

ОТНОШЕНИЯ. КЛЮЧИ. СВЯЗИ.

Базовые понятия реляционных БД: отношения, ключи и их виды, связи и их виды. Индексы. Целостность и консистентность данных, триггеры.

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

Содержание

1. Отношения.
2. Ключи.
3. Индексы.
4. Связи.
5. Ссылочная целостность данных.
6. Триггеры.

ОТНОШЕНИЯ

Определение (сначала научно)

Отношение, сущность (relation, entity) R степени n – подмножество декартового произведения множеств D_1, D_2, \dots, D_n ($n \geq 1$). Исходные множества D_1, D_2, \dots, D_n называются **доменами** (в СУБД – «тип данных»).

	D_1	D_2	D_{\dots}		
	supplier	address	product	quantity	price
Наб. зн. 1	"Рога и копыта" Ltd	Далёкое-далёко, 999	Веники	200	1500
Наб. зн. 2	"Рога и копыта" Ltd	Далёкое-далёко, 999	Ложки	100	500
Наб. зн. ...	"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

Определение (теперь «на пальцах»)

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

Отношение

Схема отношения
(в UML-нотации)

class Delivery

Delivery

«column»

supplier: VARCHAR(255)
address: VARCHAR(255)
product: VARCHAR(255)
quantity: BIGINT
price: DOUBLE

Определение (теперь «на пальцах»)

Заголовок (схема)
отношения

Значение
отношения

Атрибуты отношения
(«столбцы таблицы»)

Мощность
отношения (число
«строк таблицы»)

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

Тело отношения (кортежи НЕ
упорядочены, порядок задаётся
операторами сортировки)

Кортеж отношения
(«строка таблицы»,
«запись таблицы»)

«В жизни всё не так, как на самом деле»

Теория реляционных БД
может считаться

В реальности в таблицу БД
можно поместить 2+
одинаковые строки. Этого не
делают, но всё же...

- В таблице **нет** двух одинаковых строк.
- У таблицы **есть столбцы, соответствующие атрибутам** отношения.
- Каждый **атрибут-столбец** имеет уникальное **имя**.
- Порядок строк в таблице **произвольный**.

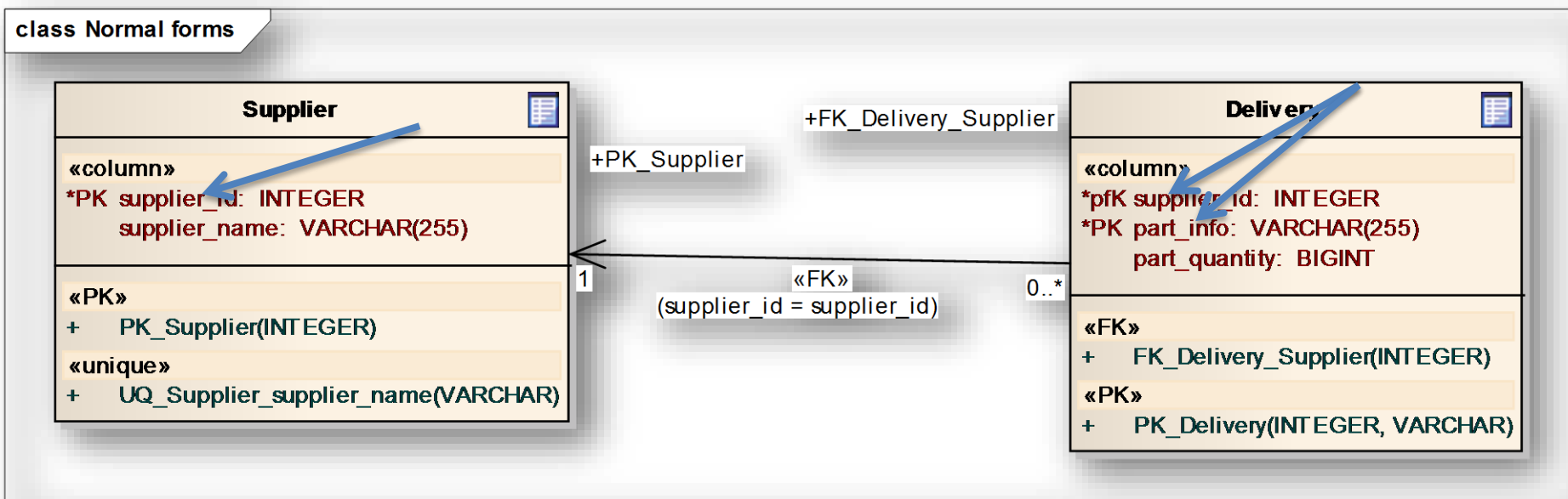
ОК

Например, JOIN'ы в MySQL
запросто возвращают таблицы с
2+ одинаковыми именами.

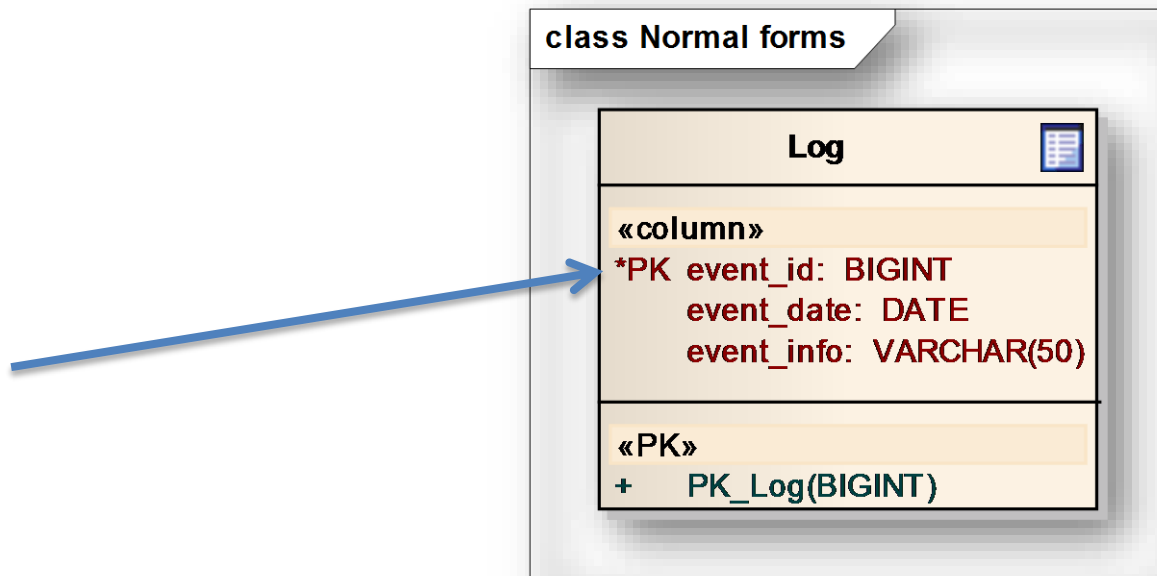
ОК

КЛЮЧИ

Ключ (key) – атрибут (или совокупность атрибутов) отношения, обладающий некоторыми специфическими свойствами, зависящими от вида ключа.



Первичный ключ (primary key, PK) – минимальное множество атрибутов, являющееся подмножеством заголовка данного отношения, составное значение которых **уникально определяет кортеж** отношения.



Первичный ключ – ВАЖНО!

Значение первичного ключа **не может** повторяться в двух и более строках.

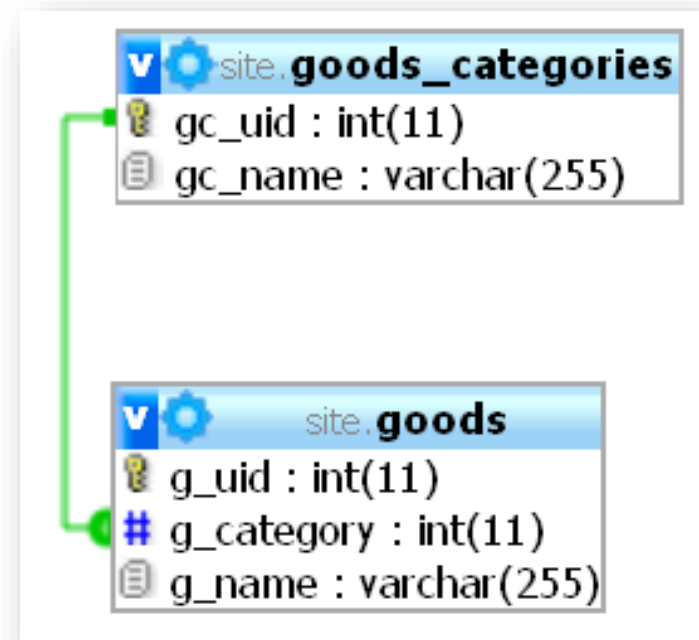
sp_uid Идентификатор страницы	sp_parent Ссылка на родительскую страницу	sp_name Имя страницы
1	NULL	Главная
2	1	О фирме
3	1	Услуги
4	3	Юрлицам
5	3	Физлицам
6	5	Test

Первичный ключ – ВАЖНО!

Первичный ключ служит для:

- однозначной идентификации строки таблицы;
- организации связей между таблицами.

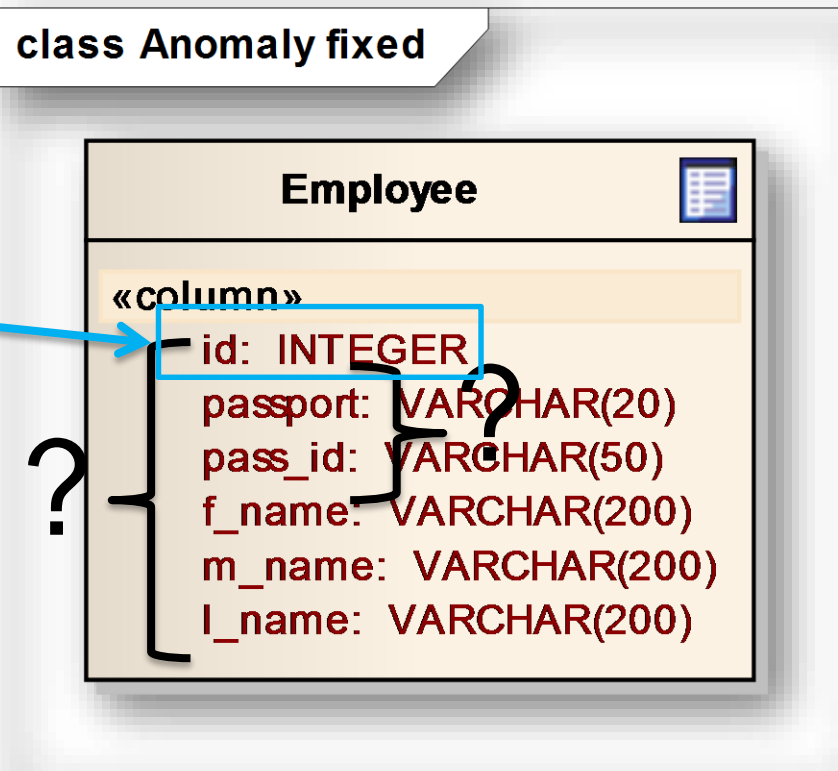
sp_uid Идентификатор страницы	sp_parent Ссылка на родительскую страницу	sp_name Имя страницы
1	NULL	Главная
2	1	О фирме
3	1	Услуги
4	3	Юрлицам
5	3	Физлицам
6	5	Test



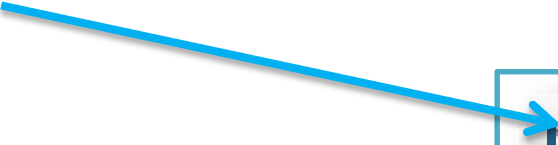
Первичный ключ – ВАЖНО!

Из всех вариантов нужно выбирать в качестве первичного ключа **самое «короткое» поле** или **самую «короткую» комбинацию полей.**

!!!



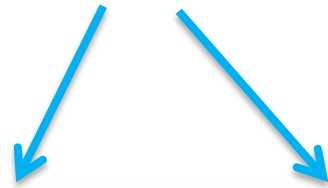
Простой первичный ключ (simple primary key) – первичный ключ, состоящий из **единственного поля** таблицы (атрибута отношения).



passport	fml_name
MA334455	Сидоров С.С.
MM991122	Сидоров С.С.
MM998877	Петров П.П.
MP112233	Иванов И.И.
MP334455	Иванов И.И.

Составной первичный ключ

Составной первичный ключ (compound PK, composite PK, concatenated PK) – первичный ключ, состоящий из **нескольких** полей таблицы.




passport	country	fml_name
MA998877	BY	Сидоров С.С.
MM112233	BY	Иванов И.И.
MM112233	RU	Петров П.П.

Составной первичный ключ – последовательность полей

Последовательность полей в составных РК имеет значение: СУБД может проводить поиск очень быстро целиком по всему составному РК или по **первому полю, входящему в его состав.**

Если поиск производится чаще по:

- номеру паспорта => **{passport, country}**
- стране => **{country, passport}**



passport	country	fml_name
MA998877	BY	Сидоров С.С.
MM112233	BY	Иванов И.И.
MM112233	RU	Петров П.П.

Естественный первичный ключ

Естественный первичный ключ (natural PK) – поле таблицы, хранящее полезные данные.



passport	fml_name
MA334455	Сидоров С.С.
MM991122	Сидоров С.С.
MM998877	Петров П.П.
MP112233	Иванов И.И.
MP334455	Иванов И.И.

Естественный первичный ключ

У естественных первичных ключей есть **недостатки**:

- Большой размер.
- Необходимость каскадных изменений.
- Невозможность вставки данных при отсутствии части информации.

passport

MA998877

8 байт

keys payments
passport : varchar(20)
money : double

keys employee
passport : varchar(15)
fml_name : varchar(255)

passport	fml_name	passport	money
MA334455	Сидоров С.С.	MA334455	999
MM991122	Сидоров С.С.	MA334455	5754
		MA334455	6433634
		MM991122	454534


?

passport fml_name

Какой-то мужик с бодуна, готов работать двоником, паспорта нет.

Синтетический (суррогатный) первичный ключ

Синтетический (суррогатный) первичный ключ (synthetic, surrogate PK) – искусственно добавленное в таблицу поле, единственная задача которого – быть первичным ключом.



id	passport	fml_name
1	MP334455	Иванов И.И.
2	MP112233	Иванов И.И.
3	MM998877	Петров П.П.
4	MM991122	Сидоров С.С.
5	MA334455	Сидоров С.С.
6		Какой-то мужик с бодуна,

Синтетический (суррогатный) первичный ключ

У синтетических ключей есть преимущества:

- Малый размер.
- Нет необходимости каскадных изменений.
- Можно вставить данные при отсутствии части информации.

id

1

4 байта

keys.payments
id : int(11)
money : double

keys.employee
id : int(11)
passport : varchar(15)
fml_name : varchar(255)

id	passport	fml_name	id	money
1	MP834455	Иванов И.И.	1	999
2	MP112233	Иванов И.И.	1	5754
			1	6433634
			2	454534

id	passport	fml_name
6		Какой-то мужик с бодуна, готов работать двоником,

Возможные и альтернативные ключи

Возможный ключ (possible key) – поле или совокупность полей с уникальными значениями.

Альтернативный ключ (alternate key) – поле или совокупность полей с уникальными значениями, но не в качестве первичного ключа.

На альтернативных ключах, как правило, создают уникальные индексы. Но об этом чуть позже.

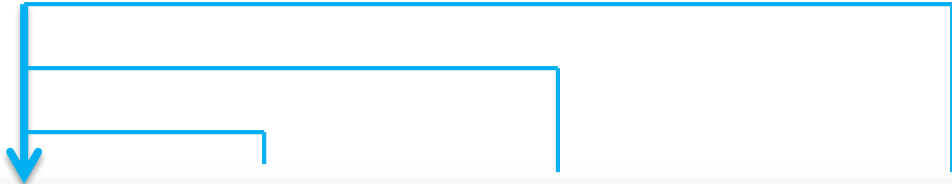
Возможные ключи

Если это – РК, ...

license_plate	vin	info
1122 EH-7	LJCPCBLCX11000237	Машинко
2233 AA-9	ZZCPCBLCX11999111	Тоже машинко

... то это – альтернативный ключ.

Интеллектуальный первичный ключ (intelligent primary key) – поле, значение которого формируется на основе значений других полей.



The diagram illustrates the formation of the 'ipk' (intelligent primary key) field. A blue line starts from the top, branches into three horizontal segments, and then drops down with arrows pointing to the 'f_name', 'm_name', and 'l_name' columns of the table below. This indicates that the 'ipk' value is derived from the concatenation of these three fields.

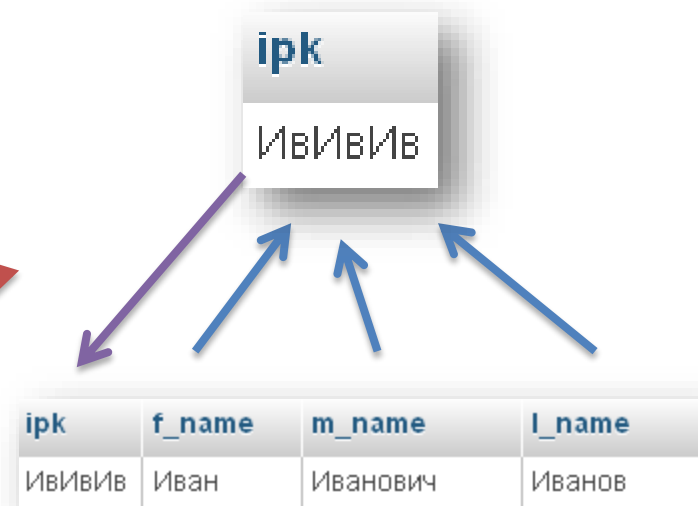
ipk	f_name	m_name	l_name
ИвИвИв	Иван	Иванович	Иванов
СиСиСи	Сидор	Сидорович	Сидоров
ПёПеПе	Пётр	Петрович	Петров
АлАлАл	Александр	Александрович	Александров

Интеллектуальный первичный ключ

С интеллектуальными ключами есть **проблемы**:

- Их нужно **генерировать** и обновлять.
- Высока **вероятность коллизий** (совпадений значений).

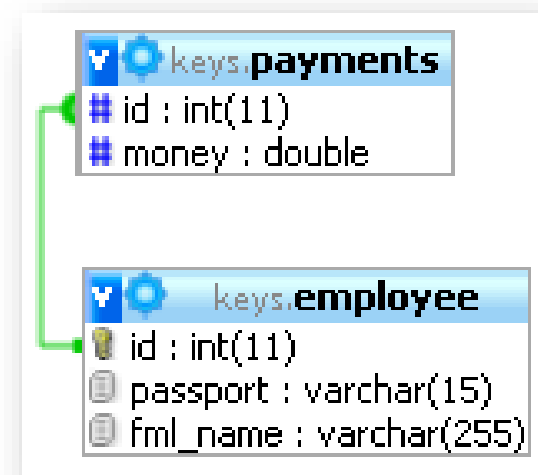
Поэтому они **почти не используются**.



АлАлАл	Александр	Александрович	Александров
АлАлАл	Алексей	Алексеевич	Алексеев

Внешний ключ

Внешний ключ (foreign key) – поле таблицы, предназначенное для хранения значения первичного ключа другой таблицы с целью организации связи между этими таблицами.



id	passport	fml_name
1	MP334455	Иванов И.И.
2	MP112233	Иванов И.И.

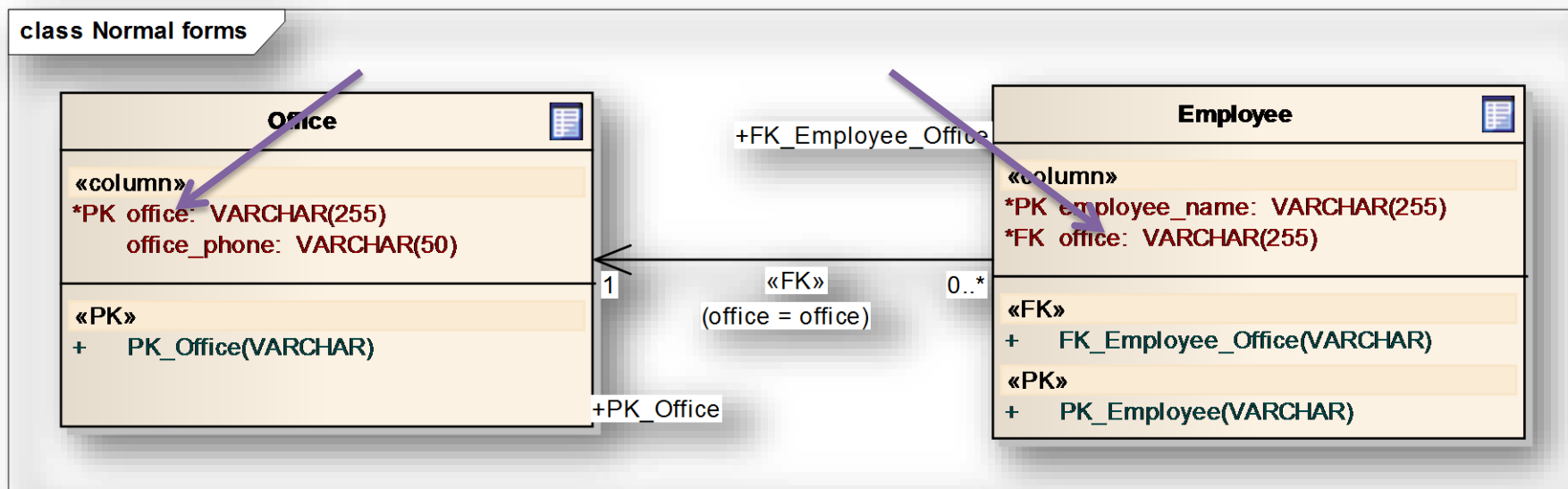
id	money
1	999
1	5754
1	6433634
2	454534

Diagram illustrating data consistency with foreign key constraints. Purple lines connect the **id** values in the **payments** table to the **id** values in the **employee** table, showing that all payments are associated with existing employees.

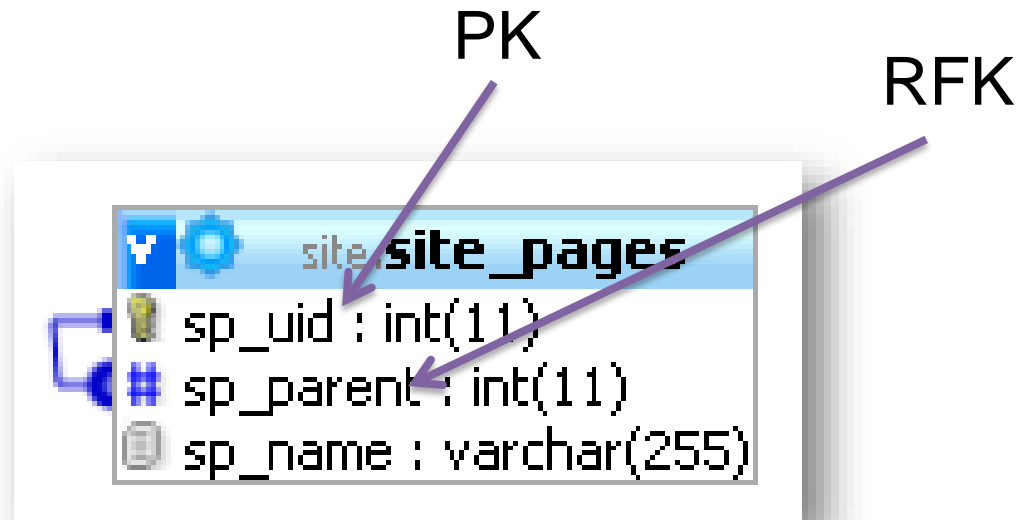
Внешний ключ – как происходит миграция

Миграция первичного ключа из родительской таблицы в дочернюю либо производится:

- **автоматически** (ErWin);
- **вручную** (Sparx EA) – создать в дочерней таблице поле такого же типа (**имя можно брать другое!!!**), что и мигрирующий РК (включая регистрочувствительность, «знаковость» и т.п.), и указать его в качестве внешнего ключа для проведённой между таблицами связи.

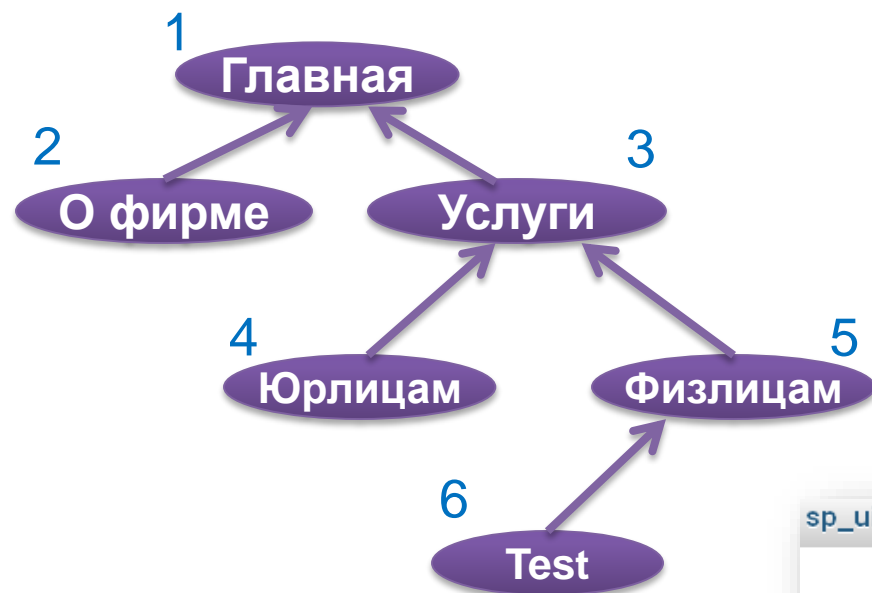


Рекурсивный внешний ключ (recursive foreign key) – внешний ключ, полученный из поля этой же таблицы.



Рекурсивный внешний ключ

Классика применения RFK – хранение древовидных структур (например, структуры сайта).



PK

RFK

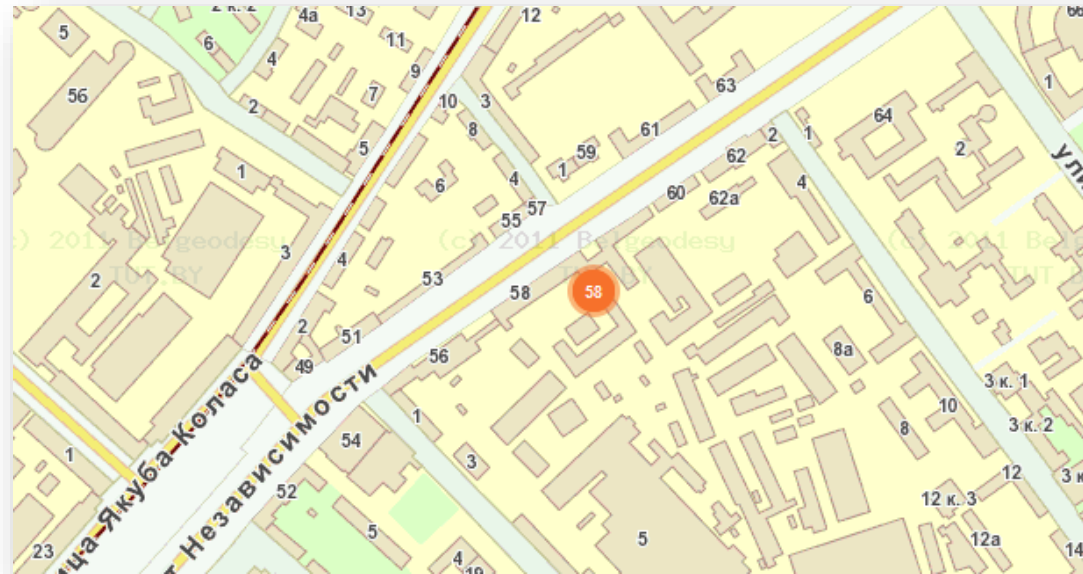
sp_uid	Идентификатор страницы	sp_parent	Ссылка на родительскую страницу	sp_name	Имя страницы
1		NULL		Главная	
2		1		О фирме	
3		1		Услуги	
4		3		Юрлицам	
5		3		Физлицам	
6		5		Test	

ИНДЕКСЫ

Понятие индекса

Индекс (index) – объект БД, создаваемый с целью повышения производительности поиска данных.

Простая аналогия «из жизни» – карта города. Можно **посмотреть по карте, где находится искомый объект**, а не бродить в поисках по всему городу.



Преимущества индексов

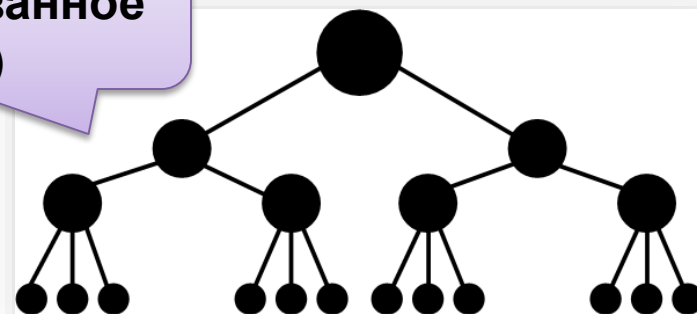
SHOW TABLE STATUS FROM `dbname`

Размер индексов позволяет разместить их **в оперативной памяти**.

Rows	Data	Index
1+ Million	>100 Mb	<20 Mb

Структура индексов **оптимизирована** для выполнения операций поиска.

B-Tree
(сбалансированное
дерево)



Индексы значительно **ускоряют операции поиска** данных в БД.

Иногда – на 2-3 порядка. Но так бывает не всегда, нужно исследовать ситуацию.

Занимают оперативную память.

Rows	Data	Index
1+ Million	>100 Mb	<20 Mb

В большинстве случаев значительно **замедляют** операции вставки, обновления, удаления, т.к. требуется обновлять сам индекс.

Иногда – в разы и на порядки. С этим можно «бороться», отключая индексы на момент вставки большого числа записей. В MySQL – так:

```
ALTER TABLE `table_name`  
DISABLE KEYS;
```

А потом:

```
ALTER TABLE `table_name`  
ENABLE KEYS;
```

Какие индексы создавать?

Признаки того, что **индекс нужен**:

- операции **чтения** из таблицы выполняются гораздо **чаще**, чем операции модификации;
- **поле или совокупность полей** часто фигурируют в запросах в секции **WHERE**;
- исследование показало, что **наличие индекса повышает производительность**.

Column	Type	Collation
<u>n_uid</u> ?	int(11)	
n_rubric	int(11)	
n_dt ?	int(11)	
n_title ?	text	utf8_general_ci
n_annotation ?	text	utf8_general_ci
n_author ?	varchar(255)	utf8_general_ci
n_text ?	text	utf8_general_ci
n_source	text	utf8_general_ci

Какие индексы создавать?

Проведём **исследование**. Для таблицы со следующей структурой **сначала не будем создавать индексов, кроме первичного ключа:**

Column	Type	Collation
<u>n_uid</u>	int(11)	
n_rubric	int(11)	
n_dt	int(11)	
n_title	text	utf8_general_ci
n_annotation	text	utf8_general_ci
n_author	varchar(255)	utf8_general_ci
n_text	text	utf8_general_ci
n_source	text	utf8_general_ci

Какие индексы создавать?

«Разброс» порядка 0.1 секунды

Выполним по 1000 раз:

Среднее время (с):

INSERT по 1000 строк

0.027289

SELECT * from `news` where `n_rubric`='...'

4.035899

SELECT * from `news` where `n_rubric`='...' AND
`n_dt`>='...' AND `n_dt`<='...'

4.065648

SELECT * from `news` where `n_dt`>='...' AND `n_dt`<='...'

4.508579

SELECT * from `news` where `n_title`='...' AND `n_dt`>='...'
AND `n_dt`<='...'

4.207702

SELECT * from `news` where `n_title`='...' AND
`n_author`='...' AND `n_dt`>='...' AND `n_dt`<='...'

4.187432

SELECT * from `news` where `n_title`='...' AND
`n_author`='...'

4.210264

SELECT * from `news` where `n_title`='...'

4.173025

SELECT * from `news` where `n_author`='...'

4.161251

Какие индексы создавать?

Создадим следующие индексы:

- **n_rubric**
- **n_rubric, n_dt**
- **n_title, n_dt**
- **n_dt, n_author**

n_rubric, т.к. есть {n_rubric, n_dt}, и первое поле в индексе МОЖЕТ ИСПОЛЬЗОВАТЬСЯ как «самостоятельно проиндексированное»

Что здесь лишнее и почему?

n_author	varchar(255)	utf8_general_ci
n_text	text	utf8_general_ci
n_source	text	utf8_general_ci

Какие индексы создавать?

«Разброс» порядка 20 секунд!!!

Снова выполним по 1000 раз: Среднее время (с):

INSERT n

0.027

0.896445

SELECT *

4.036

1.200757

SELECT *

4.066

0.207999

`n_dt`>='...' AND `n_dt`<='...'

SELECT * from `news` where `n_dt`>='...' AND `n_dt`<='...'

4.509

4.318613

SELECT * from `news` where `n_title`='...' AND `n_dt`>='...' AND `n_dt`<='...'

4.208

0.003918

SELECT * from `news` where `n_title`='...' AND `n_author`='...' AND `n_dt`>='...' AND `n_dt`<='...'

4.187

0.000468

SELECT * from `news` where `n_title`='...' AND `n_author`='...'

4.210

0.000909

SELECT * from `news` where `n_title`='...'

4.173

0.000279

SELECT * from `news` where `n_author`='...'

4.161

6.567766

Индекс «сработал», просто в выборку попадает очень много записей, извлечение которых отнимает время.

При поиске по этому полю применимых индексов нет.

Какие индексы создавать?

Итак, в общем случае индексы лучше создавать, чем не создавать 😊.

Поля и их комбинации, на которых лучше создать индексы, определяются исходя из наиболее часто выполняемых запросов на чтение.

Проверить, какие индексы использует СУБД, можно (в MySQL) командой **EXPLAIN** запрос;

```
EXPLAIN SELECT * from `news` where `n_author`='...'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	news	ALL	NULL	NULL	NULL	NULL	1000000	Using where

Виды индексов

Поле с уникальным
индексом

Поле с
НЕуникальным
индексом

Уникальные (unique) –
запрещают вставку
одинаковых значений в поле
таблицы. Как правило,
создаются на
«альтернативном ПК».

Неуникальные (non-unique) –
просто индексы ☺, созданные
для ускорения поиска.

wui	wnui
1	2
3	2
4	2
7	3
9	3

Виды индексов

Кластерные (cluster index) – строятся на поле, по значению которого упорядочена таблица. В таблице может быть только один кластерный индекс (и это, как правило – ПК).

Некластерные (non-cluster index) – строятся на произвольном неупорядоченном поле таблицы.

Здесь может быть кластерный индекс.

Здесь может быть НЕкластерный индекс.



sn_uid	sn_dt	sn_title
1	118758071	Title 64379878
2	121669144	Title 64046656
3	1569875561	Title 344103318
4	1631586361	Title 647355496
5	388175780	Title 8939791
6	160759710	Title 288420257

Виды индексов

Простые (simple index) – строятся на одном поле.

Составные, сложные (complex index) – строятся на нескольких полях или даже на выражениях.

Одно поле,
простой индекс.

Два поля,
составной индекс.



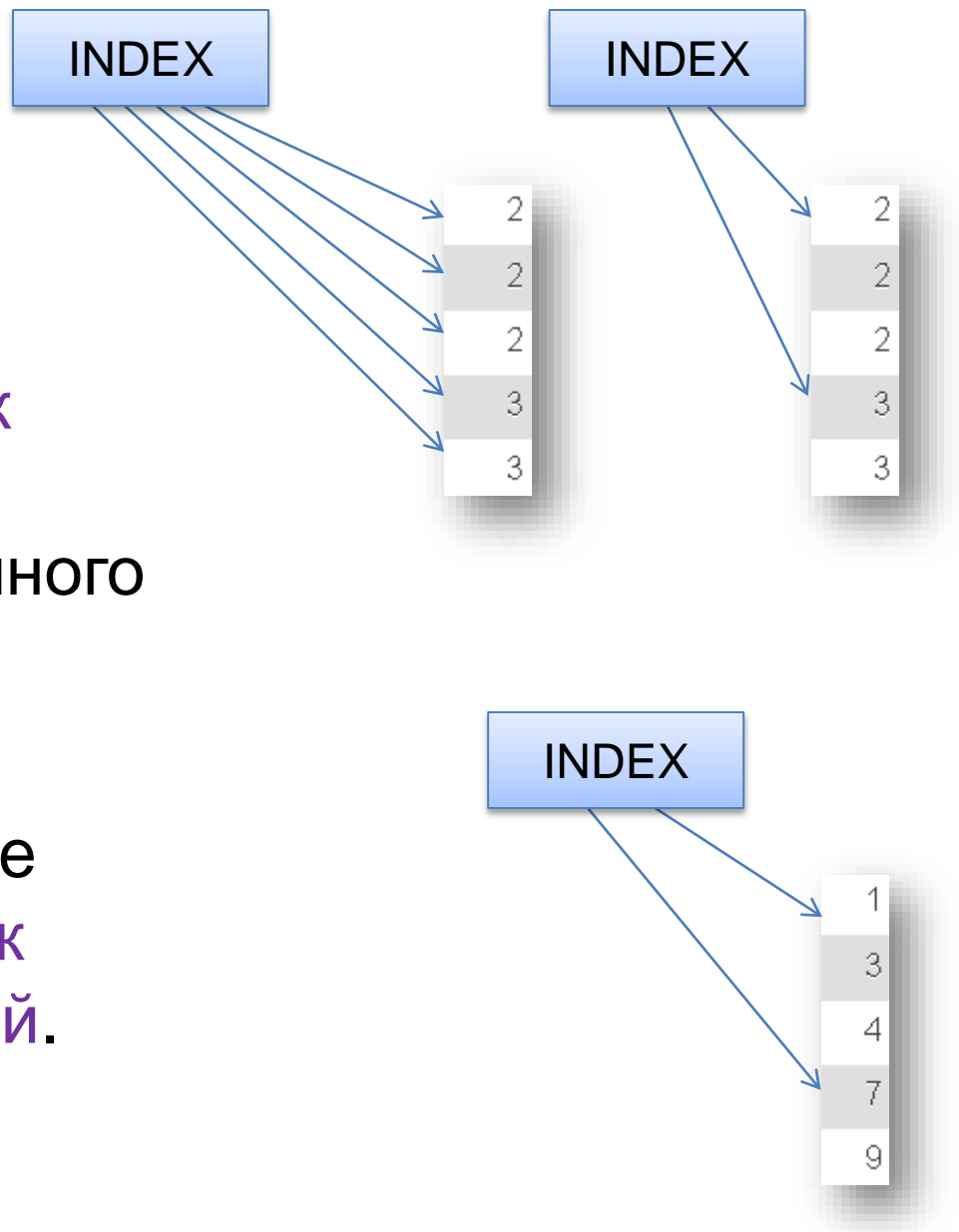
sn_uid	sn_dt	sn_title
1	118758071	Title 64379878
2	121669144	Title 64046656
3	1569875561	Title 344103318
4	1631586361	Title 647355496
5	388175780	Title 8939791
6	160759710	Title 288420257

Сложный индекс по
UPPER(`sn_title`) (**НЕ**
поддерживается MySQL)

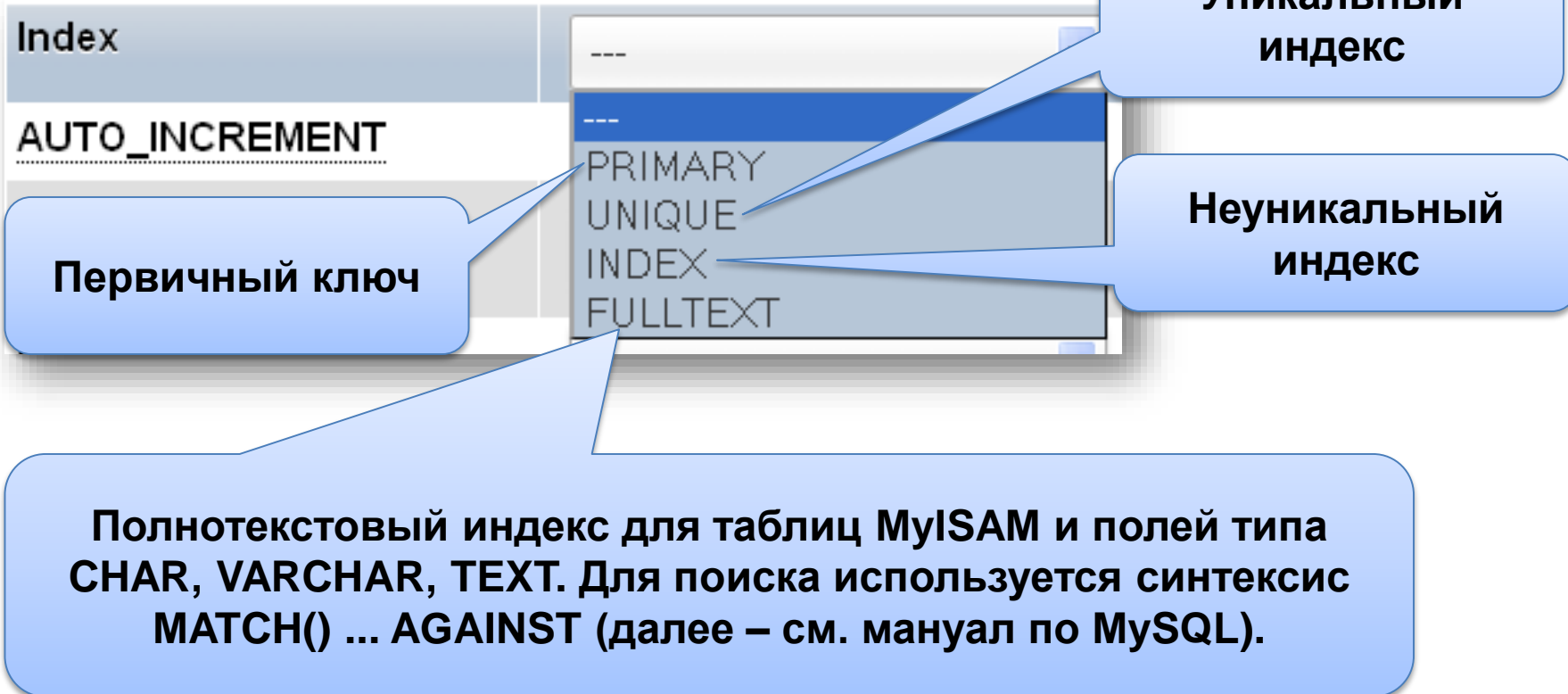
Виды индексов

Плотные (dense index) – указывают на конкретную запись в таблице или блок записей с одинаковыми значениями индексируемого поля.

Неплотные, редкие (sparse index) – указывают на блок (отсортированных) записей.



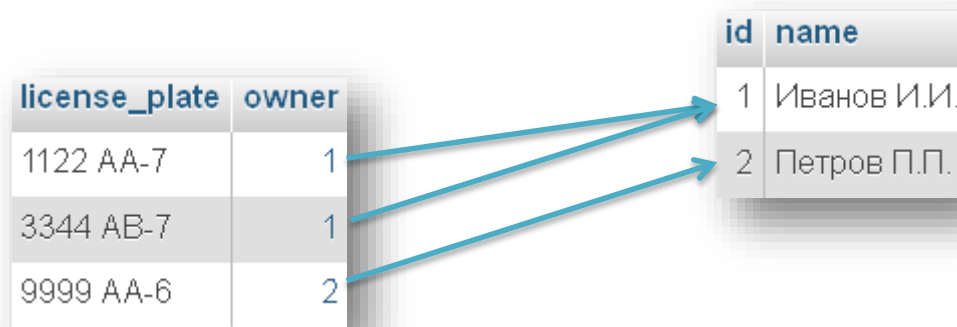
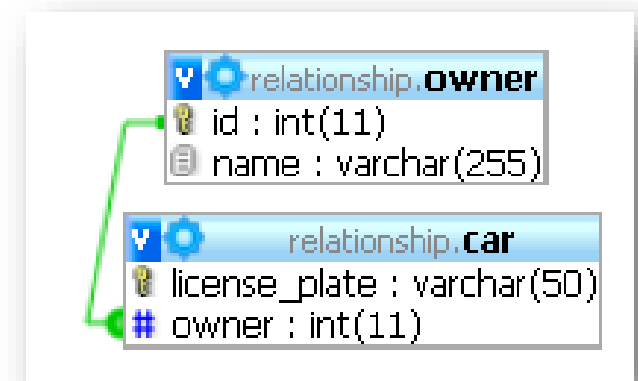
Управление индексами в MySQL



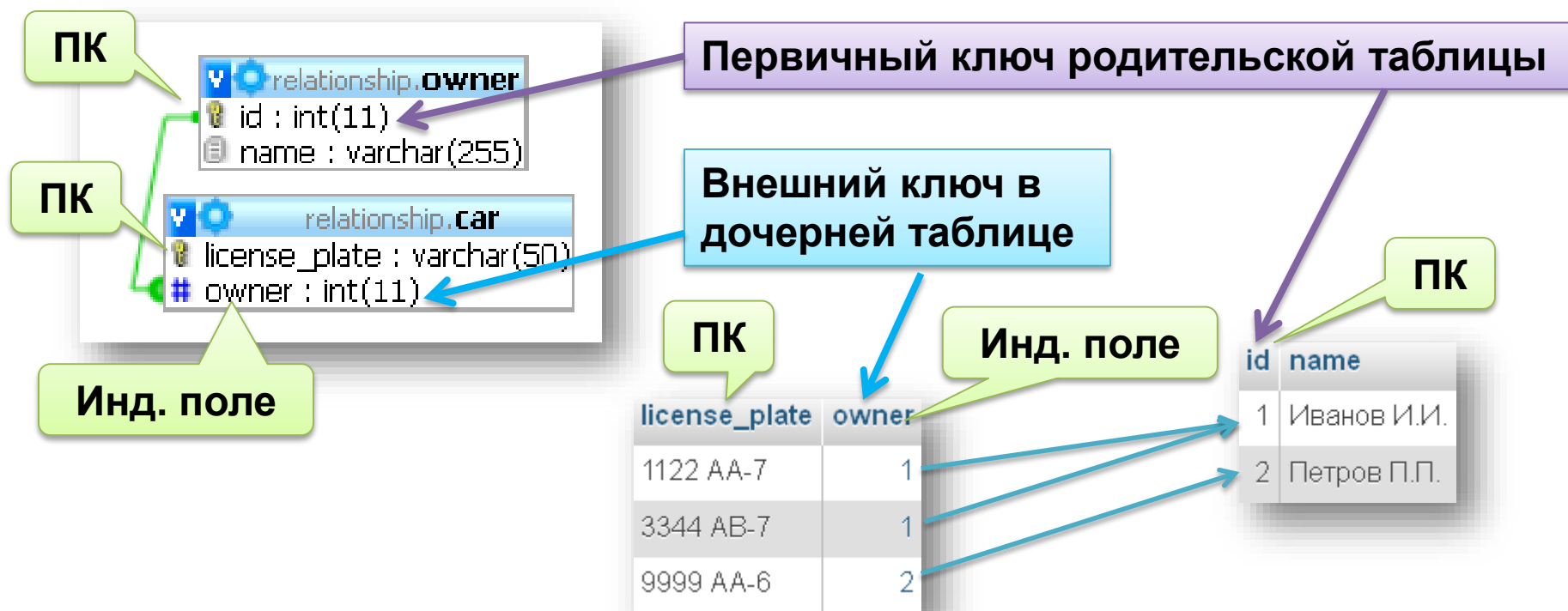
СВЯЗИ

Связь (relationship) – ассоциация, установленная между двумя и более сущностями (relations, entities).

Простая аналогия «из жизни» – указание в описании автомобиля информации о его владельце.



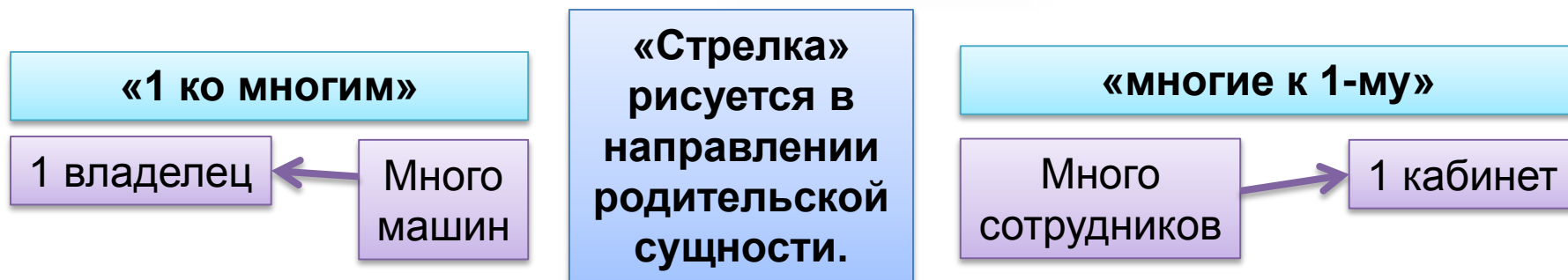
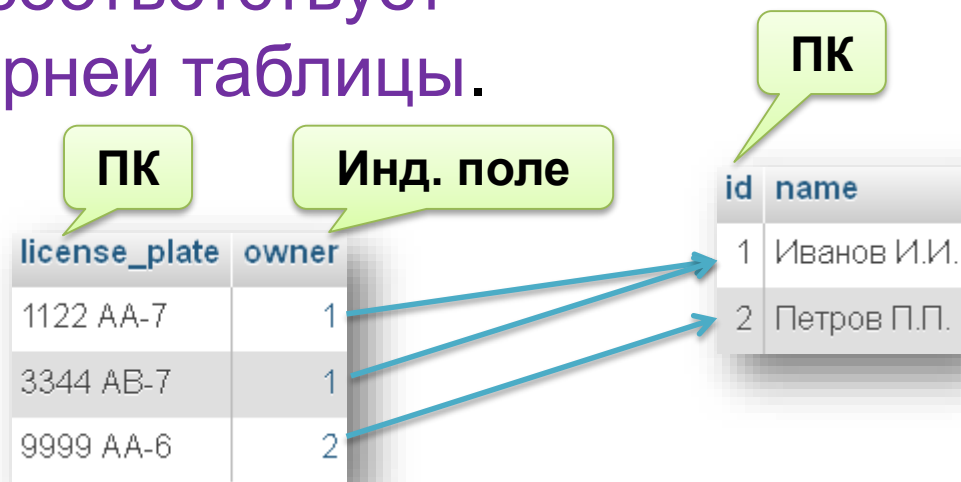
Связь организуется за счёт **миграции первичного ключа** родительской таблицы в дочернюю таблицу. Получившееся в результате поле называется **внешним ключом** (foreign key).



Виды связей: «1 ко многим» («многие к 1-му»)

Связь «1 ко многим» («многие к 1-му») определяет ситуацию, когда **одной записи** родительской таблицы соответствует **несколько записей** дочерней таблицы.

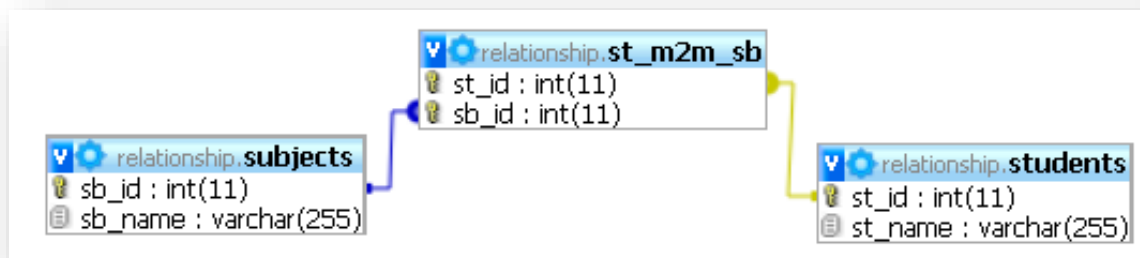
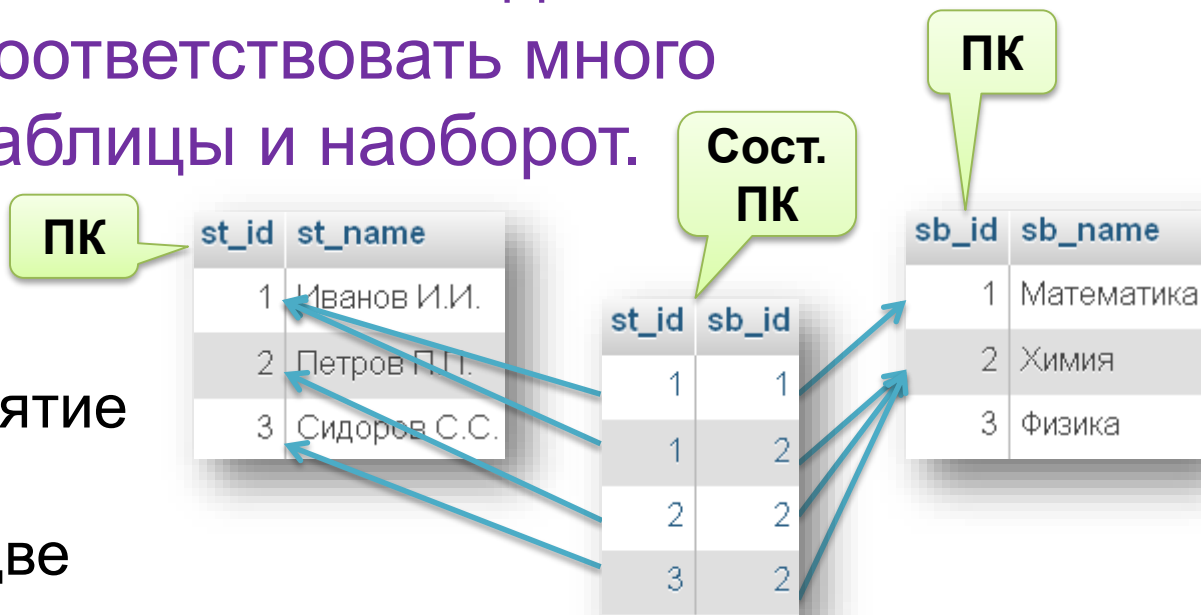
Разница между «1 ко многим» и «многие к 1-му» — исключительно в том, «кто является главным».



Виды связей: «многие ко многим»

Связь «многие ко многим» определяет ситуацию, когда **любой** записи одной таблицы может соответствовать много записей другой таблицы и наоборот.

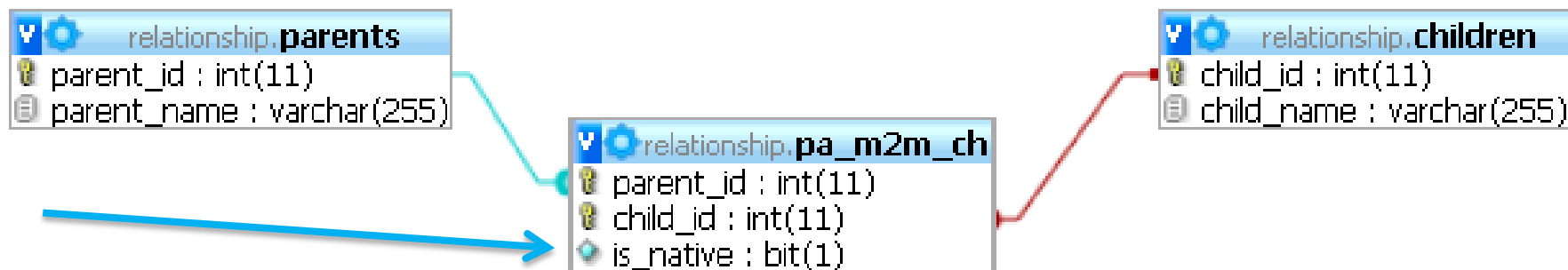
Связь «М-М» – исключительно «человеческое» понятие и в реальных БД реализуется через две связи «1-М».



Виды связей: «многие ко многим»

Связь «М-М» может также отражать некие свои свойства, не характерные для связываемых сущностей.

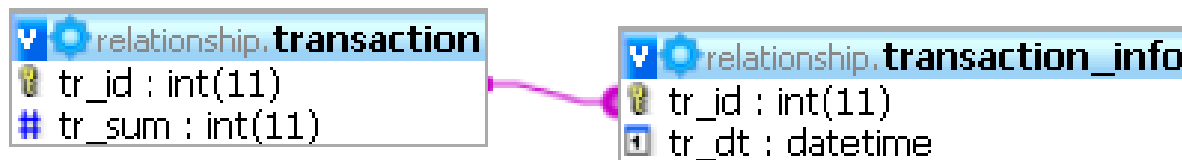
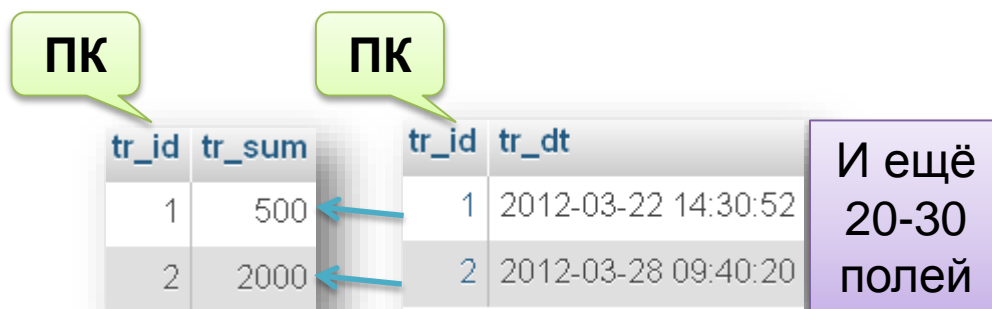
Например, в связи «родители-дети» можно отразить признак родства, который является именно свойством связи, но не родителя или ребёнка.



Виды связей: «1 к 1-му»

Связь «1 к 1-му» определяет ситуацию, когда **любой** записи одной таблицы может соответствовать **ровно одна** запись другой таблицы и наоборот.

Наличие неаргументированной связи «1-1» в простых БД – признак ошибки в формировании структуры.



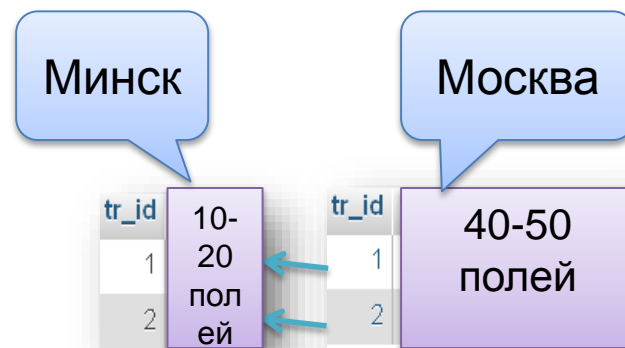
Виды связей: «1 к 1-му»

Связь «1-1» оправдана в том случае, если:

- Нужно разнести **очень часто и очень редко обрабатываемые данные** по разным таблицам для ускорения работы.
- Получилась **«мега-таблица»**, количество полей в которой «упирается» в ограничения СУБД.
- Нужно **разнести данные по удалённым друг от друга местам**, где они активно обрабатываются.

tr_id	tr_sum	tr_id	tr_dt	И ещё 20-30 полей
1	500	1	2012-03-22 14:30:52	
2	2000	2	2012-03-28 09:40:20	

tr_id	10-20 полей	tr_id	40-50 полей
1		1	
2		2	

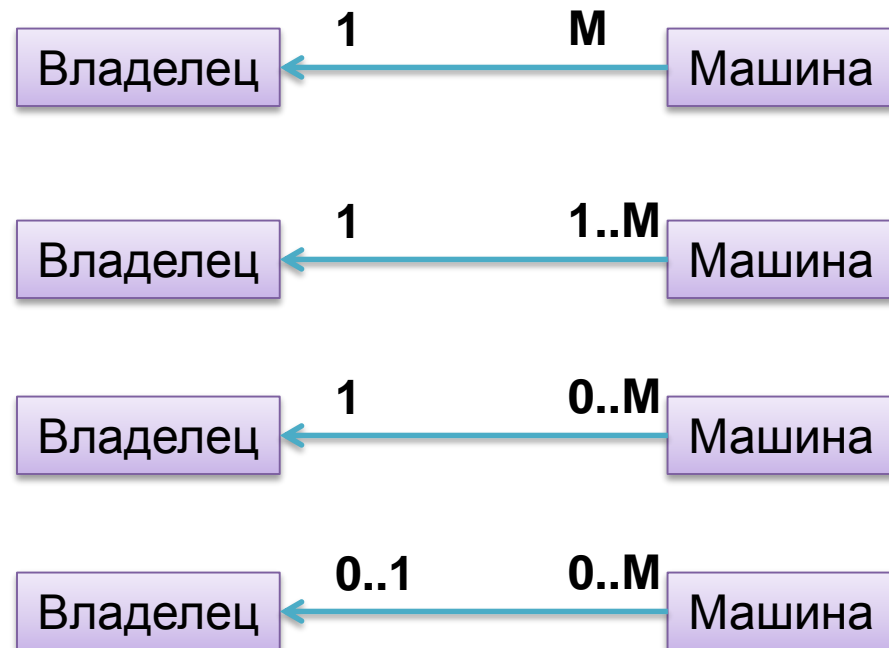


Мощность связи

Мощность (кардинальность) связи (relationship cardinality) – указание **возможного числа записей** в таблице с каждой стороны связи.

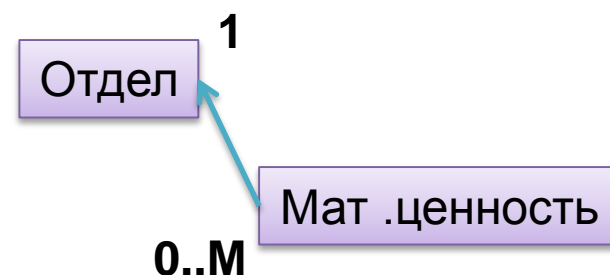
При серьёзном проектировании указывают обе (нижнюю и верхнюю) границы с каждой стороны связи.

В чём разница таких записей?



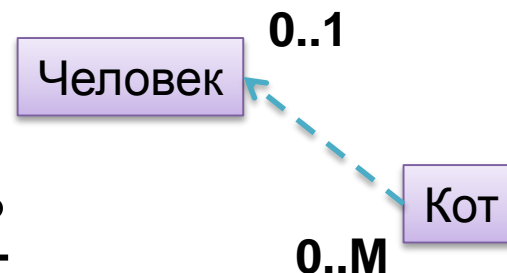
Идентифицирующая связь

(identifying relationship) определяет ситуацию, когда запись в дочерней таблице обязана быть связана с записью в родительской таблице.



НЕидентифицирующая связь

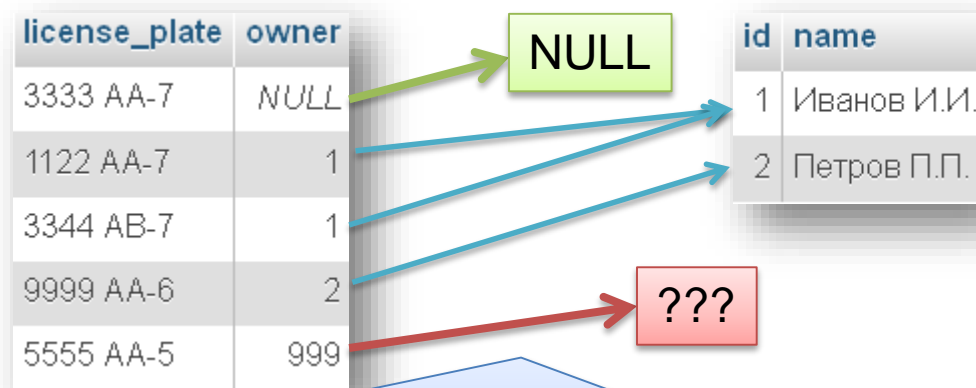
(non-identifying relationship) определяет ситуацию, когда запись в дочерней таблице может быть НЕ связана с записью в родительской таблице.



ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ ДАННЫХ

Ссылочная целостность (referential integrity) – необходимое качество реляционной БД, заключающееся в отсутствии в любом её отношении внешних ключей, ссылающихся на несуществующие кортежи.

Простым языком: «если ключ на что-то ссылается, это что-то должно существовать».



При наличии ЯВНО прописанных между таблицами связей СУБД не допустит такой ситуации.

Каскадные операции

Каскадные операции (cascade operations) – специальные ограничения БД, описывающие её поведение в случае удаления записи из родительской таблицы или изменения её первичного ключа.

Каскадное удаление	Каскадное обновление	Установка пустых ключей	Установка значения по умолчанию	Запрет каскадной операции
Записи в дочерней таблице удаляются	Значения внешних ключей в дочерней таблице обновляются	Во внешние ключи в дочерней таблице выставляется значение NULL	Во внешние ключи в дочерней таблице выставляется значение по умолчанию	Запись из родительской таблицы нельзя удалить, пока ей соответствуют записи в дочерней таблице

Каскадное удаление

При **каскадном удалении** записи дочерней таблицы, соответствующие удаляемой записи родительской таблицы, тоже удаляются.

id	name
1	Иванов И.И.
2	Петров П.П.



license_plate	owner
3333 AA-7	NULL
1122 AA-7	1
3344 AB-7	1
9999 AA-6	2

Каскадное обновление

При обновлении первичного ключа родительской таблицы **внешние ключи** соответствующих записей в дочерней таблице принимают это же новое значение.

555

id	name
1	Иванов И.И.
2	Петров П.П.

=>

license_plate	owner
3333 AA-7	NULL
1122 AA-7	1
3344 AB-7	1
9999 AA-6	2

555

555

Установка пустых ключей

При удалении записи из родительской таблицы **во внешние ключи** соответствующих записей дочерней таблицы устанавливается значение **NULL**.

id	name
1	Иванов И.И.
2	Петров П.П.



license_plate	owner
3333 AA-7	NULL
1122 AA-7	1
3344 AB-7	1
9999 AA-6	2

NULL

NULL

Установка значения по умолчанию

При удалении записи из родительской таблицы **во внешние ключи** соответствующих записей дочерней таблицы устанавливается значение по умолчанию.

id	name
1	Иванов И.И.
2	Петров П.П.



license_plate	owner
3333 AA-7	NULL
1122 AA-7	1
3344 AB-7	1
9999 AA-6	2

X

X

Запрет каскадной операции

Пока существует хотя бы одна запись в дочерней таблице, соответствующая некоей записи в родительской таблице, эту запись из родительской таблицы нельзя удалить и/или нельзя изменить её первичный ключ.

id	name
1	Иванов И.И.
2	Петров П.П.

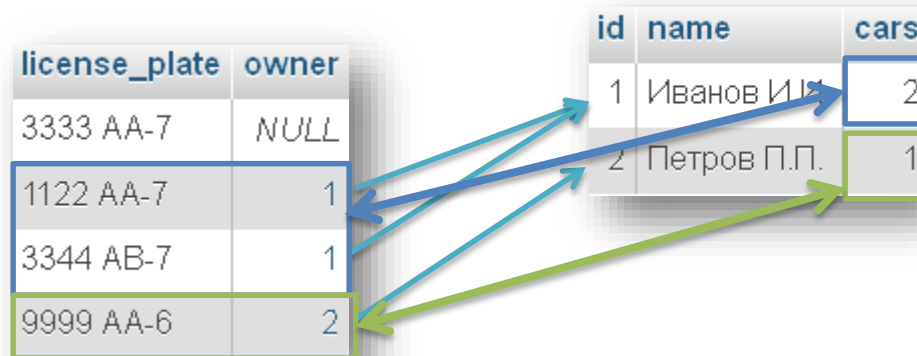


license_plate	owner
3333 AA-7	NULL
1122 AA-7	1
3344 AB-7	1
9999 AA-6	2

Изменение ПК и удаление записи запрещено

Консистентность данных (data consistency, data validity) – согласованность данных друг с другом: ссылочная целостность и внутренняя непротиворечивость.

Простым языком:
фрагменты данных в БД
не должны противоречить
друг другу.



Консистентность данных

Консистентность данных, как правило, обеспечивается созданием **триггеров**, контролирующих операции с таблицами и корректирующих соответствующие данные или блокирующих операцию.

О триггерах – чуть позже.

license_plate	owner
3333 AA-7	NULL
1122 AA-7	1
3344 AB-7	1
9999 AA-6	2

=>

id	name	cars
1	Иванов И.И.	2
2	Петров П.П.	1

1

ТРИГГЕРЫ

Понятие триггера

Триггер (trigger) – элементарная программа, написанная на некотором расширении языка SQL, и используемая **для обеспечения консистентности данных**.

Более подробно о триггерах – в разделах, посвящённых языку SQL.

	Вставка	Обновление	Удаление
Перед	BEFORE INSERT	BEFORE UPDATE	BEFORE DELETE
После	AFTER INSERT	AFTER UPDATE	AFTER DELETE

Задачи триггеров

С использованием триггеров, как правило, решаются задачи, которые нельзя решить только с использованием ссылок, например:

- Обновление **агрегированных данных** («сколько у владельца машин»).
- Сложный **запрет удаления** («в системе должен быть хотя бы один администратор»).
- Контроль **числовых значений** («рост – от 50 до 300 см»).
- ...

СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Отношения. Ключи. Связи.

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