



часть третья




# БАЗЫ ДАННЫХ



# РЕЛЯЦИОННАЯ МОДЕЛЬ

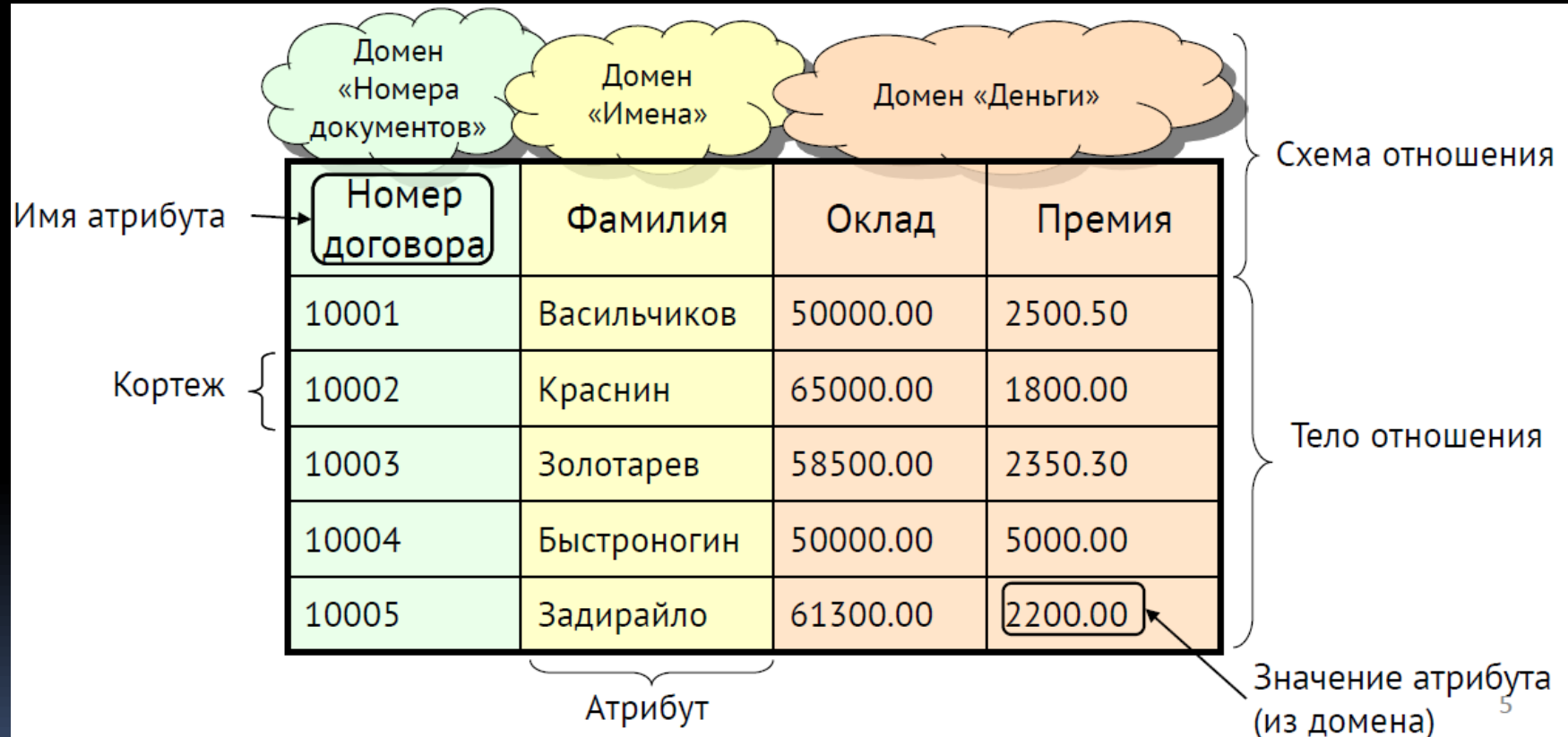
Реляционная модель предусматривает организацию данных исключительно в виде таблиц.



# РЕЛЯЦИОННАЯ МОДЕЛЬ

Таблица представляет собой множество именованных **атрибутов**, или столбцов, и множество **записей** (кортежей), или строк. Очень часто столбец называется **полем** таблицы (*field*). Пересечение строки и столбца образуют **ячейку** таблицы (*cell*). Набор допустимых значений столбца — **домен** (domain) — характеризуется определенным типом данных, например *символьным* или *целым*. Строки же и представляют собой данные.

# РЕЛЯЦИОННАЯ МОДЕЛЬ



# РЕЛЯЦИОННАЯ МОДЕЛЬ

Официальные термины	Альтернативный вариант1	Альтернативный вариант 2	Объектная модель
Отношение	Таблица	Файл	Класс
Кортеж	Строка	Запись	Объект
Атрибут	Столбец	Поле	Свойство

# РЕЛЯЦИОННАЯ МОДЕЛЬ


Реляционная модель предъявляет к таблице определенные **требования**:

- Данные в ячейках таблицы должны быть структурно неделимыми.
- Данные в одном столбце должны быть одного типа.
- Каждый столбец должен быть уникальным (недопустимо дублирование столбцов).
- Столбцы размещаются в произвольном порядке.
- Строки (записи) размещаются в таблице также в произвольном порядке.
- Столбцы имеют уникальные наименования.



# КЛЮЧИ


**Суперключ** – столбец или множество столбцов, которое единственным образом идентифицирует строку данной таблицы





# КЛЮЧИ

**Потенциальный ключ** – суперключ, который не содержит подмножества, также являющегося суперключем данного отношения








# КЛЮЧИ

**Первичный ключ** – потенциальный ключ, который выбран для уникальной идентификации сток внутри таблицы



# КЛЮЧИ

## Потенциальные ключи

- **Альтернативные (АК)** - ограничение типа UNIQUE
- **Первичный (РК, только один)** - ограничение PRIMARY KEY
  - все атрибуты РК – обязательные
  - значения РК не меняются со временем
  - *по возможности – короткий и числовой*
- Суррогатный: *ID*
- Естественные: *Номер договора, ФИО*


# КЛЮЧИ

**Внешний ключ** — это столбец или подмножество столбцов одной таблицы, который может служить в качестве первичного ключа для другой таблицы. Говорят также, что внешний ключ одной таблицы является *ссылкой* на первичный ключ другой таблицы



# КЛЮЧИ

**Пустое значение (NULL)** – указывает, что значение столбца в настоящий момент неизвестно



# Трехзначная логика

Три возможных значения выражений:

**TRUE, FALSE, UNKNOWN**

$(\text{NULL} = \text{NULL}) \rightarrow \text{UNKNOWN}$

$(\text{NULL} \neq \text{NULL}) \rightarrow \text{UNKNOWN}$

$(\text{NOT NULL} \neq \text{NULL}) \rightarrow \text{UNKNOWN}$

$(\text{NOT NULL} = \text{NULL}) \rightarrow \text{UNKNOWN}$

$(\text{FALSE OR NULL}) \rightarrow \text{UNKNOWN}$

$(\text{TRUE OR NULL}) \rightarrow \text{TRUE}$


$(\text{TRUE AND NULL}) \rightarrow \text{UNKNOWN}$

$(\text{FALSE AND NULL}) \rightarrow \text{FALSE}$



# ПРАВИЛА ЦЕЛОСТНОСТИ

**Правило целостности объектов** утверждает, что первичный ключ не может быть полностью или частично пустым, т.е. иметь значение NULL



# ПРАВИЛА ЦЕЛОСТНОСТИ

**Правило ссылочной целостности** гласит, что внешний ключ может быть либо пустым (иметь значение NULL), либо соответствовать значению первичного ключа, на который он ссылается



# ПРАВИЛА КОДДА

## 1. Явное представление данных.







# ПРАВИЛА КОДДА

## 2. Гарантированный доступ к данным.



# ПРАВИЛА КОДДА


**3. Полная обработка неопределенных значений.**





# ПРАВИЛА КОДДА

**4. Доступ к описанию БД в терминах  
реляционной модели.**





# ПРАВИЛА КОДДА

## 5. Полнота подмножества языка.



# ПРАВИЛА КОДДА

## 6. Возможность обновления представлений.



# ПРАВИЛА КОДДА

**7. Наличие высокоуровневого языка  
манипулирования данными.**





# ПРАВИЛА КОДДА

## 8. Физическая независимость данных.



# ПРАВИЛА КОДДА

## 9. Логическая независимость данных.





# ПРАВИЛА КОДДА

## 10. Независимость контроля целостности.



# ПРАВИЛА КОДДА

## 11. Дистрибутивная независимость.





# ПРАВИЛА КОДДА

## 12. **Согласование языковых уровней.**

## ПРАВИЛА КОДДА

- о. Для того чтобы систему можно было квалифицировать как реляционную СУБД, она должна использовать исключительно реляционные функции для управления базой данных.

# НОРМАЛИЗАЦИЯ

В основе **нормализации** лежит  
определенный математический аппарат,  
базирующийся на концепции  
*функциональной зависимости*


# НОРМАЛИЗАЦИЯ

Поле В таблицы функционально зависит от поля А той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля А обязательно существует только одно из различных значений поля В



# НОРМАЛИЗАЦИЯ

Главная цель нормализации — избавить реляционную таблицу от зависимостей, не связанных с первичными ключами



# НОРМАЛИЗАЦИЯ

Основные доводы в пользу **нормализации**:


- **Обеспечение целостности**
- **Создание формальной модели, как можно более независимой от специфики приложения**
- **Снижение требований к объему памяти**





# НОРМАЛИЗАЦИЯ


Существует **пять** различных уровней нормализации, но для реального приложения достаточно третьего уровня





# НОРМАЛИЗАЦИЯ

Проведение нормализации базы данных состоит в устранении избыточности данных и выявлении функциональной зависимости



# НОРМАЛИЗАЦИЯ

**Таблица представлена в первой нормальной форме (1НФ)** тогда и только тогда, когда все ее столбцы содержат только неделимые (в том смысле, что их дальнейшее разложение невозможно) значения и в ней отсутствуют повторяющиеся группы (столбцов) в пределах одной строки.

# НОРМАЛИЗАЦИЯ

**Аномалия вставки** может проявиться в том случае, когда при добавлении в таблицу с первичным ключом какой-либо записи необходимо поместить в поле первичного ключа либо пустое, либо уже существующее значение

# НОРМАЛИЗАЦИЯ

**Аномалия удаления** проявляется в тех случаях, когда возникает необходимость удалить запись и провести реорганизацию таблиц

# НОРМАЛИЗАЦИЯ

*Таблица представлена во второй нормальной форме (2НФ)* тогда и только тогда, когда она представлена в 1НФ и каждый неключевой атрибут полностью определяется первичным ключом

# НОРМАЛИЗАЦИЯ

*Таблица представлена в третьей нормальной форме (3НФ),* если она удовлетворяет определению 2НФ и не одно из неключевых полей не зависит функционально от любого другого неключевого поля

# НОРМАЛИЗАЦИЯ

*Таблица находится в нормальной форме Бойса-Кодда (НФБК), если и только если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от возможного ключа.*



# НОРМАЛИЗАЦИЯ

Определение 3НФ не совсем подходит для следующих отношений:

- 1) отношение имеет две или более потенциальных ключа;
- 2) два и более потенциальных ключа являются составными;
- 3) они пересекаются, т.е. имеют хотя бы один общий атрибут.

Для отношений, имеющих один потенциальный ключ (первичный), НФБК является 3НФ.

# НОРМАЛИЗАЦИЯ

***Многозначная зависимость.*** Поле  $A$  многозначно определяет поле  $B$  той же таблицы, если для каждого значения поля  $A$  существует хорошо определенное множество соответствующих значений  $B$ .

# НОРМАЛИЗАЦИЯ

<i>Дисциплина</i>	<i>Преподаватель</i>	<i>Учебник</i>
Информатика	Шипилов П.А.	Форсайт Р. Паскаль для всех
Информатика	Шипилов П.А.	Уэйт М. и др. Язык Си
Информатика	Голованевский Г.Л.	Форсайт Р. Паскаль для всех
Информатика	Голованевский Г.Л.	Уэйт М. и др. Язык Си

# НОРМАЛИЗАЦИЯ

*Таблица X представлена в 4НФ* тогда и только тогда, когда она представлена в НФБК и для любой многозначной зависимости  $A \rightarrow B$  в этой таблице можно сказать, что A является первичным ключом таблицы X.

# НОРМАЛИЗАЦИЯ

***Зависимость соединения без потерь*** – свойство декомпозиции, которое гарантирует отсутствие фиктивных строк при восстановлении первоначальной таблицы с помощью операций естественного соединения.

# НОРМАЛИЗАЦИЯ

*Таблицы находятся в пятой нормальной форма (5НФ)* тогда и только тогда, когда она представлена в 4НФ и не содержит зависимостей соединения.

## Алгоритм нормализации (приведение к 3НФ)

**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

# Алгоритм нормализации (приведение к 3НФ)

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  $K_1$ .



# Алгоритм нормализации (приведение к 3НФ)

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 от 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  $\{A_1, \dots, A_n\}$ .

# Нормализация на практике

STATE ABBREV	SPOP	CITY	LPOP	CPOP	PCTINC
North Carolina NC	5 млн.	Burlington	40 <u>тыс.</u>	44 <u>тыс.</u>	10%
		Raleigh	200 <u>тыс.</u>	222 <u>тыс.</u>	11%
Vermont VT	4 млн.	Burlington	60 <u>тыс.</u>	67,2 <u>тыс.</u>	12%
New York NY	17 млн.	Albany	500 <u>тыс.</u>	540 <u>тыс.</u>	8%
		New York City	14 млн.	14,7 млн.	5%
		White Plains	100 <u>тыс.</u>	106 <u>тыс.</u>	6%

# Нормализация на практике

STATE	ABBREV	SPOP	CITY	LPOP	CPOP	PCTINC
North Carolina	NC	5 млн.	Burlington	40 <u>тыс.</u>	44 <u>тыс.</u>	10%
North Carolina	NC	5 млн.	Raleigh	200 <u>тыс.</u>	222 <u>тыс.</u>	11%
Vermont	VT	4 млн.	Burlington	60 <u>тыс.</u>	67,2 <u>тыс.</u>	12%
New York	NY	17 млн.	Albany	500 <u>тыс.</u>	540 <u>тыс.</u>	8%
New York	NY	17 млн.	New York City	14 млн.	14,7 млн.	5%
New York	NY	17 млн.	White Plains	100 <u>тыс.</u>	106 <u>тыс.</u>	6%

# Нормализация на практике

STATE	ABBREV	SPOP
North Carolina	NC	5 млн.
Vermont	VT	4 млн.
New York	NY	17 млн.

CITY	ABBREV	LPOP	CPOP	PCTINC
Burlington	NC	40 <u>тыс.</u>	44 <u>тыс.</u>	10%
Raleigh	NC	200 <u>тыс.</u>	222 <u>тыс.</u>	11%
Burlington	VT	60 <u>тыс.</u>	67,2 <u>тыс.</u>	12%
New York City	NY	14 млн.	14,7 млн.	5%
Albany	NY	500 <u>тыс.</u>	540 <u>тыс.</u>	8%
White Plains	NY	100 <u>тыс.</u>	106 <u>тыс.</u>	6%

# Нормализация на практике

ABBREV	SPOP
NC	5 млн.
VT	4 млн.
NY	17 млн.

STATE	ABBREV
North Carolina	NC
Vermont	VT
New York	NY


CITY	ABBREV	LPOP	CPOP
Burlington	NC	40 <u>тыс.</u>	44 <u>тыс.</u>
Raleigh	NC	200 <u>тыс.</u>	222 <u>тыс.</u>
Burlington	VT	60 <u>тыс.</u>	67,2 <u>тыс.</u>
New York City	NY	14 млн.	14,7 млн.
Albany	NY	500 <u>тыс.</u>	540 <u>тыс.</u>
White Plains	NY	100 <u>тыс.</u>	106 <u>тыс.</u>



# OLTP и OLAP-системы

**On-Line Transaction Processing (OLTP)** -  
оперативная обработка транзакций

**On-Line Analitical Processing (OLAP)** -  
оперативная аналитическая обработка  
данных



# Числовые типы в PostgreSQL

Имя	Размер	Описание	Диапазон
smallint	2 байта	целое в небольшом диапазоне	-32768 .. +32767
integer	4 байта	типичный выбор для целых чисел	-2147483648 .. +2147483647
bigint	8 байт	целое в большом диапазоне	-9223372036854775808 .. 9223372036854775807
decimal	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
numeric	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
real	4 байта	вещественное число с переменной точностью	точность в пределах 6 десятичных цифр
double precision	8 байт	вещественное число с переменной точностью	точность в пределах 15 десятичных цифр
smallserial	2 байта	небольшое целое с автоувеличением	1 .. 32767
serial	4 байта	целое с автоувеличением	1 .. 2147483647
bigserial	8 байт	большое целое с автоувеличением	1 .. 9223372036854775807

# Денежные типы в Posgres Pro

Имя	Размер	Описание	Диапазон
money	8 байт	денежная сумма	-92233720368547758.08.. +92233720368547758.07



# Типы даты/времени в Postgres Pro

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
timestamp [ (p) ] [ without time zone ]	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
timestamp [ (p) ] with time zone	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1 микросекунда / 14 цифр
date	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
time [ (p) ] [ without time zone ]	8 байт	время суток (без даты)	00:00:00	24:00:00	1 микросекунда / 14 цифр
time [ (p) ] with time zone	12 байт	только время суток (с часовым поясом)	00:00:00+1459	24:00:00-1459	1 микросекунда / 14 цифр
interval [ поля ] [ (p) ]	16 байт	временной интервал	-178000000 лет	178000000 лет	1 микросекунда / 14 цифр

# Логический тип в Posgres Pro

Имя	Размер	Описание
boolean	1 байт	состояние: истина или ложь

# Геометрические типы в Postgres Pro

Имя	Размер	Описание	Представление
point	16 байт	Точка на плоскости	(x,y)
line	32 байта	Бесконечная прямая	{A,B,C}
lseg	32 байта	Отрезок	((x1,y1),(x2,y2))
box	32 байта	Прямоугольник	((x1,y1),(x2,y2))
path	16+16n байт	Закрытый путь (подобный многоугольнику)	((x1,y1),...)
path	16+16n байт	Открытый путь	[(x1,y1),...]
polygon	40+16n байт	Многоугольник (подобный закрытому пути)	((x1,y1),...)
circle	24 байта	Окружность	<(x,y),r> (центр окружности и радиус)

# Типы, описывающие сетевые адреса в PostgreSQL

Имя	Размер	Описание
cidr	7 или 19 байт	Сети IPv4 и IPv6
inet	7 или 19 байт	Узлы и сети IPv4 и IPv6
macaddr	6 байт	MAC-адреса

# Битовые строки в Posgres Pro

Имя	Размер	Описание
bit( <i>n</i> )	$n/8 + 5$ или 8 байт	Для хранения последовательности 0 и 1 длиной точно <i>n</i> символов
bit varying( <i>n</i> )	$n/8 + 5$ или 8 байт	Для хранения последовательности 0 и 1 длиной не более <i>n</i> символов

# Битовые строки в Posgres Pro

Имя	Описание
character varying( <i>n</i> ), varchar( <i>n</i> )	строка ограниченной переменной длины
character( <i>n</i> ), char( <i>n</i> )	строка фиксированной длины, дополненная пробелами
text	строка неограниченной переменной длины

# Другие типы в Posgres Pro


Имя	Псевдонимы	Описание
bytea		двоичные данные ("массив байт")
json		текстовые данные JSON
jsonb		двоичные данные JSON, разобранные
tsquery		запрос текстового поиска
tsvector		документ для текстового поиска
uuid		универсальный уникальный идентификатор
xml		XML-данные



# Перевод концептуальной модели в реляционную

Каждый класс ставится в соответствие с таблицей.

Класс может быть преобразован в группу таблиц, а группа классов — в одну таблицу








## Перевод концептуальной модели в реляционную

Каждая таблица должна иметь первичный ключ, гарантирующий, что каждая строка таблицы может быть однозначно определена.





# Перевод концептуальной модели в реляционную


В качестве **первичного ключа** рекомендуется  
использовать **суррогатный ключ**





## Перевод концептуальной модели в реляционную

Лучше всего выбирать такой размер полей,  
чтобы можно было записывать значения, от  
10 до 100 раз превышающие используемые в  
настоящий момент

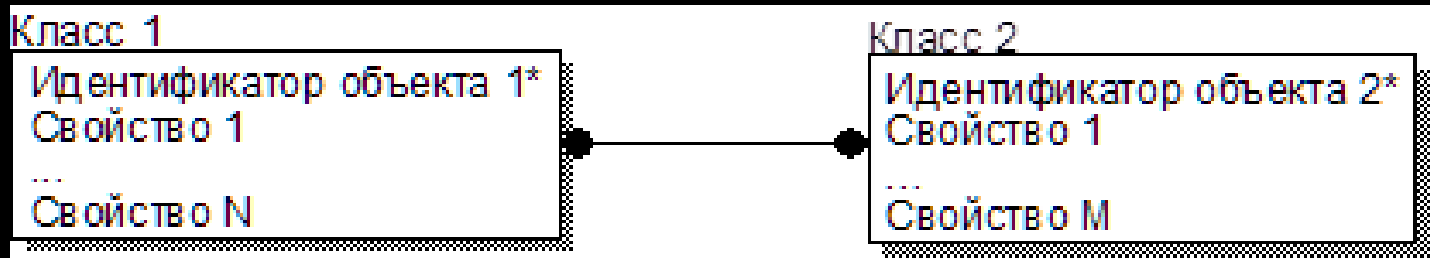


## Перевод концептуальной модели в реляционную

После определения первичных ключей для каждой таблицы они могут быть использованы для связывания таблиц в базу данных.

Связывание осуществляется за счет добавления в таблицы **внешних ключей**.

# ПРЕОБРАЗОВАНИЕ СВЯЗИ МНОГО-КО-МНОГИМ



# ПРЕОБРАЗОВАНИЕ СВЯЗИ МНОГО-КО-МНОГИМ

Имя объекта Класс1		Имя таблицы Table1
Свойство	Поле	Тип
Идентификатор объекта 1	ID_Tab1*	Number
Свойство 1	Pole1	Varchar2(20)
...	...	...
Свойство N	PoleN	Date

1

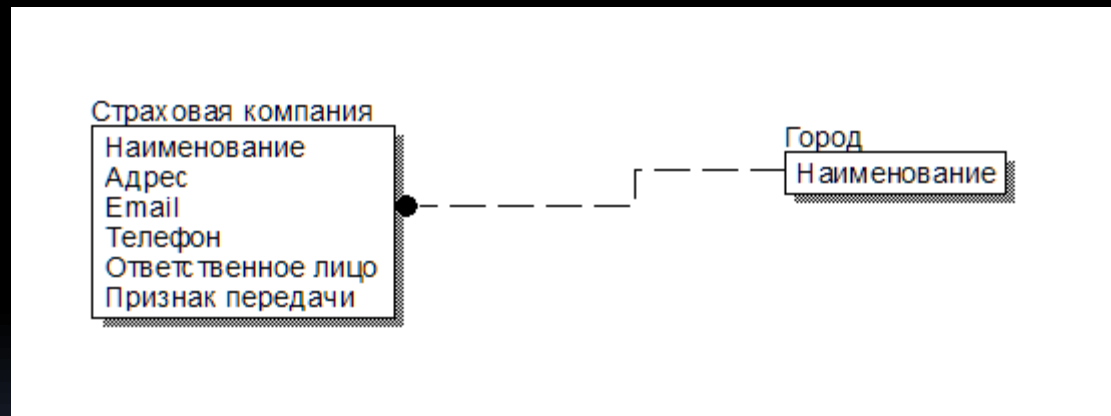
Имя объекта Класс2		Имя таблицы Table2
Свойство	Поле	Тип
Идентификатор объекта 2	ID_Tab2*	Number
Свойство 1	Pole1	Number
...	...	...
Свойство M	PoleM	Clob

1

		Имя таблицы Table1_2	
	Свойство	Поле	Тип
*		ID_Tab1*	Number
*		ID_Tab2*	Number

□

# ПРИМЕР ПЕРЕВОДА КОНЦЕПТУАЛЬНОЙ МОДЕЛИ В РЕЛЯЦИОННУЮ



# ПРИМЕР ПЕРЕВОДА КОНЦЕПТУАЛЬНОЙ МОДЕЛИ В РЕЛЯЦИОННУЮ



Имя объекта Страховая компания		Имя таблицы <u>Inscomp</u>
Свойство	Поле	Тип
Наименование	Name	Varchar2(100)
Адрес	<u>Adres</u>	Varchar2(150)
Email	Email	Varchar2(60)
Телефон	Phones	Varchar2(30)
Ответственное лицо	FIO	Varchar2(100)
Признак передачи	<u>Priz</u>	Number(1)
	<u>ID_Ins*</u>	Number
	<u>ID_City</u>	Number

Имя объекта Город		Имя таблицы City
Свойство	Поле	Тип
Наименование	Name	Varchar2(30)
	<u>ID_City*</u>	Number

1

\*





**Вопросы?**