Севастопольский государственный университет
Кафедра «Информационные системы»

Управление данными

курс лекций

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

Лекция 4

Жизненный цикл БД.

Логическое моделирование

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

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

Логическое проектирование базы данных (БД) – это **процесс создания структуры и организации данных в БД с учетом требований и целей предприятия или организации.** Включает в себя **определение сущностей** (таблиц), **атрибутов** (столбцов) **и связей между ними**, а также **определение правил и ограничений** для хранения и обработки данных.

Цель второй фазы проектирования базы данных состоит в создании эффективной и гибкой структуры данных, которая позволит эффективно хранить, обрабатывать и извлекать информацию. Структура должна удовлетворять требованиям надежности, безопасности и целостности данных.

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.

Задачи логического проектирования БД:

Определение сущностей и атрибутов: в рамках логического проектирования БД определяются сущности (таблицы) и их атрибуты (столбцы). Сущности представляют объекты или понятия, которые нужно хранить в БД, а атрибуты определяют характеристики этих сущностей.

Определение связей между сущностями: важной задачей логического проектирования БД является определение связей между сущностями. Связи определяют отношения и зависимости между сущностями и позволяют связывать данные из разных таблиц для получения полной информации.

Определение правил и ограничений: в процессе логического проектирования БД определяются правила и ограничения, которые должны соблюдаться при хранении и обработке данных. Например, можно определить ограничения на значения атрибутов, правила для обновления данных или права доступа к данным.

Оптимизация структуры данных: целью логического проектирования БД является создание структуры данных, которая будет эффективно использоваться для хранения и обработки данных. Это включает оптимизацию индексов, выбор подходящих типов данных, разделение данных на таблицы и другие методы оптимизации.

Документирование структуры и правил БД: важной задачей логического проектирования БД является документирование структуры БД и правил, которые были определены. Это позволяет легко понять и поддерживать БД в будущем, а также обеспечивает единое понимание структуры и правил среди разработчиков и пользователей.

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** - нормализация позволяет разделить данные на более мелкие и логически связанные части, что улучшает эффективность хранения и обработки данных. Нормализация включает в себя разделение данных на таблицы, определение первичных и внешних ключей и другие действия для устранения избыточности.;
- **оптимизация БД** - может включать в себя создание индексов, оптимизацию запросов, настройку параметров хранения данных и другие действия для ускорения работы с БД.
- **проверка поддержки целостности данных**;
- **документирование проекта БД** - разработчики документируют структуру БД и правила, которые были определены. Это позволяет легко понять и поддерживать БД в будущем, а также обеспечивает единое понимание структуры и правил среди разработчиков и пользователей.

ПОДХОДЫ И МЕТОДЫ К ЛОГИЧЕСКОМУ ПРОЕКТИРОВАНИЮ БД

TOP-DOWN ПОДХОД

Top-down подход к логическому проектированию БД начинается **с общего представления о системе и постепенно уточняется до более детального уровня.** В этом подходе разработчики начинают с определения общих сущностей и связей между ними, а затем постепенно добавляют дополнительные детали и атрибуты. Top-down подход позволяет разработчикам иметь общее представление о системе и ее структуре, что облегчает понимание и поддержку БД в будущем.

BOTTOM-UP ПОДХОД

Bottom-up подход к логическому проектированию БД начинается **с определения конкретных атрибутов и связей между ними, а затем постепенно объединяет их в более общие сущности.** В этом подходе разработчики начинают с низкого уровня детализации и постепенно строят более общую структуру БД. **Bottom-up подход позволяет разработчикам сосредоточиться на конкретных деталях и атрибутах,** что может быть полезно при работе с большими и сложными БД.

ER-МОДЕЛИРОВАНИЕ

ER-моделирование (Entity-Relationship Modeling) – это **метод моделирования, который используется для описания сущностей, атрибутов и связей между ними в БД.** ER-моделирование позволяет разработчикам визуализировать структуру БД и легко понять связи между сущностями.

НОРМАЛИЗАЦИЯ БД

Нормализация БД – **это процесс разделения таблиц на более мелкие и более связанные таблицы для устранения избыточности данных и обеспечения целостности БД.** Нормализация помогает улучшить эффективность и гибкость БД, а также уменьшить объем хранимых данных.

ДЕНОРМАЛИЗАЦИЯ БД

Денормализация БД – **это процесс объединения таблиц в одну или несколько таблиц для увеличения производительности и упрощения запросов.** Денормализация может быть полезна в случаях, когда требуется быстрый доступ к данным или когда объем данных невелик.

МОДЕЛЬ «СУЩНОСТЬ-СВЯЗЬ»

Модель «Сущность-связь» ***(ERD, Entity Relationship Model, ER-model)*** – представляет собой **формальную конструкцию, которая сама по себе не предписывает никаких графических средств ее визуализации.** В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь».

Диаграмма «Сущность-связь» ***(ERD, Entity-Relationship Diagram, ER-диаграмма)*** – это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы.

ER-диаграммы чаще всего применяются **для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.** ER-диаграммы полагаются на стандартный набор символов, включая **прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей**. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: **сущности выполняют роль существительных, а связи – глаголов.**

Основные характеристики

Диаграмма «Сущность-связь» (ER-диаграмма) является **абстрактным макетом базы данных, поэтому для ее моделирования был разработан ряд правил, которые облегчают переход от диаграмм к реляционным отношениям:**

- **основной элемент: сущность** – это физическое представление логической группировки данных. Сущности представляют собой объекты, данные о которых корпорация заинтересована сохранять;
- каждый **правильный (сильный) тип сущности** соответствует **базовому реляционному отношению**;
- каждая бинарная связь **типа «многие-ко-многим» также соответствует отдельному отношению**, которое должно включать в себя два внешних ключа, ссылающихся на потенциальные ключи отношений, соответствующих сущностям – участникам связи;
- связь типа **«один-ко-многим» между сильными сущностями может быть представлена с помощью внешнего ключа** и не требует отдельного отношения;
- **связь слабого объекта с сильным, от которого он зависит**, является связью типа **«многие-к-одному» и может быть представлена внешним ключом**. В некоторых случаях, когда и сильная, и подчиняющаяся ей слабая сущности представлены одним реляционным отношением, внешний ключ может ссылаться на первичный ключ своего же отношения;
- **атрибуты** сущностей приводятся к атрибутам отношений;
- **в случае более чем бинарной связи (n-арной),** обычно вводят (n + 1) отношение: по одному на каждую сущность и одну на связь.

Этапы построения ERD-диаграммы

На первом шаге производится определение сущностей. Для построения ER-диаграммы «с нуля» производиться 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.

Самые распространенные нотации (графические модели) ER-диаграмм

КЛАССИЧЕСКАЯ НОТАЦИЯ ПИТЕРА ЧЕНА (PETER CHEN NOTATION)

НОТАЦИЯ IE (INFORMATION ENGINEERING) ИНФОРМАЦИОННОГО ПРОЕКТИРОВАНИЯ: НОТАЦИЯ К. ФИНКЕЛЬШТЕЙНА, НОТАЦИЯ ДЖ. МАРТИНА ИЛИ «ВОРОНЬИ ЛАПКИ»

НОТАЦИЯ Р. БАРКЕРА (BARKER NOTATION)

НОТАЦИИ IDEF1 И IDEF1X (INTEGRATION DEFINITION FOR INFORMATION MODELING)

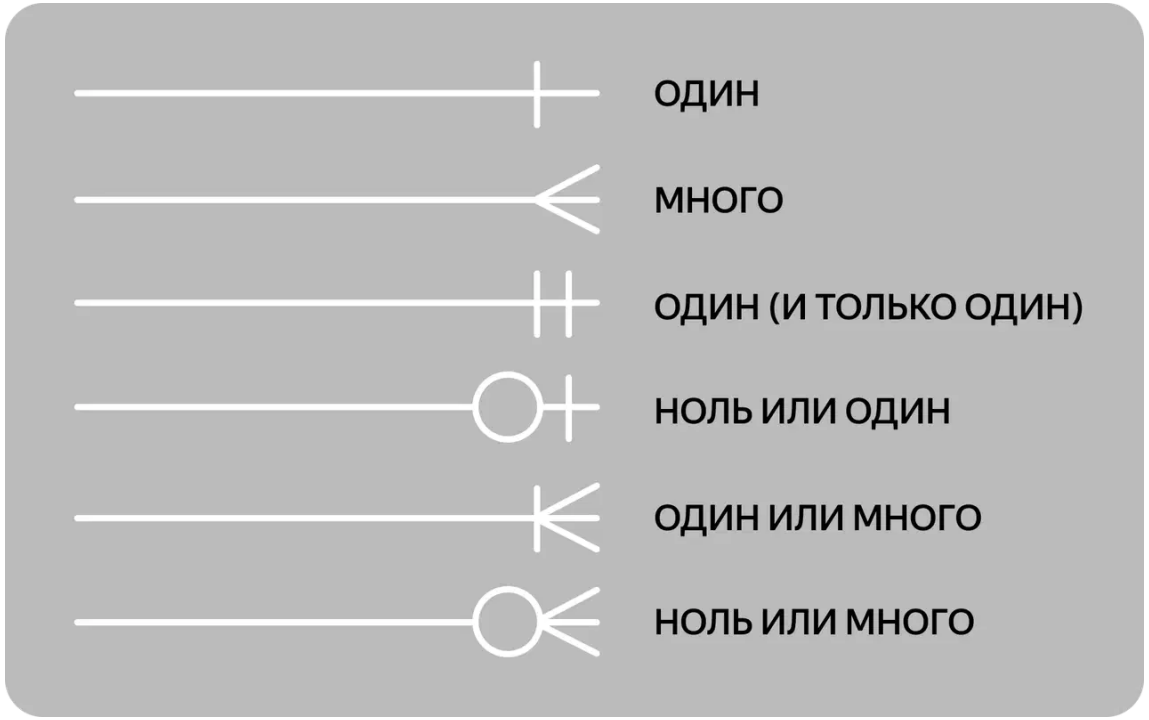
НОТАЦИЯ Ч. БАХМАНА

НОТАЦИЯ Ж.-Р. АБРИАЛЯ

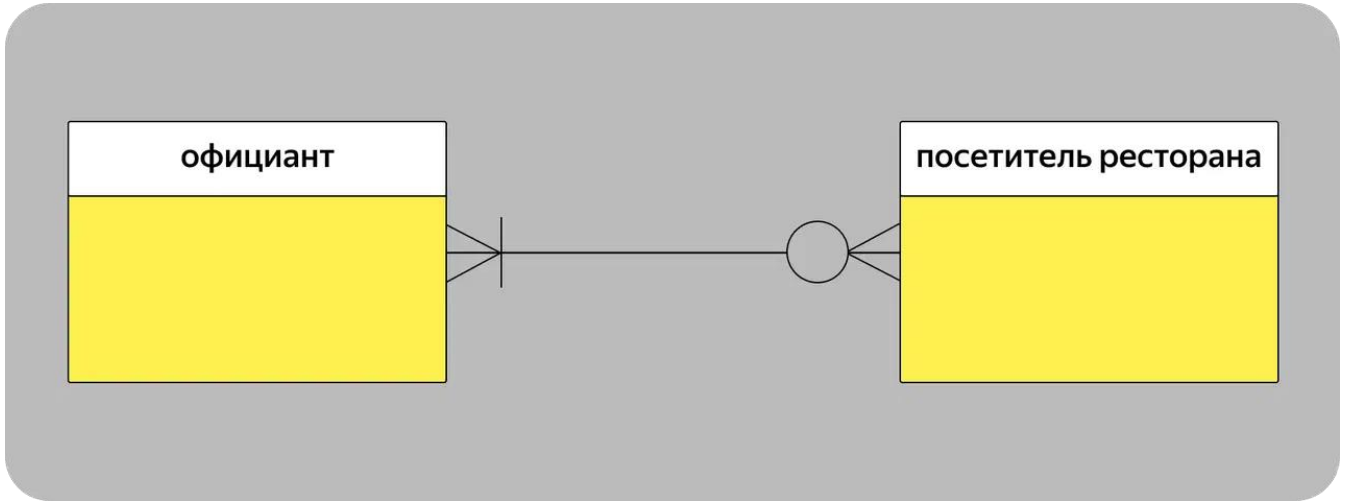
ДИАГРАММЫ КЛАССОВ UML

НОТАЦИЯ IE (INFORMATION ENGINEERING) ИНФОРМАЦИОННОГО ПРОЕКТИРОВАНИЯ: НОТАЦИЯ К. ФИНКЕЛЬШТЕЙНА, НОТАЦИЯ ДЖ. МАРТИНА ИЛИ «ВОРОНЬИ ЛАПКИ»

Родоначальником данной методологии (нотации) является **Клайв Финкельштейн**. Дальнейшее ее совершенствование связано с именами **Джеймса Мартина** и **Чарльза Рихтера**. Так же можно встретить название – **«Воронья лапка»** («**Куринная лапка**») или **«Вилка»**, основу в данную нотацию предложена **Гордоном Эверестом**, и она также может носить название **«Перевернутая стрелка»**. По графическому отображению и семантике элементов модели, нотация IE напоминает IDEF1X. Ее отличительной особенностью является указание мощности связей не в виде буквенно-цифрового обозначений, а с помощью графических элементов.

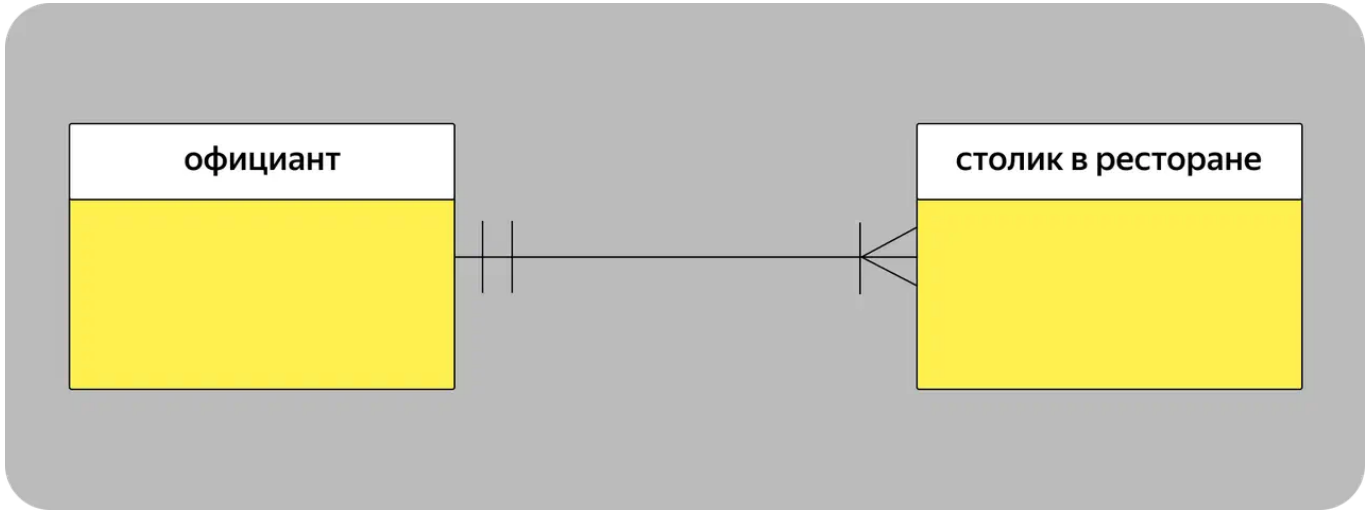


Официант может обслуживать от нуля до множества посетителей ресторана. При этом одного посетителя ресторана должен обслуживать хотя бы один официант, а могут и несколько.

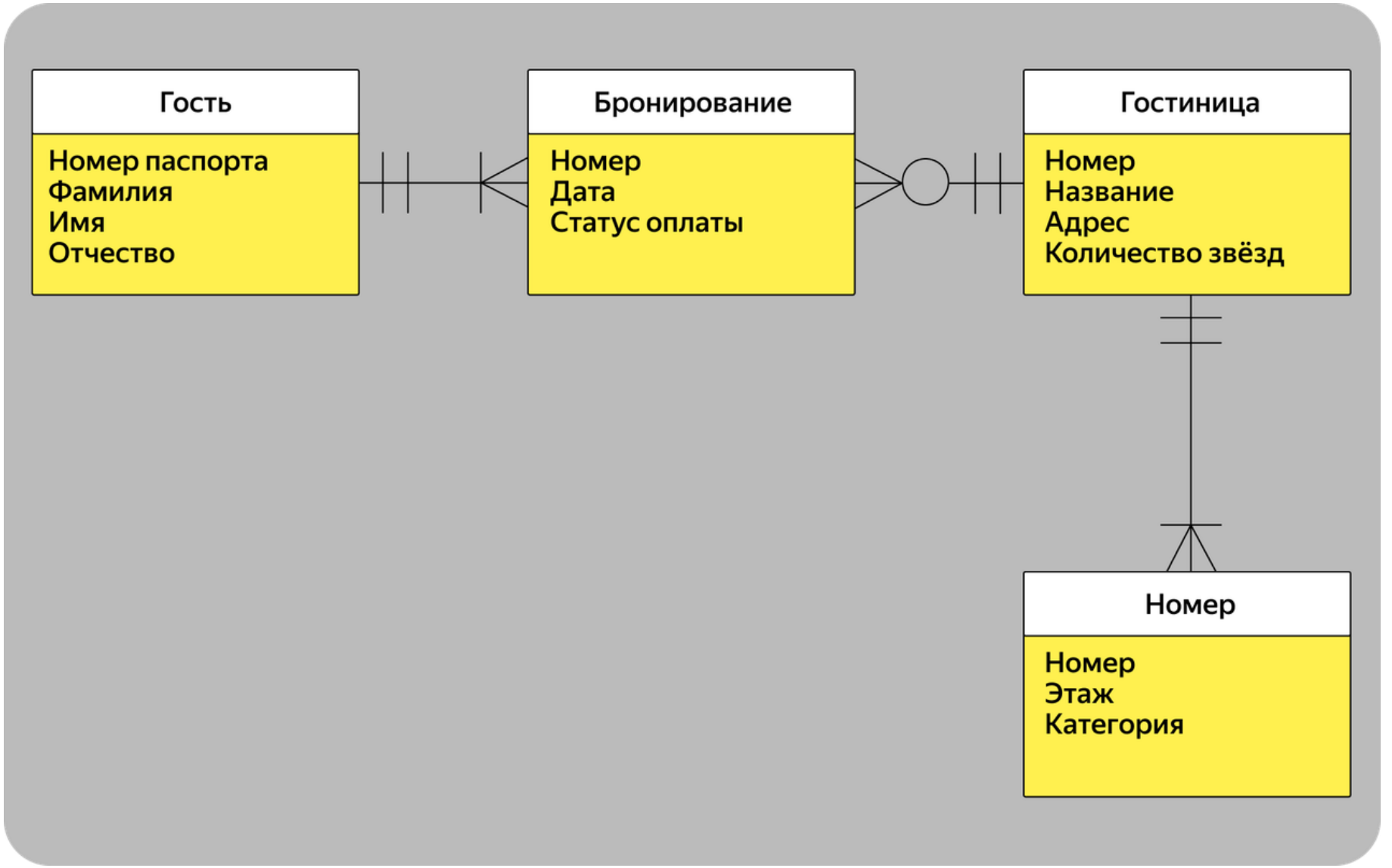


Согласно данной нотации, **сущность изображается в виде прямоугольника**, содержащем ее **имя, выражаемое существительным**. **Имя сущности** должно быть **уникальным в рамках одной модели**. При этом, имя сущности – это имя типа, а не конкретного экземпляра данного типа. Экземпляром сущности называется конкретный представитель данной сущности.

Атрибуты сущности записываются внутри прямоугольника, изображающего сущность. **Связь изображается линией**, которая соединяет две сущности, участвующие в отношении. **Множественность связи изображается в виде вилки.** Необязательные связи помечаются кружком.



За каждым столиком в ресторане закреплён только один официант. При этом за каждым официантом может быть закреплено несколько столиков

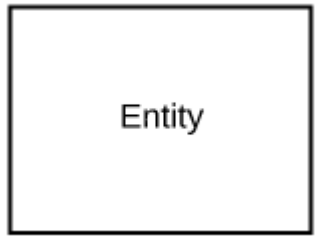


В ER-модели в нотации Мартина атрибуты сущностей перечисляют в полях под ними. За счёт этого модель занимает меньше места и её структура менее запутана

КЛАССИЧЕСКАЯ НОТАЦИЯ ПИТЕРА ЧЕНА (PETER CHEN NOTATION)

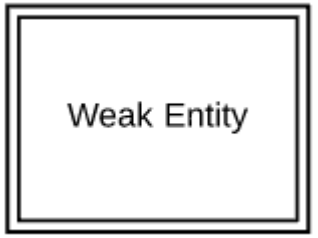
Множества сущностей изображаются в виде **прямоугольников**, **множества отношений** изображаются в виде **ромбов. Если сущность участвует в отношении, они связаны линией.** Если отношение не является обязательным, то линия пунктирная. **Атрибуты** изображаются **в виде овалов** и связываются линией с одним отношением или с одной сущностью.

Под понятием **«сущности»** подразумеваются **объекты или понятия, несущие важную информацию**. С точки зрения грамматики, они, как правило, **обозначаются существительными**, например, «товар», «клиент», «заведение» или «промоакция».



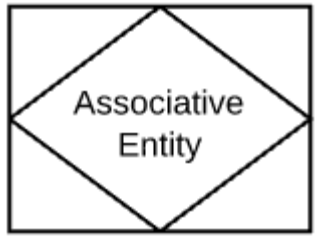
Независимая (сильная) сущность

Не зависит от других сущностей и часто называется «родительской», так как у нее в подчинении обычно находятся слабые сущности. Независимые сущности сопровождаются первичным ключом, который позволяет идентифицировать каждый экземпляр сущности.



Зависимая (слабая) сущность

Сущность, которая зависит от сущности другого типа. Не сопровождается первичным ключом и не имеет значения в схеме без своей родительской сущности.



Ассоциативная сущность

Соединяет экземпляры сущностей разных типов. Также содержит атрибуты, характерные для связей между этими сущностями.

АССОЦИАТИВНЫЕ СУЩНОСТИ

Ассоциативная сущность представляет собой данные, которые ассоциируются с отношениями между двумя или более сущностями.

Например, есть **служба проката фильмов**. Есть необходимость отслеживать инвентарь с помощью баз данных. Таким образом, создается БД и модель, в которых можно отслеживать, какие фильмы клиенты арендовали в любой момент времени.

Изначально решено **использовать отношение один ко многим (1:M). Где несколько фильмов могут быть прикреплены к одному заказчику**.

В этой модели всякий раз, **когда фильм сдается клиенту в аренду**, в столбце **customer_id устанавливается первичный ключ этого клиента**. Известно, что фильм берется напрокат, когда этот столбец установлен, и фильм доступен, если этот столбец равен null.

Запрос на проверку наличия Барби будет выглядеть так:

```
select * from movies where customer_id is null and name = 'Barbie'
```

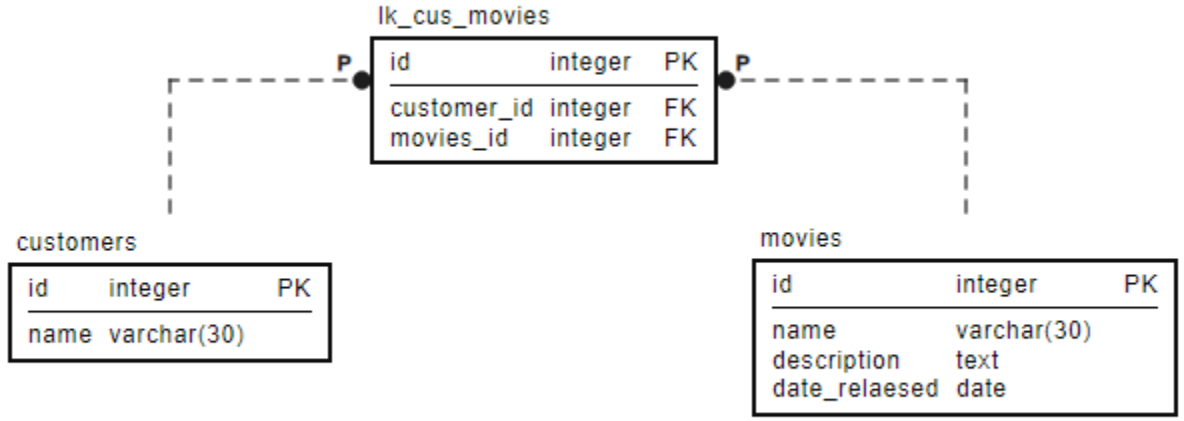
Тем не менее, проблема этой модели заключается в том, что если есть магазин фильмов, вероятно, что есть несколько копий фильма. Эта модель заставит создавать запись для каждого фильма. Допустим, есть 10 копий The Matrix, тогда база данных будет выглядеть примерно так.

id	name	customer_id
1	The Matrix	*NULL*
2	The Matrix	*NULL*
3	The Matrix	*NULL*
4	The Matrix	1
5	The Matrix	*NULL*
6	The Matrix	*NULL*
7	The Matrix	*NULL*
8	The Matrix	*NULL*
9	The Matrix	*NULL*
10	The Matrix	*NULL*

Возможно, в любом случае придется представлять каждый фильм, который есть, с помощью табличной записи, **но худшая часть этой модели** - это **количество повторяющихся данных**, которые **каждый раз сохраняются в базе данных**. Кроме того, очень вероятно, что будет более 3 столбцов с данными фильма и другими метаданными.

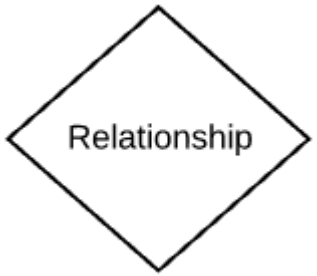
Решение по ассоциации

Ассоциативная таблица позволит представить каждую аренду как запись. В то же время можно избежать хранения повторяющихся данных. Как и у всего, есть плюсы и минусы, но в зависимости от варианта использования и задействованных данных это может быть более жизнеспособным решением.



КЛАССИЧЕСКАЯ НОТАЦИЯ ПИТЕРА ЧЕНА (PETER CHEN NOTATION)

Связи **используются в схемах «сущность-связь»** для обозначения **взаимодействия** между двумя сущностями. Грамматически связи, как правило, **выражаются глаголами**, например, «назначить», «закрепить», «отследить», и несут полезную информацию, которую невозможно получить, опираясь только на типы сущностей.

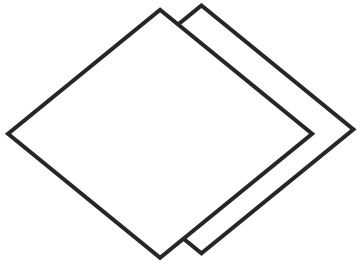


Неограниченное (обязательное отношение)

Представляет собой отношение, существующее до тех пор, пока существуют относящиеся к делу сущности.

Слабая связь

Связь между зависимой сущностью и ее «хозяином».

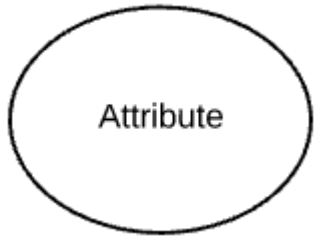


Существенно ограниченное отношение

Используется, когда соответствующие сущности взаимозависимы в системе

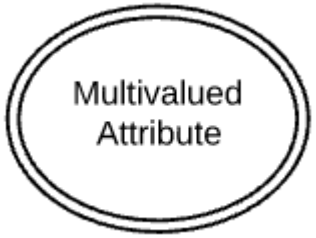
КЛАССИЧЕСКАЯ НОТАЦИЯ ПИТЕРА ЧЕНА (PETER CHEN NOTATION)

ERD-атрибуты характеризуют **сущности**, позволяя пользователям лучше разобраться в устройстве базы данных. Атрибуты **содержат информацию о сущностях**, выделенных в концептуальной ER-диаграмме.



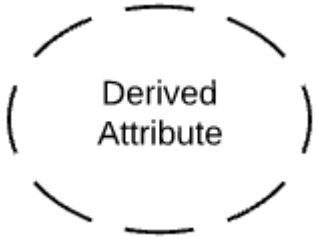
Атрибут

Характеризует сущность, а также отношения между двумя или более элементами.



Многозначный атрибут

Атрибут, которому может быть присвоено несколько значений.



Производный атрибут

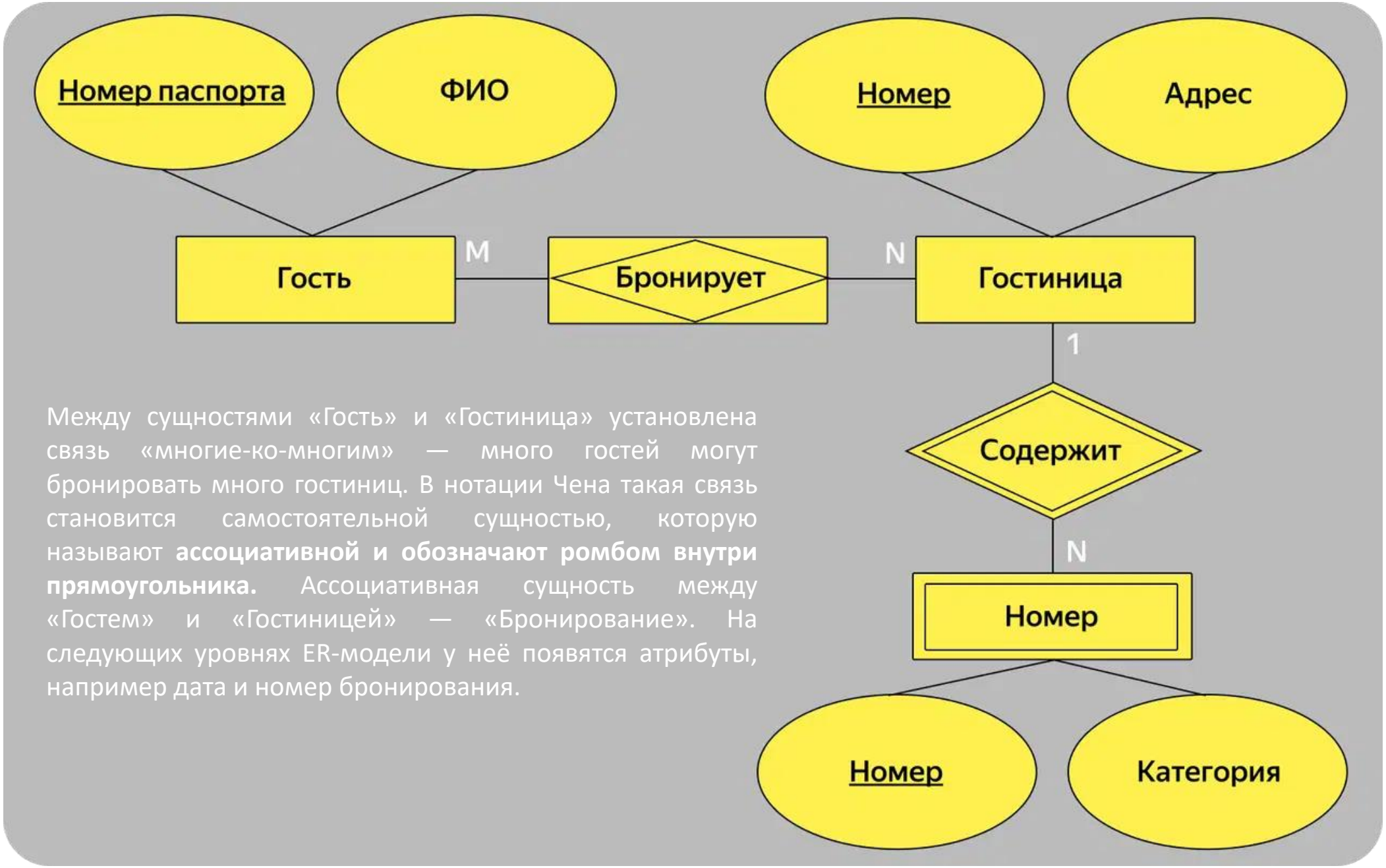
Атрибут, чье значение можно вычислить, опираясь на значения связанных с ним атрибутов.

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.



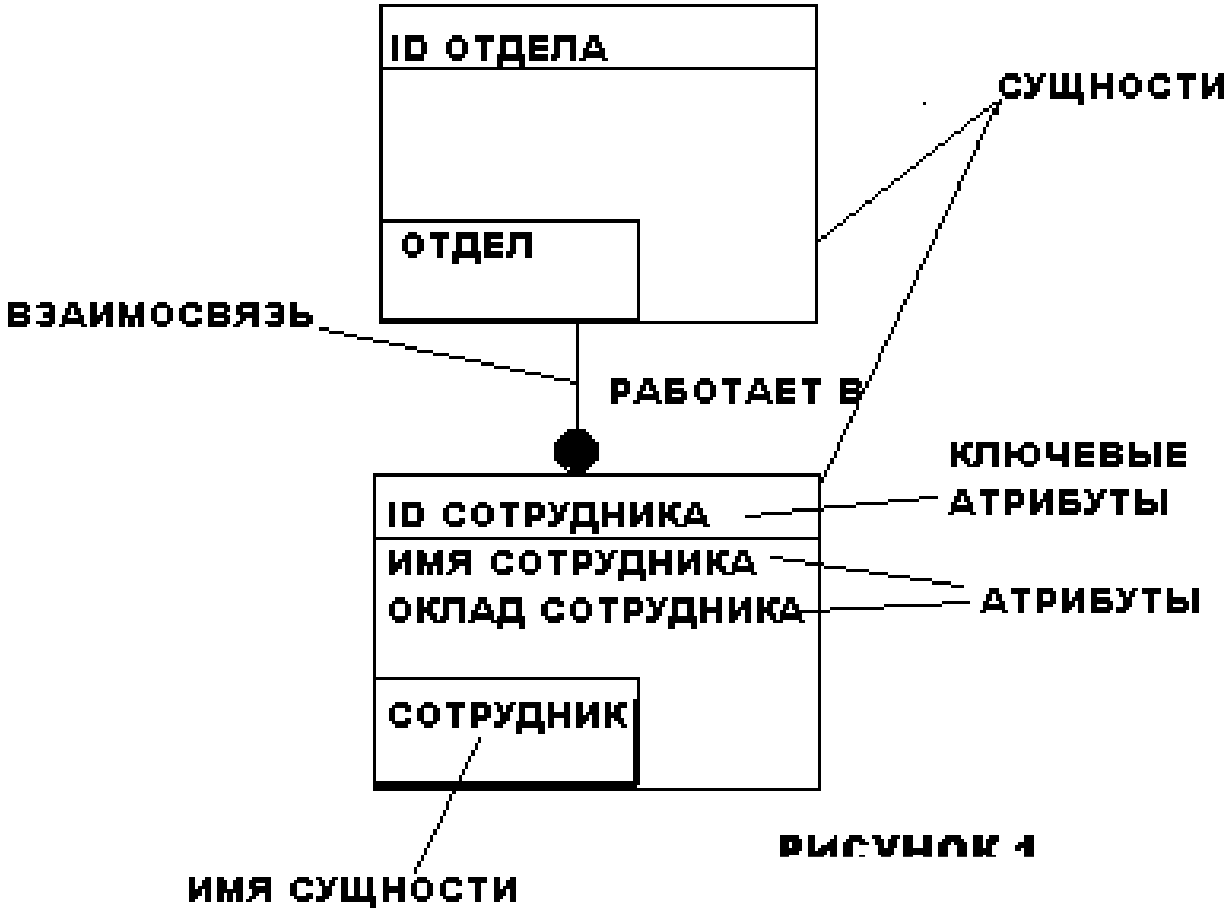
Между сущностями «Гость» и «Гостиница» установлена связь «многие-ко-многим» — много гостей могут бронировать много гостиниц. В нотации Чена такая связь становится самостоятельной сущностью, которую называют **ассоциативной и обозначают ромбом внутри прямоугольника.** Ассоциативная сущность между «Гостем» и «Гостиницей» — «Бронирование». На следующих уровнях ER-модели у неё появятся атрибуты, например дата и номер бронирования.

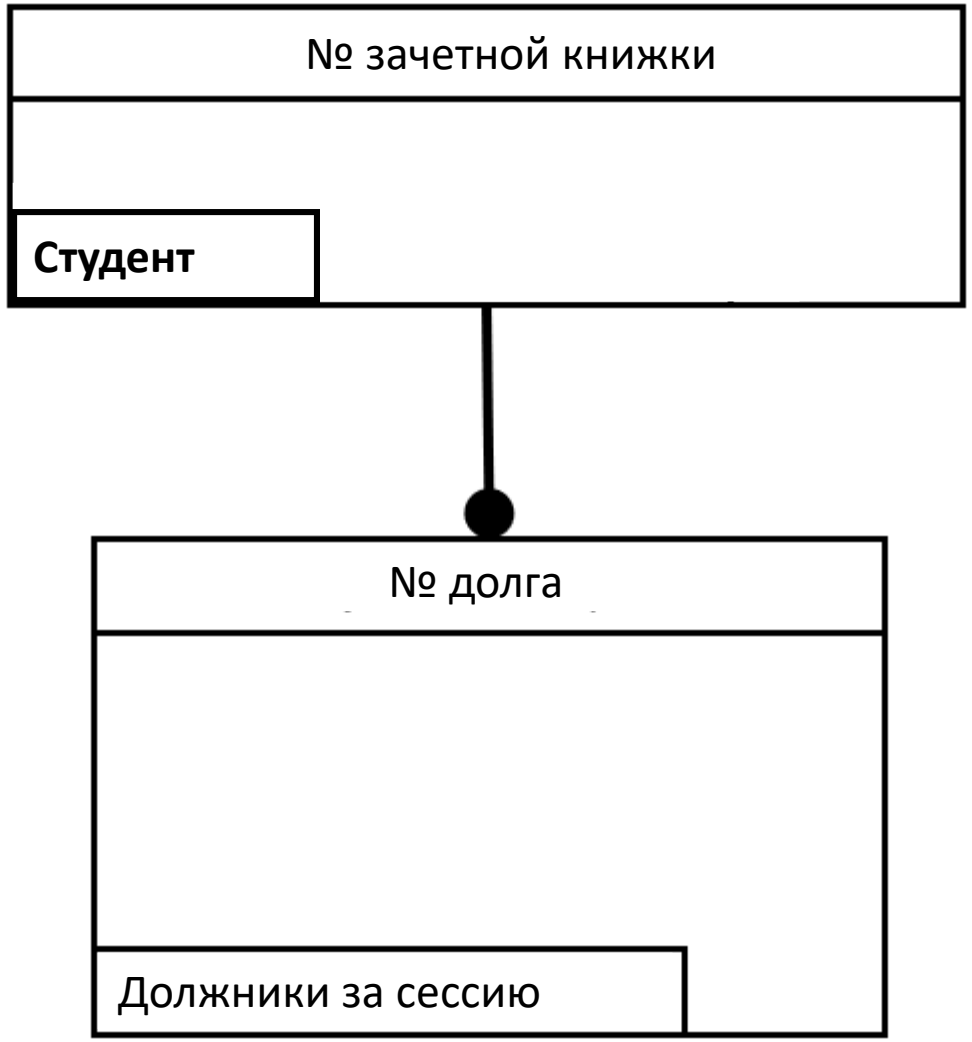
МОДЕЛЬ ОСНОВАННАЯ НА КЛЮЧАХ (KEY BASED MODEL, KB)

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.

IDEF1 (Integration Definition for Information Modeling) – данная нотация применяется **для построения модели информационных потоков**, позволяющая отображать и анализировать их структуру и взаимосвязи. Метод IDEF1, разработанный Т. Рэмей, также основан на подходе П. Чена и **позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.**


ID ОТДЕЛА
ОТДЕЛ
СУЩНОСТИ
ВЗАИМОСВЯЗЬ
РАБОТАЕТ В
КЛЮЧЕВЫЕ АТРИБУТЫ
ID СОТРУДНИКА
ИМЯ СОТРУДНИКА
ОКЛАД СОТРУДНИКА
АТРИБУТЫ
СОТРУДНИК
ИМЯ СУЩНОСТИ

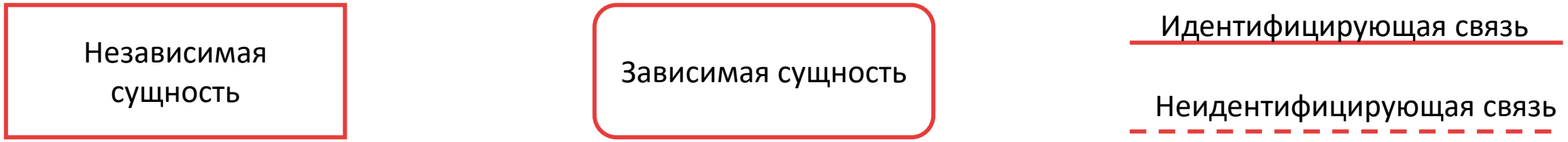

№ зачетной книжки
Студент
№ долга
Должники за сессию

ПОЛНАЯ АТРИБУТИВНАЯ МОДЕЛЬ В НОТАЦИИ IDEF1X

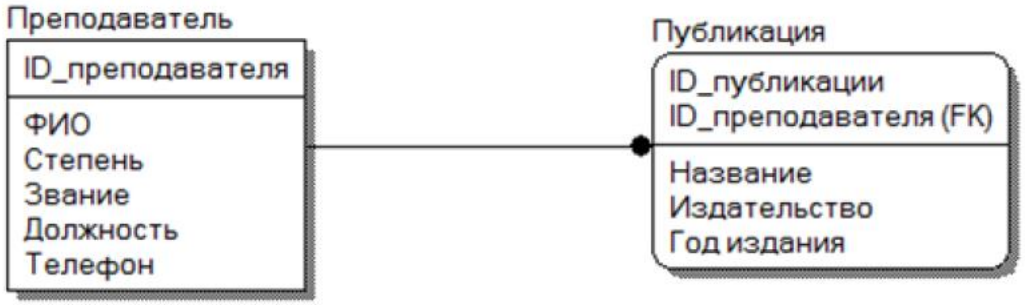
IDEF1X – предназначена для построения концептуальной схемы логической структуры реляционной базы данных, которая была бы независимой от программной платформы её конечной реализации. По сравнению с IDEF1 она дополнительно оперирует рядом понятий, правил и ограничений, такими как домены, представления,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

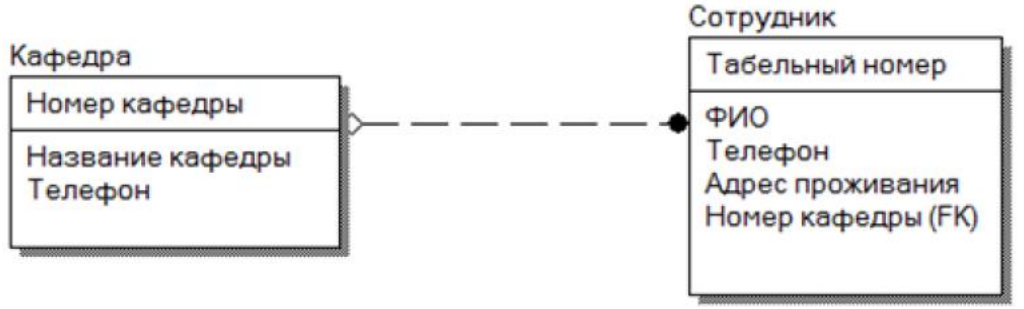
Данная методология описывает способы изображения двух типов сущностей – **независимой и зависимой**, и связей – **идентифицирующих и неидентифицирующих**.



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).**



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 → R.Y.
- **полная (неприводимая) функциональная зависимость:** функциональная зависимость R.X → R.Y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X.
- **транзитивная функциональная зависимость:** функциональная зависимость R.X -> R.Y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости R.X→R.Z и R.Z → R.Y и отсутствует функциональная зависимость R.Z→R.X.
- **многозначные зависимости в отношении R(A, B, C)** существует многозначная зависимость R.A ->> 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.

АСПЕКТ	ОПРЕДЕЛЕНИЕ	СВОЙСТВА
Логическое проектирование БД	Процесс создания схемы базы данных, определяющей структуру и связи между данными	○ определяет сущности и атрибуты ○ устанавливает связи между сущностями ○ оптимизирует структуру бд для эффективного доступа к данным
Цели и задачи	Определение требований к БД, создание структуры данных, обеспечение целостности и эффективности работы с данными	○ удовлетворение потребностей пользователей ○ минимизация избыточности и дублирования данных ○ обеспечение целостности и безопасности данных ○ оптимизация производительности запросов
Методы и подходы	ER-моделирование, нормализация, денормализация, проектирование схемы данных	○ ER-моделирование для визуализации сущностей и связей ○ нормализация для устранения избыточности данных ○ денормализация для повышения производительности ○ проектирование схемы данных для определения структуры БД
Индексы и ключи	Инструменты для ускорения поиска и обеспечения уникальности данных	○ индексы позволяют быстро находить данные по определенным атрибутам ○ ключи обеспечивают уникальность и связи между сущностями
Оптимизация	Улучшение производительности и эффективности работы с данными	○ выбор подходящих структур данных и алгоритмов ○ использование индексов и оптимизация запросов ○ учет объема данных и требований к производительности