательность и мощность.

2.2.1.2. Нотация Мартина.

<table>
<tr><th>Элемент диаграммы</th><th>Обозначает</th><th>Элемент диаграммы</th><th>Обозначает</th></tr>
<tr><td> 
</td><td>Независимая сущность</td><td colspan="2">Связи изображаются линиями, соединяющими сущности, вид линии в месте соединения с сущностью определяет и обязательность и кардинальность связи:</td></tr>
<tr><td rowspan="3"> 
</td><td rowspan="3">Зависимая сущность</td><td></td><td>Не определено, Не определено</td></tr>
<tr><td>  </td><td>0,1</td></tr>
<tr><td>  </td><td>1,1</td></tr>
<tr><td rowspan="3"> 
</td><td rowspan="3">Родительская сущность в иерархической связи</td><td>  </td><td>Не определено, N</td></tr>
<tr><td>  </td><td>0,N</td></tr>
<tr><td>  </td><td>1,N</td></tr>
<tr><td>Атрибут 1 (ключевой)
Атрибут 2
Атрибут 3 (ключевой)
Атрибут 4
…
Атрибут K</td><td>Список атрибутов приводится внутри прямоугольника, обозначающего сущность. Ключевые атрибуты подчеркиваются.</td><td colspan="2">Имя связи указывается на линии ее обозначающей.
Пример:
 
</td></tr>
</table>

2.2.1.3. Нотация IDEF1X.

<table>
<tr><th>Элемент диаграммы</th><th>Обозначает</th><th>Элемент диаграммы</th><th>Обозначает</th></tr>
<tr><td colspan="2">Сущности и связи</td><td colspan="2">Обязательность и кардинальность связи</td></tr>
<tr><td> 
</td><td>независимая сущность</td><td rowspan="4"> 
</td><td rowspan="4">Не определено, Не определено
1,1
0,1
1,N
0,N
1,точно Z (Z - произвольное число)</td></tr>
<tr><td> 
</td><td>зависимая сущность</td></tr>
<tr><td>_____</td><td>идентифицирующая связь</td></tr>
<tr><td>- - - - - - - - - - - -</td><td>неидентифицирующая связь</td></tr>
<tr><td colspan="2"> 
</td><td colspan="2">Пример.
Список атрибутов приводится внутри прямоугольника, обозначающего сущность. Атрибуты, составляющие ключ сущности, группируются в верхней части прямоугольника и отделяются горизонтальной чертой.</td></tr>
<tr><td colspan="4">Кроме того, вводится понятие “отношение категоризации”, по смыслу эквивалентное иерархической связи:
• полная категоризация - сущности-категории составляют полное множество потомков родительской сущности;
• неполная категоризация - сущности-категории составляют неполное множество потомков общей сущности.</td></tr>
</table>

2.2.1.4. Нотация Баркера.

<table>
<tr><th>Элемент диаграммы</th><th>Обозначает</th><th colspan="2">Элемент диаграммы</th><th>Обозначает</th></tr>
<tr><td>ИМЯ</td><td>Сущности обозначаются прямоугольниками</td><td colspan="2">Связи обозначаются линиями с именами, место соединения связи и сущности определяет кардинальность связи, штриховая линия определяет необязательность связи</td><td></td></tr>
<tr><td rowspan="4">Сущность 1
#атрибут 1.1
атрибут 1.2
атрибут 1.3</td><td rowspan="4">Список атрибутов приводится внутри сущности.
Ключевые атрибуты отмечаются символом #.</td><td>- - - - - - - - - - - -</td><td>0,1</td></tr>
<tr><td>————————</td><td>1,1</td></tr>
<tr><td>- - - - - - - - - -</td><td>0,N</td></tr>
<tr><td>————————</td><td>1,N</td></tr>
<tr><td>Пример:</td><td colspan="4">Сущность 1
#атрибут 1.1
атрибут 1.2
атрибут 1.3
Связь 1
Сущность 2
#атрибут 2.1
атрибут 2.2
атрибут 2.3</td></tr>
</table>

2.3. Методологии построения ER-диаграмм.

2.3.1. Методология Питера Чена.

Оригинальная работа Питера Чена (Peter Chen) «The Entity Relationship Model - Toward A Unified View of Data» («Модель сущность-связь - Унифицированное представление данных») обычно цитируется как основополагающая для методологии ERD. Однако следует отметить, что основные положения графического моделирования БД были опубликованы в ряде работ других авторов, например, в статье А. П. Г. Брауна «Modelling a Real-World System and Designing a Schema to Represent It» («Моделирование систем реального мира и проектирование схем их представления). В то же время Чен сделал больше, чем кто-либо, для популяризации этой модели и введения её в научный оборот.

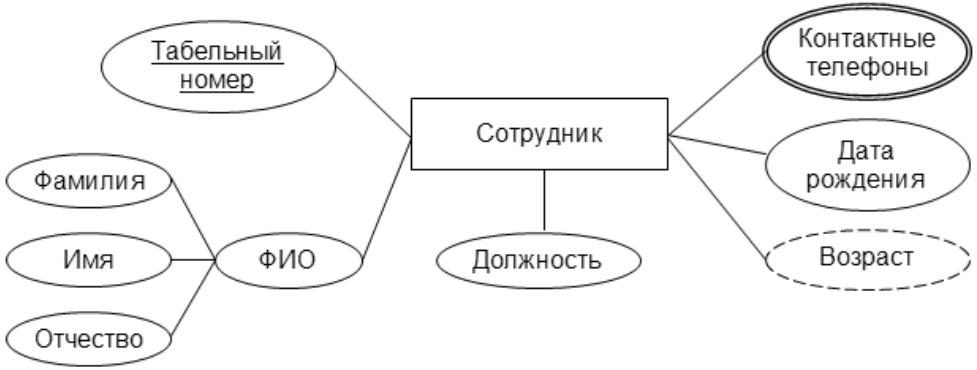
Предложенная Ченом графическая нотация ERD включает в себя следующие элементы.

2.3.1.1. Сущности.

Независимая сущность отображается на диаграммах прямоугольником с одинарной рамкой, **зависимая** – с двойной:

2.3.1.2. Атрибуты.

Атрибуты отображаются за пределами графического обозначения сущности в виде эллипсов, связанных одинарной линией с ним.



Атрибуты, **входящие в первичный ключ** сущности, выделяются подчеркиванием имени. Эллипс **многозначных** атрибутов изображается с двойным контуром, **производных** – пунктирным. Если атрибут является **составным**, то атрибуты-компоненты отображаются в виде присоединенных к нему эллипсов.

2.3.1.3. Связи.

Связь между сущностями показывается в виде ромба с указанием имени связи внутри него. При этом, если ромб имеет двойную рамку, то связь – **идентифицирующая**, одинарную – **неидентифицирующая**.

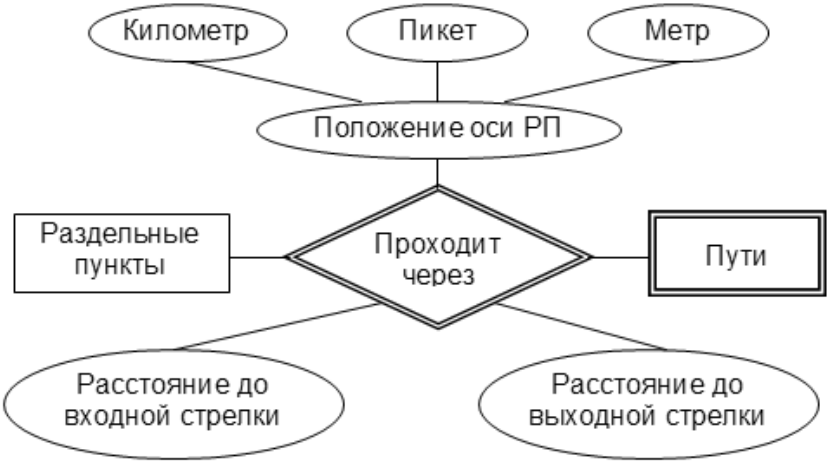
Обязательность связи отображается двойной линией. В частности на рисунке б) отдел обязательно должен состоять из сотрудников, а сотрудник (например, руководитель) необязательно входить в штат отдела. Т.е. понятие обязательности в данной нотации носит двухсторонний характер.

а) идентифицирующая связь

б) неидентифицирующая связь

2.3.1.4. Атрибуты связи.

Связь, как и сущность, может иметь собственные атрибуты, отображаемые в виде присоединенных к ней эллипсов.



2.3.1.5. Мощность (кардинальность) связи.

Мощность связи обозначается числами или буквами:

- 1 – один;
- N или M – многие;
- <число> – конкретное количество экземпляров, например, 3 или 10;
- (<Min>, <Max>) – диапазон экземпляров, где в качестве <Min> и <Max> могут использоваться предыдущие обозначения мощности.

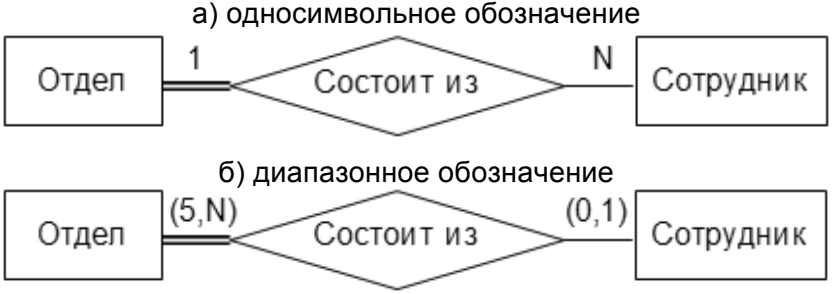
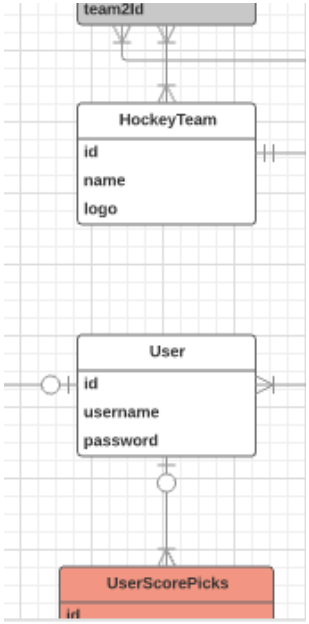


Рис. Отображение обязательности и мощности связей в нотации П. Чена.

В первом случае мощность связи (один-ко-многим) указывается традиционно, как в большинстве нотаций. Т.е. экземпляр родительской сущности (один конкретный отдел) связан с 0, 1 или более экземплярами дочерней сущности (конкретными сотрудниками). Во втором случае ее следует понимать следующим образом: экземпляр родительской сущности должен быть связан с 5 или более экземплярами дочерней сущности (отдел состоит из 5 или более сотрудников), а экземпляр дочерней сущности связан с 0 или 1 экземпляром родительской сущности (сотрудник либо не входит, либо входит в отдел).

2.3.2. Методология в нотации Мартина.

1. Необходимо правильно и полно описать предметную область будущей БД. В виде последовательного списка выписать все функции и запросы, которые будет выполнять БД.
2. В выписанном списке требований к БД, подчеркнуть все существительные. Это будущие сущности, их изображают прямоугольниками.
3. Для сущностей выписать атрибуты. Определить свойства, которые являются важными для предметной области. Атрибуты помещают внутри прямоугольника сущности, к которой атрибуты относятся.
4. Для каждой сущности выбрать ключевые атрибуты. Их отмечают подчёркиванием или жирным. При необходимости вводят искуственный ключевой атрибут (например, ID типа Counter).
5. Определяют связи, которые существуют межу сущностями. Связи обозначаются линиями. Конец линии, который подходит к сущности, связанной отношением «многие», разделяется на три части («куриная лапка»), кардинальность «один» отмечают I («штрих»), обязательность связи отмечают знаком I и O.
6. Для будущей реляционной модели, если на ER-диаграмме присутствуют связи «многие ко многим», то такую связь нужно разделить на две «один ко многим» с помощью дополнительной сущности.
7. Полученная ER-диаграмма является прототипом будущей БД: сущности являются таблицами, атрибуты – полями. Все связи должны реализовываться стандартными средствами СУБД.



3. Преобразование ER-модели в реляционную модель.

Концептуальные ER-модели позволяют более точно представить предметную область, чем реляционные и другие более ранние модели. Но в настоящее время существует немного систем управления базами данных, поддерживающих эти модели. На практике наиболее распространены системы, реализующие реляционную модель. Поэтому необходим метод перевода концептуальной модели в реляционную.

Такой метод основывается на формировании набора предварительных таблиц из ER-диаграмм. На следующем этапе состав и связи этих таблиц оптимизируются.

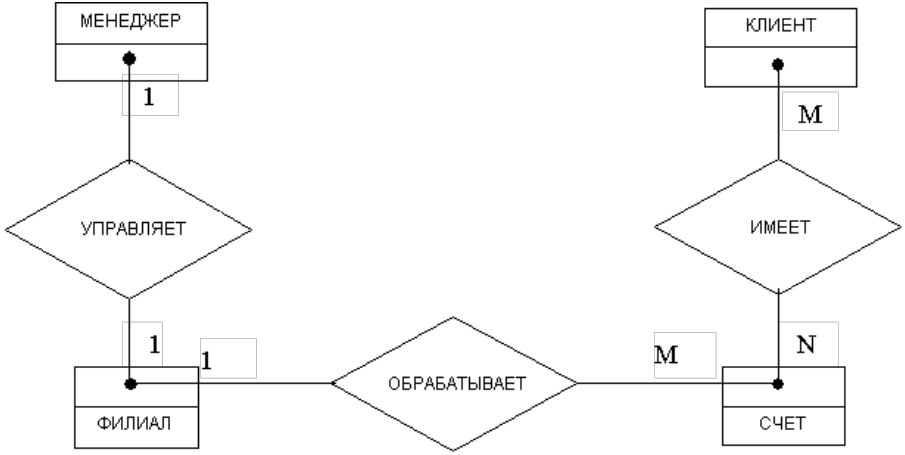
Суть преобразований кратко:

- Для каждой сущности создается таблица. Причем каждому атрибуту сущности соответствует столбец таблицы.
- Правила генерации таблиц из ER-диаграмм опираются на два основных фактора – тип связи и класс принадлежности сущности.
- Для связи типа 1:1 существуют **три** отдельных правила формирования предварительных таблиц из ER-диаграмм. Для связи типа 1:M существуют только **два** правила. Выбор одного из них зависит от класса принадлежности сущности на стороне M. Класс принадлежности сущности на стороне 1 не влияет на выбор. Для связи типа M:N класс принадлежности сущности значения не имеет и правило **одно**.

Изложим эти 6 правил преобразования для ER-модели на примере предметной области БАНК ER-модель и атрибуты сущностей которой показаны на рис. ниже.

3.0. ER-модель для преобразования.

Предметная область БАНК и наборы атрибутов сущностей.



МЕНЕДЖЕР
Номер менеджера (**НМ**)
Стаж работы (СТАЖ)
Специальность (СПЕЦ)

ФИЛИАЛ
Номер филиала (**НФ**)
Адрес филиала (АДР_Ф)

СЧЕТ
Номер счета (**НС**)
Тип счета (ТИП)
Остаток на счете (ОСТ)

КЛИЕНТ
Номер клиента (**НК**)
Ф.И.О. клиента (ФИО_К)
Социальное положение (СОЦ)
Адрес клиента (АДР_К)

Примечание: ключевые атрибуты выделены жирным шрифтом.

3.1. Правило 1.

Если связь типа 1:1 и класс принадлежности обеих сущностей является обязательным (1,1:1,1), то необходима только одна таблица. Первичным ключом этой таблицы может быть первичный ключ любой из двух сущностей.

Если на ER-диаграмме для связи 1:1 <УПРАВЛЯЕТ> класс принадлежности обоих сущностей МЕНЕДЖЕР, ФИЛИАЛ является обязательным, тогда генерируется одна таблица следующей структуры:

МЕНЕДЖЕР–ФИЛИАЛ

НМ	СТАЖ	СПЕЦ	**НФ**	АДР_Ф

Первичным ключом этой таблицы может быть выбран и первичный ключ сущности МЕНЕДЖЕР – НМ, который теперь стал потенциальным ключом.

3.2. Правило 2.

Если связь типа 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным (1,1:0,1 или 0,1:1,1), то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой класс принадлежности является необязательным, добавляется как атрибут в таблицу для сущности с обязательным классом принадлежности.

Если на ER-диаграмме для связи 1:1 <УПРАВЛЯЕТ> класс принадлежности сущности МЕНЕДЖЕР будет обязательный, а сущности ФИЛИАЛ – необязательный, тогда генерируются две таблицы следующей структуры:

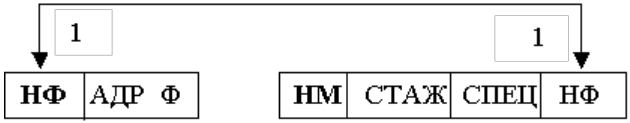
МЕНЕДЖЕР–ФИЛИАЛ

НМ	СТАЖ	СПЕЦ	НФ

ФИЛИАЛ

НФ	АДР_Ф

Сущность с необязательным классом принадлежности (ФИЛИАЛ) именуется **родительской**, а с обязательным (МЕНЕДЖЕР) – **дочерней**. Первичный ключ родительской сущности (НФ), помещаемый в таблицу, представляющую дочернюю сущность, называется внешним ключом родительской сущности. Связь между указанными таблицами устанавливается путем связи первичного и внешнего ключа и имеет вид:



Примечание. Если внешний ключ представляет связь 1:1, то должны быть запрещены его дублирующие значения.

3.3. Правило 3.

Если связь типа 1:1 и класс принадлежности обеих сущностей является необязательным (0,1:0,1), то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Если на ER-диаграмме для связи 1:1 <УПРАВЛЯЕТ> класс принадлежности обеих сущностей МЕНЕДЖЕР, ФИЛИАЛ будет необязательным, тогда генерируются три следующих таблицы:

МЕНЕДЖЕР

НМ	СТАЖ	СПЕЦ

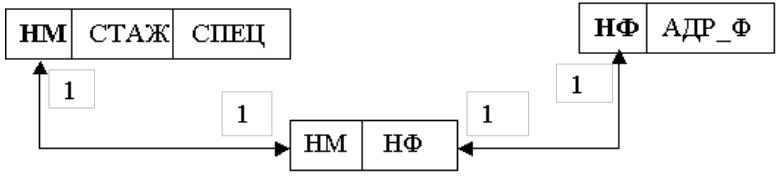
ФИЛИАЛ

НФ	АДР_Ф

МЕНЕДЖЕР–ФИЛИАЛ

НМ	**НФ**

При этом осуществляется декомпозиция связи 1:1 на две связи 1:1 следующим образом:



3.4. Правило 4

Если связь типа 1:M и класс принадлежности сущности на стороне M является обязательным (0,1:1,M или 1,1:1,M), то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне M.

Если на ER-диаграмме для связи 1:M <ОБРАБАТЫВАЕТ> класс принадлежности сущности СЧЕТ является обязательным, тогда генерируются две таблицы следующей структуры:

ФИЛИАЛ

НФ	АДР_Ф

СЧЁТ–ФИЛИАЛ

НС	ОСТ	ТИП	НФ

Связь между указанными таблицами будет иметь вид

Примечание. Если внешний ключ представляет связь 1:M, то должны быть разрешены его дублирующие значения.

3.5. Правило 5.

Если связь типа 1:M и класс принадлежности сущности на стороне M является необязательным (1,1:0,M или 0,1:0,M), то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Если на ER-диаграмме для связи 1:M <ОБРАБАТЫВАЕТ> класс принадлежности сущности СЧЁТ является необязательным, тогда генерируются три таблицы следующей структуры:

ФИЛИАЛ

НФ	АДР_Ф

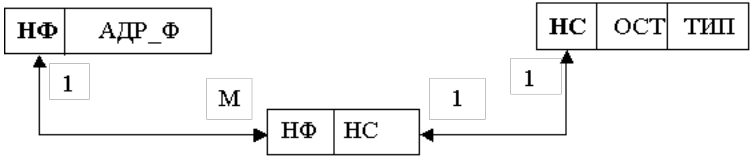
СЧЁТ

НС	ОСТ	ТИП

ФИЛИАЛ–СЧЁТ

НФ	**НС**

При этом осуществляется следующая декомпозиция связи 1:M на две связи – 1:M и 1:1:



3.6. Правило 6.

Если связь типа M:N, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

На ER-диаграмме есть связь M:N <ИМЕЕТ>, поэтому генерируются три следующих таблицы:

КЛИЕНТ

НК	ФИО_К	СОЦ_П	АДР_К

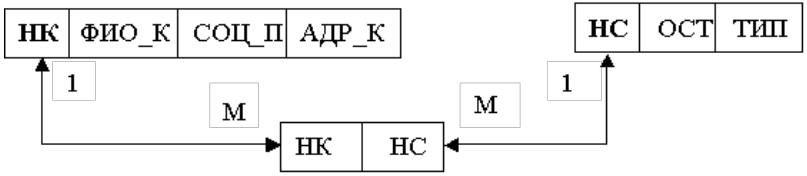
СЧЁТ

НС	ОСТ	ТИП

КЛИЕНТ–СЧЁТ

НК	**НС**

При этом осуществляется декомпозиция связи M:N на две связи 1:M следующим образом:



В таблице КЛИЕНТ–СЧЁТ клиенту, имеющему, например, три счета будут соответствовать три строки с одним и тем же номером клиента. А счет, у которого, например, два владельца, представляется двумя строками с различными номерами клиентов, владеющими этим счетом.

3.7. Итоговая реляционная модель.

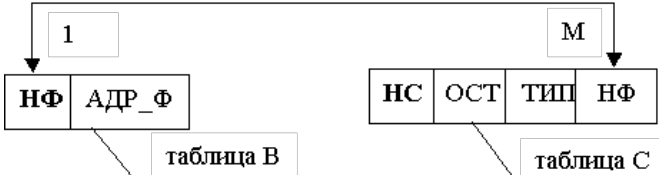
К ER-модели предметной области **БАНК**, представленной на рис. в п.3.0, применимы правила 1, 4 и 6.

По правилу 1) связь МЕНЕДЖЕР–ФИЛИАЛ представляется одной таблицей

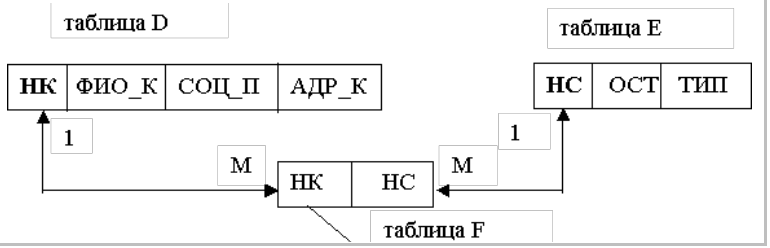
таблица A

НМ	СТАЖ	СПЕЦ	**НФ**	АДР_Ф

По правилу 4) связь ФИЛИАЛ–СЧЁТ представляется связью



По правилу 6) связь КЛИЕНТ–СЧЁТ представляется связью



Анализ состава атрибутов полученных таблиц A, B, C, D, E, F показывает, что таблица B является составной частью таблицы A, таблица E – составной частью таблицы C. Поэтому таблицы B, E можно исключить из рассмотрения. Оставшиеся таблицы A, C, D, F можно связать посредством связи первичных и внешних ключей как на рис. ниже. В результате получим реляционную модель для ER-модели предметной области **БАНК**.

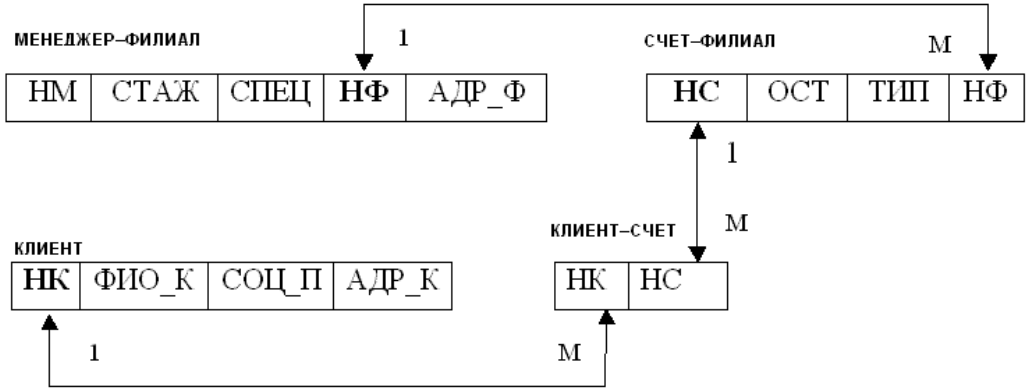


Рис. Реляционная модель предметной области БАНК.

Нормализация.

Дальнейший процесс нормализация реляционной модели приводит к декомпозиции исходных таблиц. Осуществив связь этих таблиц посредством связи первичных и внешних ключей, получим реляционную модель данных предметной области БАНК, в которой минимизирована избыточность данных.