

НОРМАЛЬНЫЕ ФОРМЫ

Аномалии операций с БД. Теория зависимостей.
Нормализация и нормальные формы. Денормализация.

Author: Svyatoslav Kulikov
Training And Education Manager
svyatoslav_kulikov@epam.com

1. Аномалии операций с БД.

2. Теория зависимостей.

3. Нормализация и нормальные формы.

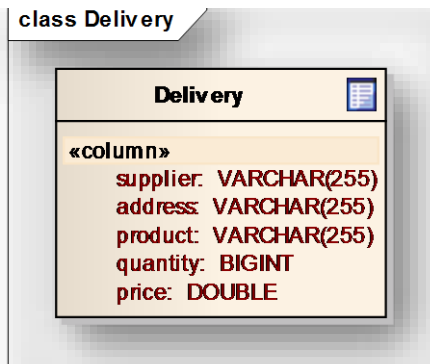
- Требования нормализации.
- Нормальные формы низких порядков.
- Нормальные формы высоких порядков.
- Пример применения нормализации.

4. Денормализация.

АНОМАЛИИ ОПЕРАЦИЙ С БД

ВАЖНО!

Для полноценного понимания материала следует помнить, что **схема отношения**

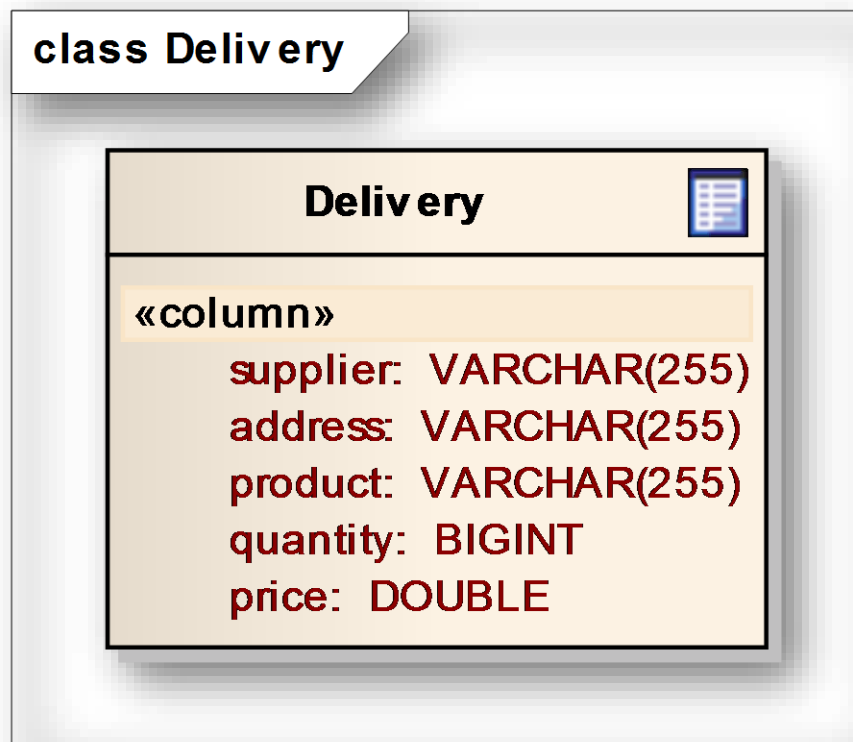


не эквивалентна **самому отношению!**

supplier	address	product	quantity	price
"Рога и копыта" Ltd	Далёкое-далёко, 999	Веники	200	1500
"Рога и копыта" Ltd	Далёкое-далёко, 999	Ложки	100	500
"Рога и копыта" Ltd	Далёкое-далёко, 999	Вилки	100	500
"Vasya and partners" Inc.	Близкое-близко, 111	Кирпичи	100000	3000000
"Vasya and partners" Inc.	Близкое-близко, 111	Корм для канареек	20	12345

Это важно знать и помнить потому, что **одной схеме может соответствовать множество отношений.**

Допустим, у нас есть отношение,
описанное такой схемой:



Создадим таблицу и заполним её данными:

Всё ли тут хорошо?

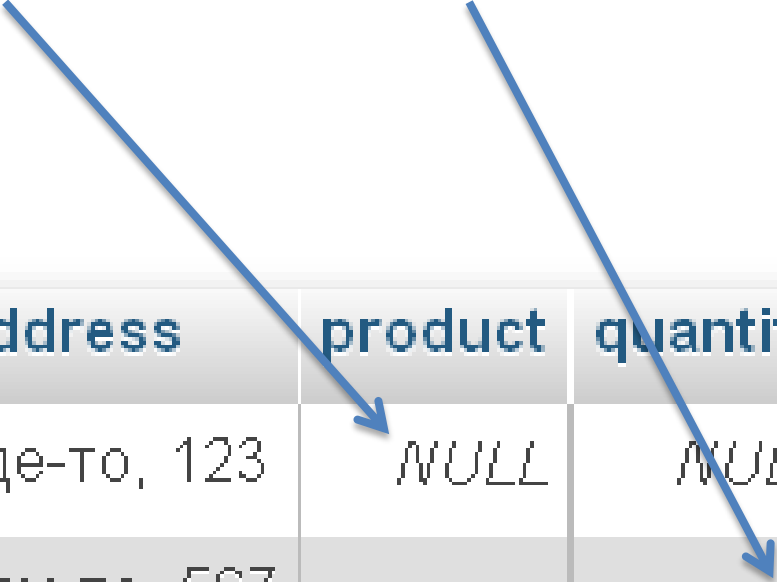
Column	Type	Collation	Attributes	Null	Default
supplier	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>
address	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>
product	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>
quantity	bigint(20)			Yes	<i>NULL</i>
price	double			Yes	<i>NULL</i>

supplier	address	product	quantity	price
"Рога и копыта" Ltd	Далёкое-далёко, 999	Веники	200	1500
"Рога и копыта" Ltd	Далёкое-далёко, 999	Ложки	100	500
"Рога и копыта" Ltd	Далёкое-далёко, 999	Вилки	100	500
"Vasya and partners" Inc.	Близкое-близко, 111	Кирпичи	100000	3000000
"Vasya and partners" Inc.	Близкое-близко, 111	Корм для канареек	20	12345

Аномалия (anomaly) – противоречие между моделью предметной области и моделью данных, поддерживаемой средствами конкретной СУБД.



Аномалия вставки – при добавлении данных, часть которых у нас отсутствует, мы вынуждены или не выполнять добавление или подставлять пустые или фиктивные данные.



supplier	address	product	quantity	price
Новый поставщик 1	Где-то, 123	NULL	NULL	NULL
Новый поставщик 2	Там-то, 567		0	0

Аномалия обновления – при обновлении данных мы вынуждены **обновлять много строк** и **рискуем часть строк «забыть обновить»**.

```
UPDATE `normalisation`.`delivery` SET  
`supplier` = '"Рога и копыта" Inc' WHERE  
`delivery`.`supplier` = '"Рога и копыта" Ltd'
```

17283545 row(s) affected.

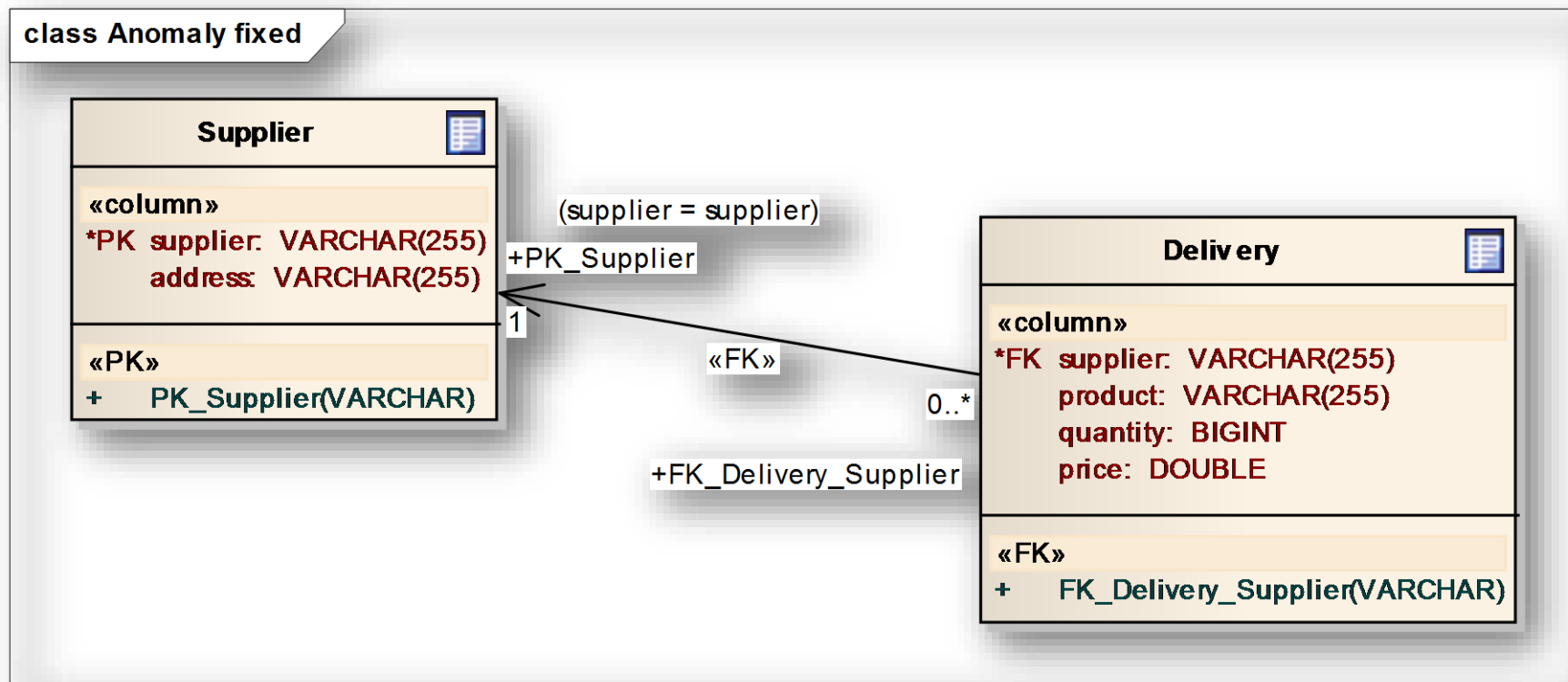
А если где-то «Ltd»
было написано с
точкой, т.е. «Ltd.»?

Аномалия удаления – при удалении части данных мы **теряем другую часть, которую не надо было удалять.**

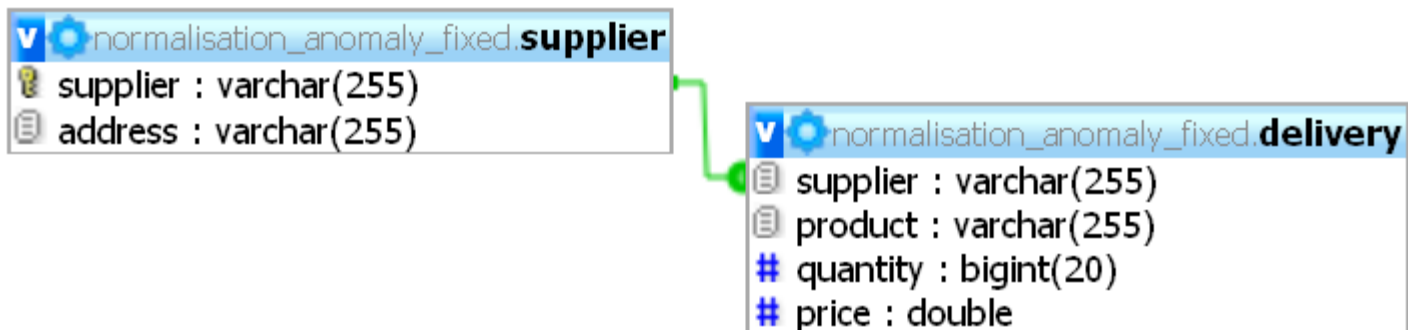
Например, сотрудничество с «Рога и копыта Ltd» прекращено, информацию о поставщике нужно удалить, и мы теряем данные о поставках:

supplier	address	product	quantity	price
"Рога и копыта" Ltd	Далёкое-далёко, 999	Веники	200	1500
"Рога и копыта" Ltd	Далёкое-далёко, 999	Ложки	100	500
"Рога и копыта" Ltd	Далёкое-далёко, 999	Вилки	100	500
"Vasya and partners" Inc.	Близкое-близко, 111	Кирпичи	100000	3000000
"Vasya and partners" Inc.	Близкое-близко, 111	Корм для канареек	20	12345

Изменив схему БД, мы получим следующую картину:



Изменив код БД, мы получим:



Column	Type	Collation	Attributes	Null	Default
<u>supplier</u>	varchar(255)	utf8_general_ci		No	None
address	varchar(255)	utf8_general_ci		Yes	NULL

Column	Type	Collation	Attributes	Null	Default
supplier	varchar(255)	utf8_general_ci		No	None
product	varchar(255)	utf8_general_ci		Yes	NULL
quantity	bigint(20)			Yes	NULL
price	double			Yes	NULL

Аномалии

Теперь мы

- Добавим возможность выполнять операции с поставками.
- Изменять информацию о поставщике.
- Удалять поставки и/или поставщиков независимо.

С удалением поставок проблем не будет, а при удалении поставщика нужно установить каскадную операцию SET NULL / RESTRICT в зависимости от желаемого поведения.

чтобы избежать нарушения необходимо каскадное удаление.

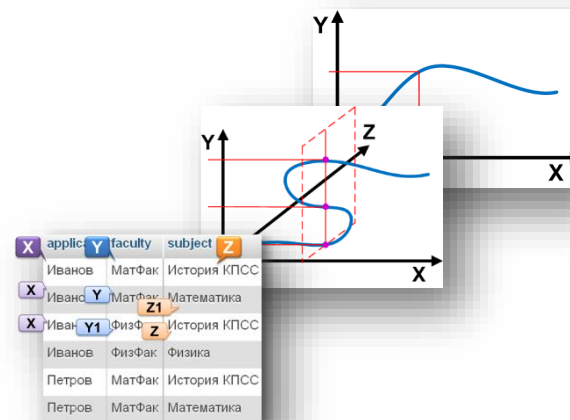
normalisation_anomaly_fixed.supplier
supplier : varchar(255)
address : varchar(255)

normalisation_anomaly_fixed.delivery
supplier : varchar(255)
product : varchar(255)
quantity : bigint(20)
price : double

И что с того!?

Нормализация, о которой скоро пойдёт речь, призвана в том числе устранить подобные аномалии.

А базируется нормализация на теории зависимостей, которую мы сейчас кратко и рассмотрим...



ТЕОРИЯ ЗАВИСИМОСТЕЙ

Функциональная зависимость

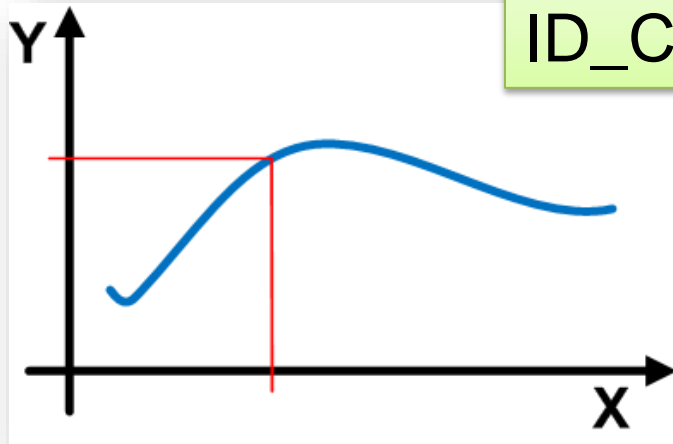
Если даны два атрибута X и Y некоторого отношения, то Y функционально зависит от X , если в любой момент времени **каждому значению X соответствует ровно одно значение Y** .

Обозначается $X \rightarrow Y$.

Примеры:

НомерПаспорта \rightarrow Фамилия

ID_Сотрудника \rightarrow СтажРаботыВФирме



Избыточная функциональная зависимость

Зависимость, заключающая в себе такую информацию, которая **может быть получена на основе других зависимостей**.

Пример:

В этой «цепи» есть
«лишнее звено»

ID_Сотрудника → НомерПаспорта → ФИО

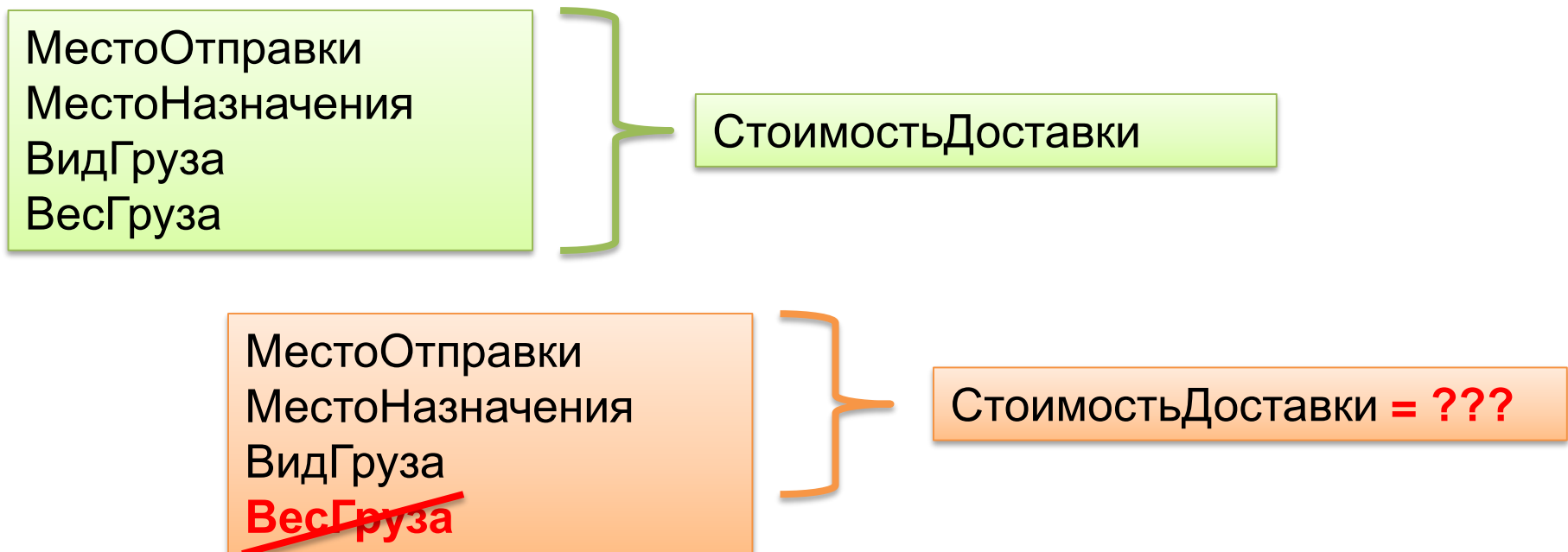
ID_Сотрудника → ФИО

Данные можно
получить напрямую

Полная функциональная зависимость

Функциональная зависимость $X \rightarrow Y$ является **ПОЛНОЙ**, если **Y не зависит функционально от любого подмножества X .**

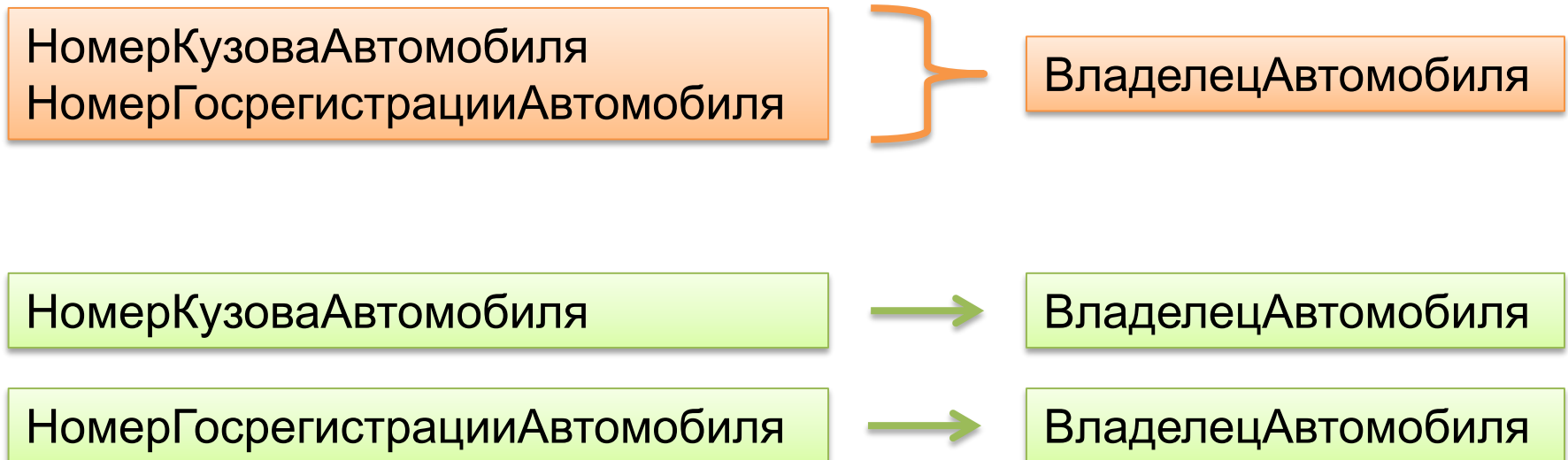
Пример:



Частичная функциональная зависимость

Функциональная зависимость $X \rightarrow Y$ является **ЧАСТИЧНОЙ**, если **Y зависит функционально от некоторого подмножества X** .

Пример:



Частый вопрос

В чём разница между **частичной** и **избыточной** зависимостью?

Частичная:

НомерКузоваАвтомобиля
НомерГосрегистрацииАвтомобиля



«Знать **A && B**» –
ошибка!
Достаточно «**A || B**»

ВладелецАвтомобиля

Избыточная:

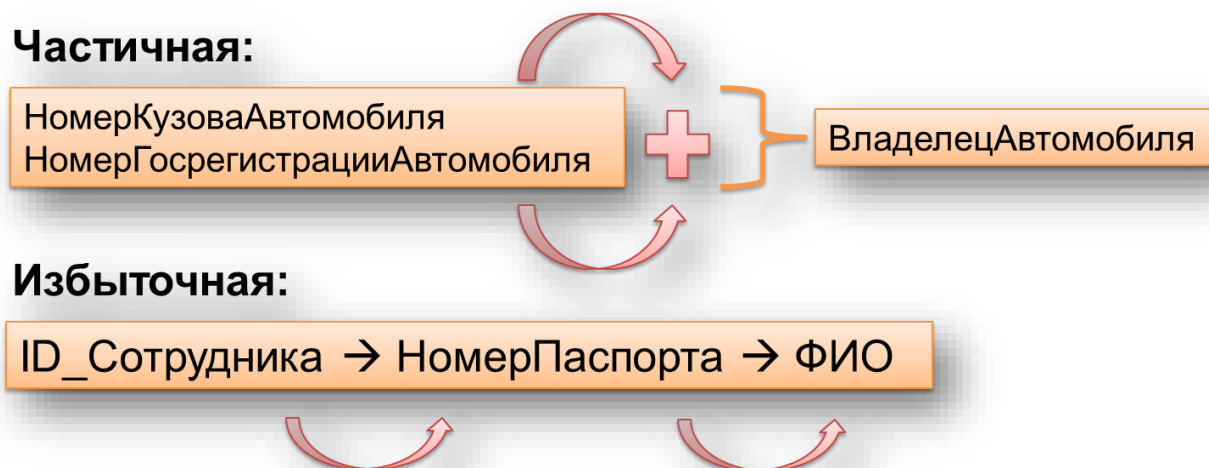
ID_Сотрудника → НомерПаспорта → ФИО

«Чтобы узнать **C**, сначала
по **A** нужно узнать **B**» –
ошибка! Из **A** можно
сразу узнать **C**.

Частый вопрос

Представьте «тупого сыщика»:

- **Частичная:** «мы знаем номер госрегистрации машины, теперь если совпадёт номер кузова, то её владелец – Пупкин»
- **Избыточная:** «мы знаем ID сотрудника, сейчас по нём мы узнаем номер паспорта, а уже по номеру паспорта – фамилию сотрудника»



Транзитивная функциональная зависимость

Функциональная зависимость $X \rightarrow Y$ является **транзитивной**, если существуют зависимости $X \rightarrow Z$ и $Z \rightarrow Y$, но отсутствует прямая зависимость $X \rightarrow Y$.

Пример:

В этой «цепи» нет
«лишних звеньев»

ID_Сотрудника \rightarrow ID_Офиса \rightarrow ТелефонОфиса

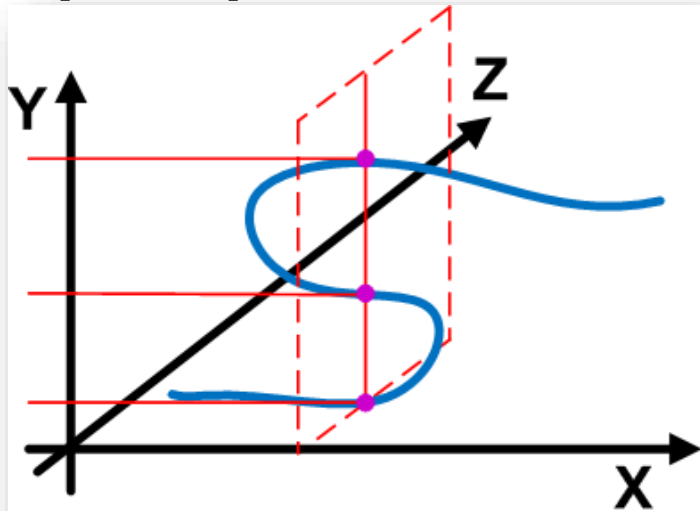
ID_Сотрудника \rightarrow ??? \rightarrow ТелефонОфиса

Так данные получить
нельзя ☹

Многозначная зависимость

Многозначная зависимость $X \twoheadrightarrow Y \mid Z$ существует в том и только в том случае, если множество значений Y , соответствующее паре значений X и Z , зависит только от X и не зависит от Z (то есть **для каждого значения атрибута X существует множество соответствующих значений атрибута Y**).

Примеры:



Проект
Сотрудник
Задание

Сотрудник числится
на проекте вне
зависимости от
выполнения
конкретного задания

Проект \twoheadrightarrow Сотрудник \mid Задание

Задание может
выполнить любой
сотрудник

Многозначная зависимость

Попробуем проще:

из того факта, что в отношении существуют кортежи $R1(X, Y, Z1)$ и $R2(X, Y1, Z)$, следует наличие кортежа $R(X, Y, Z)$.

Чтобы это условие выполнялось, необходимо:

- Абитуриент поступает на 2 и более факультета.
- У 2 и более факультетов есть общий экзамен.

Допустим, в одной таблице решили сохранить информацию об абитуриентах, факультетах и сдаваемых экзаменах.

X	applic	Y	faculty	subject	Z
	Иванов		МатФак	История КПСС	
X	Иванов	Y	МатФак	Математика	
X	Иванов	Y1	ФизФак	История КПСС	Z1
	Иванов		ФизФак	Физика	Z
	Петров		МатФак	История КПСС	
	Петров		МатФак	Математика	

Многозначная зависимость

Попробуем ещё проще:

Абитуриент
Факультет
Экзамен

Абитуриент → → Факультет | Экзамен

Одному абитуриенту
соответствует несколько
экзаменов (на любом
факультете)

Одному абитуриенту
соответствует несколько
факультетов (при любых
экзаменах)

applicant	faculty	subject
Иванов	МатФак	История КПСС
Иванов	МатФак	Математика
Иванов	ФизФак	История КПСС
Иванов	ФизФак	Физика
Петров	МатФак	История КПСС
Петров	МатФак	Математика

Тривиальная и нетривиальная многозначная зависимость

Тривиальная:

Предмет
Книга
Лектор

Лектор и
книги не
связаны.

Предмет → → Книга | Лектор

Предмет → Лектор

Предмет
читает один
лектор.

Содержит хотя бы одну
функциональную
зависимость.

Нетривиальная:

Абитуриент
Факультет
Экзамен

Абитуриент
поступает на много
факультетов и сдаёт
много экзаменов.

Абитуриент → → Факультет | Экзамен

~~Абитуриент → Факультет~~

~~Абитуриент → Экзамен~~

Не содержит
функциональных
зависимостей.

Тривиальная и нетривиальная многозначная зависимость

Тривиальная:

Предмет
Книга
Лектор

Лектор и
книги не
связаны.

Предмет → → Книга | Лектор

Предмет → Лектор

Предмет
читает один
лектор.

Нетривиальная:

Абитуриент
Факультет
Экзамен

Абитуриент
поступает на много
факультетов и сдаёт
много экзаменов.

Абитуриент → → Факультет | Экзамен

~~Абитуриент → Факультет~~

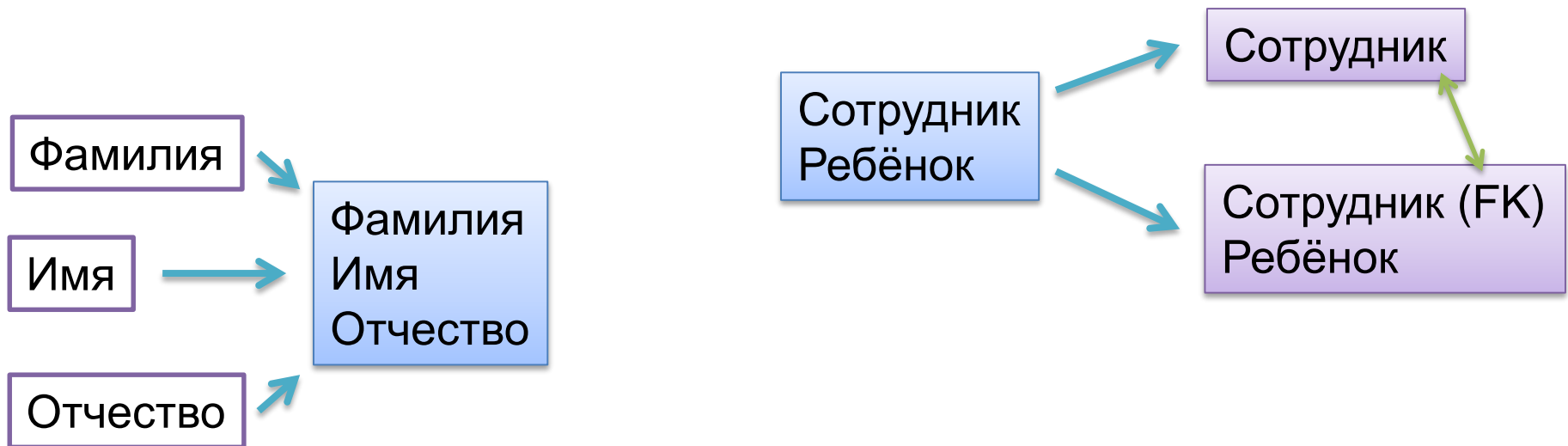
~~Абитуриент → Экзамен~~

Данные условия «предмет читает один лектор», «книги по предмету не зависят от лектора», «абитуриент может поступать на разные факультеты» и т.п. – это **ОГРАНИЧЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ**, т.е. «так решено». **И может быть «решено иначе» в других случаях.**

НОРМАЛИЗАЦИЯ И НОРМАЛЬНЫЕ ФОРМЫ

Нормализация

Нормализация (normalisation) – группировка и/или распределение атрибутов по отношениям с целью устранения аномалий операций с БД, обеспечения целостности данных и оптимизации модели БД.



ТРЕБОВАНИЯ НОРМАЛИЗАЦИИ

Как правило, **не существует «единственно правильного способа нормализации»** для достаточно сложной БД – у всех решений есть плюсы и минусы. Но желательно придерживаться следующих требований...

Эти требования могут противоречить друг другу, так что не стремитесь выполнить их все любой ценой. Выбирайте то, что важно для вашей конкретной БД!

Требование минимальности первичных ключей

Первичные ключи отношений должны быть минимальными.

!!!

Это требование
идеально
выполняется с
введением
суррогатных РК.

class Anomaly fixed

Employee

«column»

id: INTEGER

passport: VARCHAR(20)

pass_id: VARCHAR(50)

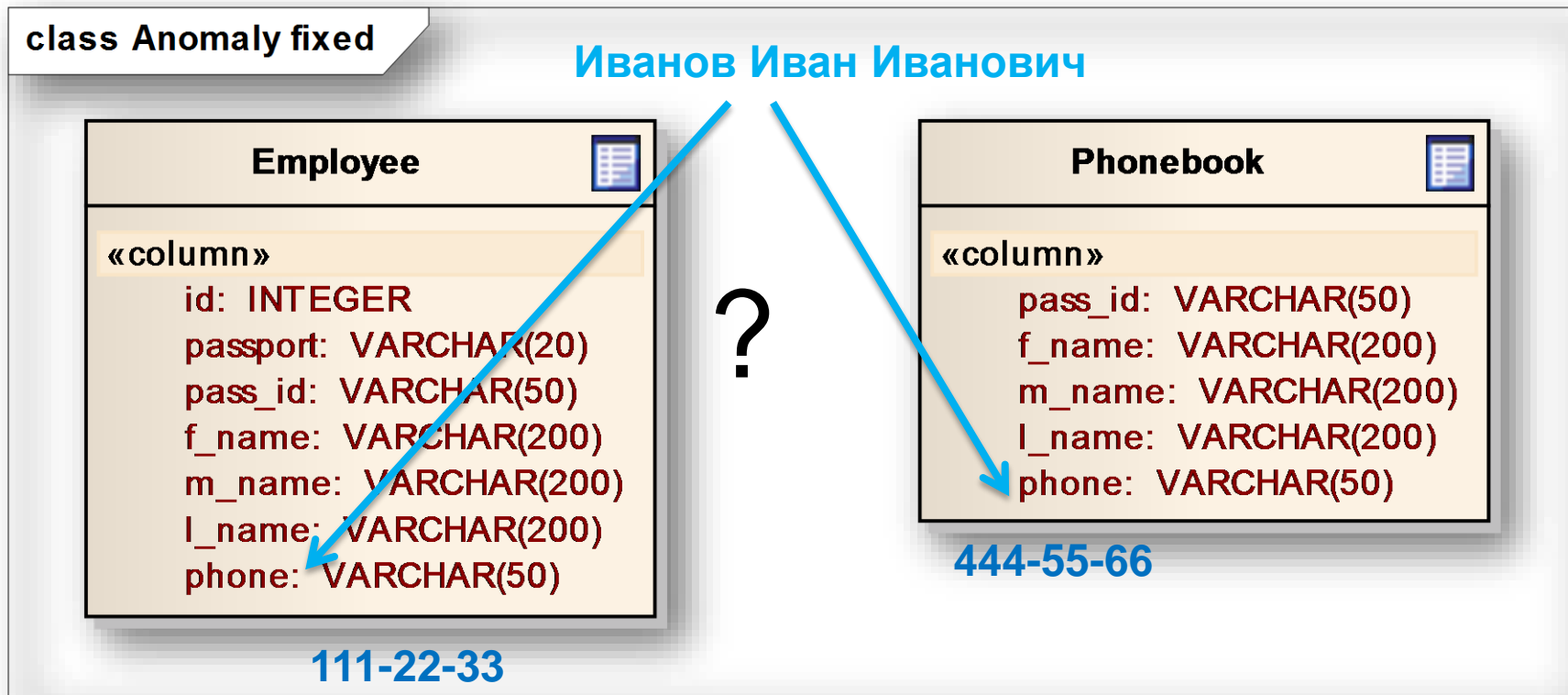
f_name: VARCHAR(200)

m_name: VARCHAR(200)

l_name: VARCHAR(200)

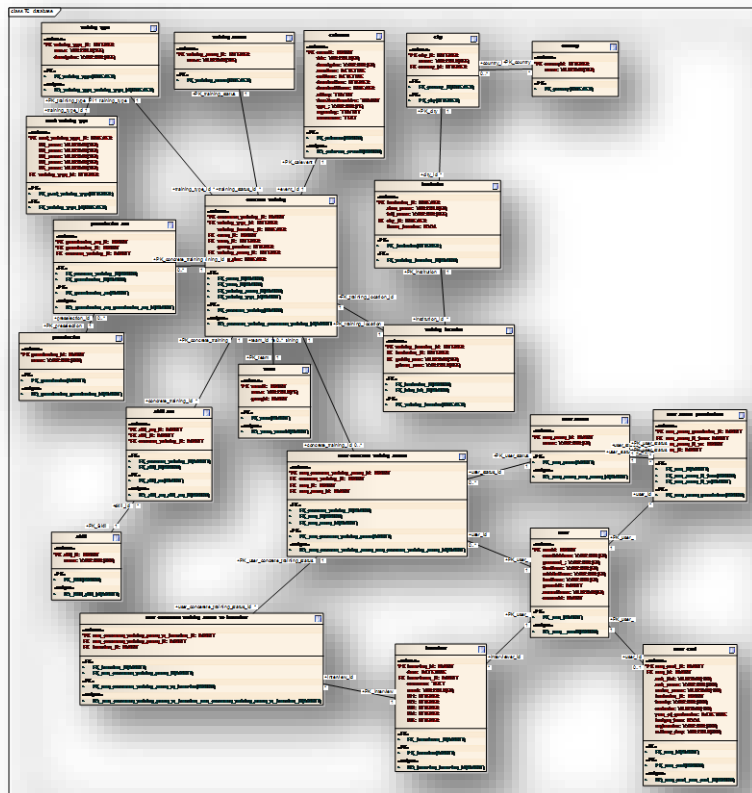
Требование надёжности данных

Модель БД должна по возможности минимизировать или устранять избыточность данных.



Требование производительности системы

Модель БД должна позволять обеспечивать необходимую производительность операций.



```
SELECT `news_keywords`.`nk_url`,  
`news_keywords`.`nk_name`,  
count(*) as 'q' from  
`news_keywords` RIGHT JOIN  
`n_m2m_nk` ON  
`news_keywords`.`nk_uid`=`n_m2m_nk`.  
`nk_uid` group by  
`n_m2m_nk`.`nk_uid` order by `q`  
desc, `news_keywords`.`nk_name`  
asc limit 100
```

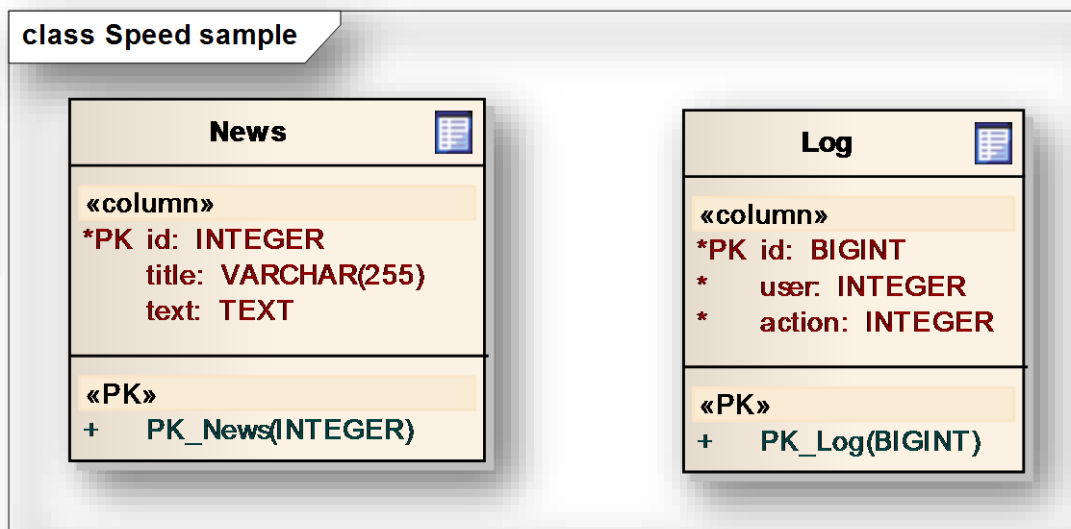
Хммм... Возможно, и обеспечивает, но...
есть сомнения.

Требование сохранения производительности

Разброс времени реакции на различные операции с данными должен быть минимальным.

Это требование выполняют крайне редко, т.к. очень часто наблюдается явный «перевес» в сторону каких-то операций при реальном использовании БД.

Read = 30K/D
Write = 5/D
Update = 2/D

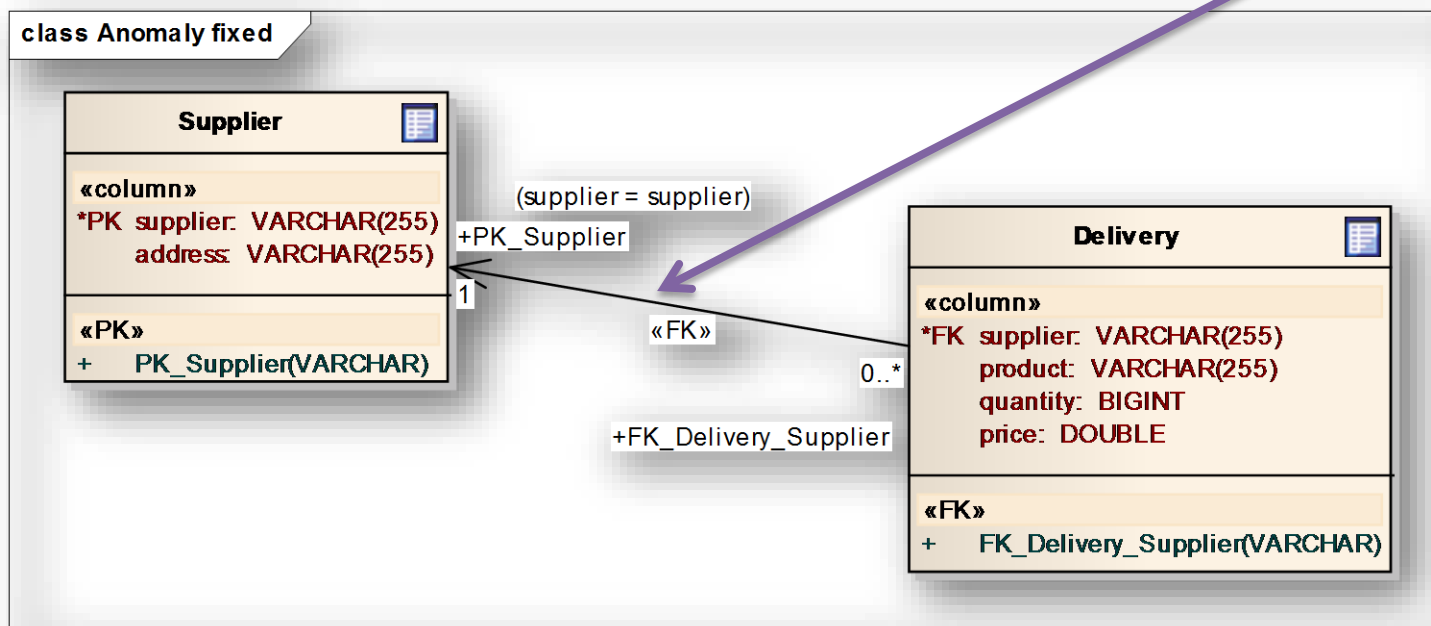


Read = 5/D
Write = 200K/D
Update = -

Требование непротиворечивости данных

Модель БД должна минимизировать вероятность возникновения противоречивости данных при любых операциях с данными.

Иными словами – связи должны быть установлены ЯВНО!



Модель БД должна быть **способной к адаптации** в случае необходимости внесения изменений.

Это достигается за счёт:

- Мнемоничных имён.
- Комментариев.
- Документации.
- Схемы в общепринятой нотации.
- Отсутствия глупых ограничений.

«abc» -- плохо,

К полям, таблицам, хорошо

Сначала её не пишут.

а потом прок
тех, кто в сво
не на

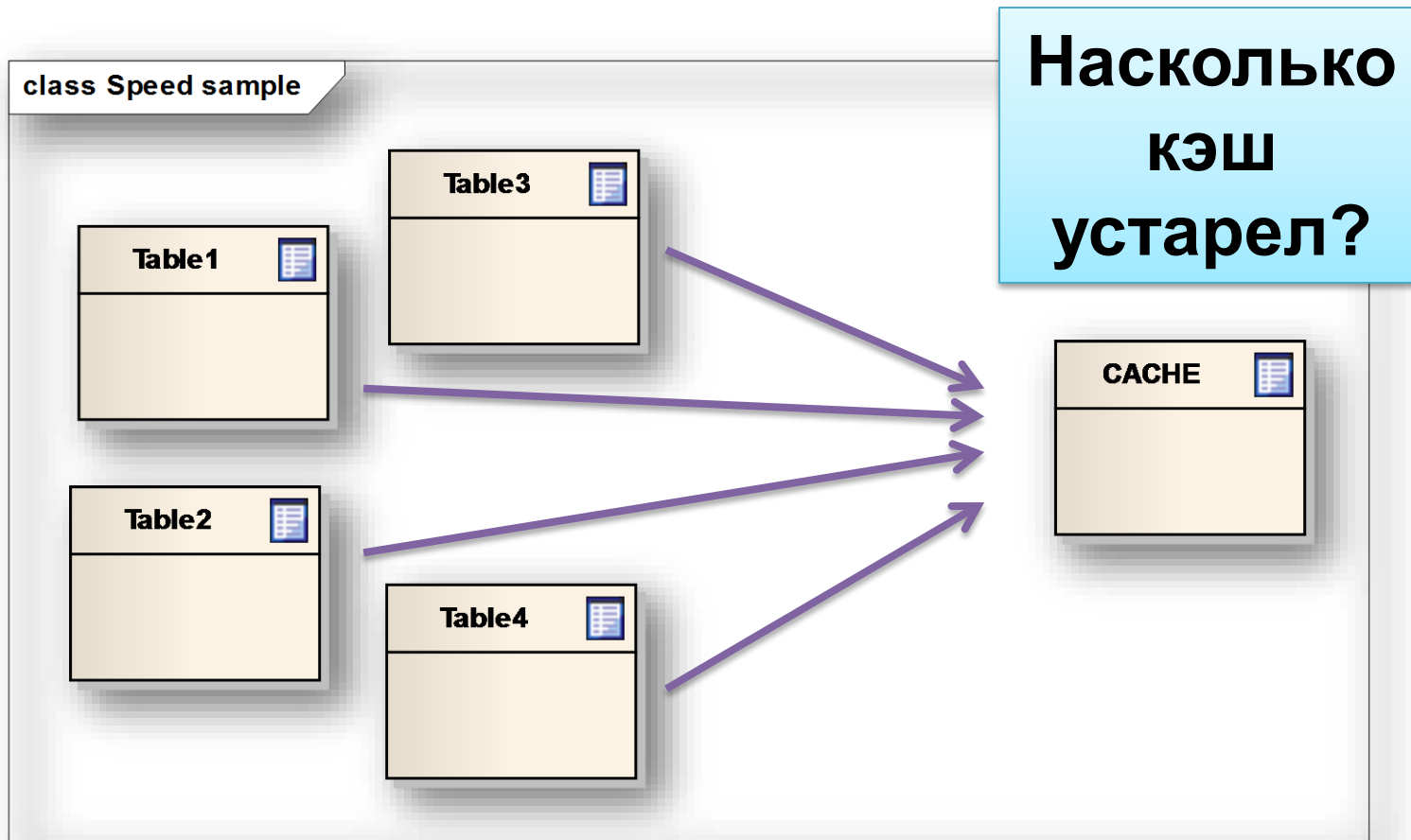
Хорошо подходит UML
или IDEF0.

И при этом ещё нет
документации 😊 --
вообще жуть
получается.

... вроде ID
пользователя
размером в 1 байт.

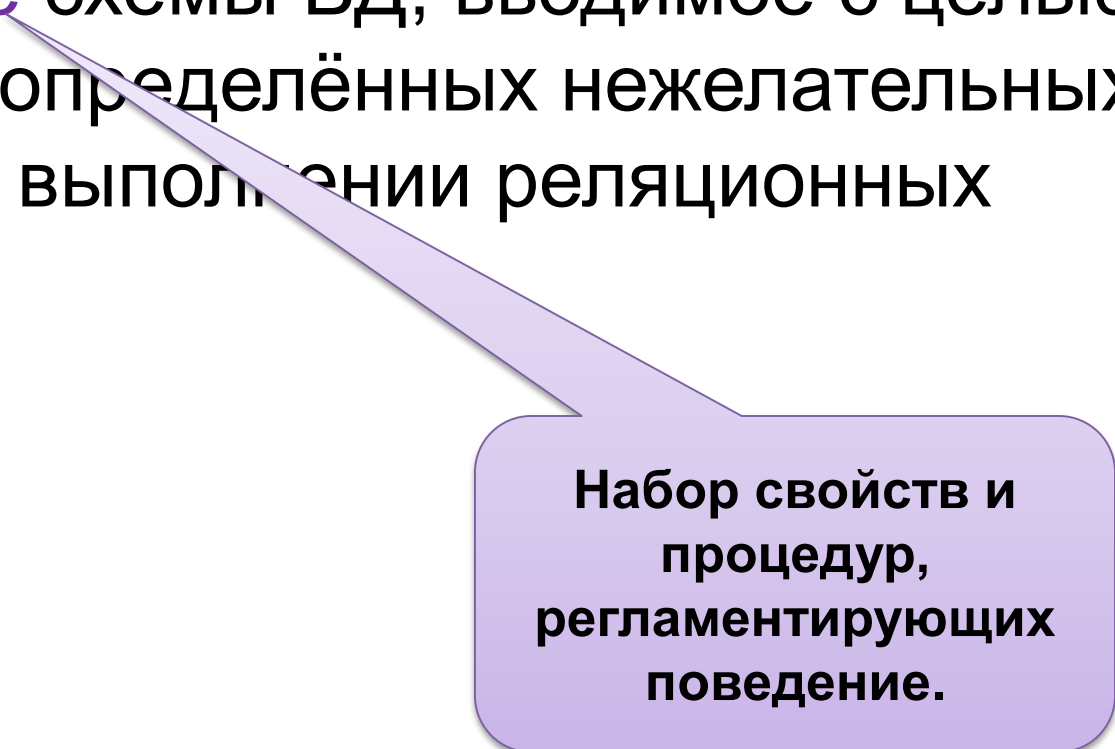
Требование актуальности данных

В каждый момент времени БД должна содержать актуальный набор данных.



НОРМАЛЬНЫЕ ФОРМЫ НИЗКИХ ПОРЯДКОВ

Нормальная форма (НФ, normal form) – **ограничение** схемы БД, вводимое с целью устранения определённых нежелательных свойств при выполнении реляционных операций.



Набор свойств и процедур, регламентирующих поведение.

Первая нормальная форма (1НФ)

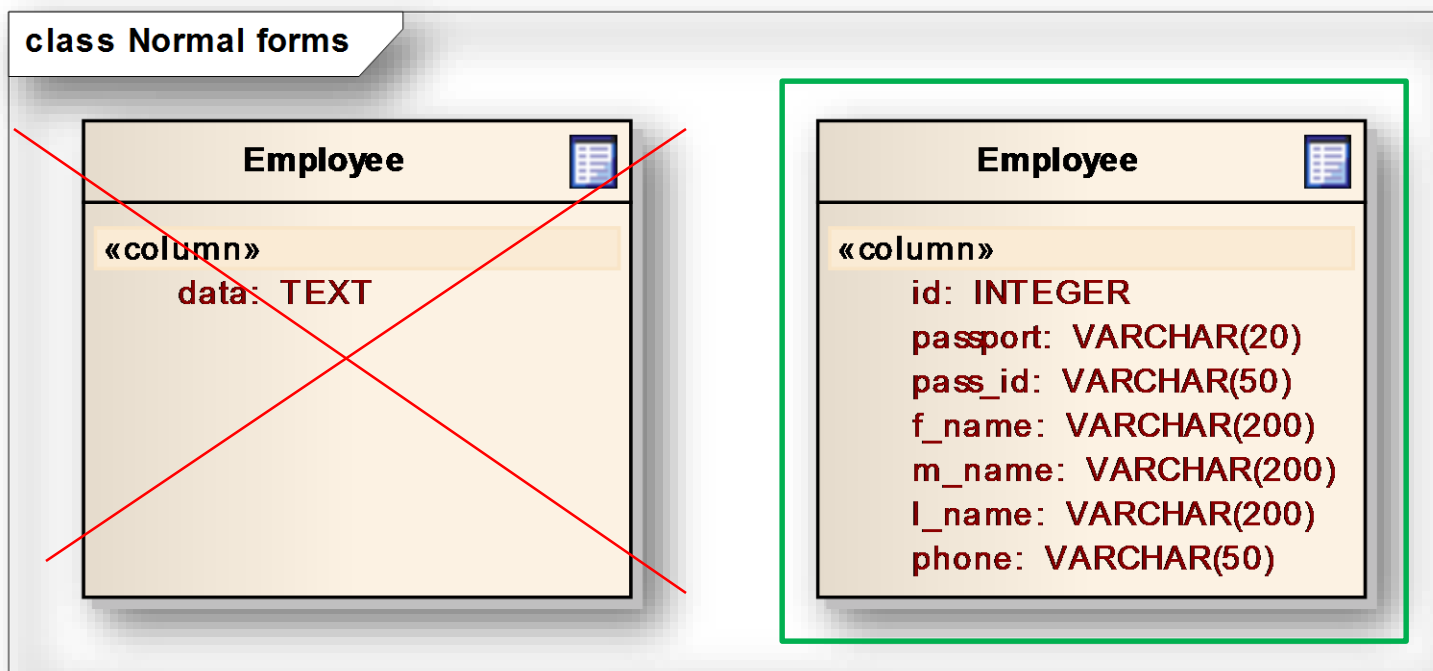
Отношение находится в **1НФ**, если все его атрибуты являются **атомарными**, т.е. не имеют компонентов.



Атрибут будет считаться атомарным, если в предметной области **не существует операции**, для выполнения которой понадобилось бы извлечь часть атрибута.

Первая нормальная форма (1НФ)

На рисунке слева представлена схема **не**нормализованного (до 1НФ) отношения, справа – нормализованного.



Первая нормальная форма (1НФ)

Внимание! Атомарность должна соблюдаться **на уровне БД**, т.е. с помощью БД не должно выполняться никаких операций над частью поля.

Поэтому, например, хранение в поле **сериализованных данных**, которые **всегда добавляются, обновляются, удаляются и извлекаются ЦЕЛИКОМ**, хоть и является признаком хорошего тона, но имеет право на существование.

umeta_id	user_id	meta_key	meta_value
11	1	wp_user_level	10
10	1	wp_capabilities	a:1:{s:13:"administrator";s:1:"1"}



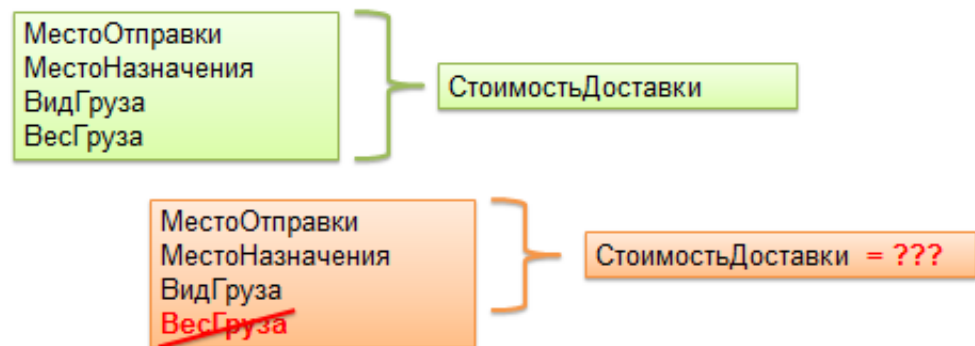
Вторая нормальная форма (2НФ)

Отношение находится во **2НФ**, если оно находится в 1НФ, и при этом любой атрибут, не входящий в состав ПК, функционально полно зависит от ПК.

Полная функциональная зависимость

Функциональная зависимость $X \rightarrow Y$ является **ПОЛНОЙ**, если **Y не зависит функционально от любого подмножества X**.

Пример:

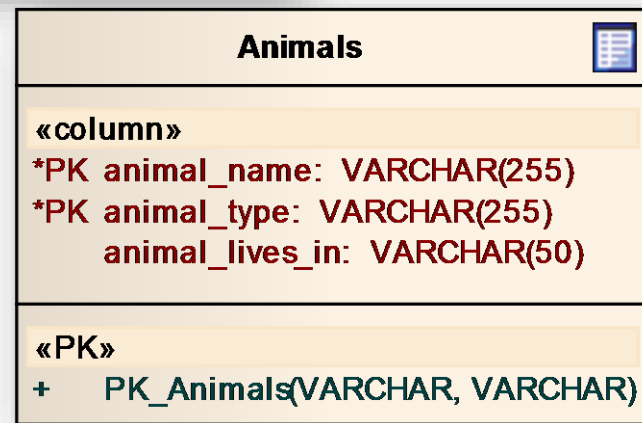


Вторая нормальная форма (2НФ)

Представим, что у нас есть следующее отношение:

lives_in зависит только от **type**, а не от {**name**, **type**}

class Normal forms

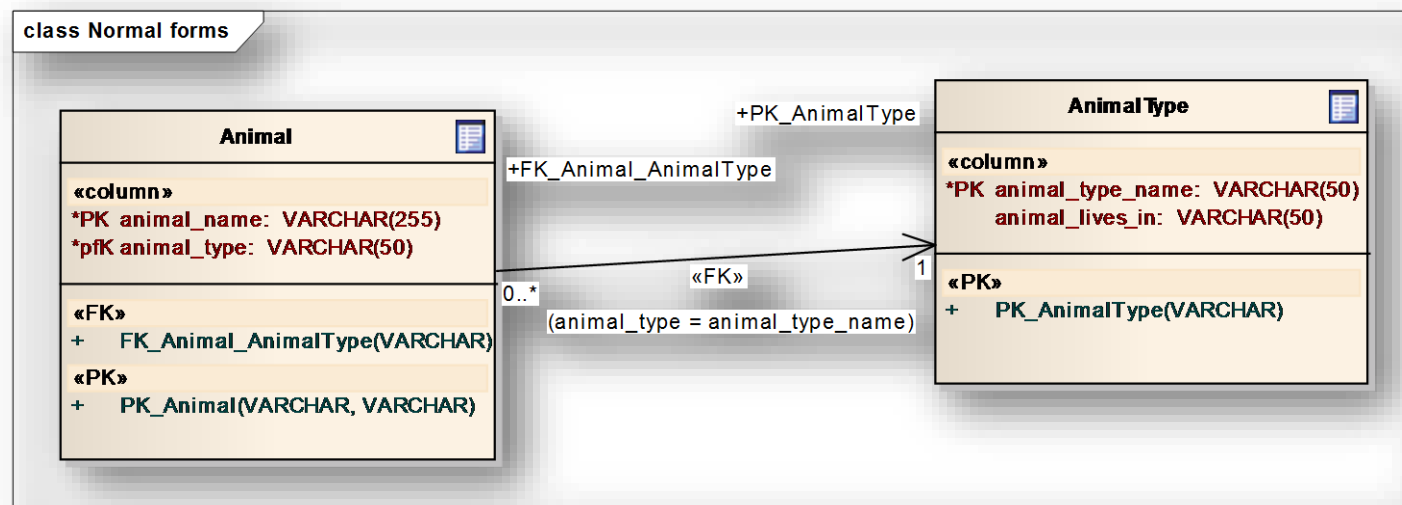


В силу человеческой или программной ошибки очень легко получить вот такой бред:

animal_name	animal_type	animal_lives_in
Пушок	Кот	Аквариум
Йааааазь!	Рыбина	На диване

Вторая нормальная форма (2НФ)

Если мы исправим модель БД (приведём ко 2НФ), всё будет хорошо:



animal_name	animal_type
Пушок	Кот
Йаааазь!	Рыбина

animal_type_name	animal_lives_in
Кот	На диване
Рыбина	Аквариум

Третья нормальная форма (3НФ)

Отношение находится в **3НФ**, если оно находится во 2НФ и при этом любой его неключевой атрибут нетранзитивно (напрямую) зависит от первичного ключа.

Транзитивная функциональная зависимость

Функциональная зависимость $X \rightarrow Y$ является **транзитивной**, если **существуют зависимости $X \rightarrow Z$ и $Z \rightarrow Y$, но отсутствует прямая зависимость $X \rightarrow Y$.**

Пример:

В этой «цепи» нет
«лишних звеньев»

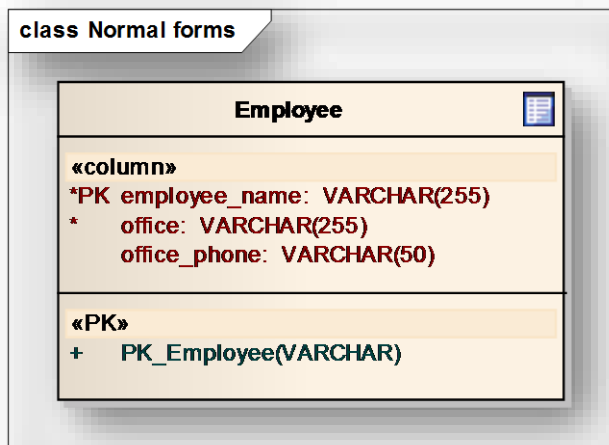
ID_Сотрудника \rightarrow ID_Офиса \rightarrow ТелефонОфиса

ID_Сотрудника \rightarrow ??? \rightarrow ТелефонОфиса

Так данные получить
нельзя ☹

Третья нормальная форма (3НФ)

На практике нарушение 3НФ проще всего отследить ещё и по **бессмысленному дублированию данных в разных строках**.



Сотрудник (PK) → Офис → ТелефонОфиса

Транзитивная зависимость

Сотрудник (PK) → Офис

??????? (PK) → ТелефонОфиса

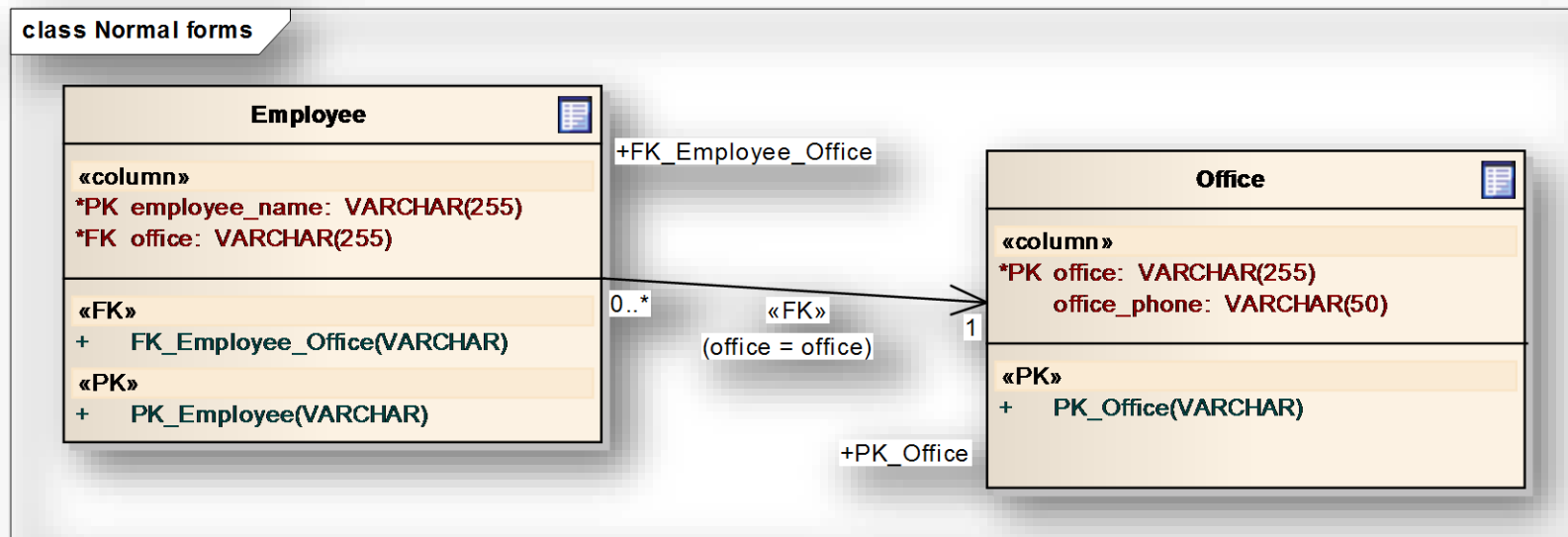
**Должно
быть так**

А в реальности об офисе будет 20-30 полей, и все они продублируются столько раз, сколько в офисе сотрудников.

employee_name	office	office_phone
Иванов И.И.	Офис-1	111-22-33
Петров П.П.	Офис-1	111-22-33
Сидоров С.С.	Офис-2	999-88-77

Третья нормальная форма (3НФ)

Приведём отношение к 3НФ:



Сотрудник → Офис

Офис → ТелефонОфиса

Зависимости **НЕ**транзитивные

Третья нормальная форма (3НФ)

С бессмысленным дублированием данных об офисе (напомним – могут быть десятки столбцов) теперь **тоже всё хорошо – дублирования нет:**

employee_name	office
Иванов И.И.	Офис-1
Петров П.П.	Офис-1
Сидоров С.С.	Офис-2

office	office_phone
Офис-1	111-22-33
Офис-2	999-88-77

**И тут ещё
10-20
полей**

НОРМАЛЬНЫЕ ФОРМЫ ВЫСОКИХ ПОРЯДКОВ («БОНУСНЫЙ» МАТЕРИАЛ)

Disclaimer

Приведение отношения к нормальным формам высоких порядков требуется в достаточно редких, хорошо обоснованных случаях.



Не страшно, если вы не запомните эти формы. Главное – **понять их, знать о них, представлять решаемые с их помощью задачи.**

Нормальная форма Бойса-Кодда (НФБК)

Отношение находится в **НФБК**, когда детерминанты всех функциональных зависимостей являются потенциальными ключами.

Детерминант – любой атрибут, от которого функционально-полно зависит некоторый другой атрибут.

То, что стоит «слева от стрелочки» в выражениях вида $N \rightarrow M$ (здесь N – детерминант). Он может быть сложным: $\{A, B\} \rightarrow C$.

МестоОтправки
МестоНазначения
ВидГруза
~~ВесГруза~~

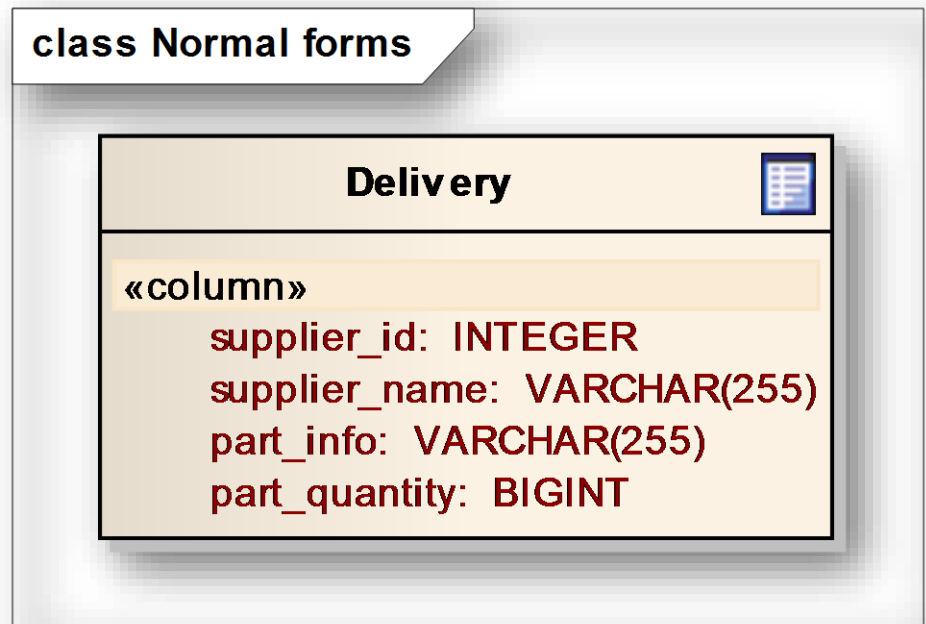
СтоимостьДоставки = ???

Нормальная форма Бойса-Кодда (НФБК)

Теперь ещё проще, «на пальцах»: отношение находится в 3НФ, но **НЕ** находится в НФБК, если оно имеет **несколько составных потенциальных ключей**, у которых есть **общие атрибуты**:

Требуется хранить данные о поставках деталей.

- Наименования поставщиков являются уникальными.
- Каждый поставщик имеет свой уникальный номер.
- **Поставщик не поставляет дважды одну и ту же деталь.**



Нормальная форма Бойса-Кодда (НФБК)

Рассмотрим возможные ключи и наличие зависимостей:

Возможные ключи:

{supplier_id, part_info}

{supplier_name, part_info}

Зависимости:

supplier_id → supplier_name

supplier_name → supplier_id

{supplier_id, part_info} → part_quantity

{supplier_name, part_info} → part_quantity

{supplier_id, part_info} → supplier_name

{supplier_name, part_info} → supplier_id

class Normal forms

Delivery

«column»

supplier_id: INTEGER

supplier_name: VARCHAR(255)

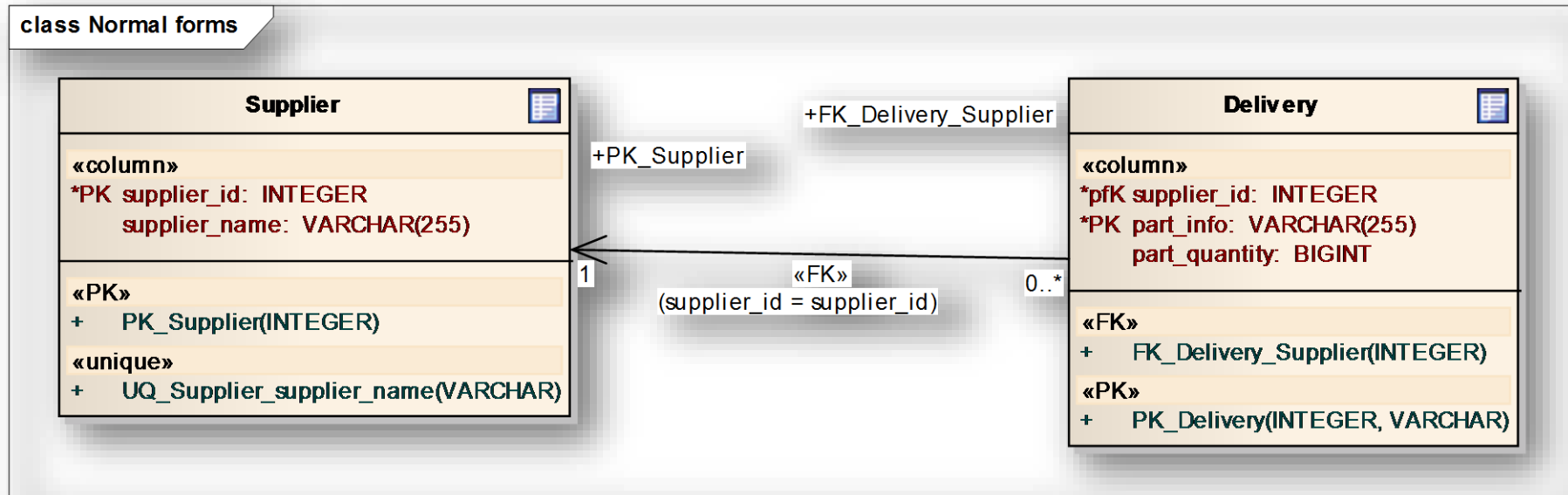
part_info: VARCHAR(255)

part_quantity: BIGINT

Детерминанты
supplier_id и
supplier_name НЕ
являются
потенциальными
ключами.

Нормальная форма Бойса-Кодда (НФБК)

Изменим схему БД:



`supplier_id` → `supplier_name`
`supplier_name` → `supplier_id`

`{supplier_id, part_info}` → `part_quantity`

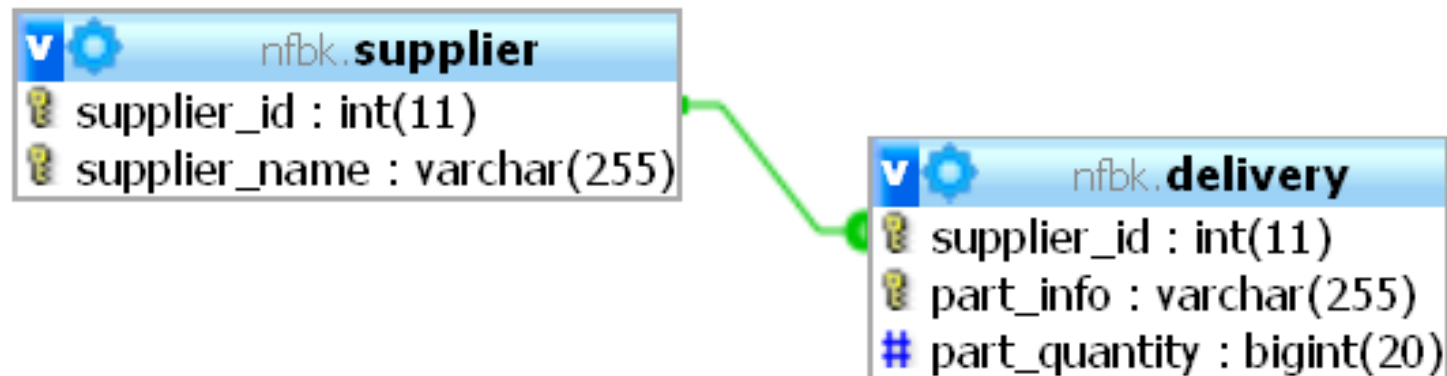
**Детерминанты всех функциональных
зависимостей являются
потенциальными ключами!**

Нормальная форма Бойса-Кодда (НФБК)

Зачем это нужно?

НФБК позволяет:

- Устранить избыточность хранения данных.
- Упростить операции с отдельными таблицами.
- Устраняет аномалии вставки, обновления, удаления.



Четвёртая нормальная форма (4НФ)

Отношение находится в **4НФ**, если оно находится в 3НФ или НФБК и не содержит нетривиальных многозначных зависимостей (т.е. все его зависимости являются функциональными от ключа).

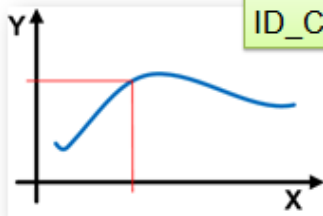
Функциональная зависимость

Если даны два атрибута X и Y некоторого отношения, то Y функционально зависит от X , если в любой момент времени **каждому значению X соответствует ровно одно значение Y** .

Обозначается $X \rightarrow Y$.

Примеры:

НомерПаспорта \rightarrow Фамилия
ID_Сотрудника \rightarrow СтажРаботыВФирме



Тривиальная и нетривиальная многозначная зависимость

Тривиальная:

Предмет
Книга
Лектор

Лектор и
книги не
связаны.

Предмет \rightarrow Книга | Лектор

Предмет \rightarrow Лектор

Предмет
читает один
лектор.

Содержит хотя бы одну функциональную зависимость.

Нетривиальная:

Абитуриент
Факультет
Экзамен

Абитуриент \rightarrow Факультет | Экзамен

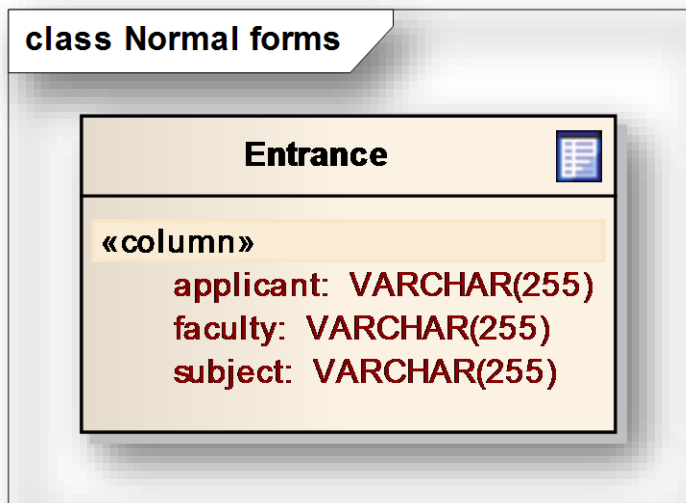
~~Абитуриент \rightarrow Факультет~~

~~Абитуриент \rightarrow Экзамен~~

Не содержит функциональных зависимостей.

Четвёртая нормальная форма (4НФ)

Рассмотрим уже знакомый пример:



applicant	faculty	subject
Иванов	МатФак	История КПСС
Иванов	МатФак	Математика
Иванов	ФизФак	История КПСС
Иванов	ФизФак	Физика
Петров	МатФак	История КПСС
Петров	МатФак	Математика

Здесь присутствует
нетривиальная
многозначная
зависимость:

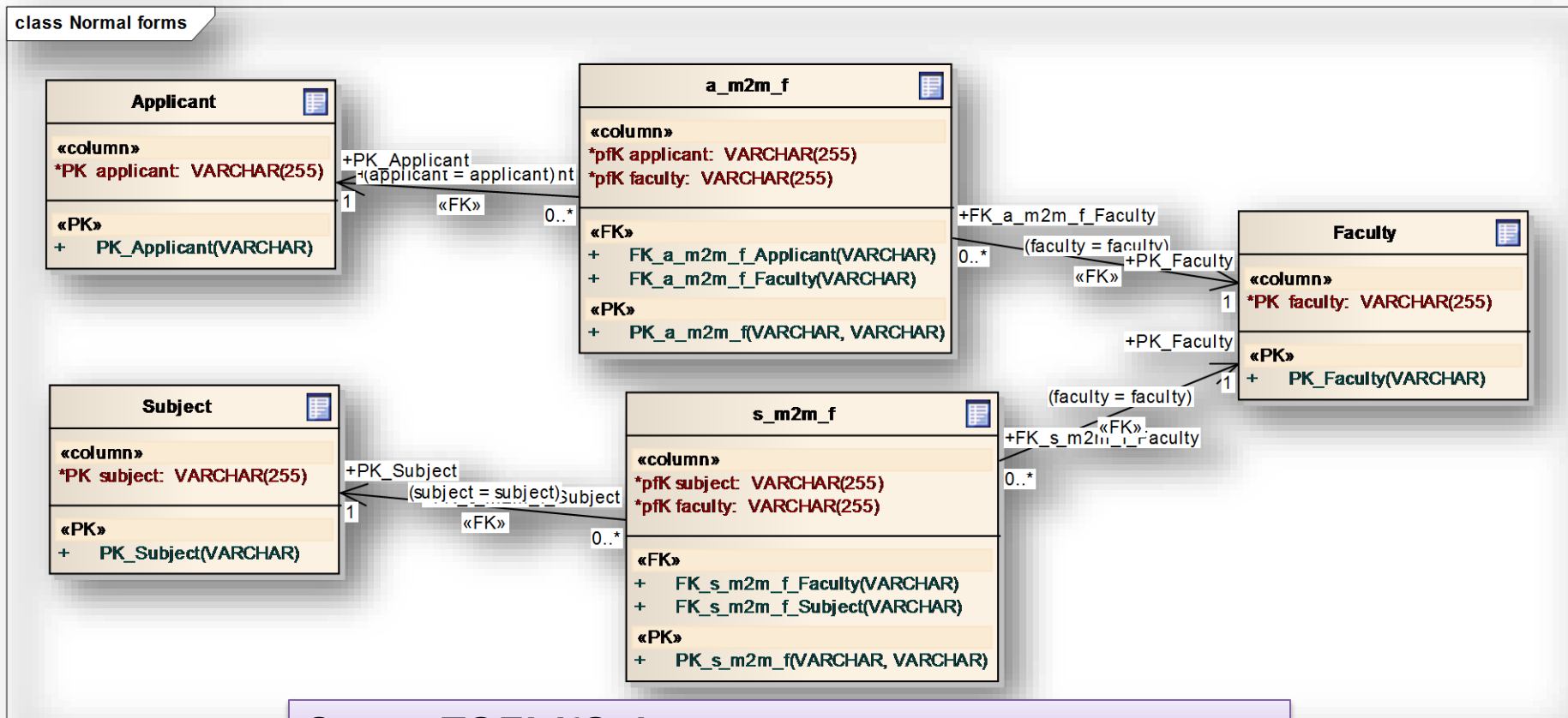
Абитуриент → → Факультет | Экзамен

~~Абитуриент → Факультет~~

~~Абитуриент → Экзамен~~

Четвёртая нормальная форма (4НФ)

Чтобы устранить нетривиальную многозначную зависимость, нужно изменить модель БД так:



Здесь ТОЛЬКО функциональные зависимости.

Пятая нормальная форма (5НФ)

Отношение находится в **5НФ**, если оно находится в 4НФ и любая **многозначная зависимость** соединения в нём является **тривиальной**.

Признак того, что это **не выполняется**: у отношения есть несколько пересекающихся составных потенциальных ключей.



Пятая нормальная форма (5НФ)

Здесь не будем «копать математику», а рассмотрим пример:

it

device	os	version
comp	windows	server
comp	linux	user
phone	windows	user
comp	windows	user

do

device	os
comp	windows
comp	linux
phone	windows

dv

device	version
comp	server
comp	user
phone	user

ov

os	version
windows	server
linux	user
windows	user

Получим три проекции отношения:

Пятая нормальная форма (5НФ)

Если теперь «собрать назад» исходную таблицу с помощью попарных JOIN'ов, получится:

```
SELECT * from `dv` join `do` on  
`dv`.`device`=`do`.`device`
```

```
SELECT * from `dv` join `ov` on  
`dv`.`version`=`ov`.`version`
```

```
SELECT * from `do` join `ov` on  
`do`.`os`=`ov`.`os`
```

dv

device	version
comp	server
comp	user
phone	user

do

device	os
comp	windows
comp	linux
phone	windows

ov

os	version
windows	server
linux	user
windows	user

device version device os

comp	server	comp	linux
comp	server	comp	windows
comp	user	comp	linux
comp	user		
phone	user		

device version os version

comp	user	linux	user
phone	user	linux	user
comp	server	windows	server
comp	user	windows	user
phone			

device os os version

comp	linux	linux	user
comp	windows	windows	server
comp	windows	windows	user
phone	windows	windows	server
phone	windows	windows	user

Пятая нормальная форма (5НФ)

Итак, мы получили:

5

device	version	device	os
comp	server	comp	linux
comp	server	comp	windows
comp	user	comp	linux
comp	user	comp	windows
phone	user	phone	windows

5

device	version	os	version
comp	user	linux	user
phone	user	linux	user
comp	server	windows	server
comp	user	windows	user
phone	user	windows	user

5

device	os	os	version
comp	linux	linux	user
comp	windows	windows	server
comp	windows	windows	user
phone	windows	windows	server
phone	windows	windows	user

А было:

4

device	os	version
comp	windows	server
comp	linux	user
phone	windows	user
comp	windows	user

Нетривиальная
многозначная зависимость
соединения: **после JOIN'а
таблиц появляются
лишние строки.**

Пятая нормальная форма (5НФ)

А вдруг сработает «тройной JOIN»?!

```
SELECT distinct `dv`.device, `ov`.`os`, `dv`.`version`  
from `dv` join `do` on `dv`.`device`=`do`.`device` join  
`ov` on `do`.`os`=`ov`.`os`
```

Ожидали:

Получили:

4

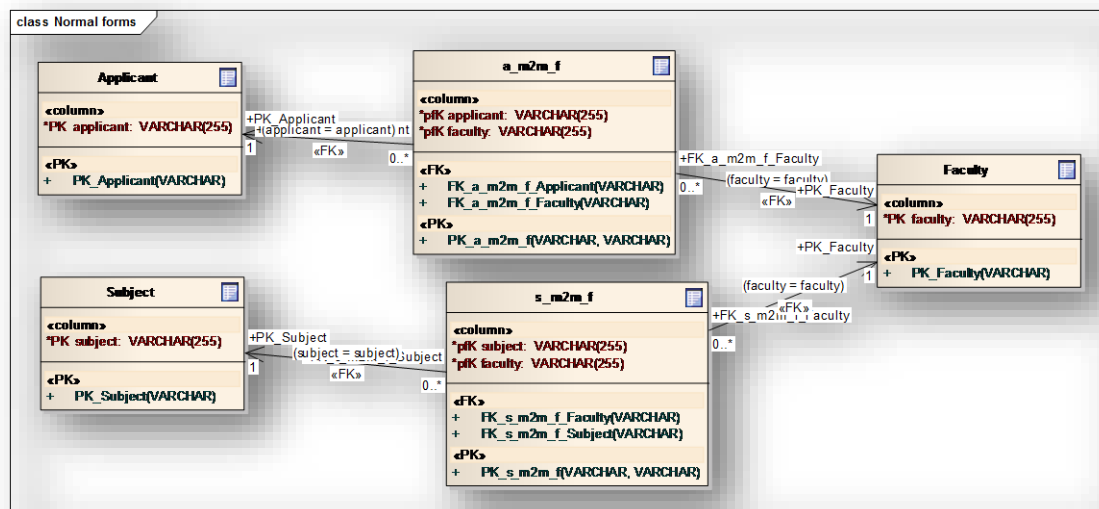
device	os	version
comp	windows	server
comp	linux	user
phone	windows	user
comp	windows	user

5

device	os	version
comp	linux	server
comp	windows	server
comp	linux	user
comp	windows	user
phone	windows	user

Пятая нормальная форма (5НФ)

Что делать? Декомпозировать отношение подобным образом, как мы делали это в случае 4НФ: **разнести атрибуты в отдельные отношения, между которыми установить связи «многие ко многим» и, если нужно, наложить ограничения с помощью триггеров.**



Доменно-ключевая нормальная форма (ДКНФ)

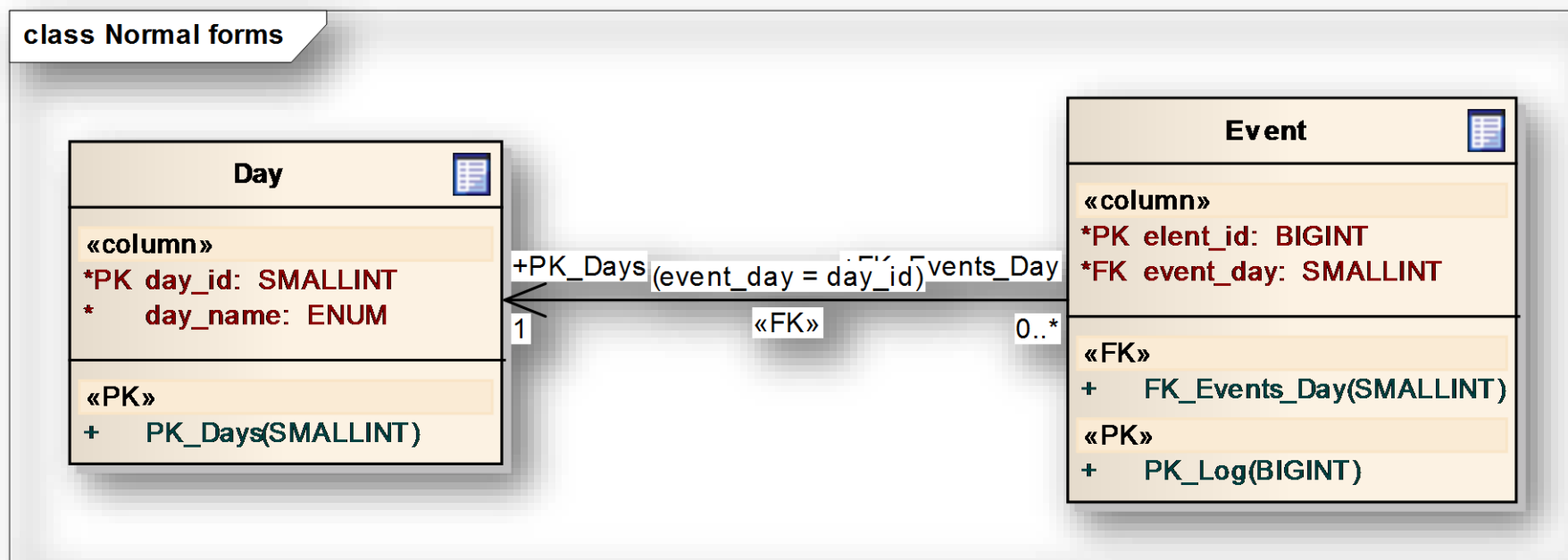
Отношение находится в **ДКНФ**, если его структура не допускает аномалий вставки, обновления и удаления, а также позволяет контролировать значения атрибутов там, где это имеет смысл.

Иными словами:

- Проведите связи и укажите каскадные операции.
- Правильно используйте типы данных.

Доменно-ключевая нормальная форма (ДКНФ)

Пример: есть некоторый лог, в котором в т.ч. указывается день, в который произошло событие.



Доменно-ключевая нормальная форма (ДКНФ)

Что будет, если «добавить день перед понедельником»?

Всё ОК, СУБД это учтёт, данные не нарушатся.

Column	Type	Collation	Attributes	Null	Default
<u>day_id</u>	smallint(6)			No	None
day_name	enum('Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота', 'Воскресенье')	utf8_general_ci		No	None

Column	Type	Collation	Attributes	Null	Default
<u>elent_id</u>	bigint(20)			No	None
event_day	smallint(6)			No	None

Связь с запретом каскадных операций

normalisation_dknf.day

day_id : smallint(6)

day_name : enum('Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота', 'Воскресенье')

Уникальное значение

Ограничение на уровне типа данных

normalisation_dknf.event

elent_id : bigint(20)

event_day : smallint(6)

Шестая нормальная форма (6НФ)

Отношение находится в **6НФ** в случае, если оно находится в 5НФ, и его проекции не приводят к потере «темпоральных» (временных) данных.

class Normal forms	
Log	
«column»	
event_date:	DATE
event_info:	VARCHAR(50)

event_date	event_info
2012-03-01	Иванову И.И. выдан ключ от кабинета
2012-03-01	Иванов И.И. сдал ключ от кабинета
2012-03-01	Иванову И.И. выдан ключ от кабинета
2012-03-01	Иванов И.И. сдал ключ от кабинета

Шестая нормальная форма (6НФ)

Проекции этого отношения с последующей «сборкой» JOIN'ом дадут:

event_date

2012-03-01

event_info

Иванову И.И. выдан ключ от кабинета

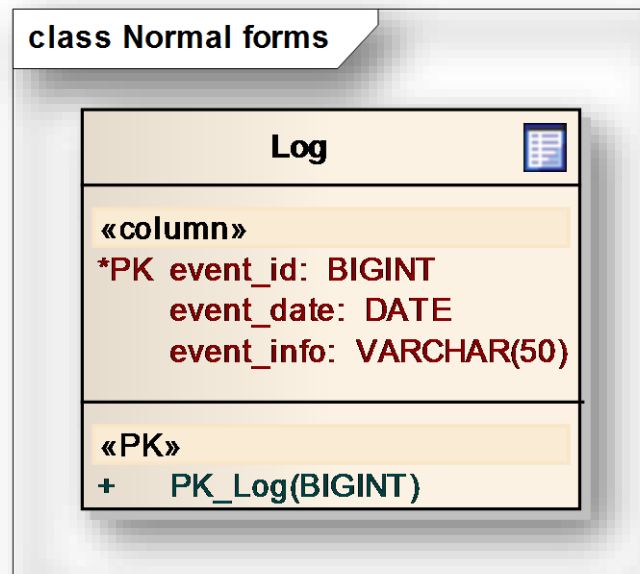
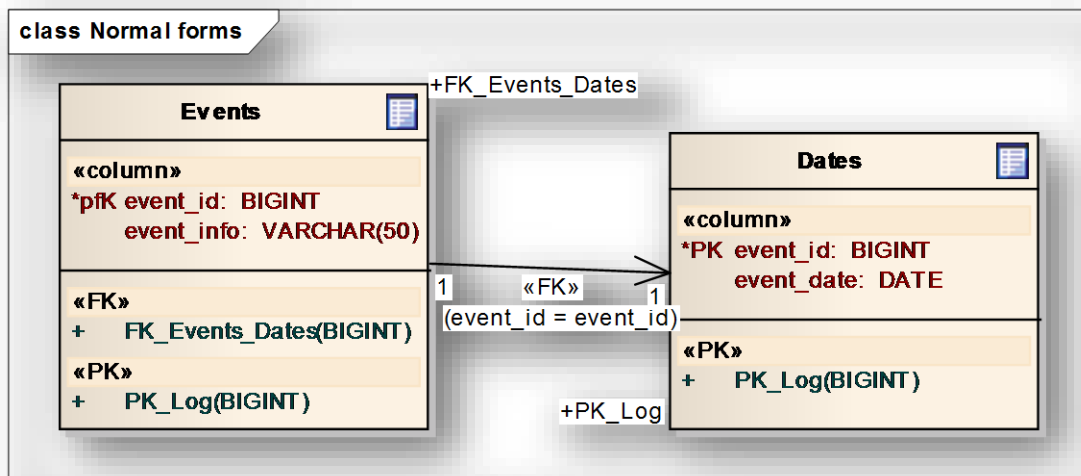
Иванов И.И. сдал ключ от кабинета

event_date	event_info
2012-03-01	Иванову И.И. выдан ключ от кабинета
2012-03-01	Иванов И.И. сдал ключ от кабинета

Информация о том, что Иванов И.И. приходил в этот день в кабинет дважды, утеряна.

Шестая нормальная форма (6НФ)

Что делать? Либо **декомпозировать** отношение так, как это было в случае 4НФ, либо **добавлять в отношение атрибуты** (например, идентификатор события).



Краткий справочник по нормальным формам

1НФ – все атрибуты отношения атомарны.

2НФ – 1НФ + нет частичных функциональных зависимостей.

3НФ – 2НФ + нет транзитивных зависимостей неключевых атрибутов от ключа.

НФБК – 3НФ + детерминанты всех функциональных зависимостей являются потенциальными ключами.

4НФ – 3НФ + нет нетривиальных многозначных зависимостей.

5НФ – 4НФ + восстановление из проекций воссоздаёт исходное отношение.

ДКНФ – отношение не имеет аномалий модификации.

6НФ – 5НФ + проекции сохраняют временные данные.

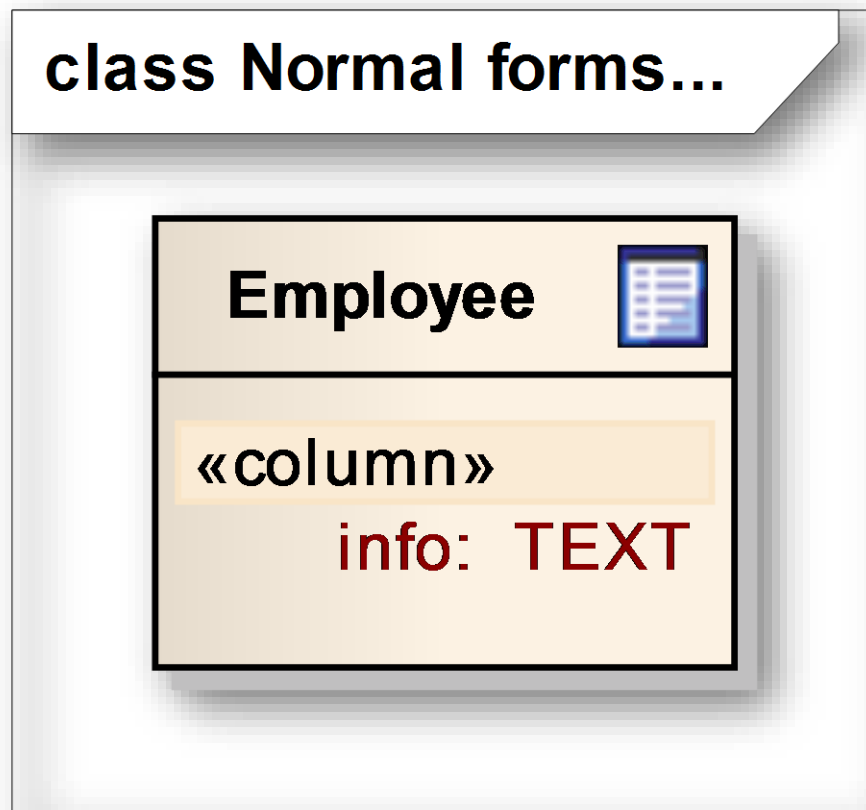
ПРИМЕР ПРИМЕНЕНИЯ НОРМАЛИЗАЦИИ

Немного ближе к реальности

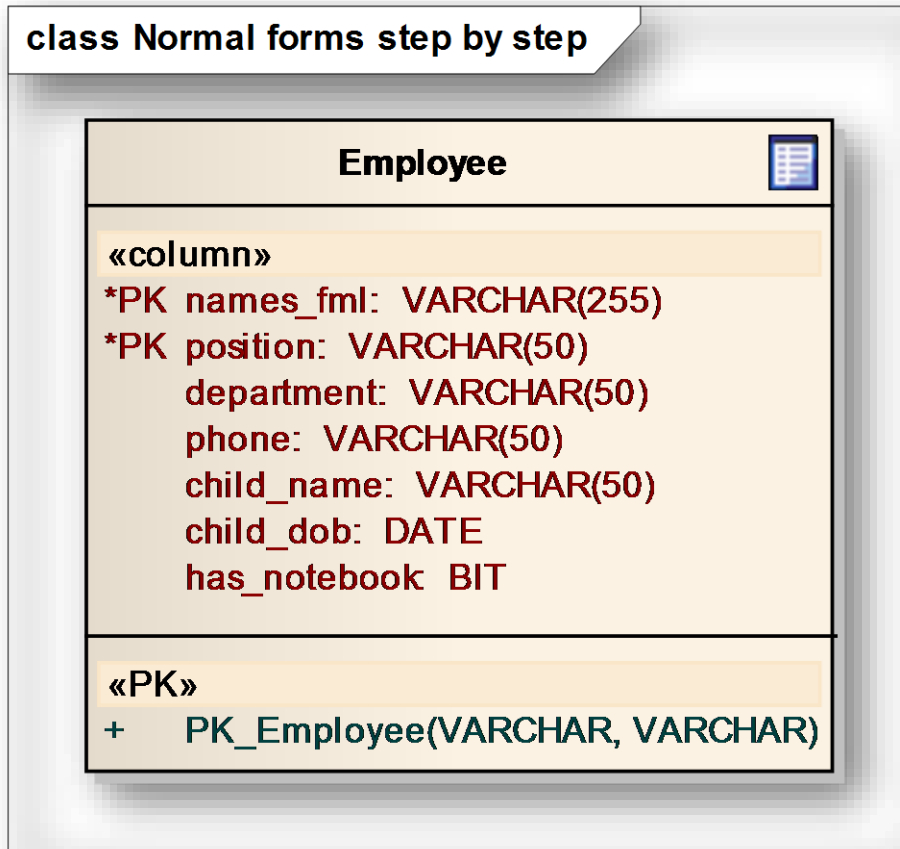
1. В большинстве случаев нормализацию завершают на 3НФ.
2. Через полгода реальной разработки БД человек физически не может придумать модель, противоречащую 3НФ, не сломав себе мозг.



1. Ужасное отношение.

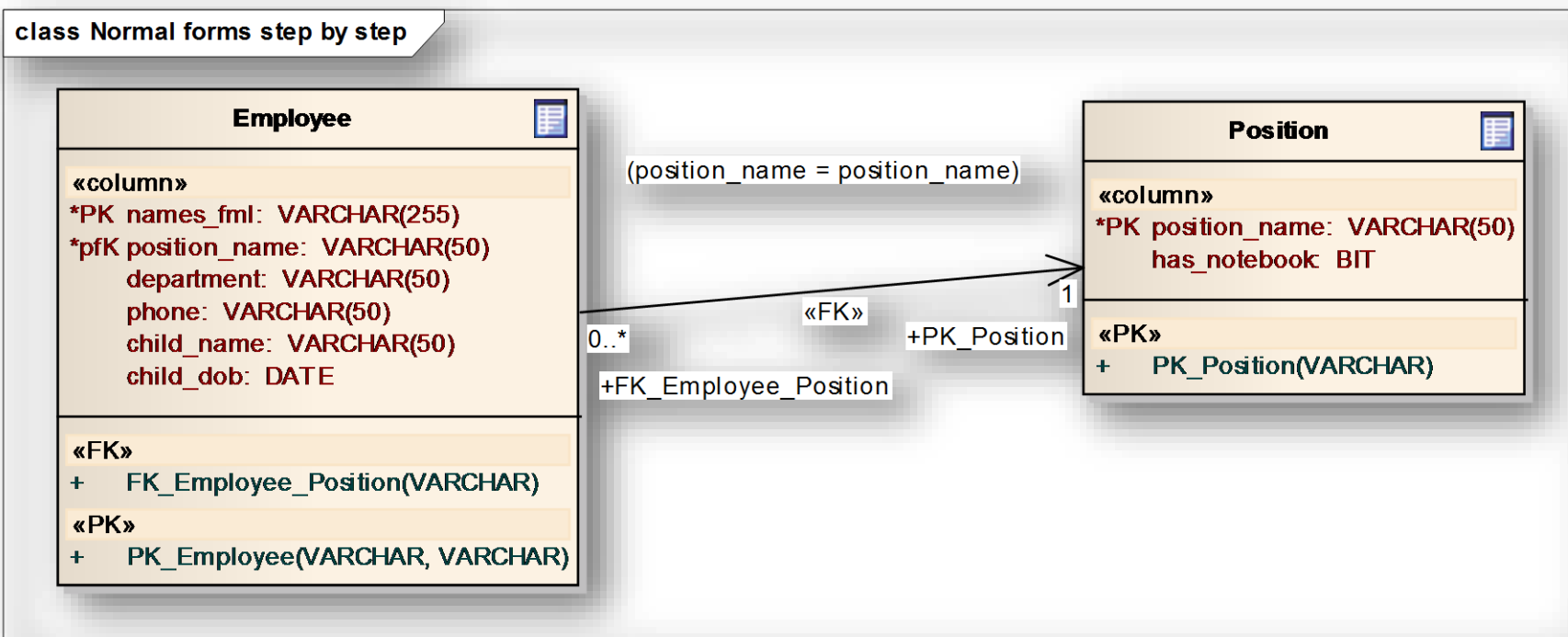


2. Приводим к 1НФ, атомизируем атрибуты:

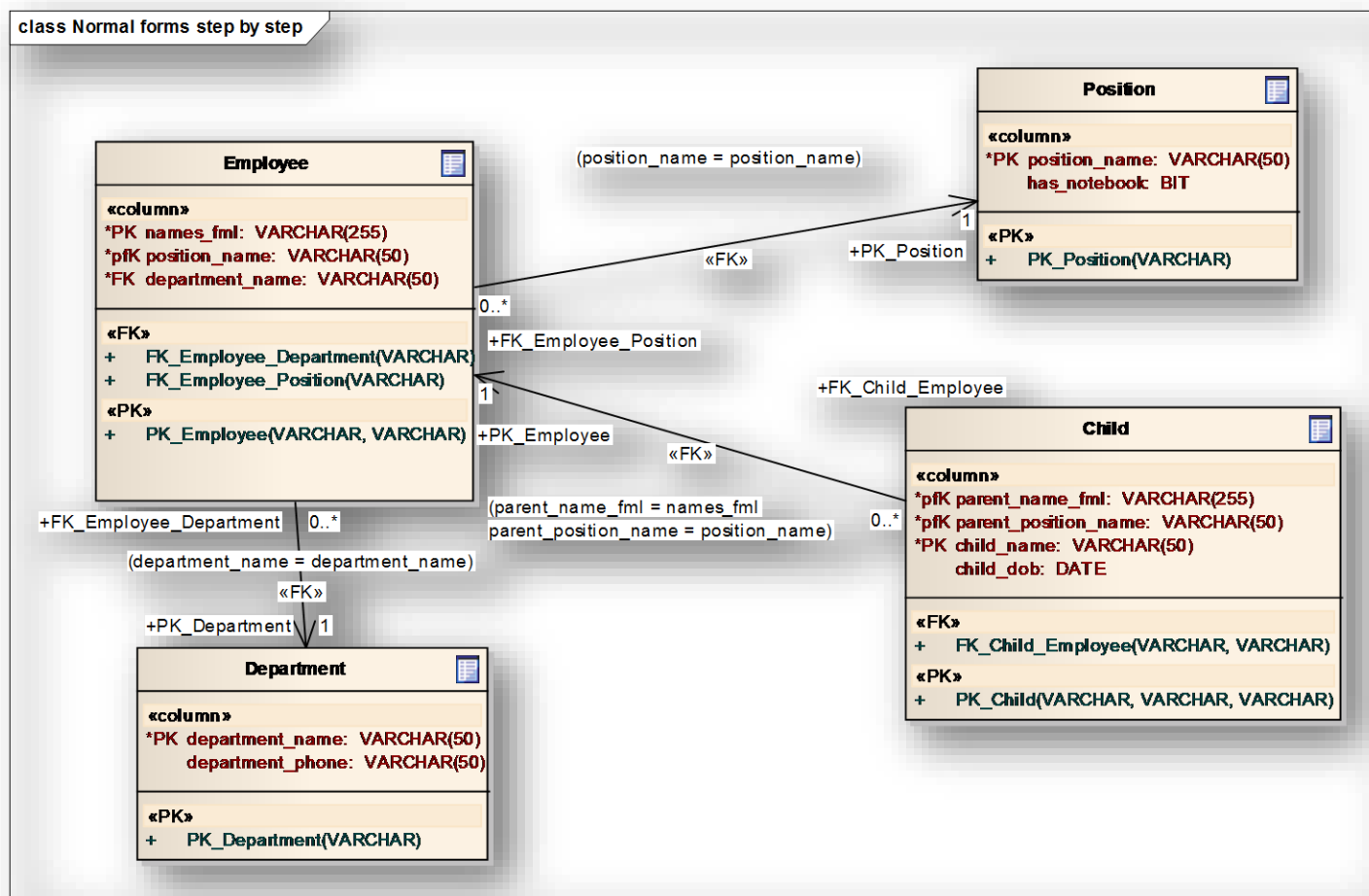


Нормальные формы по шагам

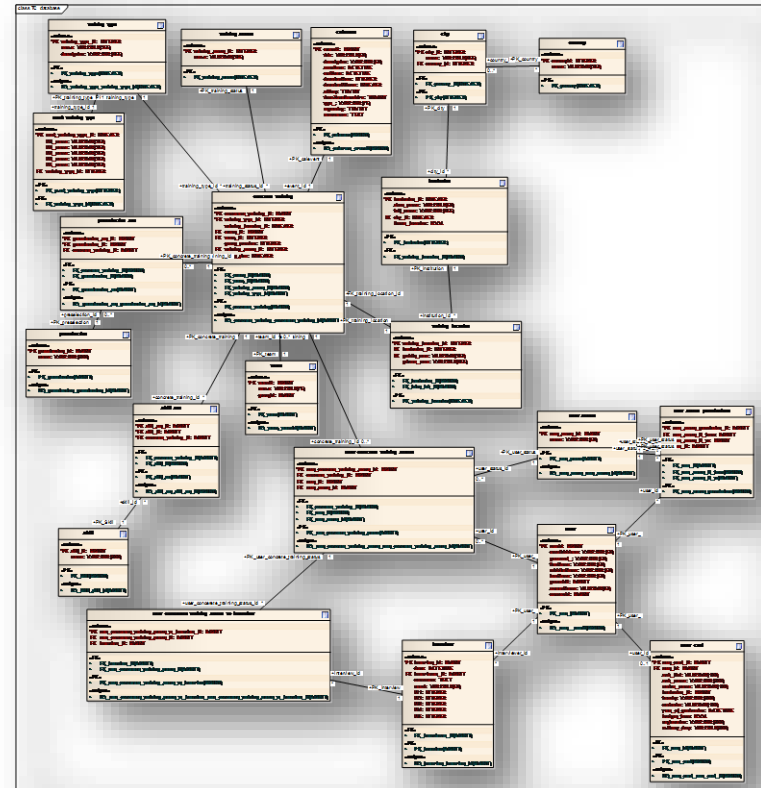
3. У начальника есть ноутбук, у подчинённого – нет. Убираем частичную Ф3. Получаем 2НФ.



4. Убираем транзитивные ФЗ, получаем 3НФ.



5. Добавляем забытые атрибуты, повторяем шаги 1-4 😊:

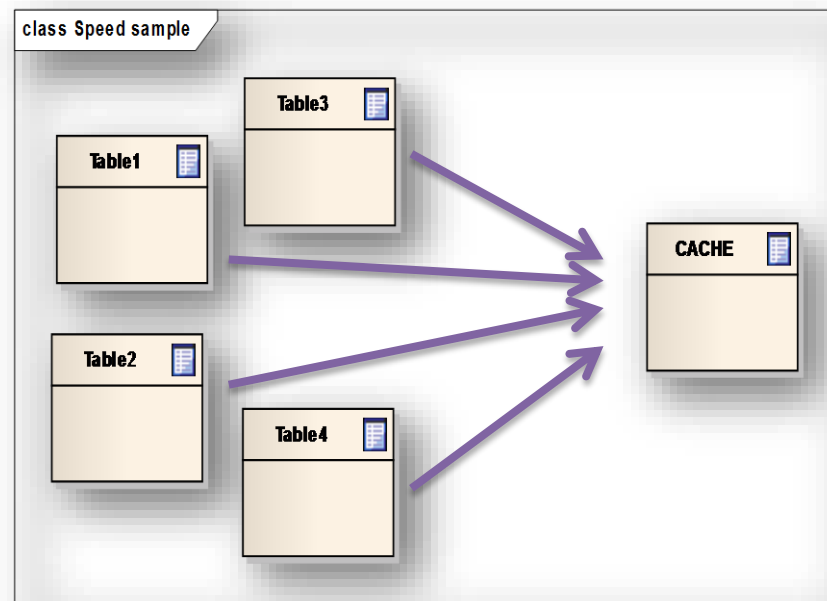


ДЕНОРМАЛИЗАЦИЯ

Здесь всего один слайд, не пугайтесь 😊

Денормализация (denormalization) – процесс приведения отношения к состоянию, нарушающему те или иные нормальные формы.

Денормализация выполняется, в основном, для создания «кэширующих таблиц», некоторые операции с которыми могут выполняться намного быстрее, чем с набором исходных таблиц.



СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Нормальные формы

Author: Svyatoslav Kulikov
Training And Education Manager
svyatoslav_kulikov@epam.com