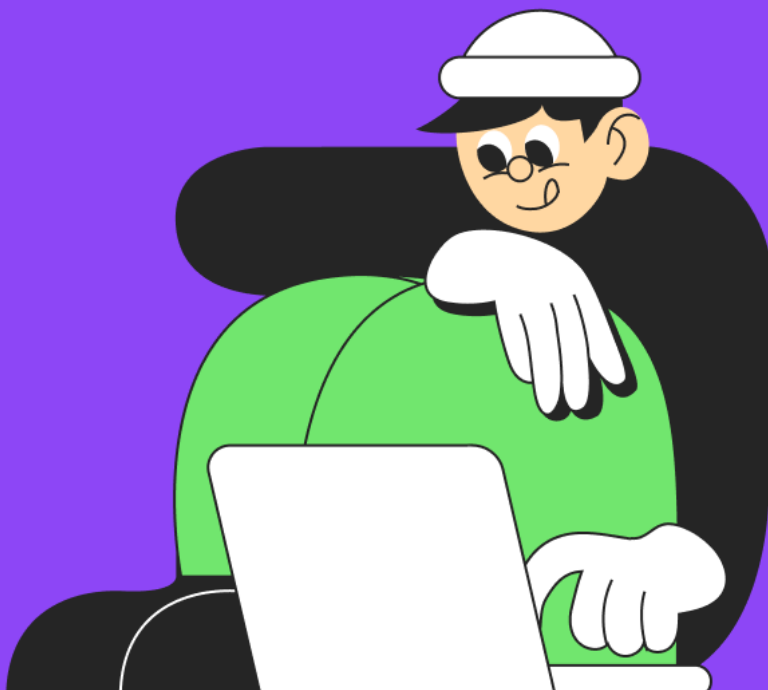
GeekBrains

Создание структуры базы данных

Знакомство с базами данных



Оглавление

Приветствие

На этом уроке

Структуры баз данных

- Структура таблицы «Люди»
- Структура таблицы «Телефоны»
- Структура таблицы «Адреса»

Типы связей

Кейс «Аренда автомобилей»

- Связующая таблица «Сделка»
- Сценарий использования

Как проектировать структуры таблиц?

Заключение

[00:01:36]

Приветствие

И. Шафигуллин: добрый день, друзья. Мы рады приветствовать вас на третьей лекции курса «Знакомство с базами данных». Сегодня со структурами баз данных вас познакомит Александр Сагун.

А. Сагун: добрый день! Возможно, вы меня уже видели, вместе с Александром Волчеком я рассказывал про профессии, которые вы будете изучать в рамках курса «Разработчик». Немного расскажу о себе: я исполнительный директор GeekBrains, до этого долго работал в IT. Начинал как junior software engineer в 2009 году, потом занимался проектами автоматизации в крупных компаниях (например, в «Техносиле» и X5 Retail Group), работал в нефтегазовой сфере. Был руководителем образовательной компании «Бизнес Молодость». Сейчас у меня есть свой проект, который я совмещаю с работой в GeekBrains.

[00:02:56]

На этом уроке

И. Шафигуллин: сегодня у нас будет несколько задач, но сперва вспомним, чему успели научиться. На первой лекции мы разбирались с моделями данных, которые находятся внутри баз. Начали с иерархических структур, посмотрели, как с ними работали до появления компьютеров. На второй лекции узнали, как извлекать данные из таблиц, и на уровне псевдокода посмотрели на язык запросов SQL. Разобрали, как можно объединять данные из нескольких таблиц с помощью разных видов соединения: inner join, left join, right join и full join.

Сегодня мы узнаем, как создать структуру базы данных. Поймём, почему важно заранее продумывать, как она будет выглядеть, и какие связи будут между таблицами.

[00:03:55]

Структуры баз данных

И. Шафигуллин: на прошлом занятии мы работали с телефонным справочником. У нас было три таблицы: «Люди», «Телефоны» и «Адреса». Мы смотрели на них как на таблицы, заполненные данными. А можно ли смотреть по-другому? Можно ли представить их так, чтобы было удобнее взглянуть на их структуру?

В этом нам поможет Александр. То, что мы делали, он приведёт в структурный вид, более удобный с точки зрения анализа. А затем мы перейдём к конкретным задачам.

А. Сагун: это занятие — финальное во вводной части. После него вы должны определиться, кем хотите быть: разработчиком, тестировщиком, менеджером продуктов, менеджером проектов или аналитиком. Но проектирование структуры баз данных будет полезно для всех этих профессий.

Если допустить ошибку при проектировании здания (например, не продумать нагрузку или этажность), со строительством будут сложности. Также и с базами данных: если сразу не

продумать структуру таблиц, их размеры и связи, переделки будут сложными и дорогими. Так что этому этапу важно уделять внимание.

[00:06:24]

Структура таблицы «Люди»

Сейчас мы рассмотрим структуру таблиц, с которой вы работали в прошлый раз: «Люди», «Телефоны» и «Адреса».

Чтобы построить структуру таблиц, выделим колонки «Поле», «Тип», «Длина» и «Комментарий»:

- **Поле** — поля из исходных таблиц. Раньше они были написаны по горизонтали, а мы разложим их по вертикали.
- **Тип** — типизация данных, которые лежат в поле.
- **Длина** — количество памяти, которое отводится под хранение поля.
- **Комментарий** — пояснение для поля.

Итак, двигаемся по полям таблицы «Люди»:

id	ФИО	Д/р	Статус
1	Иванов И.И.	2.12.90	женат
2	Иванов И.И.	9.18.01	холост
3	Петров П.П.	4.23.83	женат
4	Васильев В.В.	5.21.98	холост

Первое поле — «Id», уникальный идентификатор записей. Например, если нам нужна запись под номером 4, мы знаем, что она только одна, и понимаем, что за ней стоит.

И. Шафигуллин: в таблице «Люди» идентификаторы были для нас критически важны, потому что с их помощью мы связывали людей и телефоны, людей и адреса. Мы не добавляли идентификаторы в таблицы «Телефоны» и «Адреса», потому что нам не нужно было обращаться к конкретным записям в них, мы всегда обращались только через таблицу «Люди». Но в реальных базах данных в каждой таблице должен быть уникальный идентификатор, чтобы мы могли её с чем-то связать. Например, сейчас мы не можем связать «Телефоны» и «Адреса», потому что у них нет уникальных идентификаторов.

А. Сагун: вернёмся к структуре таблицы «Люди». Первым полем будет people id (сократим до p_id). Тип — int (integer — целочисленный). Длина — 10 символов. Комментарий — уникальный идентификатор человека.

Поле	Тип	Длина	Комментарий
p_id	int	10	уникальный идентификатор человека

Следующее поле — «ФИО». Пока я не буду разделять его на поля с фамилией, именем и отчеством, но обычно, когда проектируют структуры данных, эти три поля выделяют отдельно.

И. Шафигуллин: расскажи, для чего это нужно.

А. Сагун: например, чтобы найти, сколько в вашем телефонном справочнике Ивановых. Если мы выделим фамилию в одну колонку, а имя и отчество — в две другие, сможем отобрать всех Ивановых по полю «Фамилия». Если мы этого не сделаем, задача будет выполнима на языке SQL-запросов, но понадобятся дополнительные операторы, которые будут искать маски. Это не очень правильно, база данных дополнительно нагружается.

И. Шафигуллин: я правильно понимаю, что мы говорим про разные сценарии использования? Если мы хотим заложить возможность поиска, фильтрации или выборки данных отдельно по фамилии, имени или отчеству, то нам нужно разделить эти данные изначально. Сначала нам может показаться, что это никому не нужно, но если появится такая необходимость, переделать таблицу будет сложнее, чем построить её так изначально. При этом такое решение не сделает таблицу более тяжёлой, потому что количество символов будет тем же.

А. Сагун: верно. Но пока мы не будем переделывать таблицу «Люди» и возьмём поле «ФИО». Тип — текстовый, под него заложим 100 символов.

Поле	Тип	Длина	Комментарий
p_id	int	10	уникальный идентификатор человека
FIO	text	100	ФИО

Следующее поле — «День рождения». Для хранения такой информации в базах данных есть отдельный тип — date. Приложения умеют по-разному показывать дату в зависимости от страны: например, в Америке сначала показывают месяц, потом день и год, а в России — день, месяц и год. Тип date занимает 8 символов (8 байт). Комментарий — дата рождения.

Поле	Тип	Длина	Комментарий
p_id	int	10	уникальный идентификатор человека
FIO	text	100	ФИО
Birth_date	date	8	Дата рождения

И. Шафигуллин: на этом поле можно проиллюстрировать то, о чём мы говорили раньше. Представим, что у нас нет отдельной системы по управлению базами данных (СУБД), и мы работаем просто в Excel. Например, нам нужно найти людей, родившихся в 90-х. Если у нас данные записаны в одной ячейке через точку, нам будет тяжело сделать выборку. Придётся писать отдельную программу, которая разобьёт дату: убедится, что всё написано в нужном порядке, вычленит год и будет выводить данные.

Если же мы воспользуемся принципом, о котором Александр говорил применительно к ФИО, и раздельно пропишем год рождения, месяц и дату, легко сможем сделать выборку и найти людей, которые родились в 90-е, в феврале или в какое-то определённое число.

Но когда мы говорим про СУБД, нет необходимости проводить эти операции с датой: программа уже умеет работать с типом date и позволяет выполнять с ней операции.

А. Сагун: да, тип date позволяет использовать операторы неравенства (больше/меньше). Можно задать условия выборки: дата рождения человека больше 01.01.1990, но меньше 01.01.2000. В этом диапазоне мы сможем найти всех людей, рождённых в 90-е. Но в Excel для фильтрации придётся делить дату на год, месяц и день.

Продолжаем разбирать поля. Следующее — «Статус».

Поле	Тип	Длина	Комментарий
p_id	int	10	уникальный идентификатор человека
FIO	text	100	ФИО
Birth_date	date	8	Дата рождения
Status	text	10	Семейное положение

Структура таблицы «Люди»

В таблице «Люди» семейное положение написано текстом — женат/холост. Можно ещё добавить статус «разведён». Но такие значения лучше не хранить в тексте. Можно сделать отдельный справочник «Статус», в котором будут идентификаторы: id 1 — холост, id 2 — женат / замужем, а id 3 — разведён. А затем связать его с базой данных по полю Status_id. Но мы сейчас не будем этим заниматься.

И. Шафигуллин: какие проблемы могут быть из-за записи, которой мы сейчас пользуемся? Если с этой таблицей будет работать человек, ему будет не принципиально, написано «холост» или «Холост», есть ли опечатки, пробел или запятая в конце записи. Но если человек пишет программу, которая будет работать с этими данными, или использует СУБД, важно пользоваться тем решением, о котором говорил Александр. Из-за опечаток и других неточностей программа будет работать неправильно: либо пропускать запись, либо ломаться на её месте.

Все возможные состояния нужно вносить в отдельную таблицу (как мы вынесли телефоны и адреса). Если количество статусов увеличится, можно будет просто расширить таблицу со статусами.

А. Сагун: задача разработчика, который проектирует БД, — снизить число ошибок из-за человеческого фактора.

Например, оператор в МФЦ может вбивать семейное положение человека вручную, и допускать в нём ошибки. Но возможен и другой сценарий: разработчик создал справочник, и в интерфейс программы, с которой работает оператор, добавил возможность выбирать семейное положение из нескольких вариантов. Например, из выпадающего списка или из радиобаттонов:

Оператору проще, а в БД значительно снижается количество ошибок.

И. Шафигуллин: но всё это возможно, если мы вынесем статус в отдельную таблицу.

Вернёмся к таблице, которую мы составляли. В ней мы представили структуру таблицы «Люди». В таком виде она хороша для анализа структуры БД и для ТЗ человеку, который будет составлять базу. Здесь есть столбцы (атрибуты), есть типы столбцов, длина, и комментарий, чтобы человек, составляющий БД, понимал, что вы хотели. Если вы будете проджектом или продакт-менеджером, с такими таблицами вы будете работать часто.

[00:15:30]

Структура таблицы «Телефоны»

А. Сагун: теперь составим структуру таблицы «Телефоны». Добавлю id телефона (t_id), сделаю его целочисленным, длина — 10, комментарий — «уникальный идентификатор телефона».

Поле	Тип	Длина	Комментарий
t_id	int	10	уникальный идентификатор телефона

И. Шафигуллин: давай добавим id и в нашу таблицу «Телефоны», чтобы не было путаницы.

t_id	Чей телефон	Телефон	Комментарий
1	1	123	личный
2	1	124	рабочий
3	1	125	для поездок
4	1	126	интернет
5	1	127	старый
6	2	527	личный
7	3	234	личный
8	3	235	рабочий
9	4	456	личный

Таблица «Телефоны» с новой колонкой — t_id

А. Сагун: мы идём от простого к сложному. Теперь у нас есть идентификатор и столбец «Чей телефон» — ссылка, которая ведёт на человека из таблицы «Люди». Новое поле я назову не «Чей телефон», а «p_id». Оно целочисленное (int), длина — 10. Комментарий — «уникальный идентификатор человека».

Поле	Тип	Длина	Комментарий
t_id	int	10	уникальный идентификатор телефона
p_id	int	10	уникальный идентификатор человека

И. Шафигуллин: поменяем нашу запись «Чей телефон» в таблице на «p_id». Перейдём с человеческого языка на технический язык системы, которую будем создавать:

t_id	p_id	Телефон	Комментарий
1	1	123	личный
2	1	124	рабочий
3	1	125	для поездок
4	1	126	интернет
5	1	127	старый
6	2	527	личный
7	3	234	личный
8	3	235	рабочий
9	4	456	личный

Таблица «Телефоны» с новой колонкой — p_id

А. Сагун: следующее поле — телефон. Чтобы было проще, Ильнар указал трёхзначные телефоны. Так мог бы выглядеть телефон сотрудника в корпоративной базе данных. Нормальные телефоны в международном стандарте состоят из трёх частей:

- **Код страны** — показывает, к какой стране относится номер телефона: +7 в России, +375 в Белоруссии, +1 в Америке.
- **Код оператора** — 3 символа по международному стандарту: например, 926, 915 или 495.
- **Номер** — от 4 до 7 символов. Количество знаков зависит от размера номерной базы. Например, для населённого пункта, где меньше 10 000 абонентов, достаточно четырёхзначных номеров. Но по всей России номера семизначные. В Китае или Индии может быть больше знаков, потому что там население стремительно увеличивается.

У меня чешутся руки разделить поле «Телефон» на эти 3 части.

И. Шафигуллин: для чего нам это будет нужно?

А. Сагун: чтобы найти в нашей базе людей с номерами из Америки или из любой другой страны. Но лучше пока не будем усложнять таблицу, оставим как есть. А вот в задачах, которые будем решать дальше, так и сделаем.

Что ещё может быть в телефоне? Знак +. Соответственно, мы уже не можем сказать, что тип данных целочисленный и возьмём текстовый. А какую указать длину? Если я укажу 100, система зарезервирует 100 байт. Если вы внесёте информацию всего на 1-2 байта, СУБД всё равно оставит под неё 100 байт. Чтобы лишнее пространство не съедалось, нужно выбирать оптимальную длину поля. Посчитаем, сколько нам нужно для телефона:

- код страны — до 4-х символов (например, +375),
- код оператора — до 4-х символов,
- номер — 7 символов.

Получается, нужно 15 символов, но чтобы не скупиться, добавим ещё 1 и зарезервируем 16 байт. Все люди в IT любят кратное 8.

Поле	**Тип**	**Длина**	**Комментарий**
t_id	int	10	уникальный идентификатор телефона
p_id	int	10	уникальный идентификатор человека
tel	text	16	номер телефона

И. Шафигуллин: не надо делать большой запас памяти, это перекликается с тем, что мы разбирали по массивам на первых лекциях. Тогда мы приводили аналогию, что массив — это гардероб в театре или парковка в торговом центре. Не нужно делать гардероб на 200 мест, если театр вмещает 50 человек. Здесь то же самое — не нужно перерасходовать память.

При этом важно продумать, какой минимум информации нужно заложить. Я бы заложил на код страны два знака, потому что вспомнил Россию (+7) и Америку (+1), но не подумал про Белоруссию (+375) и Британию (+44). Для номеров из этих стран в моей базе не хватило бы места.

Старайтесь обращать внимание на такие мелочи, чтобы не пришлось переделывать готовую базу данных из-за нескольких забытых символов.

А. Сагун: последнее наше поле — комментарий. Назовём его comment. Тип данных — текст. Длина — 100 символов, думаю, этого будет достаточно.

Поле	**Тип**	**Длина**	**Комментарий**
t_id	int	10	уникальный идентификатор телефона
p_id	int	10	уникальный идентификатор человека
tel	text	16	номер телефона
comment	text	100	комментарий

Структура таблицы «Телефоны»

Готово. Получилась вторая таблица, описывающая логическую структуру. Мы чуть-чуть затронули физическую структуру: то, что касается типизации полей и их длины. При этом

подумали, сколько памяти нужно резервировать, чтобы не съедать лишнее, и предусмотрительно заложили небольшой запас. Стараемся подходить профессионально.

[00:34:52]

Структура таблицы «Адреса»

Следующая таблица, для которой нужно описать структуру, — «Адреса». Здесь есть поле «Чей адрес». Мы уже понимаем, что оно относится к людям, поэтому переименуем его в p_id. Это поле мы завели в самом начале лекции, а теперь оперируем им.

p_id	Адрес	Комментарий
1	Можга	Место рождения
1	Казань	По прописке
1	Москва	Рабочий
2	СПб	По прописке
3	Москва	По прописке
4	Белгород	По прописке

Таблица «Адреса» с новой колонкой — p_id

У адреса, как и у телефона, есть определённая структура хранения информации. Мы знаем, что в него входит страна, область (субъекты РФ: республика, край, область, крупный город), затем город. Может произойти задвоение: Москва — субъект РФ и крупный город. Если правильно записывать адрес, придётся написать про Москву дважды.

Внутри города есть улица, дом, строение (или корпус). Внутри здания есть офисы или квартиры (какие-то пронумерованные помещения). У каждой квартиры в нашей стране есть уникальный номер за счёт этой адресной структуры.

Если вы будете соприкасаться с разработкой приложений, которые работают с российскими адресами, вам пригодится система КЛАДР. Она работает так: сначала вы вводите субъект, например, Краснодарский край. Дальше, когда будете писать город, система станет предлагать вам не все города России, а только те, которые входят в Краснодарский край. Потом вы будете выбирать из улиц этого города и так далее.

И. Шафигуллин: вернёмся к таблице со структурой. Думаю, вы уже поняли, как это работает, поэтому я попрошу вас доделать её самостоятельно. Подумайте, какие нужны поля, какими будут тип и длина, какой комментарий. А затем посмотрите, что получилось у нас.

А. Сагун: здесь всё по аналогии: уникальный идентификатор адреса — a_id, «Чей адрес» заменили на p_id (связь с таблицей «Люди»), «Адрес» переименовали в «Город» (city), потому что речь всё-такие идёт про город, и добавили комментарии.

Поле	Тип	Длина	Комментарий
a_id	int	10	уникальный идентификатор адреса
p_id	int	10	уникальный идентификатор человека
city	text	100	адрес
comment	text	100	комментарий

Структура таблицы «Адреса»

И. Шафигуллин: как и семейное положение, города лучше вынести в отдельный справочник, чтобы их написания не разнились и не было опечаток. Для этого нужно будет сделать таблицу-справочник с городами, а в «Адреса» добавить связь.

А. Сагун: и поля «Комментарий» из таблиц «Адреса», и «Телефоны» мы на самом деле использовали для обозначения типов.

У одного человека может быть несколько телефонов: домашний, рабочий, второй домашний и второй рабочий. Всё это — типизация телефонов, с ней тоже делают справочники.

То же с адресами — у человека их может быть несколько: регистрации, фактический, рабочий (офисный). У офиса может быть адрес фактического нахождения и юридический. А ещё бывает почтовый, на который отправляют корреспонденцию. Поэтому в таблице «Адреса» я бы добавил колонку «Тип адреса», сделал бы справочник с типами адресов и связал бы их. Аналогично и с «Телефонами».

И. Шафигуллин: попробуйте прочувствовать разницу. Нас на прошлых занятиях это не волновало: мы писали таблицу для себя. В целом, человек может в ней разобраться, но системе сложнее. Если мы хотим создать систему, в которой будет мало ошибок из-за человеческого фактора, лучше создавать справочники и делать связи.

А. Сагун: подведём итоги. Мы добавили идентификаторы и рассмотрели типы данных. Также есть тип для даты и времени — TimeStamp. Это «отсечка» уникального момента, когда что-то произошло (например, была создана запись для картотеки). Пояснили, почему важно думать про длину полей и использовать справочники там, где в таблице повторяются значения. Узнали, для чего в интерфейсе выпадающие списки, чекбоксы (выбор нескольких вариантов) и радиобаттоны (выбор одного варианта). Подробно поговорили о том, как дробить поля с ФИО и телефонами.

[00:50:07]

Типы связей

И. Шафигуллин: у нас есть структура каждой таблицы, и мы постоянно говорим, что они связаны между собой. Расскажи, пожалуйста, какие могут быть связи? И приведи примеры того, как разные записи в таблицах могут быть между собой связаны.

А. Сагун: выделяют три типа связи.

1. **Один к одному.** Простой пример — у ребёнка может быть только одна мама.
2. **Один ко многим.** Мужчина в течение жизни может быть женат на нескольких женщинах. Либо женщина может быть замужем за несколькими мужчинами. Если продолжать пример с ребёнком, у него может быть одна мама, но у мамы — двое-трое детей. У семьи из трёх человек может быть один адрес регистрации. Правильная структура хранения предполагает отдельную таблицу, где будет идентификатор мужа и адрес, идентификатор жены и адрес, а также идентификатор ребёнка и тот же адрес.
3. **Многие ко многим**. У мужа из прошлого примера может быть несколько адресов. Тогда в структуру хранения добавится ещё один его идентификатор и ещё один адрес. Получится, что адрес может относиться к нескольким людям, а у одного человека может быть много адресов.

И. Шафигуллин: в таблицах, с которыми мы работаем, есть гражданин Иванов с id 1. К нему относятся сразу несколько городов: место рождения, по прописке, рабочий.

То же — в обратном порядке для Москвы. Если мы посмотрим по p_id, увидим, что здесь живёт Иванов и Петров. То есть Москва относится сразу к нескольким людям. Получается, и с одной стороны «ко многим», и с другой. Связь — многие ко многим.

p_id	ФИО	Д/р	Статус
1	Иванов И.И.	2.12.90	женат
2	Иванов И.И.	9.18.01	холост
3	Петров П.П.	4.23.83	женат
4	Васильев В.В.	5.21.98	холост

Таблица «Люди»

p_id	Адрес	Комментарий
1	Можга	Место рождения
1	Казань	По прописке
1	Москва	Рабочий
2	СПб	По прописке
3	Москва	По прописке
4	Белгород	По прописке

Таблица «Адреса»

Если бы мы выделили таблицу со статусами (холост, женат, разведён), какое отношение было бы между людьми и их семейным положением?

А. Сагун: думаю, вы уже догадались, что правильный ответ — многие ко многим. Потому что в течение жизни семейное положение человека меняется: сперва он холост, потом может быть женат, потом — разведён. И если мы проектируем приложение, которое будет работать долго, мы должны учитывать, что ситуация может меняться, а новые записи — появляться.

При этом статусы относятся к большому количеству людей, потому что все они либо холосты, либо женаты (замужем), либо разведены. Получается связь многие ко многим.

И. Шафигуллин: посмотрим на примере наших таблиц. У нас есть Иванов, и мы можем указать для него только один статус. Казалось бы, связь — один ко многим (человек один, а статусов много). Но если мы будем хранить не только текущее состояние, а записывать, что случилось с Ивановым, Петровым и Васильевым, их статус может измениться: был холост, затем женился. Или наоборот: женился и развёлся. Получается соотношение многое ко многому.

А. Сагун: для правильного хранения, в таблицу «Люди» я бы добавил поле «Дата изменения статуса». Например, если мы заводим в таблицу информацию о человеке, в день его рождения он будет холост. Когда он получит свидетельство о браке, мы изменим его статус и добавим дату, в которую он женился. Записи «холост» и «женат» будут относиться к одному человеку, но будут отличаться по дате. Вы не говорили про уникальные ключи?

И. Шафигуллин: говорили, что у нас есть первичный ключ, вводили его как идентификатор. И говорили, что, если мы используем ключ в другой таблице, он будет внешним, связывающим таблицы.

А. Сагун: если в таблицу «Люди» мы добавим для одного человека две строки выбора, по-хорошему нужно сделать и другую таблицу, где ключ будет составным. Либо вы делаете уникальный идентификатор для этих записей (для каждой записи при изменении семейного положения будет создаваться уникальный номер), либо надо будет объединить ключи идентификатор человека и дату, в которую происходит изменение статуса. Это будет составной ключ.

И. Шафигуллин: но мы уже забегаем вперёд. Если пока картина не выстраивается, не переживайте. В качестве упражнения попробуйте подумать, как связаны между собой «Люди» и «Телефоны», «Люди» и «Адреса»?

А затем перейдём к разбору реального более сложного кейса.

[00:59:22]

Кейс «Аренда автомобилей»

А. Сагун: для семинара мы подготовили два кейса из реальной жизни. У меня есть проект, связанный с арендой автомобилей, и на его примере я покажу кусочек программы, в которой хранится информация. На её основе обсудим, как описать структуру хранения данных, какие должны быть связи между таблицами, чего нам хватает и не хватает. Рабочий файл с этим кейсом будет приложен к лекции. Чтобы у нас был плавный переход, возьму структуру записи, похожую на предыдущий пример. Работать мы будем с тремя таблицами:

1. **Транспортное средство** — то, что сдаётся в аренду.

Car_ID	Name	VIN	Create_date	Manufact_date	Brand	Model	V_volume
1	T0001	ZVJDJDJ838823jsd	20.12.2020 18:34:07	11/20/20	Kia	K5	2
2	N0002	ZJDHJSDH88HH888	01.11.2020 11:11:11	11/1/20	Mercedes-Benz	E220d	2,2
5	H0089	ZDJDJDSKJ898JKKK	05.08.2020 10:45:23	11/11/70	Lada	Kopeika	1

2. **Клиенты** — люди, которые берут транспортное средство (ТС) в аренду.

Client_ID	First_name	Family_name	Middle_name	Birth_date	Passport_ser	Passport_num	Sex
8	Александр	Иванов	Петрович	1/19/00	2323	343434	M
10	Анна	Скуратова	Владимировна	6/1/81	4343	354345	F
20	Николай	Пряников	Оскарович	12/31/70	8988	312321	M

3. **Сделка** — товарно-денежные отношения. Здесь мы собираем информацию о том, кто и когда взял ТС, сколько за это заплатил.

Deal_ID	Description	Manager	Create_date	Amount	Currency	Start_date	End_date	Status	Car_id	Client_id
80	Первая	Костин	09.12.2021	10000	RUB	12/9/20	12/9/22	1	1	8
81	Вторая	Пермин	09.12.2021	50000	RUB	12/9/20	12/9/23	2	2	10
82	Третья	Пермин	09.12.2021	50000	RUB	12/9/21	12/9/23	1	2	20

От такой записи мы переходим к описанию структуры таблиц с их логической и физической моделью. Начинаем с того, что у нас есть транспортное средство и клиент.

Параметр	Ключ	Тип	Длина	Комментарий
Car_ID	Да	Int	10	Уникальный идентификатор ТС
Name		Text	10	Позывной ТС
VIN		Text	16	VIN номер ТС
Create_date		TimeStamp	12	Дата создания записи
Manufact_date		Date	8	Дата производства ТС
Brand		Text	30	Бренд ТС
Model		Text	20	Модель ТС
V_volume		Real	10	Объем двигателя
Client_id		Int	10	Уникальный идентификатор клиента

Структура таблицы «Транспортное средство»

Параметр	Тип	Длина	Комментарий
Client_ID	Int	10	Уникальный идентификатор Клиента
First_name	Text	30	Имя клиента
Family_name	Text	30	Фамилия
Middle_name	Text	30	Отчетсво
Birth_date	Date	8	Дата рождения
Passport_ser	Text	10	Серия паспорта
Passport_num	Text	20	Номер паспорта
Sex	Text	1	Пол

Структура таблицы «Клиент»

Напрашивается, что к «Транспортному средству» надо добавить поле «Client_id», которое есть в таблице «Клиент» и является уникальным идентификатором. Таблицы будут связаны по этому полю.

Напрашивается и колонка ключевого поля — «Ключ». В ней будут признаки «да» и «нет». Например, для «Car_ID» ключ будет «да» — это уникальный идентификатор, который обозначает, что с одним номером есть только одна машина.

И. Шафигуллин: давай сделаем шаг назад. У нас была задача: сформировать базу данных так, чтобы наш бизнес в сфере аренды автомобилей работал. Пока у тебя появились две таблицы: с клиентами и с транспортными средствами. Почему ты выделил именно их? Какие параметры в них ты решил выбрать?

А. Сагун: чтобы автоматизировать или писать какое-то приложение для бизнеса, нужно понять, что в нём есть, как он функционирует. Как правило, в любой деятельности при анализе выделяют объекты, которыми этот бизнес оперирует. В арендном бизнесе это транспортные средства: машины, велосипеды, самокаты и так далее. Для двух последних я не буду заполнять VIN-номер, который есть у автомобилей в ПТС, а укажу, например, серийный номер (для этого добавлю в таблицу со структурой ещё одно поле). А чтобы разделять транспортные средства, сделаю справочник. Это взгляд на объект.

Второй объект (иногда их называют сущностями) — это клиент, человек, который хочет воспользоваться транспортным средством. У клиента тоже есть набор параметров. Я взял минимальный. В реальном бизнесе в карточке клиента хранятся визуальные образы паспорта и прав, информация о водительском стаже и так далее. Но пока мы возьмём небольшой набор полей, которого достаточно, чтобы идентифицировать конкретного клиента.

Есть транспортное средство, есть клиент, они связаны по полю «Client_id». Когда клиент берёт в аренду машину, в «Транспортном средстве» записывается его идентификатор. Каршеринговый бизнес устроен так: сегодня, например, машину взял Ильнар, заплатил, покатался, поставил и заблокировал. Затем машину взял следующий человек. Получается, в этом поле мне постоянно нужно перезаписывать идентификатор клиента.

Перед тем как взять машину в каршеринге, важно её осмотреть, и, если есть повреждение, сообщить о нём поддержке. В свою очередь, менеджеру поддержки нужно будет выяснить, кто ездил на машине раньше. Если в нашей истории идентификаторы бывших клиентов уже стёрлись, мы сами виноваты и будем платить за повреждение из своего кошелька. Нужно было сделать структуру, из которой будет понятно, что какой-то набор клиентов в какой-то период времени ездил на этом автомобиле. Здесь напрашивается третья сущность — сделки.

И. Шафигуллин: к ней пока не переходи. Думаю, не всем пока привычно смотреть на это как на таблицу с данными. В столбце «Параметры» перечислены атрибуты нашей базы данных. Например, если мы возьмём какую-то машину, у неё будет:

Параметр
Car_ID
Name
VIN
Create_date
Manufact_date
Brand
Model
V_volume
Client_id

- идентификатор в бизнесе,
- имя,
- VIN-код,
- дополнительные данные,
- **идентификатор человека**, который сейчас ездит на этой машине.

Соответственно, если первый клиент вернул машину, и на ней собирается ездить следующий, что мы можем сделать в рамках структуры? Мы можем это поле в конкретной записи переписать: заменить Client_id первого клиента на Client_id второго. Но из-за этого мы потеряем всю историю.

[01:08:27]

Связующая таблица «Сделка»

А. Сагун: а решить это можно, добавив ещё одну таблицу. Мы берём сущность «Сделка» и добавляем её между «Транспортным средством» и «Клиентами».

Параметр	Тип	Длина	Комментарий
Deal_ID	Int	10	Уникальный идентификатор сделки
Description	Text	10	Описание сделки
Manager	Text	16	Сотрдуник создавший
Create_date	TimeStamp	12	Дата создания записи
Amount	Real	8	Сумма сделки
Currency	Text	30	Валюта сделки
Start_date	TimeStamp	12	Дата начала сделки
End_date	TimeStamp	12	Дата окончания сделки
Status	int	10	Статус сделки
Car_id	Int	10	Уникальный идентификатор ТС
Client_id	Int	10	Уникальный идентификатор клиента

Структура таблицы «Сделка»

У сделки есть:

- **Deal_ID** — уникальный идентификатор.
- **Description** — описание.
- **Manager** — создавший сделку сотрудник.
- **Create_date** — дата создания с типом данных TimeStamp, потому что здесь важен точный момент времени. Например, если каршеринговая машина нарушит ПДД, и штраф отправят в компанию, нужно будет найти клиента, который арендовал машину в момент нарушения.
- **Amount** — сумма сделки с типом данных real (числа со знаками после запятой). Это важно, потому что сделка может быть с копейками.
- **Currency** — валюта сделки с типом данных text. Если мы захотим использовать наше приложение не только в России, где всё в рублях, а, например, в Казахстане, мы столкнёмся с другой валютой. Если в этом поле будет стоять 1000 — одна в тенге, а другая в рублях — будет существенная разница из-за курса обмена.
- **Start_date** и **End_date** — время начала и окончания сделки. Здесь тоже подойдёт TimeStamp с длиной 12 символов.
- **Status** — статус сделки. Здесь будут идентификаторы с отдельным справочником.
- **Car_id** и **Client_id** — уникальный идентификатор ТС и клиента. В таблицах «Транспортные средства» и «Клиенты» они являются уникальными ключами, а в «Сделке» — внешними, потому что есть внешняя связь.

Чтобы правильно настроить связи в таблицах:

1. В таблице «Транспортные средства» убираем последнюю запись (Client_id / Int / 10 Уникальный идентификатор клиента). Получается, что связь по ней больше невозможна.

2. Связываем таблицу «Клиенты» и «Сделка» по полю Client_id.

3. Связываем таблицу «Транспортные средства» и «Сделка» по полю Car_id.

Теперь нам будет понятно, в какой сделке какой клиент арендовал какой автомобиль.

И. Шафигуллин: может показаться странным, что мы выделяем отдельные таблицы. Почему бы не сделать всё в одной? И можно ли обойтись без сделок? Технически, да, но пришлось бы делать много похожих записей: у нас была одна машина, мы её сперва отдали одному клиенту, а потом другому. Но это плохая практика — добавлять много записей с практически дублирующейся информацией.

Лучше хранить в таблице уникальную информацию. Например, в «Транспортных средствах» мы храним информацию о бренде, модели, объёме двигателя и других уникальных параметрах машины. Так и в «Клиентах»: есть номер и серия паспорта, дата рождения клиента — они не будут меняться, и их не придётся дублировать.

Получается, мы выделили сущность «Транспортные средства» и добавили туда необходимые данные. Выделили сущность «Клиент» и тоже добавили необходимые данные. По логике, нам нужно выделить ещё одну сущность, которая будет связывать клиентов и транспортные средства — «Сделки». В ней будет храниться необходимые данные о сделках, информация не будет дублироваться.

[01:20:55]

Сценарий использования

И. Шафигуллин: давай посмотрим на конкретный сценарий использования. Ты приводил хороший пример со штрафом. Как это работает?

А. Сагун: мы получаем уведомление о штрафе. В договоре прописано, что, если водитель на нашем транспортном средстве нарушает ПДД, он несёт ответственность и компенсирует ту сумму штрафа, которую мы оплачиваем.

В штрафе указан госномер, а не VIN-номер, бренд или модель машины. Отсюда вопрос: как нам по нашей структуре данных определить, к какой машине из автопарка относится штраф? Какое добавить поле с каким типом данных, длиной и комментарием?

И. Шафигуллин: без нового поля точно не обойтись. Из штрафа мы знаем время и место нарушения, а также госномер машины, на которой оно было совершено. Сопоставив эти данные с нашими таблицами, мы бы нашли тысячу машин, сданных в аренду в это время, и не отыскали бы нужную.

А. Сагун: нам нужно добавить поле с государственным регистрационным номером. Назовём его RegNum. Оно точно не будет ключевым. В нём будут храниться буквы и цифры: в российских номерах 3 цифры, 2 буквы и 3 символа кода региона; в британских и американских номерах по-другому, в арабских вообще может быть вязь. Символы очень разнообразные, поэтому тип данных лучше оставить текстовым. Количество символов в российских номерах — 7-8, но со временем оно может увеличиться, поэтому сделаем небольшой запас и укажем длину 16 символов.

Параметр	Ключ	Тип	Длина	Комментарий
Car_ID	Да	Int	10	Уникальный идентификатор ТС
Name		Text	10	Позывной ТС
VIN		Text	16	VIN номер ТС
Create_date		TimeStamp	12	Дата создания записи
Manufact_date		Date	8	Дата производства ТС
Brand		Text	30	Бренд ТС
Model		Text	20	Модель ТС
V_volume		Real	10	Объем двигателя
Reg_num		Text	16	Гос номер

Что происходит теперь? Мы ищем таблицу с госномером, который указан в штрафе, находим и идём в таблицу «Сделки». По этому идентификатору машины находим все сделки, в которых участвовал автомобиль, и ищем нужную по времени штрафа. Ограничиваем время сделки по Start_date и End_date: указываем, что время начала сделки меньше, чем время штрафа, а время окончания сделки больше, чем время штрафа.

У нас остаётся одна сделка, в которой мы видим идентификатор клиента. Отправляем ему информацию о нарушении на телефон или e-mail, говорим, что оплатили штраф и просим его компенсировать. Выставляем счёт, а он его оплачивает.

[01:31:07]

Как проектировать структуры таблиц?

И. Шафигуллин: здорово, спасибо! Теперь расскажи о том, как обновлять таблицы. Например, сперва нам казалось, что информации достаточно, а потом стало ясно, что не хватает госномера, телефона, электронной почты. Как быть? Как сделать так, чтобы таблицу не пришлось постоянно переделывать?

А. Сагун: то, что я сейчас скажу, будет особенно полезно для будущих проджект-менеджеров. Вы будете работать внутри проекта или будете приходить на проект как подрядчик. И вашей задачей будет собрать информацию о том, как работает бизнес. То есть провести бизнес-анализ.

Ваша задача — пообщаться с ключевыми сотрудниками и понять, какие бывают кейсы на всём пути следования клиента: откуда он берётся, какая информация о нём важна бизнесу. Номер телефона, электронная почта, данные паспорта, адрес — всё это нужно, чтобы проверить клиента, убедиться, что он не мошенник, который угонит автомобиль.

Что происходит дальше? Мы заключаем с клиентом сделку. На этом этапе необходимо понять, как он выбирает автомобиль. Например, клиент может позвонить и сказать: «Я хочу белый Mercedes с красным салоном». И у менеджера должна быть возможность посмотреть в информационной системе, есть ли подходящая машина. Или клиент может запросить машину с колёсами 21 радиуса. Надо будет добавлять поля в описание машин на основании того, как клиенты их выбирают.

Людям, которые закупают и обслуживают машины, тоже нужно знать определённую информацию. Например, пробег автомобиля, потому что от него зависит прохождение

техобслуживания, замена масла, расходных материалов. Получается, нужно будет поговорить с людьми, которые закупают и обслуживают автомобили, узнать у них, на что важно обратить внимание.

Вообще, что такое автомобиль, чем он описывается? Кроме VIN-номера и регистрационного номера есть свидетельство транспортного средства (СТС). У этого документа тоже есть идентификатор, и его тоже нужно сохранять. Он всегда уникальный для каждого автомобиля, у него есть срок начала и окончания действия, другие нюансы. В нашей базе данных его тоже не хватает. Не указан цвет. Прописан объём двигателя, но непонятно, на каком топливе ездит автомобиль: бензин, дизель, газ? Тип двигателя тоже можно было бы реализовать в справочнике.

Также нужно добавить информацию о коробке передач (она может быть механической и автоматической). Не все умеют ездить на механике, а аренда автомата может быть дороже, поэтому важно оставить клиенту выбор, показать, что сколько стоит.

Нужна информация о стоимости суток аренды, о тарификации. Менеджеру она поможет посчитать сумму сделки. Например, аренда автомобиля на день может стоить 5000 рублей. При аренде на 90 дней сутки будут стоить 3000 рублей. Так что тарификация — отдельная сущность, которая должна быть у менеджера.

Не буду усложнять дальше, подведу итог. Когда вы начинаете решать такую задачу, важно выделить сущности. В нашем случае это автомобиль, клиент, менеджер, сделка, ценовые условия, документы на автомобиль (ПТС, СТС, номер автомобиля, фотография, состояние, в котором он выдан и принят обратно), стоянка хранения, офисы, в которых выдаются автомобили.

И. Шафигуллин: и надо указывать местоположение, где он сейчас находится.

А. Сагун: да, например, человек хочет взять его в аэропорту. Если мы договорились, что привезём автомобиль в аэропорт, в сделке должно быть место выдачи и место сдачи. Возможно, клиент захочет взять автомобиль в Москве и сдать его в Санкт-Петербурге. Соответственно, нужно будет передать информацию петербургскому офису о том, когда и в какой точке нужно будет забрать автомобиль, осмотреть его и загрузить в базу документы о том, что с ним всё в порядке.

И. Шафигуллин: вы должны выделить сущности и понять процесс. Что происходит от момента коммуникации клиента с компанией до завершения этой коммуникации? Какой путь он проходит, какая информация его сопровождает, что нужно, чтобы решить бизнес-кейсы, которые возникают при аренде автомобилей?

А. Сагун: вы должны стать человеком, который знает, как работает всё в компании. Я сам прошёл этот путь, когда занимался автоматизацией бизнеса. Иногда происходили парадоксальные вещи: ко мне, IT-сотруднику, приходили коммерческий директор и директор по логистике, просили помочь разобраться с какой-то ситуацией. Я говорил, что автоматизирую бизнес, а не управляю им. Они обращались ко мне, потому что я знал, как работают все подразделения.

И. Шафигуллин: пример со штрафами получился очень показательным. Мы посмотрели на сценарий использования базы данных, поняли, чего нам не хватает, и что нужно скорректировать. Здорово, спасибо.

[01:40:02]

Заключение

И. Шафигуллин: если что-то было непонятно, обязательно пересмотрите лекцию в записи. Попробуйте поиграться с таблицами, подумать, какие таблицы нужно добавить, какие связи настроить. И обязательно ломайте всё, что собираете: продумывайте жизненные кейсы и смотрите, в какой ситуации ваша база может сломаться, как её доработать. Подумайте, что лучше вынести в справочники, какие параметры нужно добавить, какие сущности могут возникнуть. Всё это поможет вам познакомиться с процессом проектирования баз данных.

А. Сагун: подведём итог. Мы взяли таблицы из предыдущей лекции и превратили их в логическую структуру с физическим описанием параметров. Разобрали примеры с типами полей, поняли, когда и что нужно дробить.

Затем мы разобрали пример с арендой автомобилей: связали сущности «Транспортное средство» и «Клиент» и поняли, почему такое решение не оптимально. Добавили сущность «Сделка», сделали её связующей, чтобы новый функционал был более полезен для бизнеса.

Система, над которой мы работали, — сердце бизнеса. Если она будет недоступна час или день, бизнес встанет. Менеджеры не смогут выдавать и принимать машины, компания начнёт терять деньги. Поэтому решения, которые мы принимаем, очень важны. От них зависит стабильная, правильная, оптимальная работа бизнеса.

Надеюсь, лекция была понятной и полезной, и вам захотелось подумать над другими сценариями. В следующий раз, когда будете брать каршеринг, или заказывать такси, подумайте, как бы вы описали в базе данных новую ситуацию, связанную с ними. Когда будете пользоваться сервисом доставки еды или справочником адресов, подумайте, что за ними стоит? Как хранится информация? Как бы вы её организовали?

И. Шафигуллин: на этом заканчиваем нашу лекцию. У нас будет ещё один бонусный разбор задачи от Александра. А на сегодня всё, увидимся на семинарах.

А. Сагун: до свидания, хорошего вам дня.