НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

МИЭМ НИУ ВШЭ

ИНФОРМАЦИОННЫЙ ПРАКТИКУМ.
ОСНОВЫ ПРОЕКТИРОВАНИЯ БД.
ИНСТРУМЕНТЫ ПРОЕКТИРОВАНИЯ БД.

Лекция 2

Белов Александр Владимирович

Москва, 2021

Содержание

Реляционная модель. Основные понятия (1)

В основе реляционной модели лежит понятие **«отношение» (relation)**. С математической точки зрения **отношение** – это некая структура, которая формально описывает свойства различных объектов и их взаимосвязи. Понятие отношения широко используется в теории множеств.
Понятие «отношение» и реляционная алгебра лежат в основе теории реляционных баз данных.

При этом реляционная модель данных представляется набором таблиц, используемых для представления данных и связей между ними. **Таблица** – это неформальное понятие и используется для наглядной, интуитивно понятной иллюстрации отношения.

Каждая таблица содержит набор **записей (строк),** представляющих данные определенного формата. Каждая запись содержит фиксированное количество **полей (столбцов/атрибутов)**.

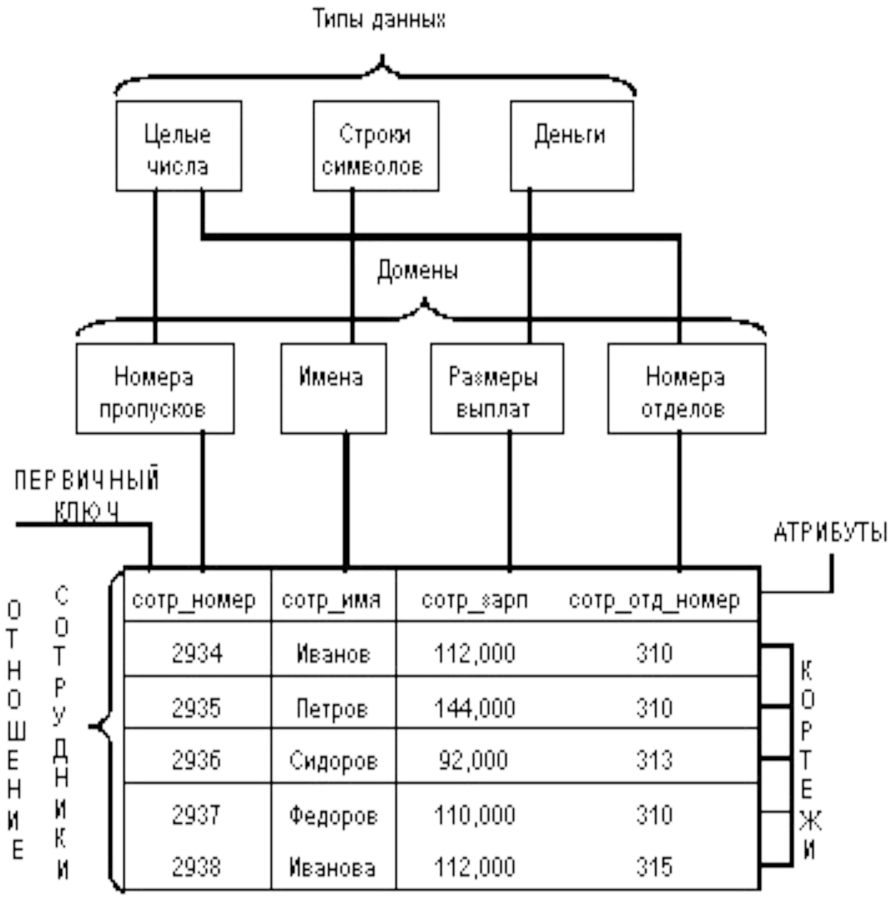
Реляционная модель. Основные понятия (2)

Тип данных – интуитивно, по аналогии с ЯП
Пример: строки символов, целые числа и "деньги"

Домен - потенциально допустимое множество значений типа
Пример: Элемент домена «Имена» не может начинаться с «Ъ»
Данные сравнимы, если относятся к одному домену (Oracle использует домены с v7.0)

Схема отношения - поименованное множество пар {атрибут, домен} или, в общем случае, {атрибут, тип}

Схема БД - набор именованных схем отношений



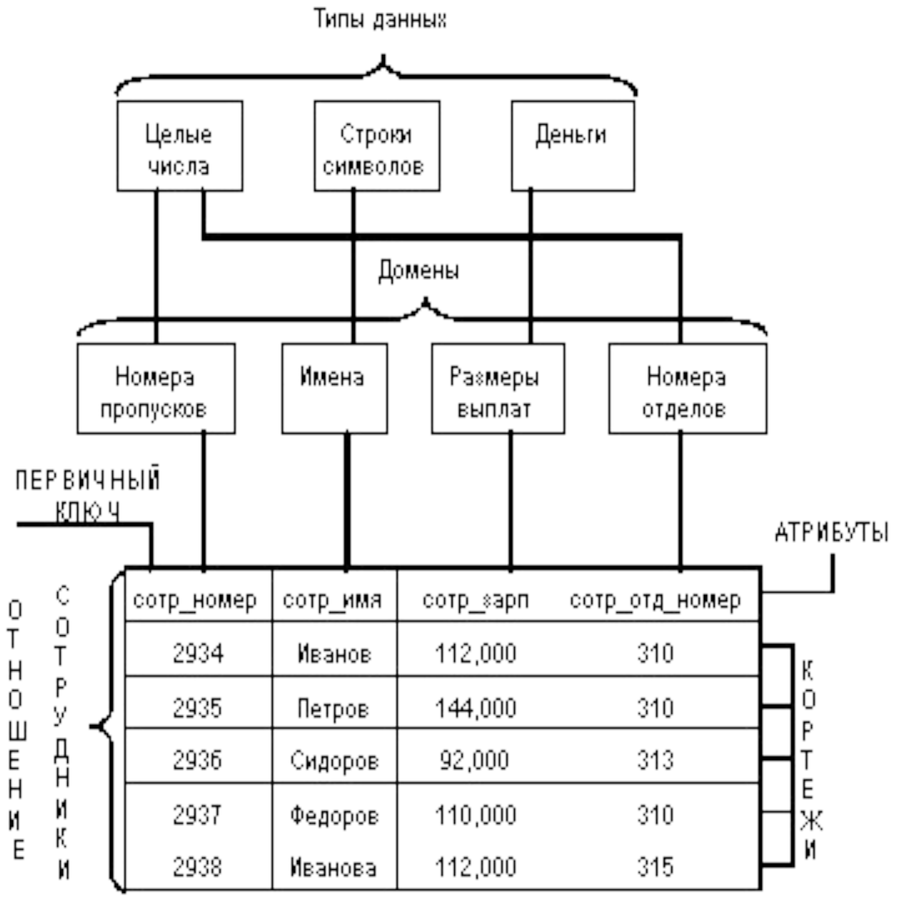
Реляционная модель. Основные понятия (2)

Кортеж – набор именованных значений заданного типа в форме {атрибут, значение}

Отношение - множество кортежей, соответствующих одной схеме.

Таблица - графическое представление отношения (см. пр. «СОТРУДНИКИ»):
заголовок - схема отношения
строки - кортежи отношения-экземпляра
столбцы - атрибуты.

Реляционная база данных (РБД) – набор отношений (таблиц)



Ключи

Суперключ (Superkey) – множество, состоящее из одного или нескольких атрибутов, которые позволяют уникально идентифицировать сущность из множества сущностей.

Пример: *ИНН – позволяет уникально идентифицировать клиента, т.е. позволяет отличить один экземпляр сущности Клиент от другого экземпляра.*

Суперключи, которые образуют множество, у которого нет подмножества, состоящего из суперключей. называют **ключами-кандидатами (candidate keys)** или **потенциальными ключами.**

Будем называть **первичным ключом** (**Primary key)** выбранный ключ-кандидат, который будет использоваться для уникальной идентификации сущности во множестве сущностей**.**

Множество сущностей, в котором нельзя выбрать атрибуты, которые могут являться первичными ключами, называется **слабым (weak).**

Множество сущностей, которое имеет первичный ключ, называется **сильным (strong).**

Сильное множество является **главной (доминирующей) сущностью.** Слабое – **зависимым**, подчиненным множеством сущностей.

Преобразование E-R модели в табличную форму

E-R-диаграмма может быть преобразована в набор таблиц с уникальными именами. Каждая строка в таблице соответствует одному экземпляру множества сущностей/отношений, а столбец – атрибуту сущности/отношения

Если имеются слабые (A) и сильные (B) множества сущностей, то мощность отношения R между ними – M:1.

Пример. Сущность Transaction (Tr_num, Date, Amount) – слабая, Account (Acc_num, Balance) – сильная. Отношение между ними – Log (Acc_num, Tr_num)

Т.к. R не имеет описательных атрибутов, то для построения таблицы A необходимо добавить первичный ключ из таблицы B.

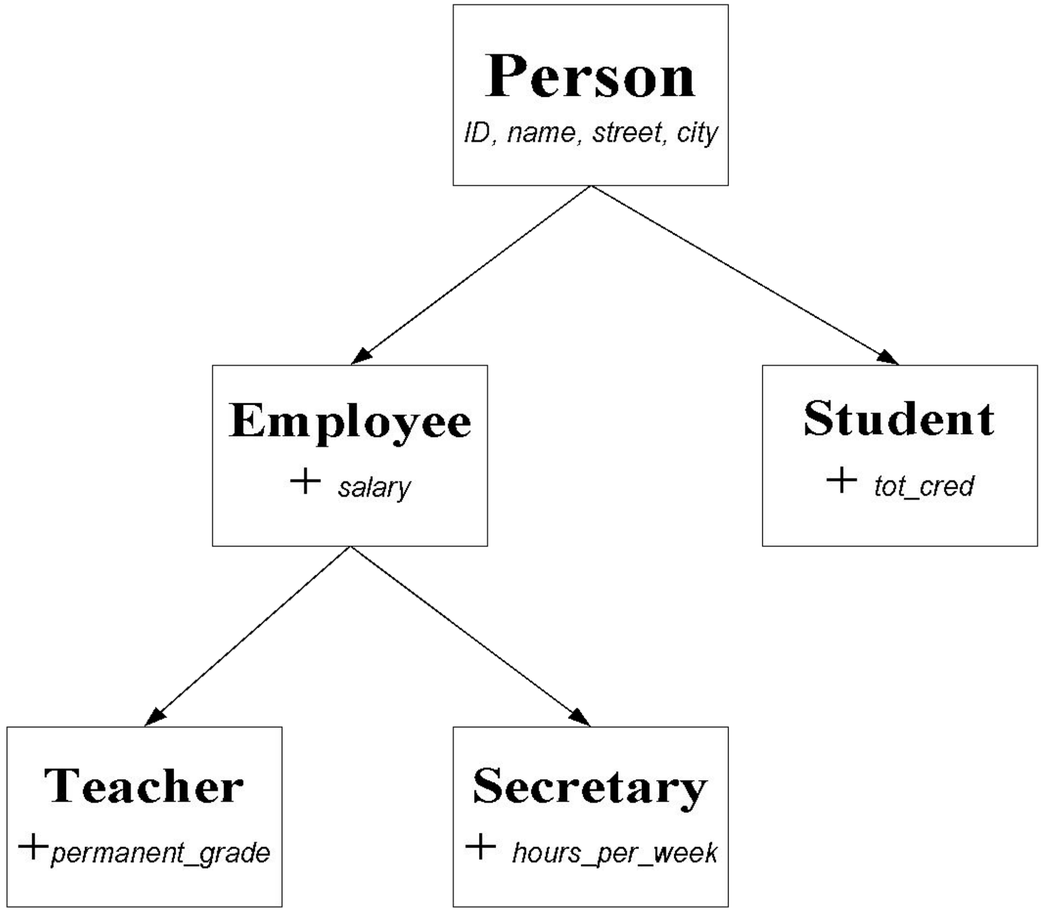
Пример. Transaction (Tr_num, Date, Amount) преобразуется в таблицу вида Transaction (Acc_num , Tr_num, Date, Amount). Эта таблица – избыточна

В общем случае отношение между слабым и сильным множеством сущностей будет всегда избыточным.

Методы проектирования РБД. Детализация (Specialization)

Детализация – это метод проектирования схемы БД «сверху вниз», который предполагает добавление новых атрибутов сущности/отношения на каждом шаге детализации.

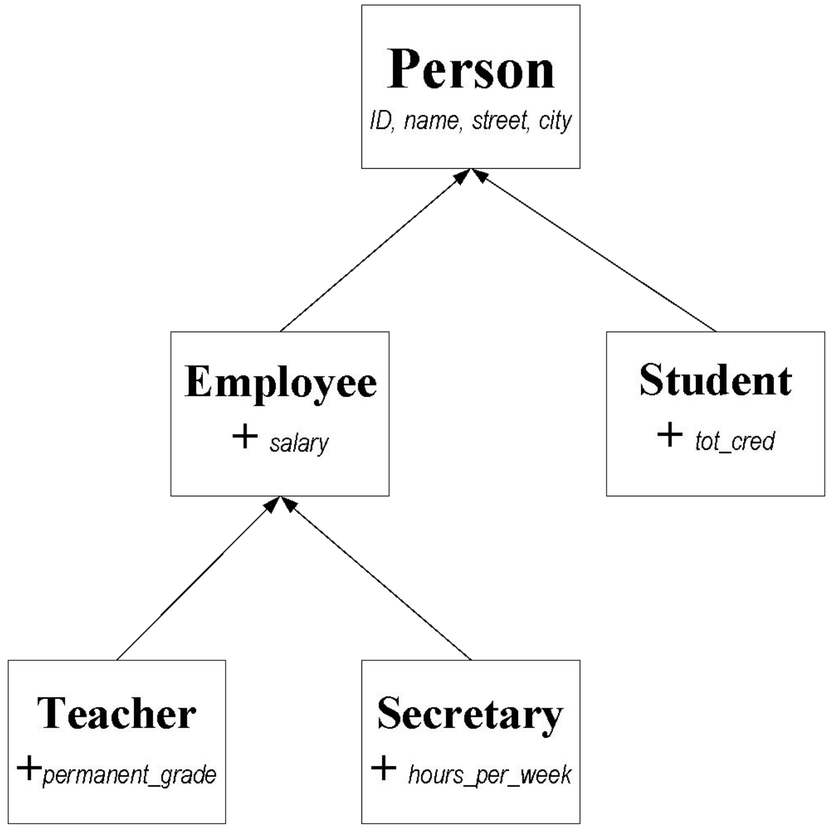
Пример. Сущности Student (*ID, name, street, city, tot_cred*) и Employee (*ID, name, street, city, salary*) отличаются атрибутами – *tot_cred* и *salary*. Сущность Employee (*ID, name, street, city, salary*) может быть детализирована. В результате появятся две новых сущности - Teacher и Secretary с общими атрибутами: *ID, name, street, city, salary* и атрибутом *permanent_grade* и *hours_per_week, соответственно.*



Методы проектирования РБД. Обобщение (Generalization)

Обобщение – это метод проектирования схемы БД, который предполагает образование нового отношения между сущностями, которые имеют общие признаки. Подход к проектированию использует принцип восходящего проектирования «снизу вверх»
.

Пример. Сущности Saving-account и Loan-account имеют общие атрибуты – *acc_num* и *balance*. Образуется новая сущность/отношение, которое будет родительским по отношению к дочерним сущностям.



Преобразование E-R модели в табличную форму с использованием методов Generalization/Specialization

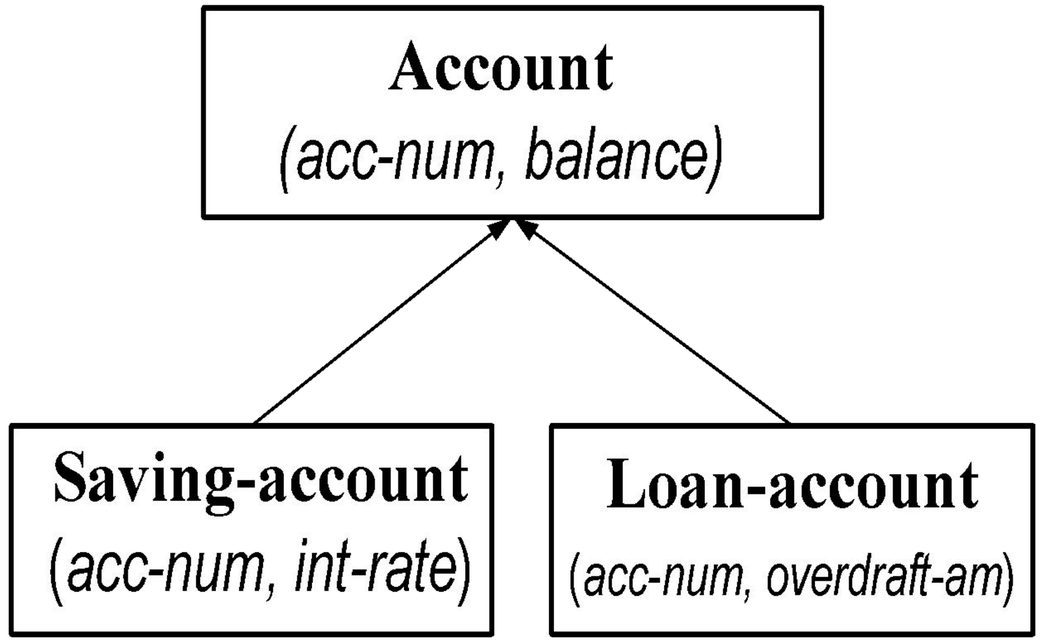
Существует два метода преобразования E-R диаграммы в табличную форму с использованием **операции обобщения**:

- Создать таблицу родительскую и две дочерних с первичным ключом из родительской.

Пример. Сущности Account (*acc-num, balance*). Saving-account (*acc-num, int-rate*) и Loan-account (*acc-num, overdraft-am*).

- Не создавать родительскую таблицу. В каждую из дочерних таблиц добавить атрибуты из родительской.

Пример. Saving-account (*acc-num, balance, int-rate*) и Loan-account (*acc-num, balance, overdraft-am*).



Вопросы для самоконтроля

1. Что является результатом логического проектирования БД?
2. Чем отличается первичный ключ отношения от потенциального ключа?
3. Если рассматривать ER-диаграмму как граф, то что означает, с т.зрения, структуры БД:

-Несвязанный граф

-Циклический граф

Проектирование реляционной БД

Цель проектирования реляционной базы данных - создать набор отношений, который позволит хранить информацию без ненужной избыточности, а также легко извлекать информацию.

Для достижения указанной цели используется метод нормализации отношений. Чтобы определить находится ли схема отношения в одной из желаемых нормальных форм, нам нужна информация о реальном объекте, которое мы моделируем с помощью базы данных.

Проблемы при проектировании реляционной БД

Ошибки в проектировании БД могут привести к:

- избыточности данных (повторение одних и тех же данных в разных структурных объектах);
- потере данных при операциях модификации отношений;
- неадекватному представлению информации в БД (построена неправильная модель предметной области)

Аномалии модификации отношений

Аномалия - неадекватность модели данных предметной области, (что говорит на самом деле о том, что логическая модель данных попросту неверна!).

Т.к. аномалии проявляют себя при выполнении операций, изменяющих состояние базы данных, то различают следующие виды аномалий:

- **Аномалии вставки (INSERT)**
- **Аномалии обновления (UPDATE)**
- **Аномалии удаления (DELETE)**

Пример аномалий баз данных

Рассмотрим отношение в БД «Отдел кадров», имеющее следующие атрибуты: **СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ (*Н_СОТР, ФАМ, Н_ОТД, ТЕЛ, Н_ПРО, ПРОЕКТ, Н_ЗАДАН*).** Представим это отношение в виде таблицы, отражающей состояние предметной области

Н_СОТР	*ФАМ*	*Н_ОТД*	*ТЕЛ*	*Н_ПРО*	*ПРОЕКТ*	*Н_ЗАДАН*
1	Иванов	1	11-22-33	*1*	Космос	1
1	Иванов	1	11-22-33	*2*	Климат	1
2	Петров	1	11-22-33	*1*	Космос	2
3	Сидоров	2	33-22-11	*1*	Космос	3
3	Сидоров	2	33-22-11	*2*	Климат	2

Аномалия вставки

В отношение **СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ** нельзя вставить данные о сотруднике, который пока не участвует ни в одном проекте.
Точно также нельзя вставить данные о проекте, над которым пока не работает ни один сотрудник.
Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Аномалия обновления

Фамилии сотрудников, наименования проектов, номера телефонов повторяются во многих кортежах отношения. Поэтому если сотрудник меняет фамилию, или проект меняет наименование, или меняется номер телефона, то такие изменения необходимо *одновременно* выполнить во всех местах, где эта фамилия, наименование или номер телефона встречаются, иначе отношение станет некорректным (например, один и тот же проект в разных кортежах будет называться по-разному). Таким образом, обновление базы данных одним действием реализовать невозможно.

Причина аномалии - избыточность данных, порожденная тем, что в одном отношении хранится разнородная информация.

Аномалия удаления

При удалении некоторых данных может произойти потеря другой информации. Например, если закрыть проект "Космос" и удалить все строки, в которых он встречается, то будут потеряны все данные о сотруднике Петрове.
Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

Нормализация отношений

Для улучшения структуры отношений, чтобы предотвратить появление аномалий, используется механизм нормализации отношений.

Нормализация отношений – это формальный аппарат ограничений на их формирование, который позволяет устранить дублирование данных, обеспечить их непротиворечивость и уменьшить затраты на поддержание базы данных.

На практике наиболее часто используются понятия первой (1НФ), второй (2НФ), третьей (3НФ) нормальных форм.

1-я нормальная форма

Определение 1. Отношение называется **нормализованным или приведенным к 1НФ**, если все его атрибуты простые или атомарные (неделимые).

Отношение, находящееся в 1НФ, будет иметь следующие свойства:

- В отношении нет повторяющихся записей;
- Кортежи не упорядочены
- Атрибуты не упорядочены и различаются по наименованиям
- Все значения атрибутов атомарные

Любое отношение автоматически находится в 1НФ.

Функциональная зависимость (1)

Нормализация основана на понятии функциональной зависимости атрибутов отношения.

Определение 1. Пусть **R**-отношение. Множество атрибутов **Y** ***функционально зависимо*** от множества атрибутов **X** (**X** ***функционально определяет Y***) тогда и только тогда, когда *для любого состояния* отношения **R** для любых кортежей r_1, r_2 из того, что $\mathbf{r_1.X=r_2.X}$ следует что $\mathbf{r_1.Y=r_2.Y}$ (т.е. во всех кортежах, имеющих одинаковые значения атрибутов **X**, значения атрибутов **Y** также совпадают *в любом состоянии* отношения **R**). Символически функциональная зависимость записывается

$$X \longrightarrow Y$$

Функциональная зависимость (2)

Функциональная зависимость – это семантическое понятие. Она возникает, когда по значениям одних данных в предметной области можно определить значения других данных.

Зависимость атрибутов от ключа отношения:

$$\{Н\_СОТР, Н\_ПРО\} \longrightarrow Н\_ОТД$$

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

$$Н\_СОТР \longrightarrow ФАМ$$

Зависимость наименования проекта от номера проекта:

$$Н\_ПРО \longrightarrow ПРОЕКТ$$

2-я нормальная форма

Определение 2. Отношение **R** находится во ***второй нормальной форме*** (***2НФ***) тогда и только тогда, когда отношение **R** находится в 1НФ и *нет неключевых атрибутов, зависящих от части сложного ключа*. (***Неключевой атрибут*** - это атрибут, не входящий в состав никакого потенциального ключа).

Замечание. Если потенциальный ключ (potential key, набор ключей, который может уникально идентифицировать кортеж в отношении) отношения является простым, то отношение автоматически находится в 2НФ.

Пример.

Отношение **СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ** не находится в 2НФ, т.к. есть атрибуты, зависящие от части сложного ключа:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника, является зависимостью от части сложного ключа:

Н_СОТР $\longrightarrow$ **ФАМ,** ***Н_СОТР*** $\longrightarrow$ **Н_ОТД**

Для того, чтобы устранить зависимость атрибутов от части сложного ключа, нужно произвести декомпозицию отношения на несколько отношений. При этом те атрибуты, которые зависят от части сложного ключа, выносятся в отдельное отношение.

Понятие логической модели данных

Отношение **СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ** декомпозируем на три отношения:

- **СОТРУДНИКИ_ОТДЕЛЫ**,
- **ПРОЕКТЫ**,
- **ЗАДАНИЯ**.

Отношения, полученные в результате декомпозиции, находятся в 2НФ. Действительно, отношения **СОТРУДНИКИ_ОТДЕЛЫ** и **ПРОЕКТЫ** имеют простые ключи, следовательно автоматически находятся в 2НФ, отношение **ЗАДАНИЯ** имеет сложный ключ, но единственный неключевой атрибут **Н_ЗАДАН** функционально зависит от всего ключа {***Н_СОТР***, ***Н_ПРО***}.

Часть аномалий обновления устранена.

Фамилии сотрудников и наименования проектов теперь хранятся без избыточности. Если сотрудник сменит фамилию или проект сменит наименование, то такое обновление будет произведено в одном месте.

3-я нормальная форма

Атрибуты называются ***взаимно независимыми***, если ни один из них не является функционально зависимым от другого.

Определение 3. Отношение **R** находится в ***третьей нормальной форме*** (***3НФ***) тогда и только тогда, когда отношение находится в 2НФ и *все неключевые атрибуты взаимно независимы*.

Пример. Отношение **СОТРУДНИКИ_ОТДЕЛЫ** не находится в 3НФ, т.к. имеется функциональная зависимость неключевых атрибутов (зависимость номера телефона от номера отдела): **Н_ОТД ⟶ТЕЛ**

Для того, чтобы устранить зависимость неключевых атрибутов, нужно произвести декомпозицию отношения на несколько отношений. При этом *те неключевые атрибуты, которые являются зависимыми,* выносятся в отдельное отношение.

Отношение **СОТРУДНИКИ_ОТДЕЛЫ** декомпозируем на два отношения - **СОТРУДНИКИ** (***Н_СОТР***, **ФАМ**, **Н_ОТД**) и **ОТДЕЛЫ** (***Н_ОТД***, **ТЕЛ**).

Вывод. Все обнаруженные аномалии отношения устранены. Реляционная модель, состоящая из четырех отношений **СОТРУДНИКИ**, **ОТДЕЛЫ**, **ПРОЕКТЫ**, **ЗАДАНИЯ**, находящихся в третьей нормальной форме, является моделью, адекватной описанной предметной области.

Вопросы для самоконтроля

1. В каких случаях (пример) отношение может находиться не в 1НФ?
2. Что такое функциональная зависимость?
3. Что подразумевается под дублированием информации? Почему дублирование данных может указывать на плохой дизайн реляционной базы данных?

Методы и инструменты проектирования БД

- Инструменты проектирования ИС в части БД основываются на методах проектирования
- Инструменты проектирования БД относятся к CASE-средствам (Computer Aided Software Engineering)
- Преимущество CASE–средств проектирования – это разграничение процессов проектирования и разработки
- Методы проектирования БД – структурный подход и объектно-ориентированный

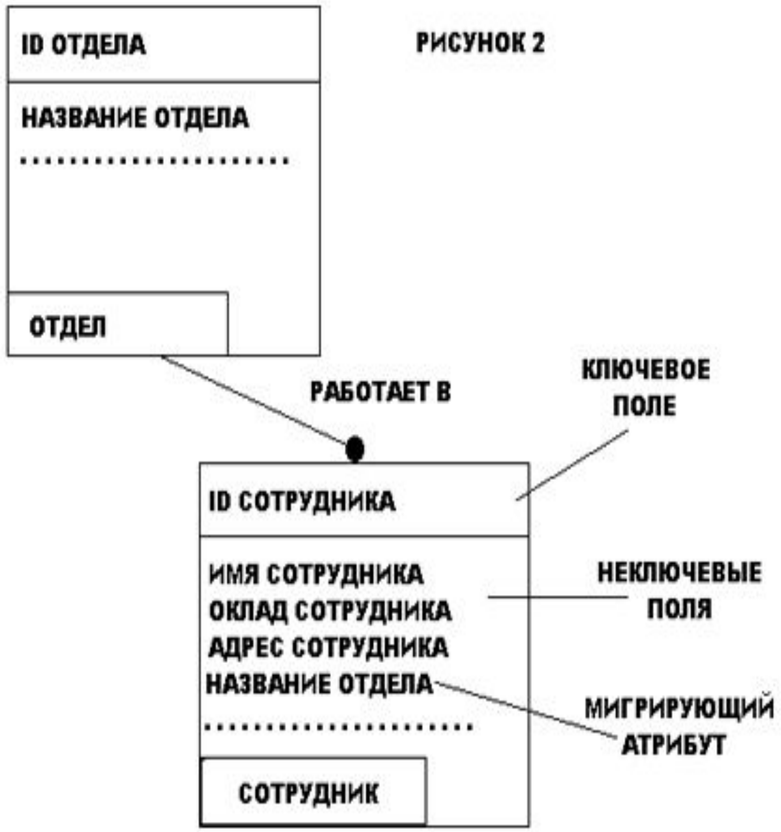
Структурный подход к проектированию БД

Структурные методы проектирования ИС:

- **SADT** (Structured Analysis and Design Technique). Методология, использующая графический язык для процесса моделирования ИС
- **DFD** (Data Flow Diagram). Описывает информационные потоки, источники данных, хранилища данных в ИС
- **ERD** (Entity-Relationship Diagram). Описывает структуру БД. Наиболее известная нотация моделирования БД - IDEF1.

IDEF1X (Integrated DEFinition) - нотация применяемая для построения информационной модели, отображающей структуру и содержание информационных потоков, разработки на её основе базы данных. Данная нотация предназначена для разработки реляционных баз данных на основе модели «сущность-связь».

IDEF1X. Основные объекты



Сущность описывается в диаграмме IDEF1X графическим объектом в виде прямоугольника.

Каждый прямоугольник, отображающий собой сущность, разделяется горизонтальной линией на часть, в которой расположены ключевые поля и часть, где расположены неключевые поля. Верхняя часть называется ключевой областью, а нижняя часть областью данных.

Ключевая область объекта СОТРУДНИК содержит поле "Уникальный идентификатор сотрудника", в области данных находятся поля "Имя сотрудника", "Адрес сотрудника", "Телефон сотрудника" и т.д.

IDEF1X. Основные объекты

РИСУНОК 1

Связи в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями. Связи это суть глаголы, которые показывают, как соотносятся сущности между собой. Ниже приведен ряд примеров связи между сущностями:

- Отдел <состоит из> нескольких Сотрудников
- Самолет <перевозит> нескольких Пассажиров.
- Сотрудник <пишет> разные Отчеты.

Сущности в IDEF1X

Сущность **Отдел** называется ***родительской***, а **Сотрудник** - ***дочерней***. Связи отображаются в виде линии между двумя сущностями с точкой на одном конце и глагольной фразой, отображаемой над линией.

Если сущности в IDEF1X диаграмме связаны, связь передает ключ (или набор ключевых атрибутов) дочерней сущности. Эти атрибуты называются ***внешними ключами***. Внешние ключи определяются как атрибуты первичных ключей родительского объекта, переданные дочернему объекту через их связь.

Дочерняя сущность, уникальность которой зависит от атрибута внешнего ключа, называется ***зависимой*** сущностью. В примере на рис.1

Пример. Сущность **СОТРУДНИК** является зависимой сущностью потому, что его идентификация зависит от сущности **ОТДЕЛ**.

В обозначениях IDEF1X зависимые сущности представлены в виде закругленных прямоугольников.

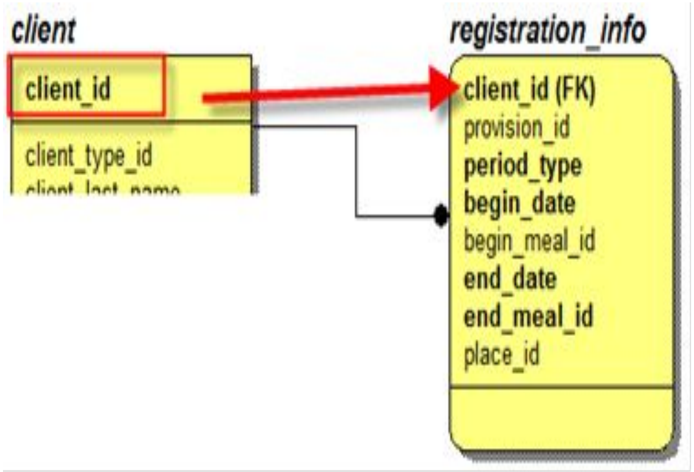
Сущности, независящие при идентификации от других объектов в модели, называются ***независимыми*** сущностями.

Пример. Сущность **ОТДЕЛ** можно считать независимой. В IDEF1X независимые сущности представлены в виде прямоугольников.

Связи в IDEF1X

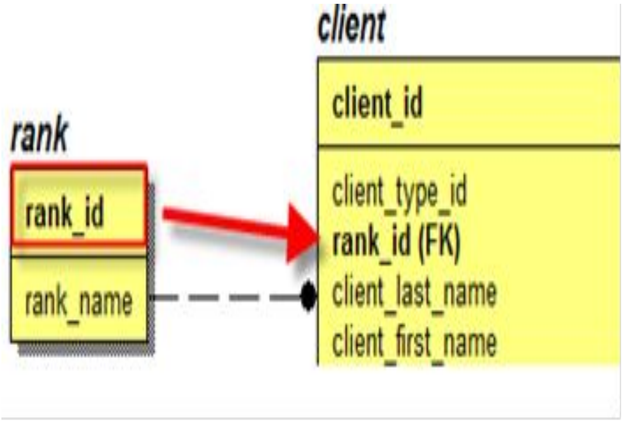
Идентифицирующая связь устанавливается между независимой (родительской) и зависимой (дочерней) сущностями. При создании идентифицирующей связи дочерняя сущность автоматически становится зависимой. В дочерней сущности новые (мигрировавшие из родительской сущности) атрибуты помечаются как внешний ключ (FK).

Идентифицирующие взаимосвязи обозначаются сплошной линией между сущностями.



Неидентифицирующие связи также связывают родительскую сущность с дочерней. Неидентифицирующие связи используются для отображения передачи в область данных дочерней сущности (под линией).

Неидентифицирующие связи отображаются пунктирной линией между объектами.

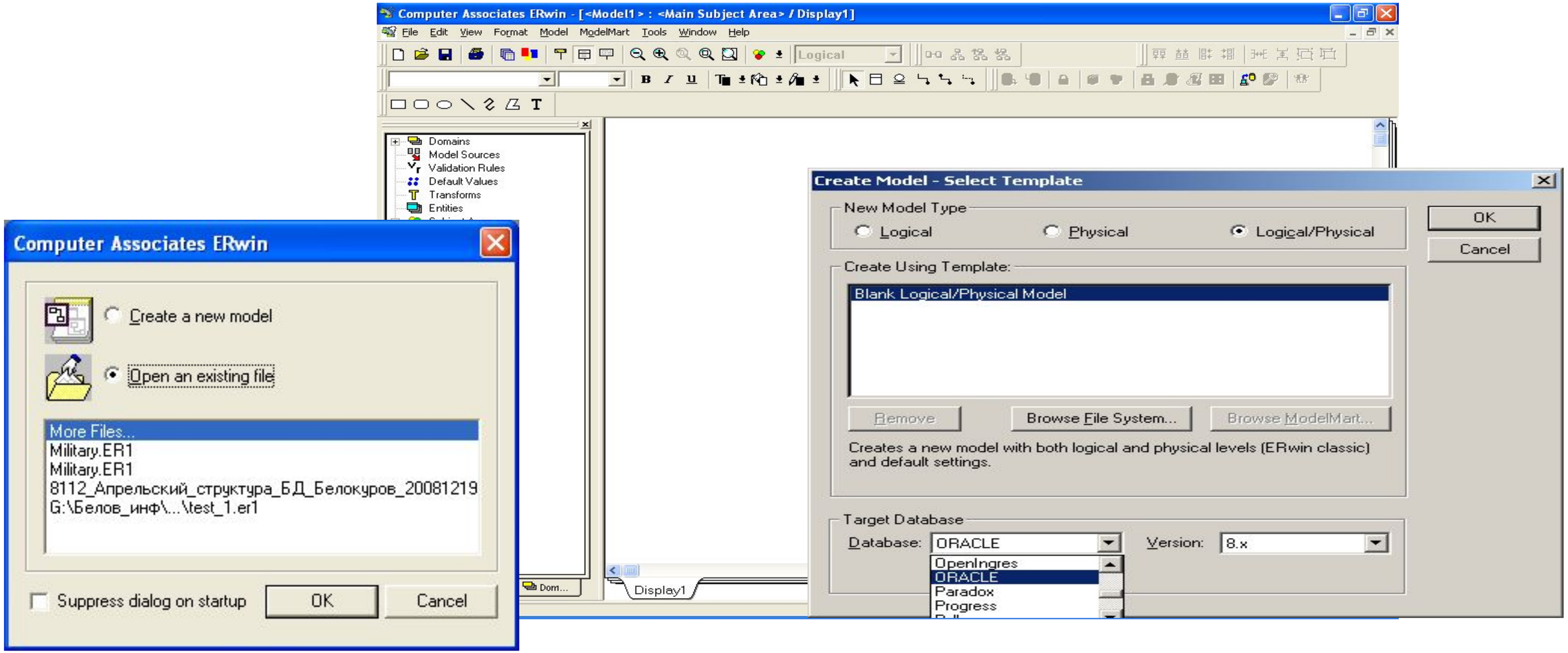


Ограничения в IDEF1X

Средства моделирования IDEF1X специально разработаны для построения информационных систем на основе реляционных СУБД. Если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы моделирования.

- IDEF1X требует от проектировщика определить ключевые атрибуты, для того чтобы отличить одну сущность от другой, в то время как объектно-ориентированные системы не требуют задания ключевых атрибутов, в целях идентификации объектов.
- В тех случаях, когда более чем один атрибут является однозначно идентифицирующим сущность, проектировщик должен определить один из этих атрибутов первичным ключом, а все остальные вторичными. И, таким образом, построенная проектировщиком IDEF1X-модель и переданная для окончательной реализации программисту является некорректной для применения методов объектно-ориентированной реализации, и предназначена только для построения реляционной БД.

ERWin – средство проектирования БД



Средства проектирования БД

Графические редакторы для проектирования БД (свободнообращаемое ПО):

- *Dia* - графический редактор для построения диаграмм. Работает под Linux и Windows
- *LucidChart (lucidchart.com)* - online редактор для построения диаграмм
- *Draw.io* - online редактор для построения диаграмм. Поддерживает ER-диаграммы

Проприетарные (коммерческие) средства: Microsoft Visio, SmartDraw, ERwin Data Modeler, IBM Rational Rose Modeler, Poseidon for UML,

Белов Александр Владимирович
Email: avbelov@hse.ru

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ