

EPAM Systems, RD Dep.
Конспект и раздаточный материал

СТЕСН.ДВ.02 Отношения. Ключи.
Связи.

REVISION HISTORY					
Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
<1.0>	Первая версия	Святослав Куликов	<19.03.2012>		

Legal Notice

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM Systems.

Содержание

1. ОТНОШЕНИЯ.....	3
1.1. ОПРЕДЕЛЕНИЕ	3
1.2. ОТНОШЕНИЯ В РЕЛЯЦИОННОЙ ТЕОРИИ И В РЕАЛЬНОСТИ	4
2. КЛЮЧИ.....	4
2.1. ОПРЕДЕЛЕНИЕ	4
2.2. ВИДЫ КЛЮЧЕЙ	4
3. ИНДЕКСЫ.....	11
3.1. ОПРЕДЕЛЕНИЕ	11
3.2. ВИДЫ ИНДЕКСОВ	14
3.3. УПРАВЛЕНИЕ ИНДЕКСАМИ В MySQL	15
4. СВЯЗИ	16
4.1. ОПРЕДЕЛЕНИЕ	16
4.2. ВИДЫ СВЯЗЕЙ	17
4.3. ИДЕНТИФИЦИРУЮЩАЯ И НЕИДЕНТИФИЦИРУЮЩАЯ СВЯЗИ	19
5. ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ ДАННЫХ	20
5.1. ОПРЕДЕЛЕНИЕ	20
5.2. КАСКАДНЫЕ ОПЕРАЦИИ	20
5.3. КОНСИСТЕНТНОСТЬ ДАННЫХ	22
6. ТРИГГЕРЫ.....	23

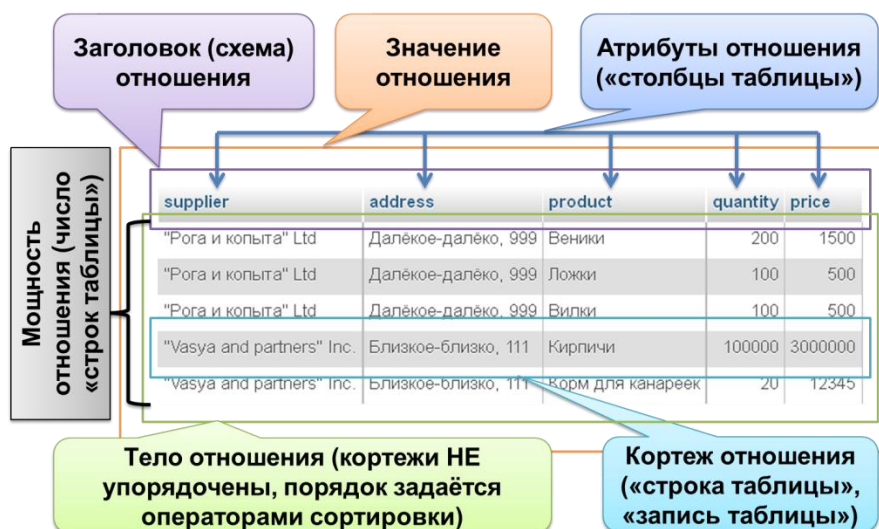
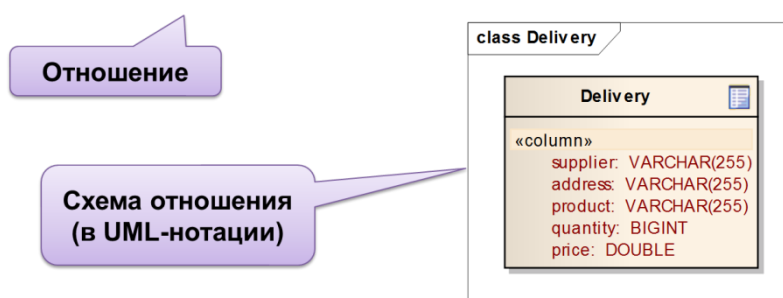
1. Отношения

1.1. Определение

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

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



1.2. Отношения в реляционной теории и в реальности

Теория реляционных БД говорит, что таблица может считаться отношением, если:

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

ОК

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

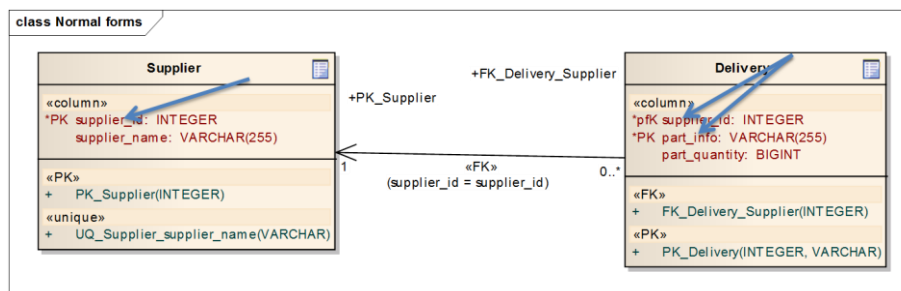
ОК

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

2. Ключи

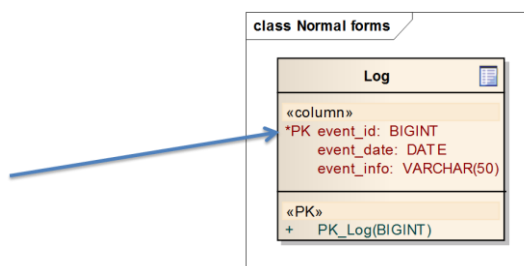
2.1. Определение

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



2.2. Виды ключей

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



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

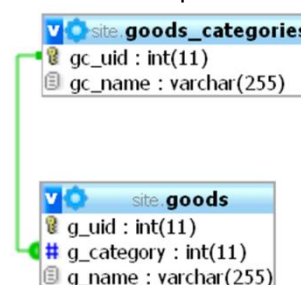
sp_uid Идентификатор страницы	sp_parent Ссылка на родительскую страницу	sp_name Имя страницы
1	NULL	Главная
2	1	О фирме
3	1	Услуги
4	3	Юристам
5	3	Физлицам
6	5	Test

Первичный ключ служит для:

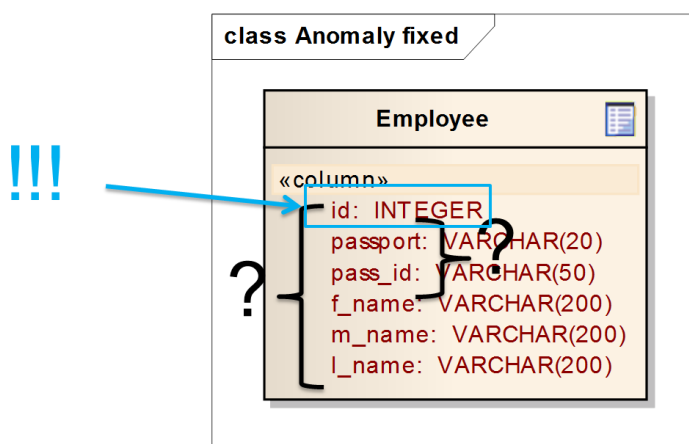
Однозначной идентификации строки
таблицы

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	Юристам
5	3	Физлицам
6	5	Test

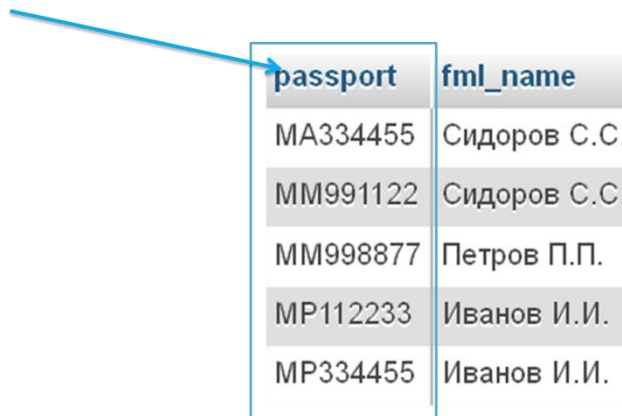
Организации связей между
таблицами



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

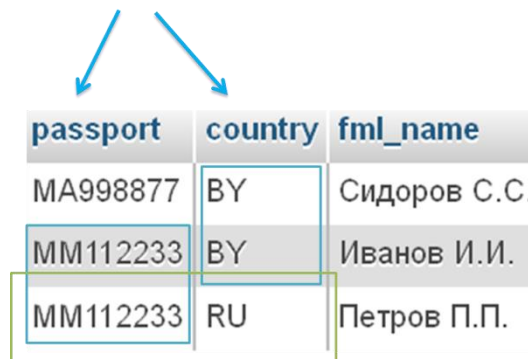


Простой первичный ключ (simple primary key) – первичный ключ, состоящий из единственного поля таблицы (атрибута отношения).



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



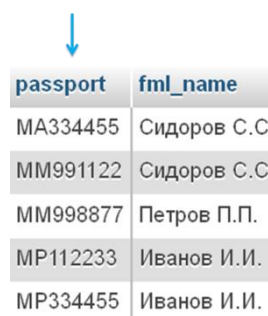
passport	country	fml_name
MA998877	BY	Сидоров С.С.
MM112233	BY	Иванов И.И.
MM112233	RU	Петров П.П.

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.

Т.о. следует определить, по какому полю поиск производится чаще, и так создать ключ:

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

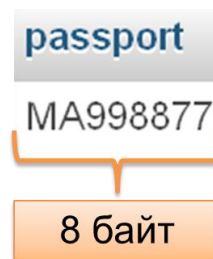
Естественный первичный ключ (natural PK) – поле таблицы, хранящее полезные данные.



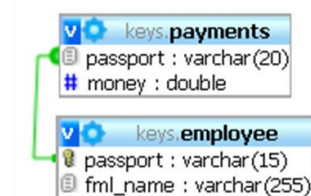
passport	fml_name
MA334455	Сидоров С.С.
MM991122	Сидоров С.С.
MM998877	Петров П.П.
MP112233	Иванов И.И.
MP334455	Иванов И.И.

У естественных первичных ключей есть **недостатки**:

Большой размер.

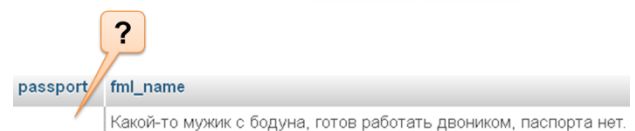


Необходимость каскадных изменений.



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

Невозможность вставки данных при отсутствии части информации.



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,

У синтетических ключей есть **преимущества**:

Малый размер.




Нет необходимости каскадных изменений.



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

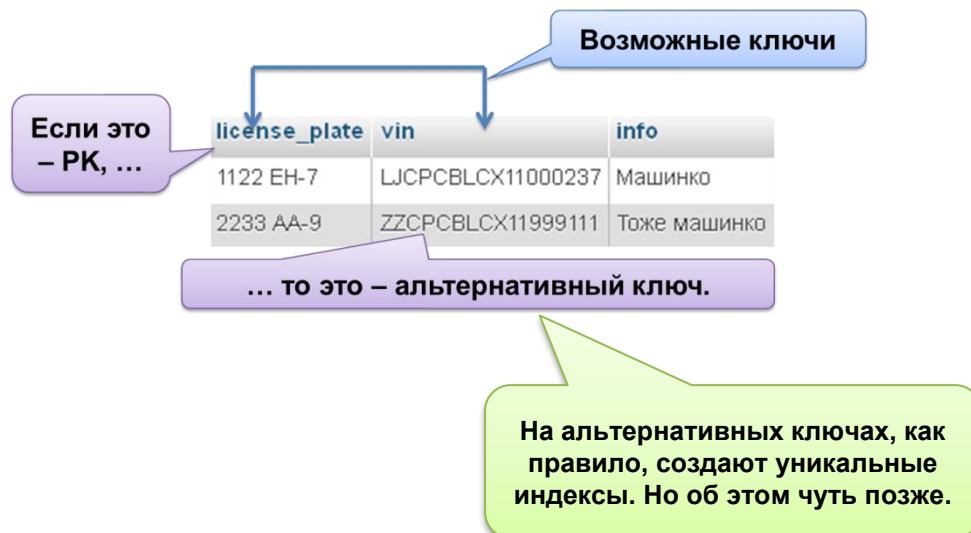
Можно вставить данные при отсутствии части информации.




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

Возможный ключ (possible key) – поле или совокупность полей с уникальными значениями, кандидат в первичные ключи.

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



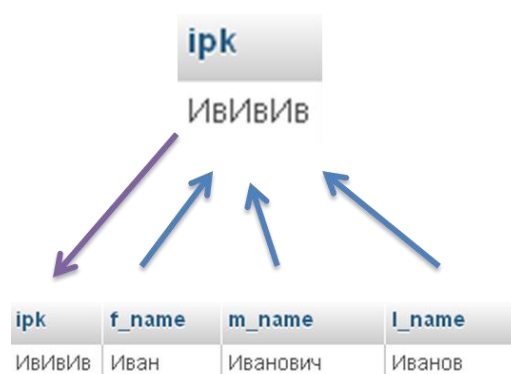
Интеллектуальный первичный ключ (intelligent primary key) – поле, значение которого формируется на основе значений других полей.



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

С интеллектуальными ключами есть проблемы из-за которых этот вид ключей практически не используется.

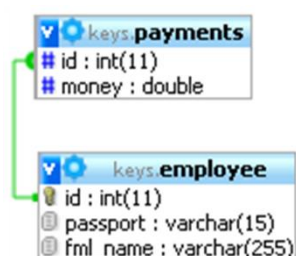
Их нужно генерировать и обновлять.



Высока вероятность коллизий (совпадений значений).

АлАлАл	Александр	Александрович	Александров
АлАлАл	Алексей	Алексеевич	Алексеев

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

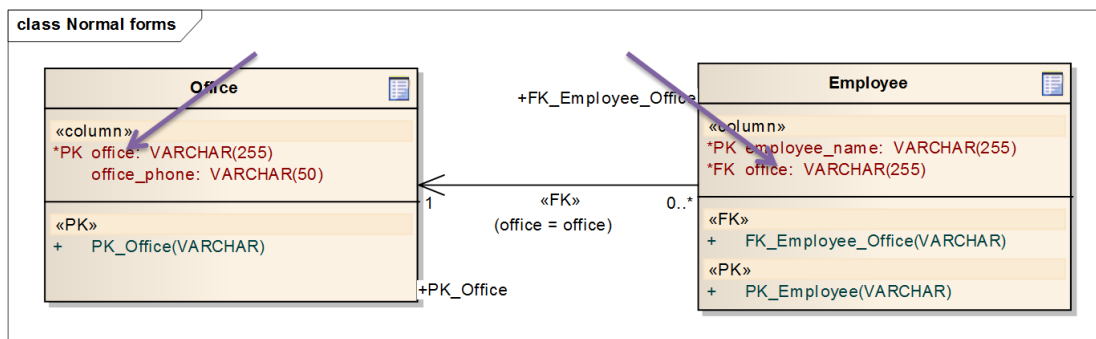


id	passport	fml_name
1	MP334455	Иванов И.И.
2	MP112233	Иванов И.И.

id	money
1	999
1	5754
1	6433634
2	454534

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.



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



PK	RFK
sp_uid Идентификатор страницы	sp_parent Ссылка на родительскую страницу
1	NULL Главная
2	1 О фирме
3	1 Услуги
4	3 Юристам
5	3 Физлицам
6	5 Test

3. Индексы

3.1. Определение

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

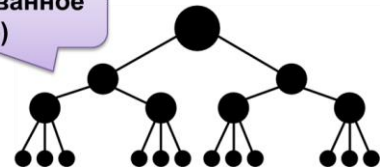
Размер индексов позволяет разместить их в оперативной памяти.

`SHOW TABLE STATUS FROM `dbname``

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.**

Недостатки индексов

Занимают оперативную память.

`SHOW TABLE STATUS FROM `dbname``

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.

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

Column	Type	Collation
n_uid	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

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

Создадим следующие индексы:

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

n_rubric лишний, т.к. есть {n_rubric, n_dt}, и первое поле в индексе может использоваться как «самостоятельно проиндексированное»

Выполним по 1000 раз	БЫЛО среднее время, с	СТАЛО среднее время, с
INSERT по 1000 строк	0.027289	0.896445
SELECT * from `news` where `n_rubric`='...'	4.035899	1.200757
SELECT * from `news` where `n_rubric`='...' AND `n_dt`>='...' AND `n_dt`<='...'	4.065648	0.207999
SELECT * from `news` where `n_dt`>='...' AND `n_dt`<='...'	4.508579	4.318613
SELECT * from `news` where `n_title`='...' AND `n_dt`>='...' AND `n_dt`<='...'	4.207702	0.003918
SELECT * from `news` where `n_title`='...' AND `n_author`='...' AND `n_dt`>='...' AND `n_dt`<='...'	4.187352	0.000468
SELECT * from `news` where `n_title`='...' AND `n_author`='...'	4.210264	0.000909
SELECT * from `news` where `n_rubric`='...'	4.173025	0.000279
SELECT * from `news` where `n_author`='...'	4.161251	6.567766

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

При поиске по этому полю применимых индексов нет.

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

3.2. Виды индексов

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

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

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.

Простые (simple index) – строятся на одном поле.

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

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

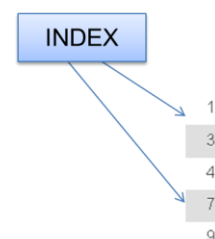
sn_uid	sn_dt	sn_title
1	118758071	Title 64379878
2	121669144	Title 64046656
3	1569875561	Title 344103318
4	1631586361	Title 647355496
5	388175780	Title 8939791
6	160759710	Title 288420257

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)

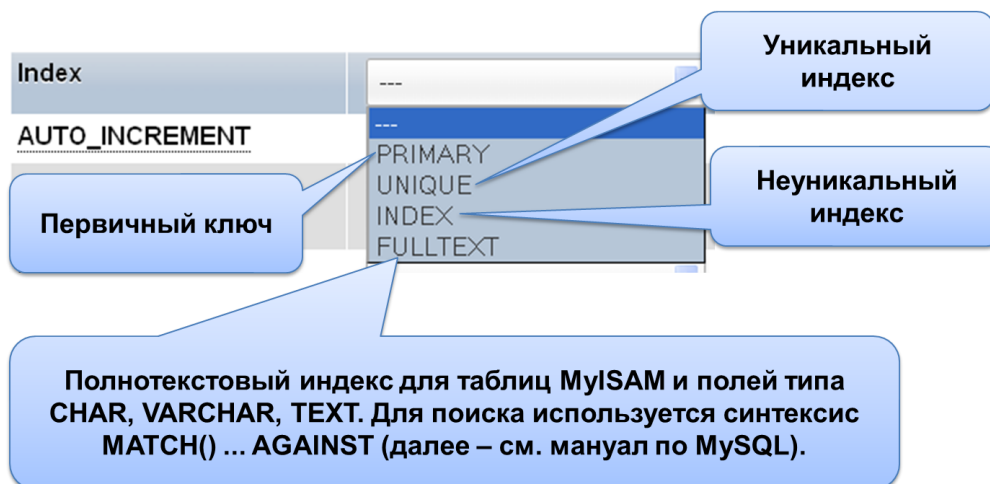
Плотные (dense index) – указывают на конкретную запись в таблице или блок записей с одинаковыми значениями индексированного поля.

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



3.3. Управление индексами в MySQL

MySQL на текущий момент поддерживает четыре типа индексов:

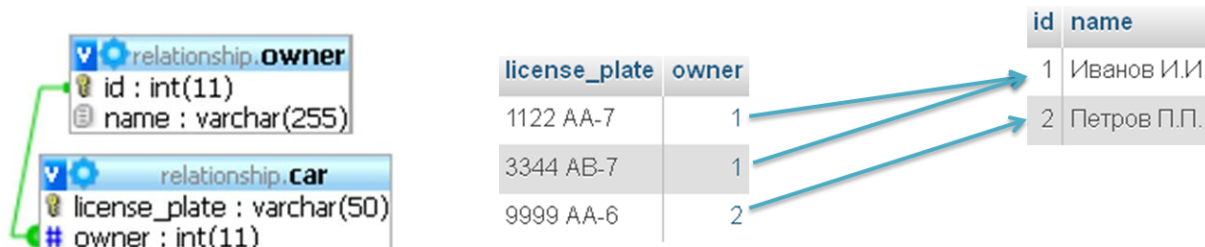


4. Связи

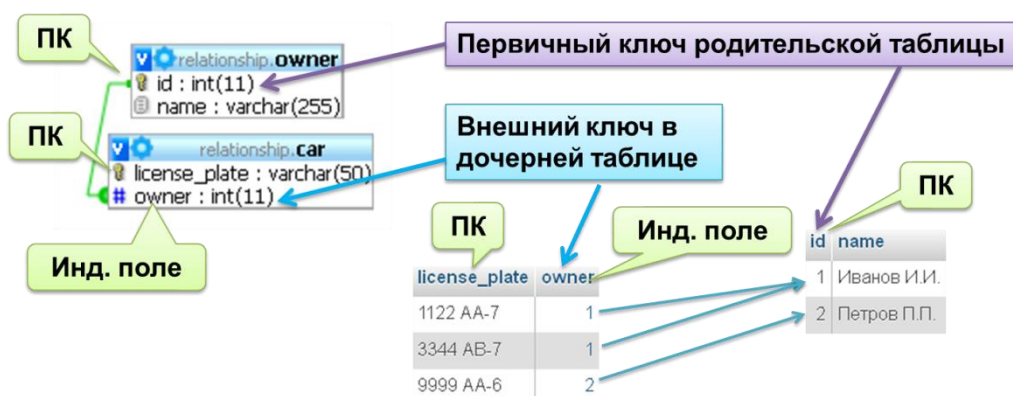
4.1. Определение

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

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



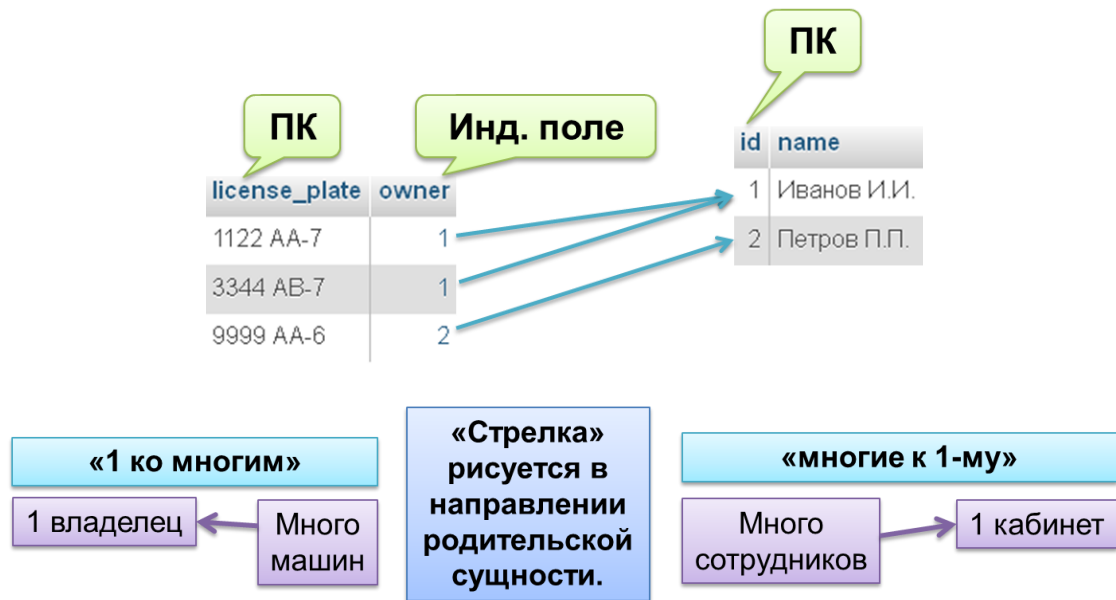
Связь организуется за счёт миграции первичного ключа родительской таблицы в дочернюю таблицу. Получившееся в результате поле называется внешним ключом (foreign key).



4.2. Виды связей

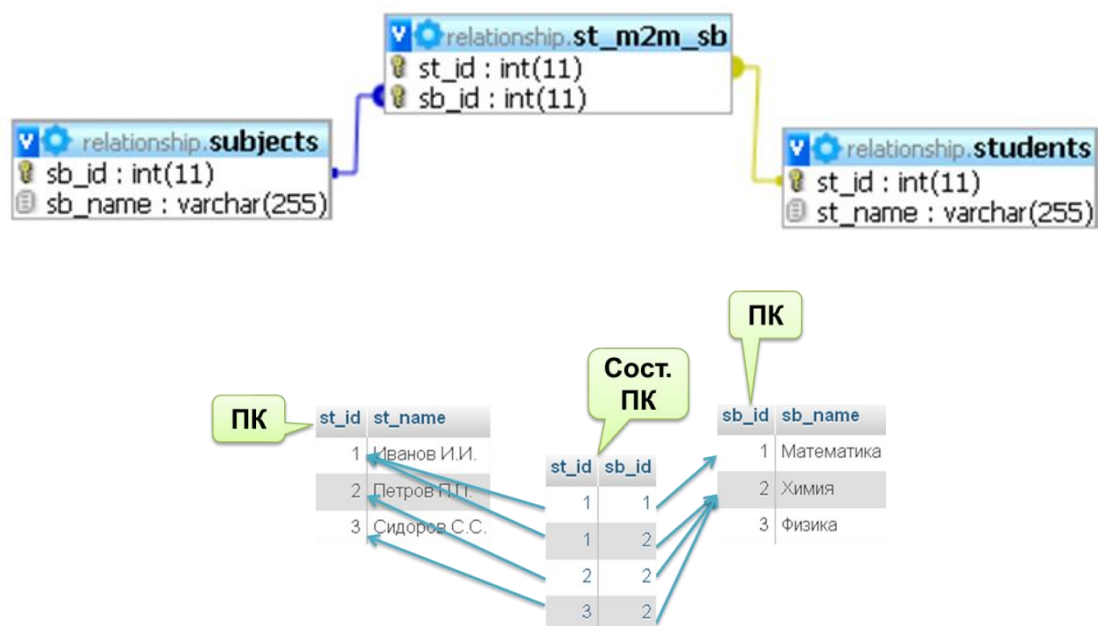
Связь «1 ко многим» («многие к 1-му») определяет ситуацию, когда одной записи родительской таблицы соответствует несколько записей дочерней таблицы.

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



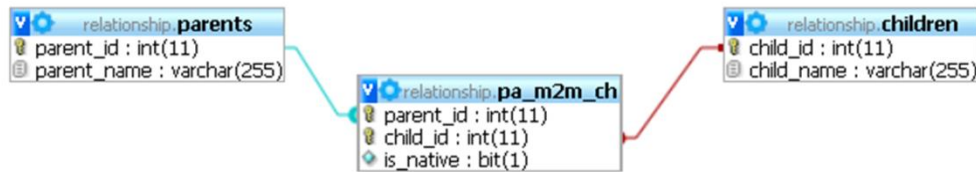
Связь «многие ко многим» определяет ситуацию, когда любой записи одной таблицы может соответствовать много записей другой таблицы и наоборот.

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



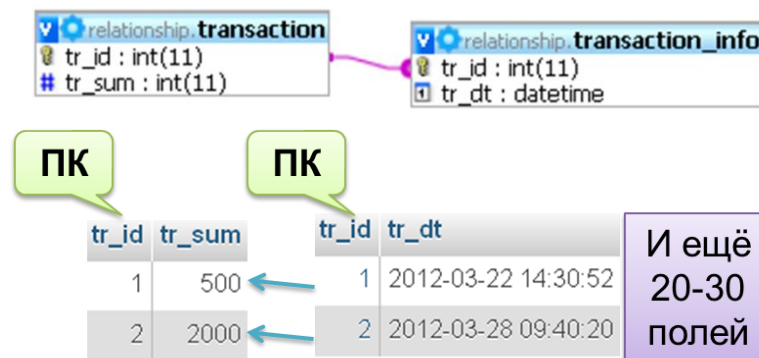
Связь «М-М» может также отражать некие свои свойства, не характерные для связываемых сущностей.

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



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.

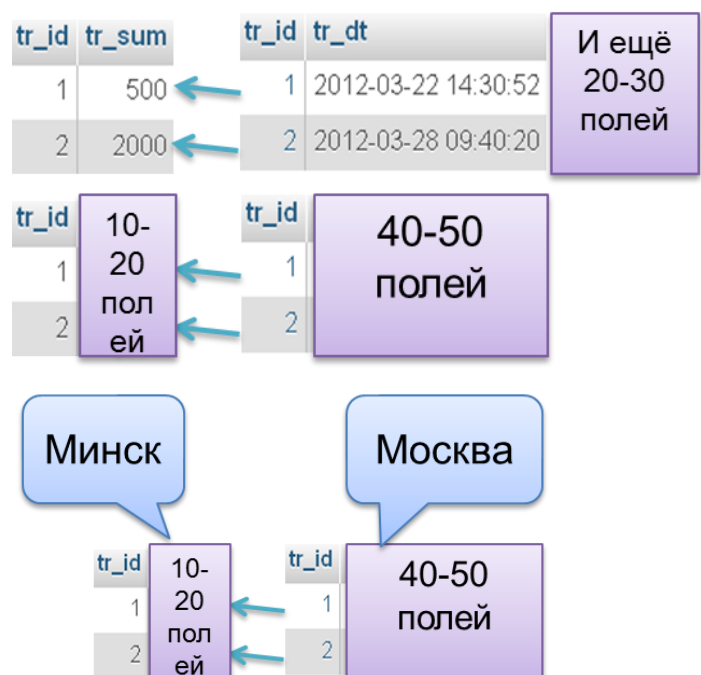


Связь «1-1» оправдана в том случае, если:

Нужно разнести очень часто и очень редко обрабатываемые данные по разным таблицам для ускорения работы.

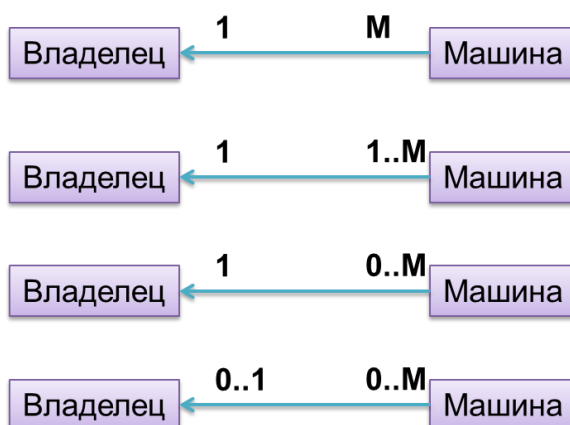
Получилась «мега-таблица», количество полей в которой «упирается» в ограничения СУБД.

Нужно разнести данные по удалённым друг от друга местам, где они активно обрабатываются.



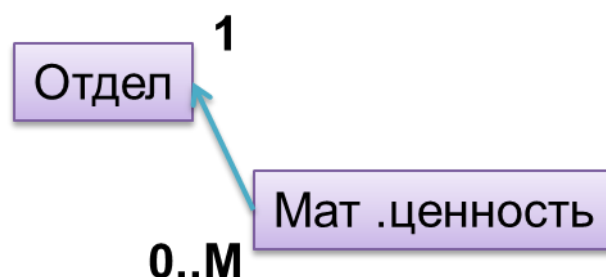
Мощность (кардинальность) связи (relationship cardinality) – указание возможного числа записей в таблице с каждой стороны связи.

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

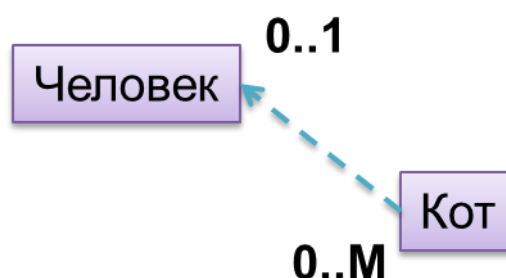


4.3. Идентифицирующая и неидентифицирующая связи

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.



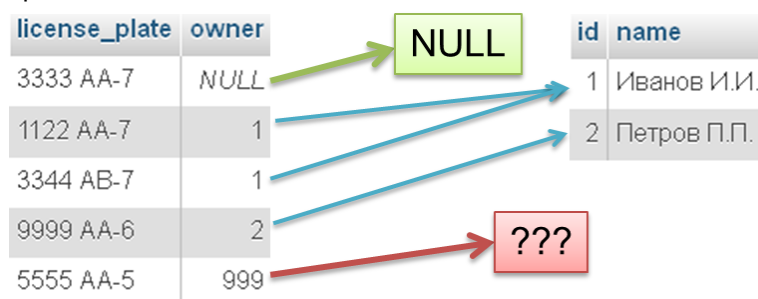
5. Ссылочная целостность данных

5.1. Определение

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:



5.2. 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.

id	name
1	Иванов И.И.
2	Петров П.П.

=>

license_plate	owner
3333 AA-7	NULL
1122 AA-7	555
3344 AB-7	555
9999 AA-6	2

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	NULL
3344 AB-7	NULL
9999 AA-6	2

Установка значения по умолчанию

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

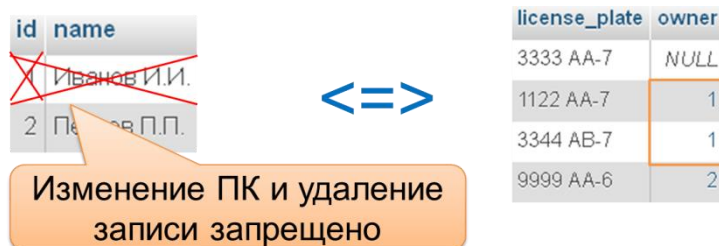
id	name
1	Иванов И.И.
2	Петров П.П.

=>

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

Запрет каскадной операции

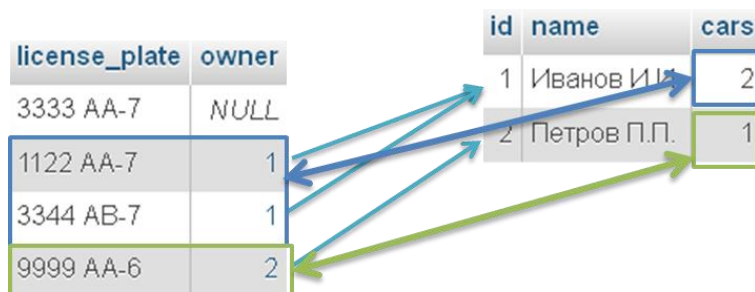
Пока существует хотя бы одна запись в дочерней таблице, соответствующая некоей записи в родительской таблице, эту запись из родительской таблицы нельзя удалить и/или нельзя изменить её первичный ключ.



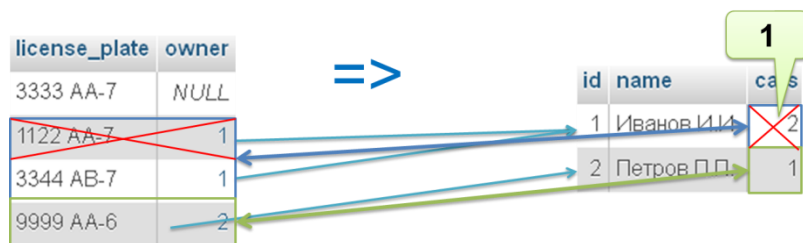
5.3. Консистентность данных

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



6. Триггеры

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.

Виды триггеров

	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»).
- ...