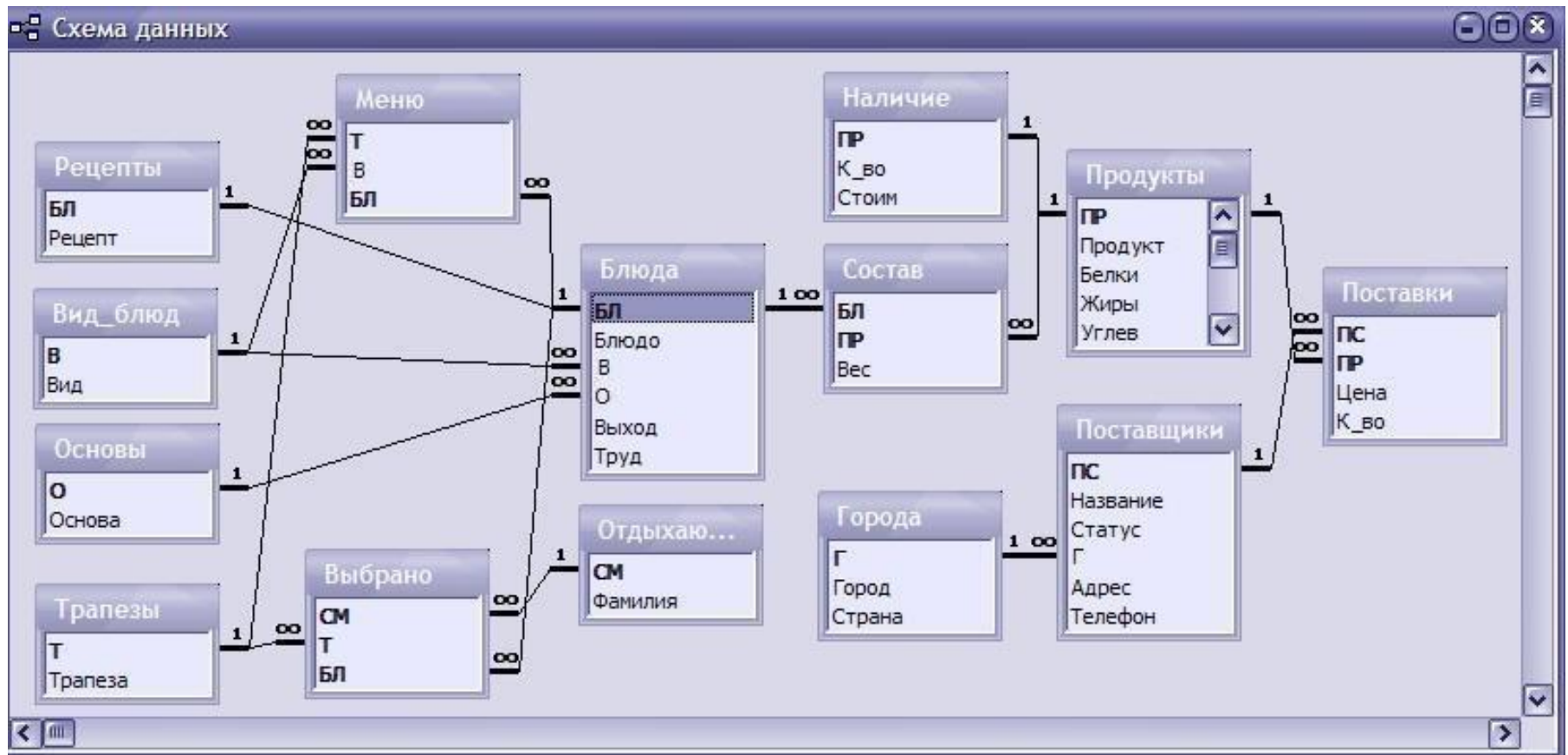


# Тема 2. Целостность баз данных

## Схема базы данных ПАНСИОН



## Целостность баз данных. Первичные ключи

Каждая сущность обладает хотя бы одним *возможным* ключом. Один из них принимается за *первичный* ключ.

### Рекомендации:

При выборе первичного ключа следует отдавать предпочтение **несоставным** **ключам** или ключам, составленным **из минимального числа атрибутов**.

## Целостность баз данных. Первичные ключи

Нецелесообразно также использовать ключи с длинными текстовыми значениями  
**(предпочтительнее использовать целочисленные атрибуты).**

## Правило целостности сущностей:

1. Не допускается, чтобы первичный ключ сущности (любой атрибут, участвующий в первичном ключе) принимал неопределенное значение. Иначе возникнет противоречивая ситуация: появится **не существующий** экземпляр стержневой сущности.

2. По тем же причинам необходимо обеспечить уникальность *первичного ключа*.

**ПРИМЕР.** Требуется хранить информацию о **наименовании поставщиков, наименовании и количестве поставляемых ими деталей,** причем **каждый поставщик может поставлять несколько деталей и каждая деталь может поставляться несколькими поставщиками.** Можно предложить хранить данные в следующем отношении:

# Целостность баз данных. Внешние ключи

Пример с поставщиками и поставками деталей.

<u>Номер поставщик</u> <u>a</u>	Поставщик	<u>Номер детали</u>	Деталь	Поставляемое количество
1	Иванов	1	Болт	100
1	Иванов	2	Гайка	200
1	Иванов	3	Винт	300
2	Петров	1	Болт	150
2	Петров	2	Гайка	250
3	Сидоров	3	Винт	1000

Приведенный способ хранения данных обладает рядом недостатков.  
Например:

1. Что произойдет, если **изменилось наименование поставщика**?
2. Как отразить факт, что **некоторый поставщик**, например Петров, временно прекратил поставки деталей?

# Целостность баз данных. Внешние ключи

Разделим данные на три таблицы - "Поставщики", "Детали", "Поставки".

Номер поставщик а	Поставщик
1	Иванов
2	Петров
3	Сидоров

Номер детали	Деталь
1	Болт
2	Гайка
3	Винт

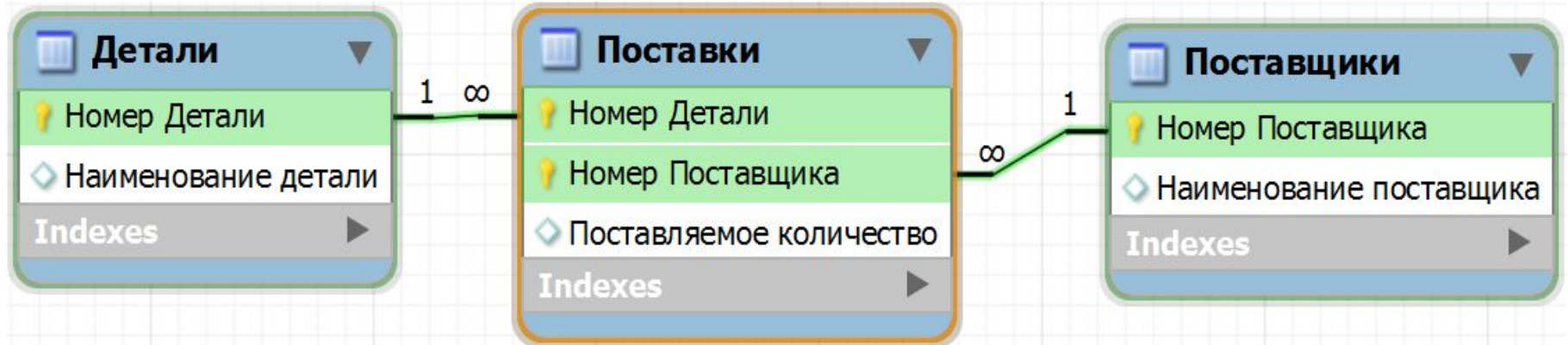
Номер поставщика	Номер детали	Поставляемое количество
1	1	100
1	2	200
1	3	300
2	1	150
2	2	250
3	3	1000

Данные таблицы свободны от недостатков, описанных выше, когда все данные предлагалось хранить в одной таблице.

## Целостность баз данных. Внешние ключи

В реляционных базах данных основными являются взаимосвязи типа "один-ко-многим". Взаимосвязи типа "многие-ко-многим" реализуются использованием нескольких взаимосвязей типа "один-ко-многим".

Сущность, входящая в связь со стороны **один**, называют **родительской сущностью**. Сущность, входящую в связь со стороны **многие**, называется **дочерней сущностью**.





## Требования к внешним ключам:

1. Если сущность С связывает сущности А и В (ассоциация), то она должна включать внешние ключи, соответствующие первичным ключам сущностей А и В.
2. Если сущность В обозначает или характеризует сущность А, то она должна включать внешний ключ, соответствующий первичному ключу сущности А.

## Целостность баз данных. Внешние ключи

**Замечание 1.** Внешний ключ может быть простым или составным.

**Замечание 2.** Хотя каждое значение внешнего ключа обязано совпадать со значениями первичного ключа в некоторой строке родительской сущности, обратное, в общем случае неверно. Например, могут существовать поставщики, не поставляющие никаких деталей.

**Замечание 3.** Внешний ключ, как правило, не обладает свойством уникальности. В дочерней сущности может быть несколько строк, ссылающихся на одну и ту же строку родительской сущности.

**Замечание 4.** Если внешний ключ все-таки обладает свойством уникальности, то связь между сущностями имеет тип "один-к-одному". Чаще всего такие сущности можно объединить в одну сущность, хотя это и не обязательно.

## Целостность баз данных. NULL-значения

В реальном мире **часто встречается ситуация, когда данные неизвестны или не полны.**

Для того чтобы обойти проблему неполных или неизвестных данных, в базах данных могут использоваться типы данных, пополненные так называемым **Null-значением**. Это некий маркер, показывающий, что **значение неопределено или неизвестно.**

## Целостность баз данных. NULL-значения

Наиболее бросающейся в глаза проблемой является **необходимость использования трехзначной логики при оперировании с данными, которые могут содержать null-значения.** В этом случае при неаккуратном формулировании запросов, даже самые естественные запросы могут давать неправильные ответы.

Практически все реализации современных реляционных СУБД позволяют использовать null-значения, несмотря на их недостаточную теоретическую обоснованность.

# Целостность баз данных. NULL-значения

## Трехзначная логика (3VL)

Определение истинности логических выражений базируется на трехзначной логике (three-valued logic, 3VL), в которой кроме значений **T** - ИСТИНА и **F** - ЛОЖЬ, введено значение **U** - НЕИЗВЕСТНО.

Трехзначная логика базируется на следующих таблицах истинности:

# Целостность баз данных. NULL-значения

Таблица истинности **AND**

<b>AND</b>	<b>F</b>	<b>T</b>	<b>U</b>
<b>F</b>			
<b>T</b>			
<b>U</b>			

# Целостность баз данных. NULL-значения

Таблица истинности **AND**

<b>AND</b>	<b>F</b>	<b>T</b>	<b>U</b>
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>U</b>
<b>U</b>	<b>F</b>	<b>U</b>	<b>U</b>

# Целостность баз данных. NULL-значения

Таблица истинности **OR**

<b>OR</b>	<b>F</b>	<b>T</b>	<b>U</b>
<b>F</b>			
<b>T</b>			
<b>U</b>			



# Целостность баз данных. NULL-значения

Таблица истинности **OR**

<b>OR</b>	<b>F</b>	<b>T</b>	<b>U</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>U</b>
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>U</b>	<b>U</b>	<b>T</b>	<b>U</b>

# Целостность баз данных. NULL-значения

## Таблица истинности **NOT**

<b>OR</b>	
<b>F</b>	
<b>T</b>	
<b>U</b>	

# Целостность баз данных. NULL-значения

Таблица истинности **NOT**

<b>OR</b>	
<b>F</b>	<b>T</b>
<b>T</b>	<b>F</b>
<b>U</b>	<b>U</b>

# Целостность баз данных. NULL-значения

Имеется несколько парадоксальных следствий применения трехзначной логики.

**Парадокс 1.** Null-значение не равно самому себе. Действительно, выражение **null = null** дает значение не ИСТИНА, а НЕИЗВЕСТНО. Значит в общем случае проверка условия **x=x** не обязательно ИСТИНА!

**Парадокс 2.** Неверно также, что null-значение не равно самому себе! Действительно, выражение **null ≠ null** также принимает значение не ИСТИНА, а НЕИЗВЕСТНО! Значит, и выражение **x ≠ x** тоже не обязательно ЛОЖЬ!

## Операции, нарушающие целостность внешних ключей (ссылочную целостность)

Целостность данных может нарушиться в результате операций, изменяющих состояние базы данных.

Таких операций три - *вставка, обновление и удаление* экземпляров сущностей.

В определении целостности участвуют две сущности - родительская и дочерняя, а в каждой из них возможны три операции - вставка, обновление, удаление. Поэтому нужно рассмотреть шесть различных вариантов:

## Операции, нарушающие целостность внешних ключей (ссылочную целостность)

	Для родительской сущности	Для дочерней сущности
Вставка		
Обновление		
Удаление		

## Операции, нарушающие целостность внешних ключей (ссылочную целостность)

	Для родительской сущности	Для дочерней сущности
Вставка	Не нарушает	<b>Может нарушить</b>
Обновление	<b>Может нарушить</b>	<b>Может нарушить</b>
Удаление	<b>Может нарушить</b>	Не нарушает

## Операции, нарушающие целостность внешних ключей (ссылочную целостность)

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

Обновление в родительской сущности.

Удаление в родительской сущности.

Вставка в дочернюю сущность.

Обновление в дочерней сущности.

С целью обеспечения ссылочной целостности **внешних** ключей в современных СУБД обеспечиваются следующие *стратегии поддержания ссылочной целостности:*



## Стратегии поддержания ссылочной целостности:

**RESTRICT (ОГРАНИЧИТЬ)** - не разрешать выполнение операции, приводящей к нарушению ссылочной целостности.

**CASCADE (КАСКАДИРОВАТЬ)** - разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других сущностях так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительской сущности и каскадно выполняется в дочерней сущности.

**IGNORE (ИГНОРИРОВАТЬ)** - выполнять операции, не обращая внимания на нарушения ссылочной целостности.

# Стратегии поддержания ссылочной целостности:

**Примечание 1.** Не все из перечисленных стратегий присутствуют в конкретных СУБД.

**Примечание 2.** В некоторых СУБД в дополнение к приведенным стратегиям пользователь может придумать свою уникальную стратегию поддержания ссылочной целостности.



```
CREATE TABLE блюда (  
    БЛ INTEGER UNSIGNED NOT NULL,  
    Блюдо VARCHAR(45) NOT NULL,  
    В CHAR NOT NULL,  
    О INTEGER UNSIGNED NOT NULL,  
    Выход VARCHAR(45),  
    Труд VARCHAR(45) NOT NULL,  
    PRIMARY KEY(БЛ),  
    CONSTRAINT FK_блюда_1 FOREIGN KEY  
        FK_блюда_1 (В)  
        REFERENCES вид_блюд (В)  
        ON DELETE RESTRICT  
        ON UPDATE CASCADE,  
    CONSTRAINT FK_блюда_2 FOREIGN KEY  
        FK_блюда_2 (О)  
        REFERENCES основы (О)  
        ON DELETE RESTRICT  
        ON UPDATE CASCADE  
);  
TYPE = InnoDB;
```

Реализация  
стратегий  
поддержания  
ссылочной  
целостности в  
MySQL

```
CREATE TABLE блюда (  
    БЛ INTEGER UNSIGNED NOT NULL,  
    Блюдо VARCHAR(45) NOT NULL,  
    В CHAR NOT NULL,  
    О INTEGER UNSIGNED NOT NULL,  
    Выход VARCHAR(45),  
    Труд VARCHAR(45) NOT NULL,  
    PRIMARY KEY(БЛ),  
    FOREIGN KEY (В) REFERENCES вид_блюд (В)  
        ON DELETE RESTRICT  
        ON UPDATE CASCADE,  
    FOREIGN KEY (О) REFERENCES основы (О)  
        ON DELETE RESTRICT  
        ON UPDATE CASCADE  
)  
TYPE = InnoDB;
```

Реализация  
стратегий  
поддержания  
ссылочной  
целостности в  
MySQL

## Правила целостности внешних ключей

После выбора внешних ключей для каждого внешнего ключа необходимо решить следующие вопросы:

А. Может ли данный внешний ключ принимать неопределенные значения (NULL-значения)?

Б. Выбор стратегий поддержания ссылочной целостности для каждого внешнего ключа.

# Правила целостности внешних ключей

Ограничения на внешний ключ для характеристик :

**NULL-значения не допустимы**

**УДАЛЕНИЕ ИЗ (цель) КАСКАДИРУЕТСЯ**

**ОБНОВЛЕНИЕ (первичный ключ цели)**

**КАСКАДИРУЕТСЯ**