Знакомство с базами данных.

Примеры баз данных:

- картотека
- результаты
- переписи
- библиотеки
- книги учёта

Базы данных нужны для хранения, обработки и быстрого извлечения необходимой информации.

Базы данных бывают:

- Иерархические
- Реляционные

При создании **иерархической** базы данных мы изначально фиксируем сценарий использования этой базы.

Иерархическая модель удобна, если она создаётся под конкретную задачу. Например, по данной модели каталога картин удобно находить все картины российских художников или конкретный жанр картин российских художников.

Главное ограничение иерархической структуры — отсутствие гибкости. Она сразу определяет сценарий использования данных.

Реляционные базы данных (от англ. Relation – связь) — базы данных, в которых данные распределены по отдельным, но связанным между собой таблицам.

- **Первичный ключ** — ID, уникальная информация, которая позволяет идентифицировать каждую конкретную запись таблицы.
- **Внешний ключ** — ID из дополнительной таблицы (уже не уникальный). Позволяет найти информацию из основной таблицы.

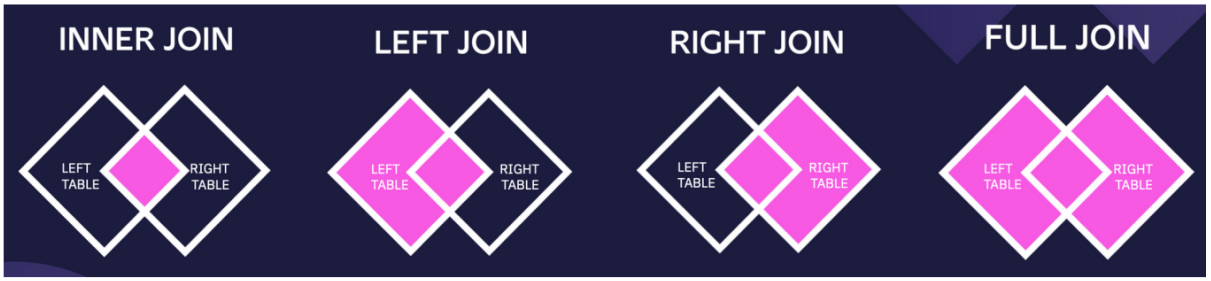
SQL – это особый язык программирования, который позволяет формулировать то, что нужно сделать с данными в таблицах.

SELECT * FROM «Общий список»

- SELECT – «выбери»;
- * – после SELECT указывается набор столбцов, * означает что мы хотим видеть все столбцы;
- FROM – «из» указываем, откуда необходимо выбрать информацию;
- Общий список – название таблицы из нашего примера.

Способы связывания таблиц между собой:

Похожи на операции со множествами:



Примеры использования INNER JOIN

JOIN — объединение, INNER — внутренний:

- общая информация из двух табличек (пересечение),
- нет «пустых» значений.

INNER JOIN Люди, Телефоны

ON id = «Чей телефон»

Сначала из разных таблиц программа собирает одну таблицу по нашему условию, из которой мы можем получать информацию.

INNER JOIN Люди, Адреса_2 ON id = «Чей адрес».

Адреса_2 — в таблице есть адрес, который не относится ни к кому из указанных в таблице людей.

Есть полное соответствие столбцов ID, «Чей адрес» и «Чей телефон»

Примеры использования LEFT JOIN

LEFT JOIN Люди, Адреса_2

ON id = «Чей адрес»

Адреса_2 — в таблице есть адрес, который не относится ни к кому из указанных в таблице людей.

Иванов И.И.	2.12.90	женат	Можга	Место рождения
Иванов И.И.	2.13.90	женат	Казань	По прописке
Иванов И.И.	2.14.90	женат	Москва	Рабочий
Иванов И.И.	9.18.01	холост	СПб	По прописке
Петров П.П.	4.23.83	женат	Москва	По прописке
Васильев В. В.	5.21.98	холост	Белгород	По прописке

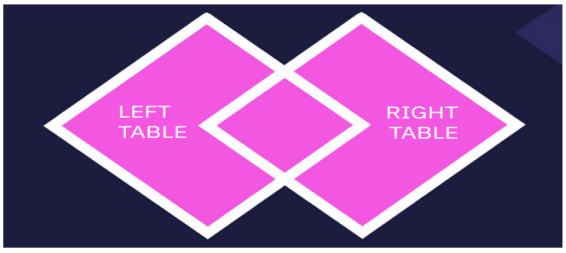
В новую таблицу переходят все данные из первой (левой) таблицы, к ним добавляются соответствующие данные из второй (правой).

INNER JOIN и LEFT JOIN вернули в данном случае одинаковый результат → у нас в левой таблице нет данных, которые выходили бы за границу пересечения.

Если мы поменяем местами таблички «Люди» и «Адреса_2», то «Адреса_2» будет слева, а «Люди» — справа.

- В левой таблице «Адреса_2» есть данные, которых нет во второй таблице. Эту запись переносим в новую таблицу. Соответствующего значения из второй таблицы нет, поэтому все ячейки с недостающими данными заполняем заглушкой «нет».
- С INNER JOIN не может быть пустых ячеек: он берёт только пересекающиеся значения.

Примеры использования FULL JOIN



Одновременное использование **LEFT JOIN** и **RIGHT JOIN**.

Берём данные, которые:

1. есть в обеих таблицах;
2. есть только в левой таблице (недостающие значения заполняем значением по умолчанию, например, «нет»);
3. есть только в правой таблице (недостающие значения заполняем значением по умолчанию, например, «нет»).

Описание структуры данных

На примере файла с прошлой лекции (телефонный справочник):

Люди

p_id	ФИО	Д/р	Статус
1	Иванов И. И.	12.02.1990	женат
2	Иванов И. И.	18.09.2001	холост
3	Петров П. П.	23.04.1983	женат
4	Васильев В. В.	21.05.1998	холост

Телефоны

t_id	p_id	Тел	Коммент
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

Адреса

a_id	p_id	Адрес	Комментарий
1	1	Можга	Место рождения
2	1	Казань	По прописке
3	1	Москва	Рабочий
4	2	Санкт-Петербург	По прописке
5	3	Москва	По прописке
6	4	Белгород	По прописке

В каждую таблицу добавлен уникальный идентификатор:

- p_id — уникальный идентификатор человека,
- t_id — уникальный идентификатор телефона,
- a_id — уникальный идентификатор адреса.

Если мы изначально хотим заложить возможность фильтрации данных по фамилии или имени, то Фамилия, Имя и Отчество должны храниться в базах данных отдельно.

Данные о статусе (семейном положении) обычно хранятся не текстом, а с помощью идентификаторов (в отдельном справочнике) → 1 — холост, 2 — женат, 3 — разведён.

Структура таблицы «Люди»:

Поле	Тип данных	Длина (в байтах)	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 «Телефоны»:

Поле	Тип данных	Длина (в байтах)	Комментарий
t_id	Int	10	Уникальный идентификатор телефона
p_id	Int	10	Уникальный идентификатор человека
tel	Text	16	Номер телефона
Comment	Text	100	Комментарий

- Добавляем внутрь таблицы уникальный идентификатор t_id.
- Номера телефонов состоят из трёх составляющих: телефонный код страны, код оператора и сам номер.
- Номер телефона представлен текстом, потому что кроме цифр мы используем знак «+».

Длина: под каждое значение резервируем одинаковое количество символов. Лучше выносить города в отдельную таблицу (справочник).

- Не надо делать большой запас и перерасходовать память.
- Лучше сразу рассчитать максимальное необходимое количество памяти для каждого типа данных и закладывать достаточное количество памяти.

Лучше выносить города в отдельную таблицу (справочник) - это избавит от проблем с опечатками: система не будет думать, понимать ли под значениями «москва» и «Москва» один город, вместо этого оператор выберет значение из списка.

Общие рекомендации по подготовке структуры хранения данных:

- Добавлять уникальные идентификаторы в каждую таблицу.
- Указывать тип данных.
- Использовать особый тип данных для даты или времени. Заранее рассчитывать длину полей.
- Использовать справочники для повторяющихся данных: семейное положение, город и других.
- Дробить информацию на несколько полей, если это соответствует вероятным будущим задачам работы с БД. Например, можно разнести значения из «ФИО», номера телефонов и т.д.

Типы связей между таблицами:

- Один к одному (Одно свидетельство о рождении соответствует одному человеку.)
- Один ко многим (Один мужчина был женат на нескольких женщинах.)
- Многое ко многому (По одному адресу зарегистрированы несколько людей, и у каждого человека, в свою очередь, может быть несколько адресов: фактический и адрес регистрации, например.)

Разбор базы данных «Аренда автомобилей»

Структура описана во вкладке «Аренда Структура записей»:

Транспортное средство

Car_ID	Name	VIN	Create_date	Manufact_date	Brand	Model	V_volume
1	T0001	ZVJDJDJ838823jsd	20.12.2020 18:34:07	20.11.2020	Kia	K5	2
2	N0002	ZJDHJSDH88HH888	01.11.2020 11:11:11	01.11.2020	Mercedes-Benz	E220d	2,2
5	H0089	ZDJDJDSKJ898JKKK	05.08.2020 10:45:23	11.11.1970	Lada	Kopeika	1

Транспортные средства - то, что сдаётся в аренду

Client_ID	First_name	Family_name	Middle_name	Birth_date	Passport_ser	Passport_num	Sex
8	Александр	Иванов	Петрович	19.01.2000	2323	343434	M
10	Анна	Скуратова	Владимировна	01.06.1981	4343	354345	F
20	Николай	Пряников	Оскарович	31.12.1970	8988	312321	M

Клиенты, которые арендуют транспортные средства

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	09.12.2020	09.12.2022	1	1	8
81	Вторая	Пермин	09.12.2021	50000	RUB	09.12.2020	09.12.2023	2	2	10
82	Третья	Пермин	09.12.2021	50000	RUB	09.12.2021	09.12.2023	1	2	20

Сделка: кто взял какое средство

Транспортное средство

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
Client_id		Int	10	Уникальный идентификатор клиента

Клиент

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

Ключ (в нашем случае, Car_ID) — уникальный идентификатор.
Для одного номера есть только одна машина.

- Одну и ту же машину в разные моменты времени могут брать в аренду разные люди. Как в этой ситуации хранить данные о клиенте:
  - Затирать данные о предыдущем клиенте новым → тогда в базе данных не будет информации о предыдущих клиентах. Это важно, если у машины обнаруживается поломка и необходимо установить, кто брал её в аренду.
  - Добавить ещё одну таблицу «Сделка».

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

В изначальной таблице было задание на поиск ошибки, а в этой таблице ошибка уже исправлена

- Тип данных TimeStamp — сочетание даты и времени.
- Поля Car_id и Client_id — внешние ключи, связывающие таблицу «Сделка» с таблицей «Транспортное средство» через Car_id и «Сделку» с «Клиентами» — через Client_id.
- Зачем делать несколько таблиц? Чтобы не дублировать одну и ту же информацию внутри одной таблицы.

Кейс: штраф

Компания, которая сдаёт машины в аренду, получает штраф за превышение скорости. Компании необходимо установить, кто из клиентов брал машину в момент, когда был выписан штраф.

1. Изначально в таблице не было поля «Регистрационный номер», и сейчас имеющейся информации недостаточно, чтобы установить, кто из клиентов превысил скорость и кто должен платить штраф.
2. По информации из штрафа мы знаем:
   - время превышения скорости,
   - номер машины,
   - место превышения скорости.

Задание: добавить в таблицу «Транспортное средство» поле (строку) с государственным номером машины.

Reg_num		Text	16 Гос номер

3. Теперь можно найти клиента, которому выписали штраф!
   - Далее из штрафа берём номер машины, находим его идентификатор, находим клиентов, которые брали машину в аренду. Далее ограничиваем время аренды: начало сделки меньше времени получения штрафа, конец сделки — больше. Таким образом находим идентификатор клиента.

Как сделать оптимальную базу данных:

- Изначально в таблице не было поля «Регистрационный номер», и сейчас имеющейся информации недостаточно, чтобы установить, кто из клиентов превысил скорость и кто должен платить штраф.
- Понять, какие бывают ситуации и кейсы.

- Понять, какая информация может понадобится в разных кейсах.
- Понять, какой путь проходит клиент от момента начала коммуникации с сервисом до завершения, какая информация сопровождает этот путь.
- На этапе создания базы данных придумывать кейсы, в которых ваша база данных может «сломаться».