

Проектирование реляционных БД на основе принципов нормализации

проект реляционной базы данных

- Это набор взаимосвязанных отношений, в которых
- определены все атрибуты,
- заданы первичные ключи отношений и
- заданы еще некоторые дополнительные свойства отношений, которые относятся к принципам поддержки целостности.

Уровни моделирования

- Сама предметная область
Модель предметной области
Логическая модель данных
Физическая модель данных
Собственно база данных и приложения

Критерии оценки качества логической модели данных

- Адекватность базы данных предметной области
- Легкость разработки и сопровождения базы данных
- Скорость выполнения операций обновления данных (вставка, обновление, удаление кортежей)
- Скорость выполнения операций выборки данных

Адекватность базы данных предметной области

- Состояние базы данных в каждый момент времени должно соответствовать состоянию предметной области.
- Изменение состояния предметной области должно приводить к соответствующему изменению состояния базы данных
- Ограничения предметной области, отраженные в модели предметной области, должны некоторым образом отражаться и учитываться базе данных.

Легкость разработки и сопровождения базы данных

- Практически любая база данных, за исключением совершенно элементарных, содержит некоторое количество программного кода в виде триггеров и хранимых процедур.
- Очевидно, что чем больше программного кода в виде триггеров и хранимых процедур содержит база данных, тем сложнее ее разработка и дальнейшее сопровождение.

Скорость операций обновления данных (вставка, обновление, удаление)

- Основными операциями, изменяющими состояние базы данных, являются операции вставки, обновления и удаления записей.
- скорость выполнения операций вставки, обновления и удаления также **уменьшается при увеличении количества индексов** у таблицы и **мало зависит от числа строк** в таблице.

Скорость операций выборки данных

- Одной из наиболее дорогостоящих операций при выполнении оператора `SELECT` является операция соединения таблиц.
- Таким образом, увеличение количества отношений приводит к замедлению выполнения операций выборки данных, особенно, если запросы заранее неизвестны.

Результаты Исследования OASIG

- • Примерно 80-90% компьютеризованных систем не обладают требуемой производительностью.
- • При разработке около 80% систем были превышены установленные для этого временные и бюджетные рамки.
- • Разработка около 40% систем закончилась неудачно или была прекращена до завершения работы.
- • Менее чем 40% систем предусматривали профессиональное обучение и повышение квалификации пользователей во всем необходимом объеме.
- • Гармонично интегрировать интересы бизнеса и используемой технологии удалось не более чем в 25% систем.
- • Только 10-20% систем отвечают всем критериям достижения успеха.

Неудачи при создании программного обеспечения были вызваны следующими причинами:

- а отсутствием полной спецификации всех требований;
- • отсутствием приемлемой методологии разработки;
- • недостаточной степенью разделения общего глобального проекта на отдельные компоненты, поддающиеся эффективному контролю и управлению.

Этапы разработки ИС

- Планирование разработки базы данных
- Определение требований к системе
- Сбор и анализ требований пользователей
- Проектирование базы данных
- Выбор целевой СУБД (необязательный этап)
- Разработка приложений
- Создание прототипов (необязательный этап)
- Реализация
- Преобразование и загрузка данных
- Тестирование
- Эксплуатация и сопровождение

Модель предметной области

- Сотрудники организации выполняют проекты.
- Проекты состоят из нескольких заданий.
- Каждый сотрудник может участвовать в одном или нескольких проектах, или временно не участвовать ни в каких проектах.
- Над каждым проектом может работать несколько сотрудников, или временно проект может быть приостановлен, тогда над ним не работает ни один сотрудник.
- Над каждым заданием в проекте работает ровно один сотрудник.
- Каждый сотрудник числится в одном отделе.
- Каждый сотрудник имеет телефон, находящийся в отделе сотрудника

Ввод дополнительных атрибутов

- О каждом сотруднике необходимо хранить табельный номер и фамилию. Табельный номер является уникальным для каждого сотрудника.
- Каждый отдел имеет уникальный номер.
- Каждый проект имеет номер и наименование. Номер проекта является уникальным.
- Каждая работа из проекта имеет номер, уникальный в пределах проекта. Работы в разных проектах могут иметь одинаковые номера.

Проектирование схемы БД

- Проектирование схемы БД может быть выполнено двумя путями:
- путем декомпозиции (разбиения), когда исходное множество отношений, входящих в схему БД заменяется другим множеством отношений (число их при этом возрастает), являющихся проекциями исходных отношений;
- путем синтеза, то есть путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД.

Процесс проектирования с использованием декомпозиции

- Процесс проектирования с использованием декомпозиции представляет собой процесс последовательной нормализации схем отношений, при этом каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.**

1НФ (Первая Нормальная Форма)

- В отношении нет одинаковых кортежей.
- Кортежи не упорядочены.
- Атрибуты не упорядочены и различаются по наименованию.
- Все значения атрибутов атомарны.

Потенциальный ключ

- *Потенциальным ключом* отношения называется набор атрибутов отношения, который полностью и однозначно (функционально полно) определяет значения всех остальных атрибутов отношения, то есть возможный ключ — это набор атрибутов, однозначно определяющий кортеж отношения
- Среди всех возможных ключей отношения обычно выбирают один, который считается главным и который называют *первичным ключом отношения*.

СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ

- ***Н_СОТР*** - табельный номер сотрудника
- **ФАМ** - фамилия сотрудника
- **Н_ОТД** - номер отдела, в котором числится сотрудник
- **ТЕЛ** - телефон сотрудника
- ***Н_ПРО*** - номер проекта, над которым работает сотрудник
- **ПРОЕКТ** - наименование проекта, над которым работает сотрудник
- **Н_ЗАДАН** - номер задания, над которым работает сотрудник
- Т.к. каждый сотрудник в каждом проекте выполняет ровно одно задание, то в качестве потенциального ключа отношения необходимо взять пару атрибутов ***{Н_СОТР, Н_ПРО}***.

Отношение

<i>Н_СОТР</i>	ФАМ	Н_ОТД	ТЕЛ	<i>Н_ПРО</i>	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		Климат	2

Аномалии обновления

- *неадекватность модели данных предметной области, либо некоторые дополнительные трудности в реализации ограничений предметной области*
- Аномалии вставки (INSERT)
- Аномалии обновления (UPDATE)
- Аномалии удаления (DELETE)

Аномалии вставки (INSERT)

- В отношении **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).

Аномалии обновления (UPDATE)

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

Аномалии удаления (DELETE)

- При удалении некоторых данных может произойти потеря другой информации.
- Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

Определение функциональной зависимости

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 $r_1, r_2 \in R$ из того, что $r_1.X = r_2.X$ следует что $r_1.Y = r_2.Y$ (т.е. во всех кортежах, имеющих одинаковые значения атрибутов X , значения атрибутов Y также совпадают в *любом* состоянии отношения R). Символически функциональная зависимость записывается

$X \rightarrow Y$.

Множество атрибутов X называется **детерминантом функциональной зависимости**, а множество атрибутов Y называется **зависимой частью**.

примеры функциональных зависимостей:

- Зависимость атрибутов от ключа отношения
- {*H_COTR*, *H_PRO*} ФАМ
- {*H_COTR*, *H_PRO*} Н_ОТД
- {*H_COTR*, *H_PRO*} ТЕЛ
- {*H_COTR*, *H_PRO*} ПРОЕКТ
- {*H_COTR*, *H_PRO*} Н_ЗАДАН
- Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:
 - *H_COTR* ФАМ
 - *H_COTR* Н_ОТД
 - *H_COTR* ТЕЛ
- Зависимость наименования проекта от номера проекта:
 - *H_PRO* ПРОЕКТ
- Зависимость номера телефона от номера отдела:
 - *H_ОТД* ТЕЛ

Математическое определение

Определение 2. Функциональная зависимость (*функция*) - это тройка объектов $\{X, Y, f\}$, где

X - множество (*область определения*),

Y - множество (*множество значений*),

f - правило, согласно которому каждому элементу $x \in X$ ставится в соответствие один и только один элемент $y \in Y$ (*правило функциональной зависимости*).

Функциональная зависимость обычно обозначается как $f: X \rightarrow Y$ или $y = f(x)$.

Замечание. Правило f может быть задано любым способом - в виде формулы (чаще всего), при помощи таблицы значений, при помощи графика, текстовым описанием и т.д.

2НФ (Вторая Нормальная Форма)

- *Определение 3.* Отношение находится во *второй нормальной форме (2НФ)* тогда и только тогда, когда отношение находится в 1НФ и *нет неключевых атрибутов, зависящих от части сложного ключа.*
- *Неключевой атрибут* - это атрибут, не входящий в состав **никакого потенциального ключа.**

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* ФАМ**
- ***Н_СОТР* Н_ОТД**
- ***Н_СОТР* ТЕЛ**
- Зависимость наименования проекта от номера проекта является зависимостью от части сложного ключа:
- ***Н_ПРО* ПРОЕКТ**

- Отношение
СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ
декомпозируем на три отношения -
СОТРУДНИКИ_ОТДЕЛЫ,
- **ПРОЕКТЫ,**
- **ЗАДАНИЯ.**

Отношение СОТРУДНИКИ_ОТДЕЛЫ

<i>Н_СОТР</i>	ФАМ	Н_ОТД	ТЕЛ
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Отношение ПРОЕКТЫ и ЗАДАНИЯ

<i>Н_ПРО</i>	ПРОЕКТ
1	Космос
2	Климат

<i>Н_СОТР</i>	<i>Н_ПРО</i>	Н_ЗАДАН
1	1	1
1	2	1
2	1	2
3	1	3
3	2	2

Анализ декомпозированных отношений

- Отношения, полученные в результате декомпозиции, находятся в 2НФ.
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}**.

Оставшиеся аномалии вставки (INSERT)

- В отношении **СОТРУДНИКИ_ОТДЕЛЫ** нельзя вставить кортеж (4, Пушкинов, 1, 33-22-11), т.к. при этом получится, что два сотрудника из 1-го отдела (Иванов и Пушкинов) имеют разные номера телефонов, а это противоречит модели предметной области.

Оставшиеся аномалии удаления (DELETE)

- При удалении некоторых данных по-прежнему может произойти потеря другой информации. Например, если удалить сотрудника Сидорова, то будет потеряна информация о том, что в отделе номер 2 находится телефон 33-22-11.

3НФ (Третья Нормальная Форма)

- *Определение 4.* Атрибуты называются **взаимно независимыми**, если ни один из них не является функционально зависимым от другого.
- *Определение 5.* Отношение находится в **третьей нормальной форме (3НФ)** тогда и только тогда, когда отношение находится в 2НФ и **все неключевые атрибуты взаимно независимы**.
- Отношение **СОТРУДНИКИ_ОТДЕЛЫ** не находится в 3НФ, т.к. имеется функциональная зависимость неключевых атрибутов (зависимость номера телефона от номера отдела):
- **Н_ОТД ТЕЛ**

Пример

- Отношение **СОТРУДНИКИ_ОТДЕЛЫ** декомпозируем на два отношения - **СОТРУДНИКИ**, **ОТДЕЛЫ**.

<i>Н_СОТР</i>	ФАМ	Н_ОТД
1	Иванов	1
2	Петров	1
3	Сидоров	2

<i>Н_ОТД</i>	ТЕЛ
1	11-22-33
2	33-22-11

Пример

- Рассмотрим отношение, моделирующее сдачу студентами текущей сессии. Структура этого отношения определяется следующим набором атрибутов:
- (ФИО, Номер зач.кн, Группа, Дисциплина, Оценка)

Приведение ко 2НФ

- Для приведения данного отношения ко второй нормальной форме следует разбить его на проекции, при этом должно быть соблюдено условие восстановления исходного отношения без потерь.
- Такими проекциями могут быть два отношения
 - (ФИО, Номер зач.кн, Группа)
 - (Номер зач.кн, Дисциплина, Оценка)

Приведение к ЗНФ

- Рассмотрим отношение, связывающее студентов с группами, факультетами и специальностями
- (ФИО, Номер зач.кн, Группа, Факультет, Специальность, Выпускающая кафедра)

Функциональные зависимости

- Номер зач.кн. -> ФИО
- Номер зач.кн. -> Группа
- Номер зач.кн. -> Факультет
- Номер зач.кн. -> Специальность
- Номер зач.кн. -> Выпускающая кафедра
- Группа -> Факультет
- Группа -> Специальность
- Группа -> Выпускающая кафедра
- Выпускающая кафедра -> Факультет

Декомпозиция

И эти зависимости образуют транзитивные группы. Для того чтобы избежать этого, мы можем предложить следующий набор отношений

- (Номер.зач.кн., ФИО, Специальность, Группа)
- (Группа, Выпускающая кафедра)
- (Выпускающая кафедра, Факультет)

Первичные ключи отношений выделены.

Алгоритм нормализации

- *Шаг 1 (Приведение к 1НФ).* На первом шаге задается одно или несколько отношений, отображающих понятия предметной области.
- По модели предметной области (не по внешнему виду полученных отношений!) выписываются обнаруженные функциональные зависимости.
- Все отношения автоматически находятся в 1НФ.

- *Шаг 2 (Приведение к 2НФ)*. Если в некоторых отношениях обнаружена зависимость атрибутов от части сложного ключа, то проводим декомпозицию этих отношений на несколько отношений следующим образом:
 - те атрибуты, которые зависят от части сложного ключа выносятся в отдельное отношение вместе с этой частью ключа.
 - В исходном отношении остаются все ключевые атрибуты

Исходное отношение: $R(K_1, K_2, A_1, \dots, A_n, B_1, \dots, B_m)$.

Ключ: $\{K_1, K_2\}$ - сложный.

Функциональные зависимости:

$\{K_1, K_2\} \rightarrow \{A_1, \dots, A_n, B_1, \dots, B_m\}$ - зависимость всех атрибутов от ключа отношения.

$\{K_1\} \rightarrow \{A_1, \dots, A_n\}$ - зависимость некоторых атрибутов от части сложного ключа.

Декомпозированные отношения:

$R_1(K_1, K_2, B_1, \dots, B_m)$ - остаток от исходного отношения. Ключ $\{K_1, K_2\}$.

$R_2(K_1, A_1, \dots, A_n)$ - атрибуты, вынесенные из исходного отношения вместе с частью сложного ключа. Ключ K_1 .

- *Шаг 3 (Приведение к 3НФ)*. Если в некоторых отношениях обнаружена зависимость некоторых неключевых атрибутов других неключевых атрибутов, то проводим декомпозицию этих отношений следующим образом:
 - те неключевые атрибуты, которые зависят других неключевых атрибутов выносятся в отдельное отношение.
 - В новом отношении ключом становится детерминант функциональной зависимости:

Исходное отношение: $R(K, A_1, \dots, A_n, B_1, \dots, B_m)$.

Ключ: K .

Функциональные зависимости:

$K \rightarrow \{A_1, \dots, A_n, B_1, \dots, B_m\}$ - зависимость всех атрибутов от ключа отношения.

$\{A_1, \dots, A_n\} \rightarrow \{B_1, \dots, B_m\}$ - зависимость некоторых неключевых атрибутов других неключевых атрибутов.

Декомпозированные отношения:

$R_1(K, A_1, \dots, A_n)$ - остаток от исходного отношения. Ключ K .

$R_2(A_1, \dots, A_n, B_1, \dots, B_m)$ - атрибуты, вынесенные из исходного отношения вместе с детерминантом функциональной зависимости. Ключ $\{A_1, \dots, A_n\}$.

Анализ критериев для нормализованных и ненормализованных моделей данных

Критерий	Отношения слабо нормализованы (1НФ, 2НФ)	Отношения сильно нормализованы (3НФ)
Адекватность базы данных предметной области	ХУЖЕ (-)	ЛУЧШЕ (+)
Легкость разработки и сопровождения базы данных	СЛОЖНЕЕ (-)	ЛЕГЧЕ (+)
Скорость выполнения вставки, обновления, удаления	МЕДЛЕННЕЕ (-)	БЫСТРЕЕ (+)
Скорость выполнения выборки данных	БЫСТРЕЕ (+)	МЕДЛЕННЕЕ (-)

Корректность процедуры нормализации - декомпозиция без потерь

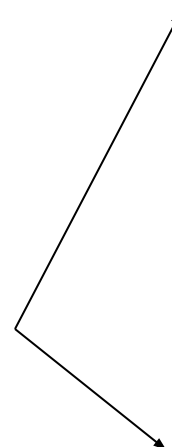
- *Определение.* Проекция $R[X]$ отношения R на множество атрибутов X называется **собственной**, если множество атрибутов X является **собственным подмножеством** множества X атрибутов отношения R (т.е. множество атрибутов X не совпадает с множеством всех атрибутов отношения R).
- *Определение.* **Собственные** проекции $R1$ и $R2$ отношения R называются **декомпозицией без потерь**, если отношение R **точно восстанавливается** из них при помощи **естественного соединения для любого состояния** отношения R :
- $R1 \text{ JOIN } R2 = R$

Пример декомпозиции с потерями

НОМЕР	ФАМИЛИЯ	ЗАРПЛАТА
1	Иванов	1000
2	Петров	1000

НОМЕР	ЗАРПЛАТА
1	1000
2	1000

ФАМИЛИЯ	ЗАРПЛАТА
Иванов	1000
Петров	1000



Естественное соединение

НОМЕР	ФАМИЛИЯ	ЗАРПЛАТА
1	Иванов	1000
1	Петров	1000
2	Иванов	1000
2	Петров	1000

Теорема Хеза

Теорема (Хеза). Пусть $R(A,B,C)$ является отношением, и A,B,C - атрибуты или множества атрибутов этого отношения. Если имеется функциональная зависимость $A \rightarrow B$, то проекции $R_1 = R[A,B]$ и $R_2 = R[A,C]$ образуют декомпозицию без потерь.

Доказательство. Необходимо доказать, что $R_1 \text{ JOIN } R_2 = R$ для любого состояния отношения R . В левой и правой части равенства стоят множества кортежей, поэтому для доказательства достаточно доказать два включения для двух множеств кортежей: $R_1 \text{ JOIN } R_2 \supseteq R$ и $R_1 \text{ JOIN } R_2 \subseteq R$.

Докажем первое включение. Возьмем произвольный кортеж $r = (a,b,c) \in R$. Докажем, что он включается также и в $R_1 \text{ JOIN } R_2$. По определению проекции, кортежи $r_1 = (a,b) \in R_1$ и $r_2 = (a,c) \in R_2$. По определению естественного соединения кортежи r_1 и r_2 , имеющие одинаковое значение a общего атрибута A , будут соединены в процессе естественного соединения в кортеж $(a,b,c) \in R_1 \text{ JOIN } R_2$. Таким образом, включение доказано.

доказательство

Докажем первое включение. Возьмем произвольный кортеж $r = (a, b, c) \in R$. Докажем, что он включается также и в $R_1 \text{ JOIN } R_2$. По определению проекции, кортежи $r_1 = (a, b) \in R_1$ и $r_2 = (a, c) \in R_2$. По определению естественного соединения кортежи r_1 и r_2 , имеющие одинаковое значение a общего атрибута A , будут соединены в процессе естественного соединения в кортеж $(a, b, c) \in R_1 \text{ JOIN } R_2$. Таким образом, включение доказано.

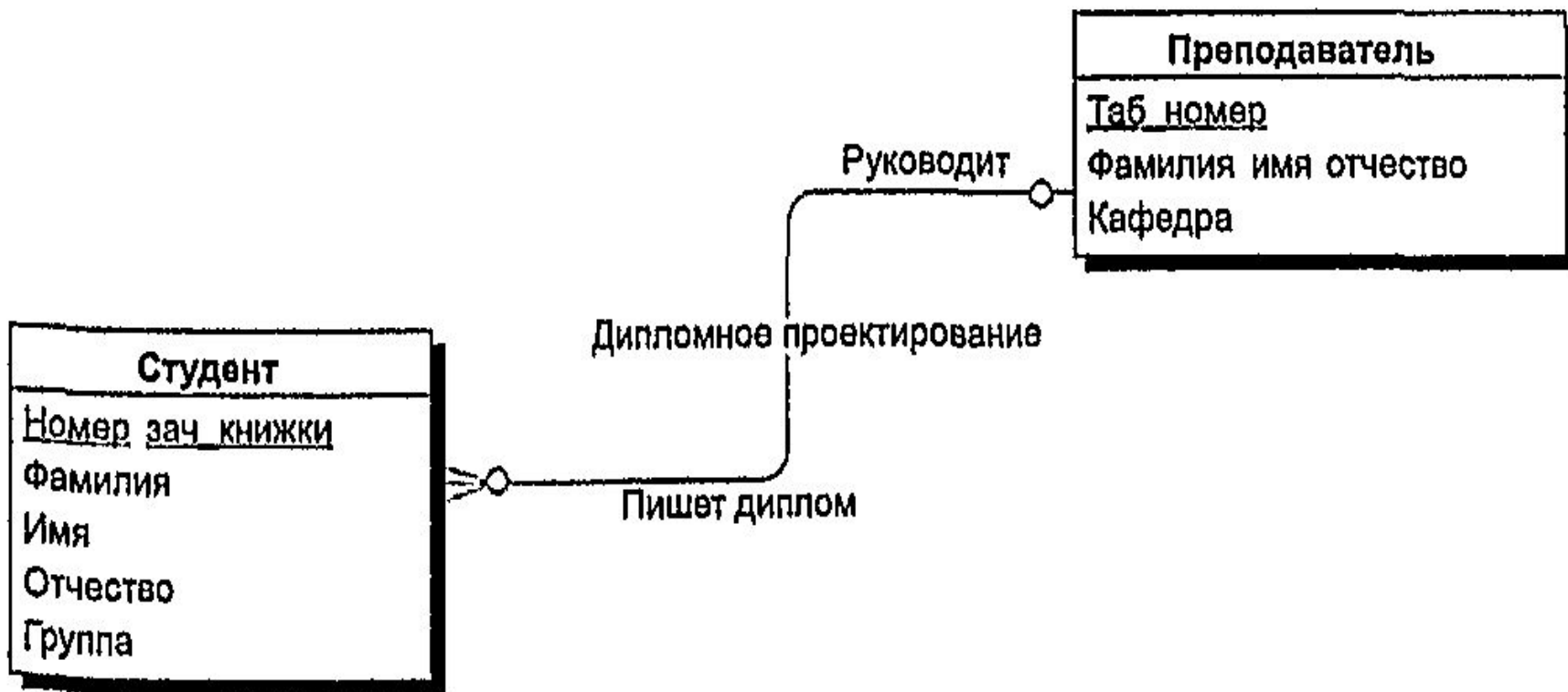
Докажем обратное включение. Возьмем произвольный кортеж $r = (a, b, c) \in R_1 \text{ JOIN } R_2$. Докажем, что он включается также и в R . По определению естественного соединения получим, что в R имеются кортежи $r_1 = (a, b) \in R_1$ и $r_2 = (a, c) \in R_2$. Т.к. $R_1 = R[A, B]$, то существует некоторое значение c_1 , такое что кортеж $r_1 = (a, b, c_1) \in R$. Аналогично, существует некоторое значение b_1 , такое что кортеж $r_2 = (a, b_1, c) \in R$. Кортежи r_1 и r_2 имеют одинаковое значение атрибута A , равное a . Из этого, в силу функциональной зависимости $A \rightarrow B$, следует, что $b = b_1$. Таким образом, кортеж $r_2 = (a, b, c) \in R$. Обратное включение доказано. Теорема доказана.

Инфологическое проектирование

Модель «сущность—связь»

- *Основные понятия:*
- *Сущность*, с помощью которой моделируется класс однотипных объектов.
- Объект, которому соответствует понятие сущности, имеет свой набор *атрибутов* — характеристик, определяющих свойства данного представителя класса.
- Набор атрибутов, однозначно идентифицирующий конкретный экземпляр сущности, называют *ключевым*.
- сущностями могут быть установлены *связи* - бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют между собой.

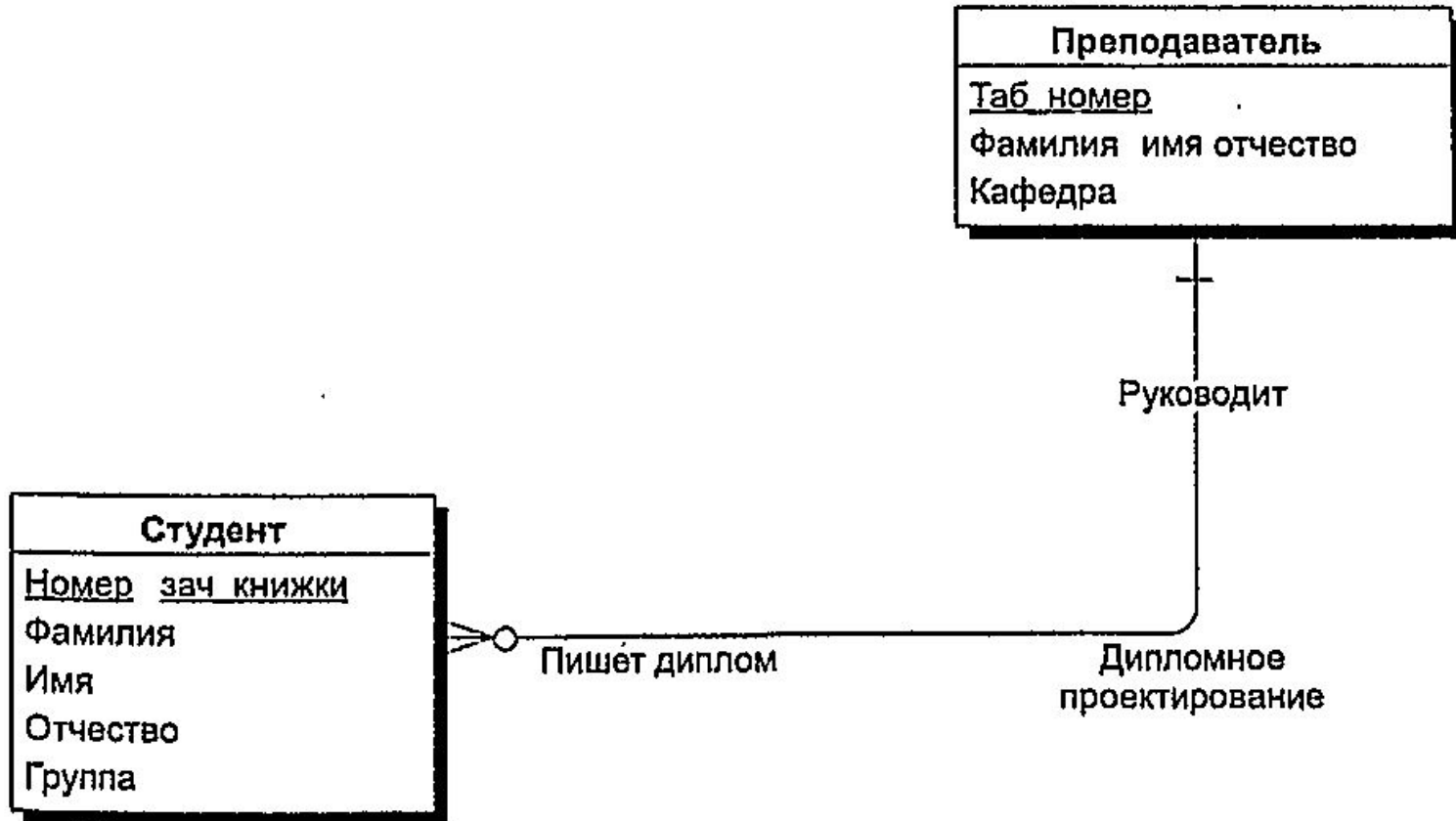
Пример



Обязательные и необязательные СВЯЗИ

- Связь любого из этих типов может быть *обязательной*, если в данной связи должен участвовать каждый экземпляр сущности, *необязательной* — если не каждый экземпляр сущности должен участвовать в данной связи. При этом связь может быть *обязательной с одной стороны и необязательной с другой стороны*.

Обозначения



Типы связей

- Связи делятся на три типа по множественности:
- *один-к-одному* (1:1),
- *один-ко-многим* (1:M),
- *многие-ко-многим* (M:M).
- Между двумя сущностями может быть задано сколько угодно связей с разными смысловыми нагрузками.

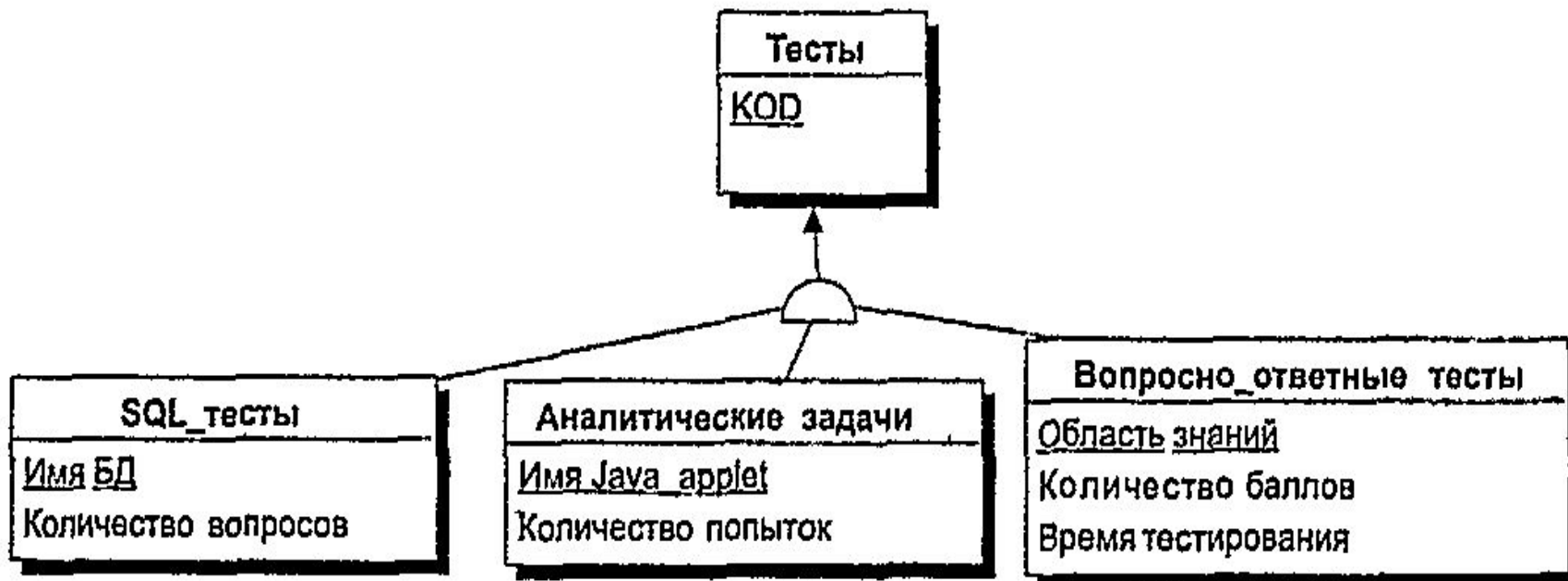
принцип категоризации сущностей

- Подтип сущности -- *сущности*, каждая из которых может иметь общие атрибуты и отношения и/или атрибуты и отношения, которые определяются однажды на верхнем уровне и наследуются на нижнем уровне.
- Все подтипы одной сущности рассматриваются как взаимоисключающие, и при разделении сущности на подтипы она должна быть представлена в виде полного набора взаимоисключающих подтипов.

супертип

- Сущность, на основе которой строятся подтипы, называется *супертипом*. Любой экземпляр супертипа должен относиться к конкретному подтипу.

Пример супертипа и подтипов



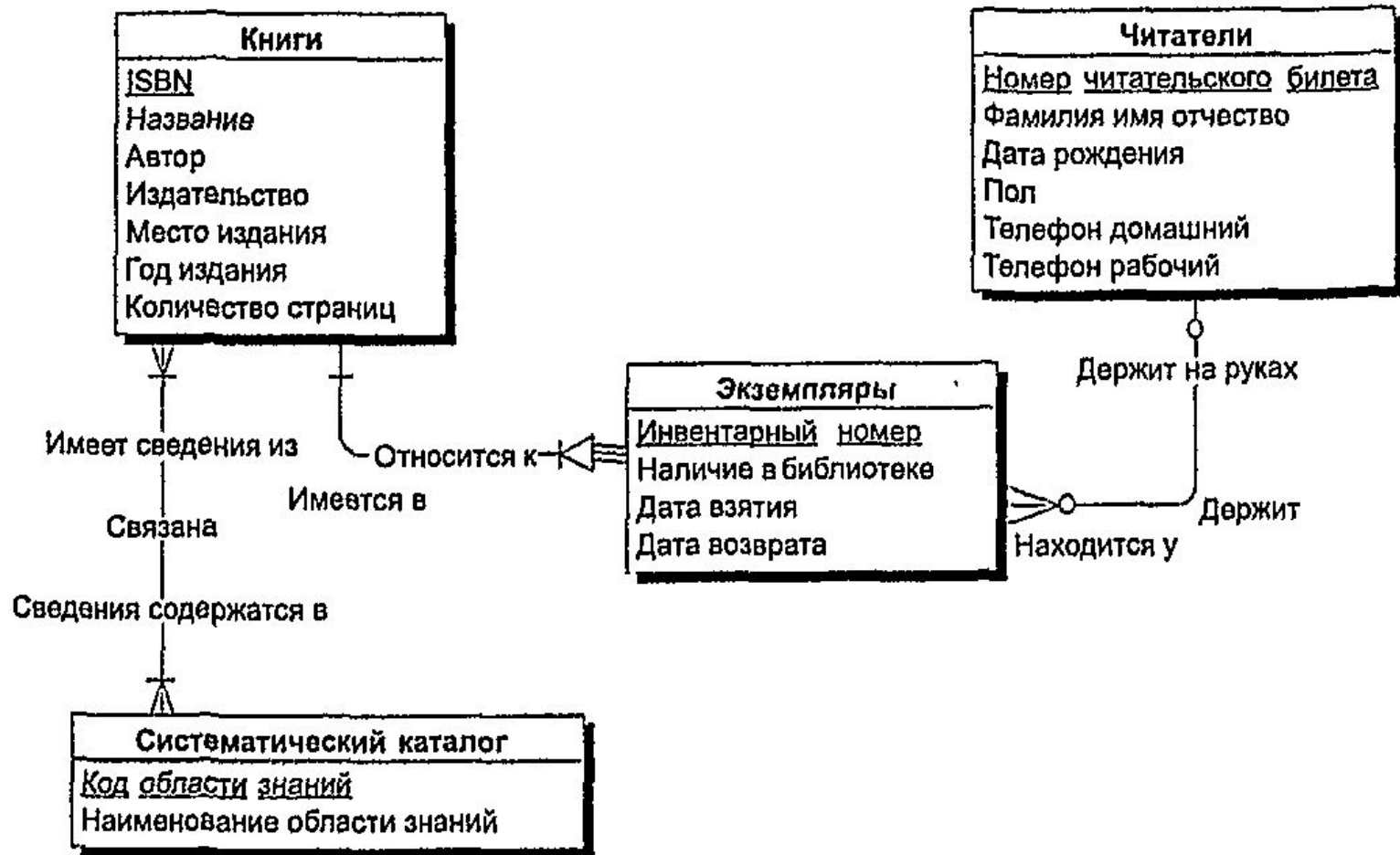
Результат проектирования

- В результате построения модели предметной области в виде набора сущностей и связей получаем связный граф. В полученном графе необходимо избегать циклических связей — они выявляют некорректность модели.

Пример инфологического проектирования

- В качестве примера спроектируем инфологическую модель системы, предназначенной для хранения информации о книгах и областях знаний, представленных в библиотеке.
- Разработку модели начнем с выделения основных сущностей.

Пример ER-модели



Преобразование ER-модели в реляционную

правила преобразования ER- модели в реляционную.

- 1. Каждой сущности ставится в соответствие отношение реляционной модели данных.
- 2. Каждый атрибут сущности становится атрибутом соответствующего отношения.

СОТРУДНИК	
<u>Табельный номер</u>	
Фамилия	
Имя	
Отчество	
Количество детей	

EMPLOYEE	
<u>T_NUM</u>	int
NAME	varchar(30)
F_NAME	varchar(30)
L_NAME	varchar(30)
COUNT_CH	varchar(30)

Преобразование сущности СОТРУДНИК к отношению EMPLOYEE

Преобразование ключей

- 4. В каждое отношение, соответствующее подчиненной сущности, добавляется набор атрибутов основной сущности, являющейся первичным ключом основной сущности. В отношении, соответствующем подчиненной сущности, этот набор атрибутов становится внешним ключом (FOREIGN KEY).

СВЯЗИ

- 5. Для моделирования необязательного типа связи на физическом уровне у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений (признак NULL).
- При обязательном типе связи атрибуты получают свойство отсутствия неопределенных значений (признак NOT NULL).

Категоризация типов

- Для отражения Категоризации сущностей при переходе к реляционной модели возможны несколько вариантов представления.
- Возможно создать только одно отношение для всех подтипов одного супертипа

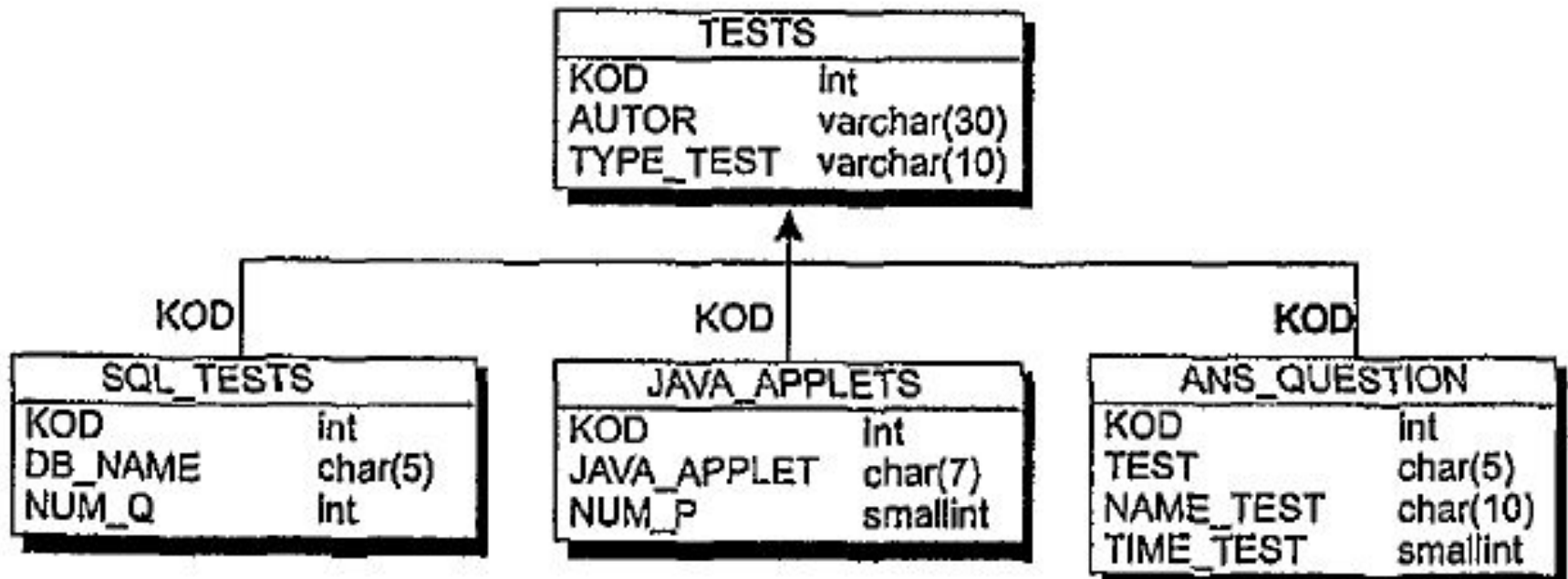
- При втором способе для каждого подтипа и для супертипа создаются свои отдельные отношения.
- Для возможности переходов к подтипам от супертипа необходимо в супертип включить идентификатор связи.

дискриминаторы

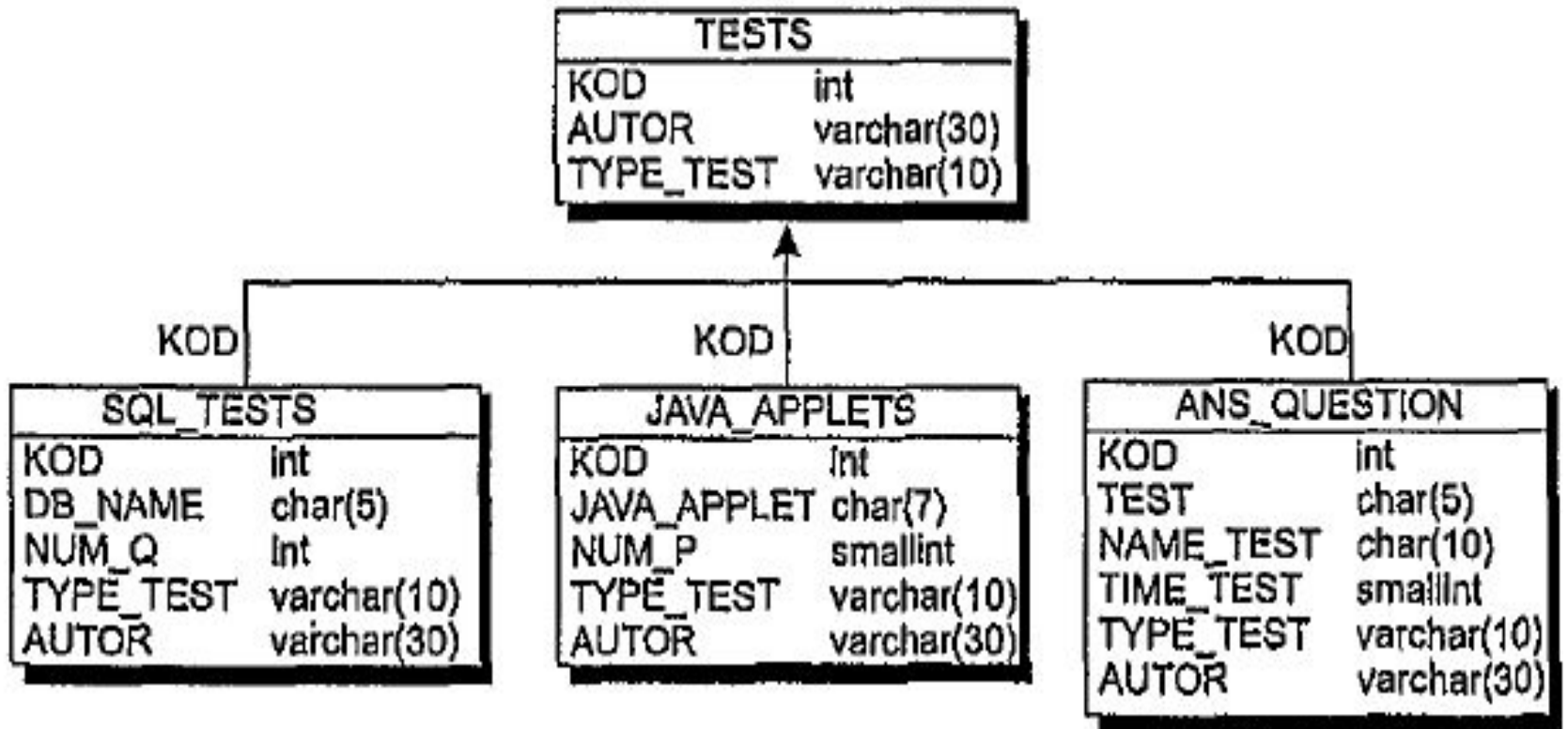
- Дополнительно при описании отношения между типом и подтипами необходимо указать тип дискриминатора
- Дискриминатор может быть взаимоисключающим или нет



Пример – наследование идентификатора суперсущности



Наследование всех атрибутов суперсущности



Разрешение связей типа «МНОГИЕ-КО-МНОГИМ».

- Это делается введением специального дополнительного связующего отношения, которое связано с каждым исходным связью «один-ко-многим», атрибутами этого отношения являются первичные ключи связываемых отношений.
- например» в схеме «Библиотека» присутствует связь такого типа между сущностью «Книги» и «Системный каталог». Для разрешения этой неспецифической связи при переходе к реляционной модели, должно быть введено специальное дополнительное отношение, которое имеет всего два атрибута;
- ISBN (шифр книги) и KOD (код области знаний).

При этом каждый из атрибутов нового отношения является внешним ключом (FORKING KEY), а вместе они образуют первичный ключ (PRIMARY KEY) повой связующей сущности.

Библиотека

