Севастопольский государственный университет
Кафедра «Информационные системы»

Управление данными

курс лекций

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

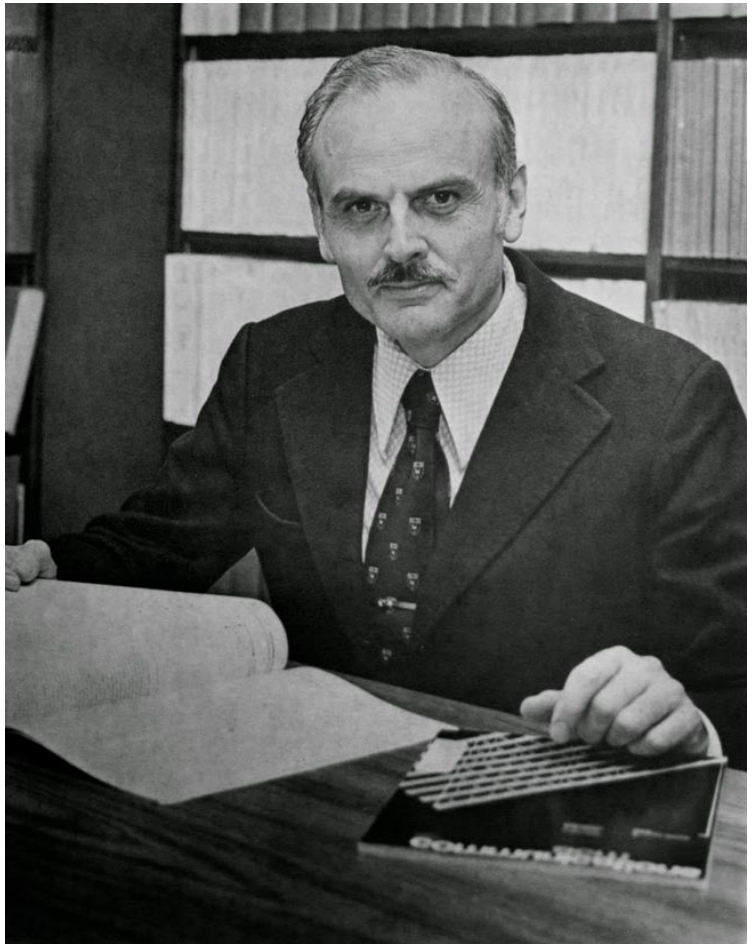
Лекция 6

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Определения понятий реляционной модели (РМД). Свойства отношений и связей. Типы ключей

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ: ИСТОРИЯ ВОПРОСА

Создателем реляционной модели является сотрудник фирмы IBM доктор **Эдгар Франк Кодд**. В статье **"A Relational Model of Data for Large Shared Data Banks"**, вышедшей в 1970 году, он показал, что **любое представление данных сводится к совокупности двумерных таблиц особого вида**, известного в математике как **отношение (relation).**



Положив теорию отношений в основу реляционной модели, Кодд обосновал реляционную **замкнутость отношений и ряда некоторых специальных операций, которые применяются сразу ко всему множеству строк отношения, а не к отдельной строке.** Указанная реляционная замкнутость означает, что **результатом выполнения операций над отношениями является также отношение, над которым в свою очередь можно осуществить некоторую операцию.** Из этого следует, что в данной модели можно оперировать **реляционными выражениями, а не только отдельными операндами в виде простых имен таблиц**.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ: ИСТОРИЯ ВОПРОСА

Одним из **основных преимуществ** реляционной модели является ее **однородность**. Все данные **рассматриваются как хранимые в таблицах и только в таблицах. Каждая строка такой таблицы имеет один и тот же формат.**

К числу достоинств реляционного подхода можно отнести:

- наличие небольшого **набора абстракций**, которые **позволяют сравнительно просто моделировать большую часть распространенных предметных областей** и допускают точные формальные определения, оставаясь интуитивно понятными;
- наличие простого и в то же время мощного **математического аппарата**, опирающегося главным образом на теорию множеств и математическую логику и **обеспечивающего теоретический базис реляционного подхода к организации БД**;
- возможность **ненавигационного манипулирования данными без необходимости знания конкретной физической организации** баз данных во внешней памяти.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ: ИСТОРИЯ ВОПРОСА

Кодд сформулировал **двенадцать правил**, которым **должна соответствовать** настоящая **реляционная база данных.**

Правило информации

- Вся информация в базе данных должна быть представлена **исключительно на логическом уровне и только одним способом** — в виде значений, содержащихся в таблицах.

Правило гарантированного доступа

- Логический доступ ко всем и каждому элементу данных **должен обеспечиваться путем использования комбинации имени таблицы, первичного ключа и имени столбца.**

Правило поддержки недействительных значений

- Должна быть реализована **поддержка недействительных значений**, которые отличаются от строки символов нулевой длины, строки пробельных символов и от нуля или любого другого числа и используются для представления отсутствующих данных независимо от типа этих данных.

Правило динамического каталога, основанного на реляционной модели

- Описание базы данных на **логическом уровне должно быть представлено в том же виде, что и основные данные**, чтобы пользователи, обладающие соответствующими правами, могли работать с ним с помощью того же реляционного языка.

Правило исчерпывающего подъязыка данных

- Реляционная система может поддерживать **различные языки и режимы взаимодействия, но должен существовать, по крайней мере, один язык, операторы которого можно представить в виде строк символов** в соответствии с некоторым четко определенным синтаксисом.

Правило обновления представлений

- Все представления, которые теоретически можно обновить, **должны быть доступны для обновления**.

Правило добавления, обновления и удаления

- Возможность работать с отношением как с одним операндом **должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных**.

Правило независимости физических данных

- Прикладные программы и утилиты для работы с данными должны **на логическом уровне оставаться нетронутыми при любых изменениях способов хранения** данных или методов доступа к ним.

Правило независимости логических данных

- Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми **при внесении в базовые таблицы любых изменений**.

Правило независимости условий целостности

- Должна существовать **возможность определять условия целостности**, специфические для конкретной реляционной базы данных.

Правило независимости распространения

- Реляционная СУБД **не должна зависеть от потребностей конкретного клиента**.

Правило единственности

- Если в реляционной системе есть **низкоуровневый язык** (обрабатывающий одну запись за один раз), **то должна отсутствовать возможность использования его** для того, чтобы **обойти правила и условия целостности**.

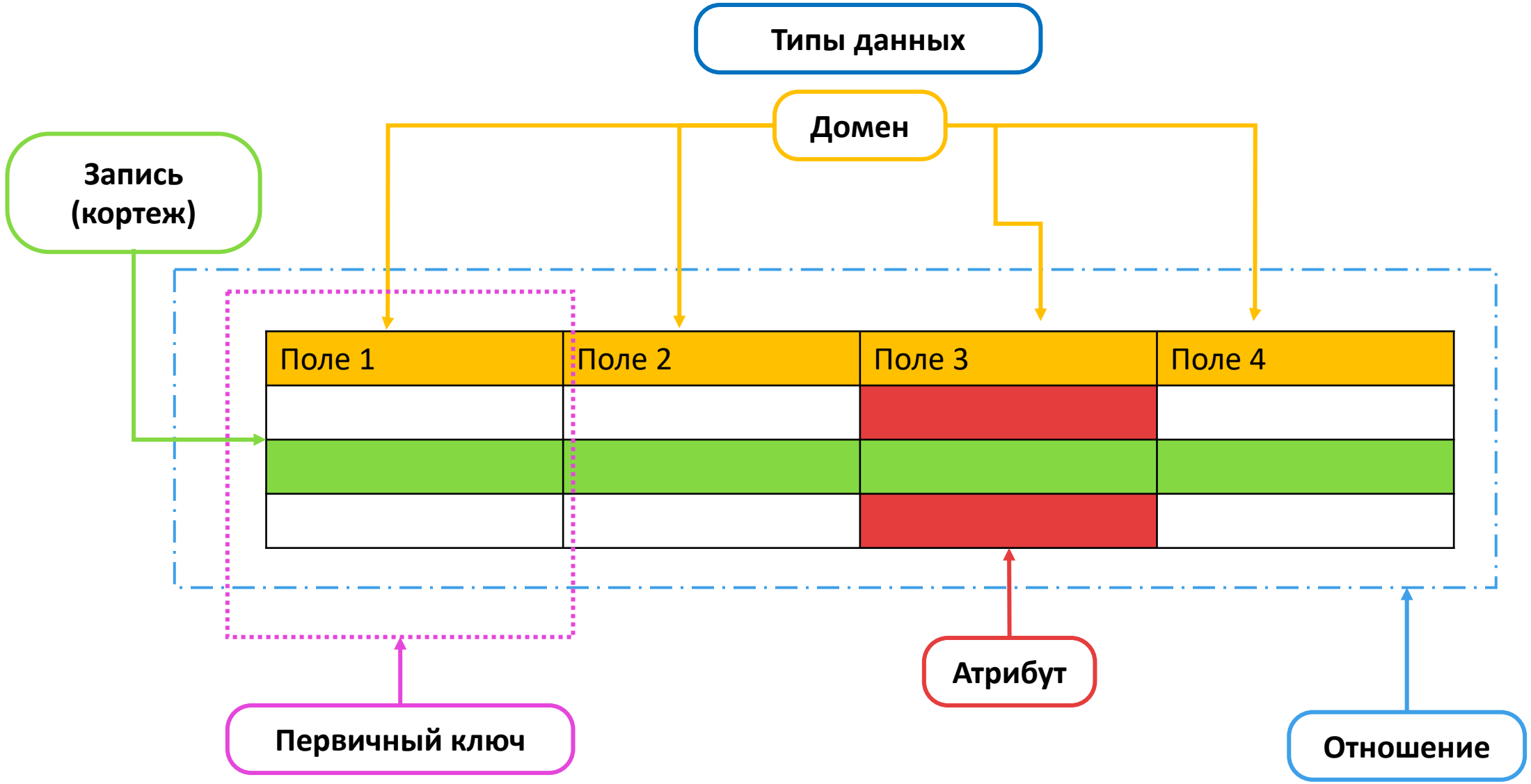
РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Реляционная модель данных представляет собой **совокупность данных, содержащихся в двухмерных таблицах,** соединённых между собой отношениями. Любые данные можно преобразовать в простую таблицу. **Такое представление является наиболее удобным и для пользователя, для и машины.**

РЕЛЯЦИОННАЯ МОДЕЛЬ СОСТОИТ ИЗ ТРЕХ ЧАСТЕЙ:

Структурная часть	• описывает, какие объекты рассматриваются реляционной моделью.
Целостная часть	• описывает ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных. **Это целостность сущностей и целостность внешних ключей.**
Манипуляционная часть	• описывает два эквивалентных способа манипулирования реляционными данными - **реляционную алгебру и реляционное исчисление**.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ



ОСНОВНЫЕ ЭЛЕМЕНТЫ РЕЛЯЦИОННОЙ МОДЕЛИ ДАННЫХ И ФОРМЫ ИХ ПРЕДСТАВЛЕНИЯ

Отношение	• Представляет собой **двумерную таблицу**, содержащую некоторые данные
Схема отношения	• Строка заголовков столбцов таблицы (**заголовок таблицы**)
Кортеж	• **Строка таблицы**, характеризующая отдельный объект
Сущность	• **Объект** любой природы **данные о котором хранятся в БД**. Данные о сущности хранятся в одном или нескольких отношениях
Домен	• **Множество допустимых значений** определенного атрибута отношений
Атрибут	• **Поименованный столбец** отношения
Значение атрибута	• **Значение в поле** записи
Тип данных	• **Тип значений элементов** таблицы. Каждый домен образует значение одного типа данных. Например, числовые или символьные
Ключ	• **Уникальное значение поля** для каждой записи, то есть, для каждой строки в таблице значение ключа будет разным.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Реляционная база данных представляет собой **совокупность двумерных таблиц. Строки** таблицы содержат **сведения об объектах**. Каждая **строка** таблицы содержит **разнообразную (разного типа) информацию**. Каждый **столбец** в таблице должен содержать только **определенный тип информации**.

Данные в реляционной таблице должны удовлетворять следующим принципам:

- **каждое значение поля должно быть атомарным**, т.е. не расчленяемым на несколько значений;
- **значения данных домена** (в одном и том же столбце) должны принадлежать к **одному и тому же типу данных**, доступному для использования в данной СУБД;
- **каждая запись в таблице уникальна**, т.е. в таблице не существует двух записей с полностью совпадающим набором значений ее полей;
- **каждое поле имеет уникальное имя**;
- **последовательность полей** в таблице **несущественна**;
- **последовательность записей** в таблице **несущественна**.

ОТНОШЕНИЯ

Отношение — это двумерная таблица, имеющая уникальное имя и состоящая из строк и столбцов, где строки соответствуют записям, а столбцы — атрибутам.

Множество кортежей называется телом отношения. Тело отношения отражает состояние сущности, поэтому во времени оно постоянно меняется. Тело отношения характеризуется **кардинальным числом, которое равно количеству содержащихся в нем кортежей.**

COMPANY	PRODUCT	DRIVERS
ООО «Темная сторона»	Печеньки	Петр
ООО «Темная сторона»	Чай	Петр
ОАО «Овощи»	Огурцы	Олег
ООО «Молочко»	Йогурт	Иван

Одной из главных характеристик отношения является его **степень (ранг или арность).** Степень отношения **определяется количеством атрибутов, которое в нем присутствует.** Отношение с одним атрибутом называется унарным, с двумя атрибутами — бинарным, с тремя — тернарным, с n атрибутами n-арным.

СВОЙСТВА И ВИДЫ ОТНОШЕНИЙ

Отношение по структуре подобно таблице, обладающей определенными **свойствами:**

- отношение **имеет имя**, которое **отличается от имен всех других отношений**;
- отношение представляется в виде **табличной структуры**;
- каждый **атрибут имеет уникальное имя**, его значения берутся из одного и того же домена;
- каждый **компонент кортежа является простым**, атомарным значением, не состоящим из группы значений;
- **упорядочение атрибутов** теоретически **несущественно**, однако оно может влиять на эффективность доступа к кортежам;
- все строки **(кортежи) должны быть различны**;
- теоретически **порядок следования кортежей не имеет значения**.

СВОЙСТВА И ВИДЫ ОТНОШЕНИЙ

В реляционной теории встречается несколько видов отношений, но не все они поддерживаются реальными системами.

Именованное отношение — это переменная отношения, определенная в СУБД посредством специальных операторов

Базовое отношение — это именованное отношение, являющееся частью базы данных

Производное отношение — это отношение, определенное посредством реляционного выражения через базовые отношения

Представление — это именованное виртуальное производное отношение, представленное в системе исключительно через определение в терминах других именованных отношений

Снимки — это отношения, подобные представлениям, но они сохраняются, доступны для чтения и периодически обновляются

Результат запроса — это неименованное производное отношение, получаемое в результате запроса, которое для сохранения необходимо преобразовать в именованное отношение

Хранимое отношение — это отношение, которое поддерживается в физической памяти

Отношения **используются для представления сущностей**, а также для **представления связей между сущностями.**

СВЯЗИ МЕЖДУ СУЩНОСТЯМИ

Сущность (entity) – **это «предмет» реального мира**, который может быть идентифицирован некоторым способом, отличающим его от других «предметов» (определение Питера Чена).

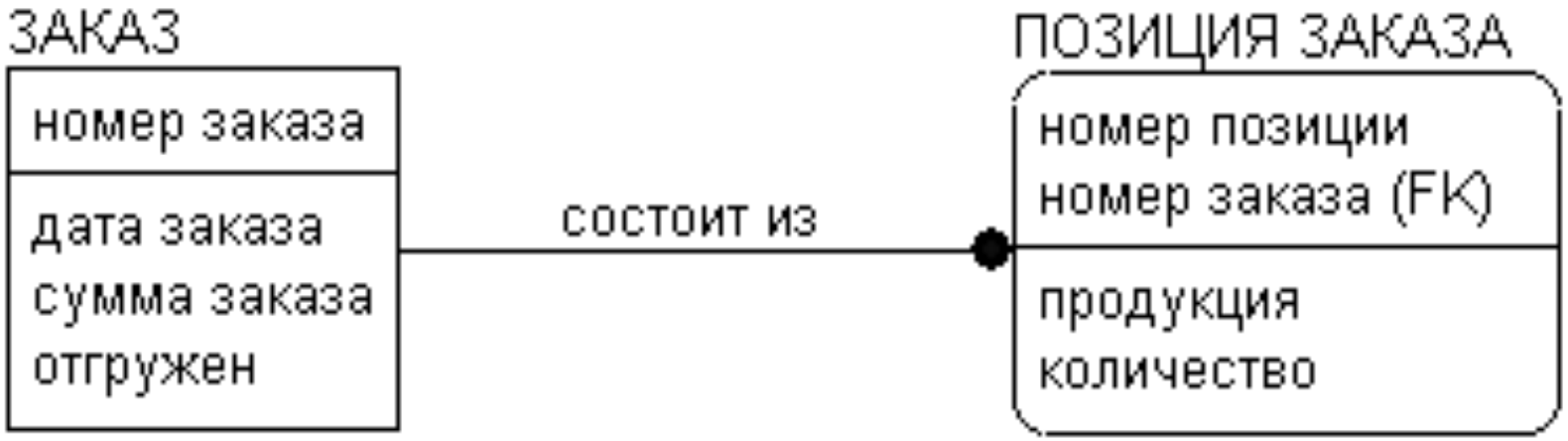
Существует два типа сущностей:

- **зависимая сущность** (dependent entity) – для определения экземпляра такой сущности необходимо сослаться на экземпляр независимой сущности, с которой связана зависимая сущность;
- **независимая сущность** (independent entity) – для определения экземпляра сущности нет необходимости ссылаться на другие сущности. Пример: сущности – заказ и позиция заказа. Для определения заказа (сущности) нет необходимости ссылаться на позиции этого заказа (другие сущности).

Связь (relationship) – это логическая **ассоциация, устанавливаемая между сущностями,** которая представляет бизнес-правило или ограничение.

СВЯЗИ МЕЖДУ СУЩНОСТЯМИ. ВИДЫ СВЯЗЕЙ

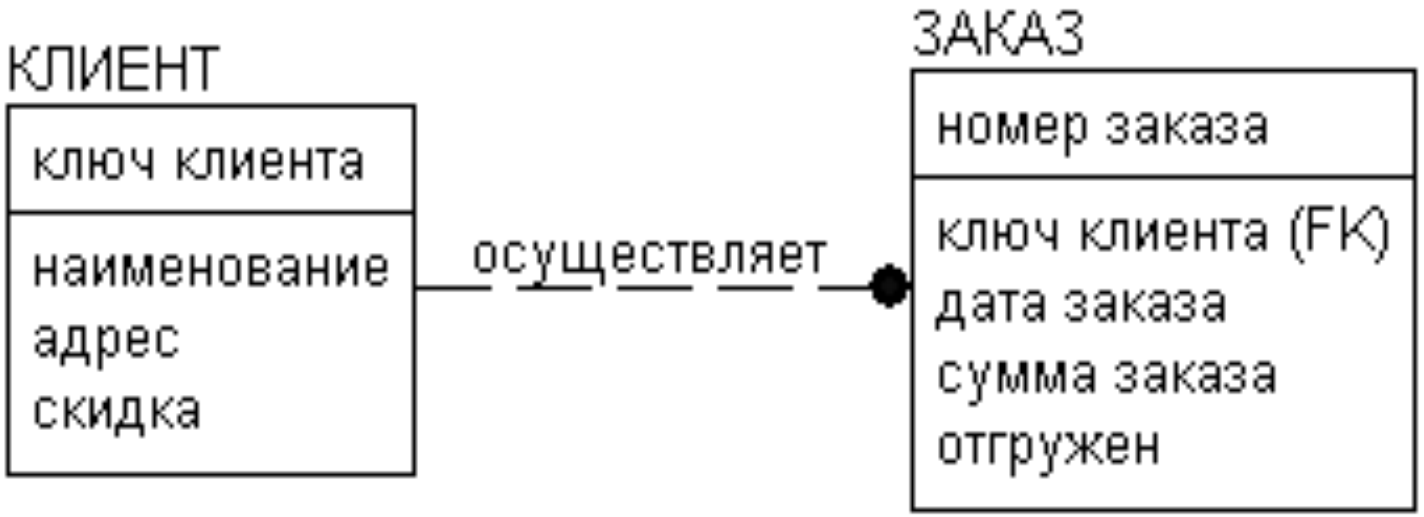
Идентифицирующая (identifying relationship) – **указывает на то, что дочерняя сущность в связи является зависимой от родительской сущности, т.е. экземпляр зависимой сущности может быть однозначно определен, только если в этом экземпляре есть ссылка на экземпляр независимой сущности.** В идентифицирующем отношении сущность-потомок всегда является зависимой от идентифицирующей сущности. Первичный ключ родительской сущности наследуется в качестве первичного ключа дочерней сущностью (внешний ключ).



Идентифицирующая связь отображается сплошной линией, причем дочерняя сущность является зависимой и поэтому отображается прямоугольником со скругленными углами.

СВЯЗИ МЕЖДУ СУЩНОСТЯМИ. ВИДЫ СВЯЗЕЙ

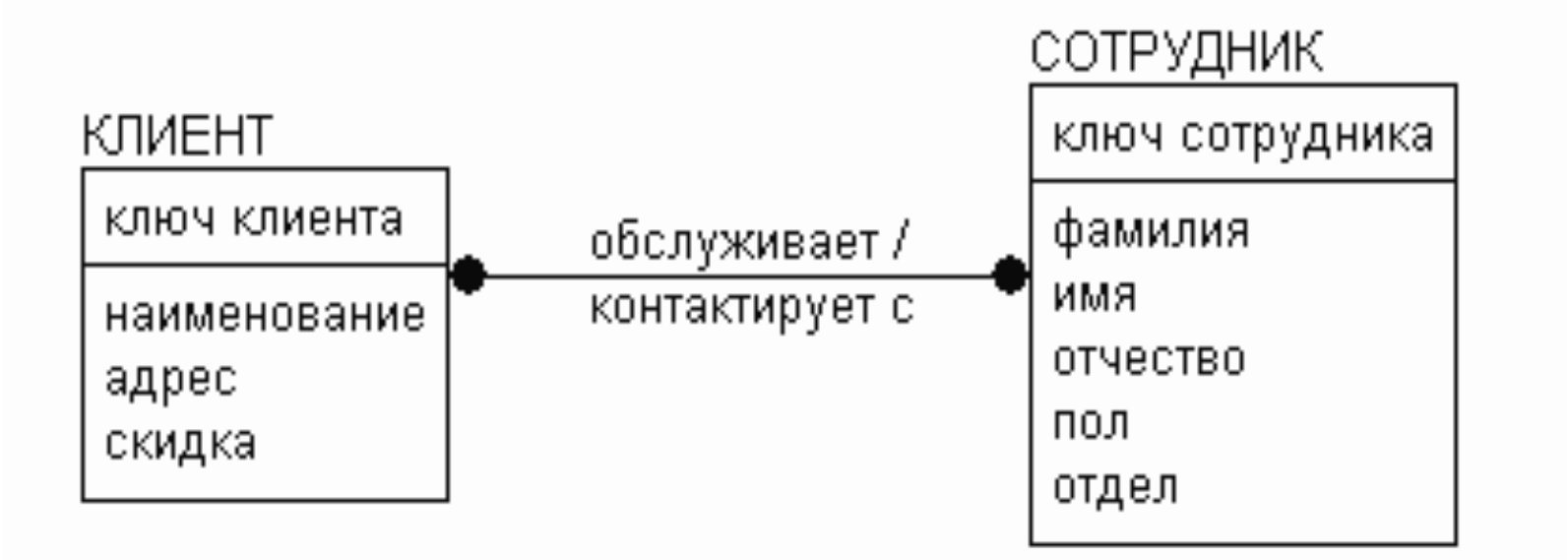
Неидентифицирующая (non-identifying relationship) **– показывает на зависимость между родительской и дочерней сущностями, при этом экземпляр дочерней сущности может быть однозначно идентифицирован без ссылки на экземпляр родительской сущности,** т.е. родительская сущность связана с дочерней сущностью, но однозначно не определяет ее. Первичный ключ родительской сущности наследуется в качестве неключевого атрибута дочерней сущности.



Неидентифицирующая связь отображается штриховой линией

СВЯЗИ МЕЖДУ СУЩНОСТЯМИ. ВИДЫ СВЯЗЕЙ

Многие ко многим – неспецифичный тип связи, **во многом свидетельствует о незавершенности анализа.** На конечных этапах моделирования данных все связи многие-ко-многим преобразуются в другие (идентифицирующие и неидентифицирующие) типы связей. **В связи многие-ко-многим не выделяется родительская и дочерняя сущность и, как правило, используется две глагольных фразы.**



СВЯЗИ МЕЖДУ СУЩНОСТЯМИ. ВИДЫ СВЯЗЕЙ

Иерархическая – **указывает на связь сущности саму на себя.** Если у родительской таблицы линия имеет **белый ромб**, то это означает, что **внешний ключ** дочерней таблицы **может содержать нулевые значения**. **Если связь является иерархической рекурсией, то первичный ключ этой сущности мигрирует в область атрибутов этой же сущности.** При этом внешнему ключу должно быть обязательно назначено имя роли.

Такой вид рекурсивной связи называется иерархической рекурсией (hierarchical recursion) или рыбный крючок (fish hook) и задает связь, когда **руководитель (экземпляр родительской сущности) может иметь множество подчиненных (экземпляров дочерней сущности), но подчиненный имеет только одного руководителя.**

РЕЛЯЦИОННЫЕ КЛЮЧИ

Ключ – это атрибут (несколько одиночных или составных атрибутов) в отношении, которые однозначно идентифицируют кортеж отношения.

Простой ключ – ключ, состоящий из **одного** атрибута.

Составной (сложный) ключ – ключ, состоящий из **нескольких** атрибутов.

Первичный ключ (primary key, PK) – это набор атрибутов, **значение которых однозначно определяют каждый экземпляров сущности.** Атрибуты, которые могут быть выбраны первичными ключами, называются **кандидатами в ключевые атрибуты** (потенциальные атрибуты).

Правила выбора первичного ключа:

- **уникальным образом идентифицировать** экземпляр сущности;
- **не использовать NULL** значений;
- **не изменяться со временем** (экземпляр идентифицируется при помощи ключа, при изменении ключа, соответственно меняется экземпляр);
- быть **как можно более короткими для использования** индексирования и получения данных.

РЕЛЯЦИОННЫЕ КЛЮЧИ

Потенциальный ключ – атрибуты, **претендующие** на роль **первичного ключа**.

Суррогатный ключ (surrogate key) – **атрибут, значение которого создается искусственно и не имеет отношения к предметной области, это произвольный номер, который уникальным образом определяет запись в сущности.** Суррогатный ключ лучше всего подходит на роль первичного ключа потому, что является коротким и быстрее всего идентифицирует экземпляры в объекте.

Альтернативный ключ (alternate key, AKn,m) – это **потенциальный ключ, который не выбран первичными.** Атрибут входит в состав альтернативного ключа n в позиции m. С помощью альтернативных ключей часто отображают различные индексы доступа к данным в конечной реализации реляционной базы. Являются основой для создания уникальных индексов.

Внешний ключ (foreign key, FK) – первичный ключ, наследуемый от родительской сущности через специфическое отношение.

ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Поддержка целостности базы данных **реализуется посредством ряда ограничений**, накладываемых на данные.

Первый тип ограничений накладывает ограничения на атрибуты: каждый атрибут определяется на своем домене, или наоборот: домен атрибута задает множество значений, которые может принимать атрибут.

Важными понятиями в теории реляционных баз данных являются категорная целостность и целостность на уровне ссылок.

Категорная целостность ограничивает набор значений первичных ключей базовых отношений – **кортеж не может записываться в БД до тех пор, пока значения его ключевых атрибутов не будут полностью определены.**

База данных, в которой **все непустые внешние ключи ссылаются на текущие значения ключей другого отношения**, обладает **целостностью на уровне ссылок.**