

Работа с многотабличной базой данных

Многотабличная БД хранит связанные данные и объединяет их ключами и связями.



Почему это важно в школьной информатике

В реальных системах сведения относятся к разным объектам: ученик, класс, предмет, оценка, учитель и дата. Разделение по таблицам хранит каждый факт в одном месте и позволяет быстро получать сводные результаты запросами SQL.



Сущности и атрибуты в реляционной модели

- Сущность — это объект предметной области: ученик, предмет, оценка, заказ или товар. Она описывает «что» хранит база, даже если сам объект существует в реальности как запись школьного журнала или списка покупок.
- Атрибуты — это свойства сущности: фамилия, имя, дата рождения, название предмета, балл, цена. Одинаковые атрибуты задают одинаковые столбцы таблицы, чтобы каждая строка сравнивалась по одному и тому же набору характеристик.
- Обычно одна таблица описывает одну сущность. Каждая строка соответствует одному экземпляру объекта, а столбцы отражают одинаковый набор параметров для всех записей, что делает структуру предсказуемой и удобной для запросов.

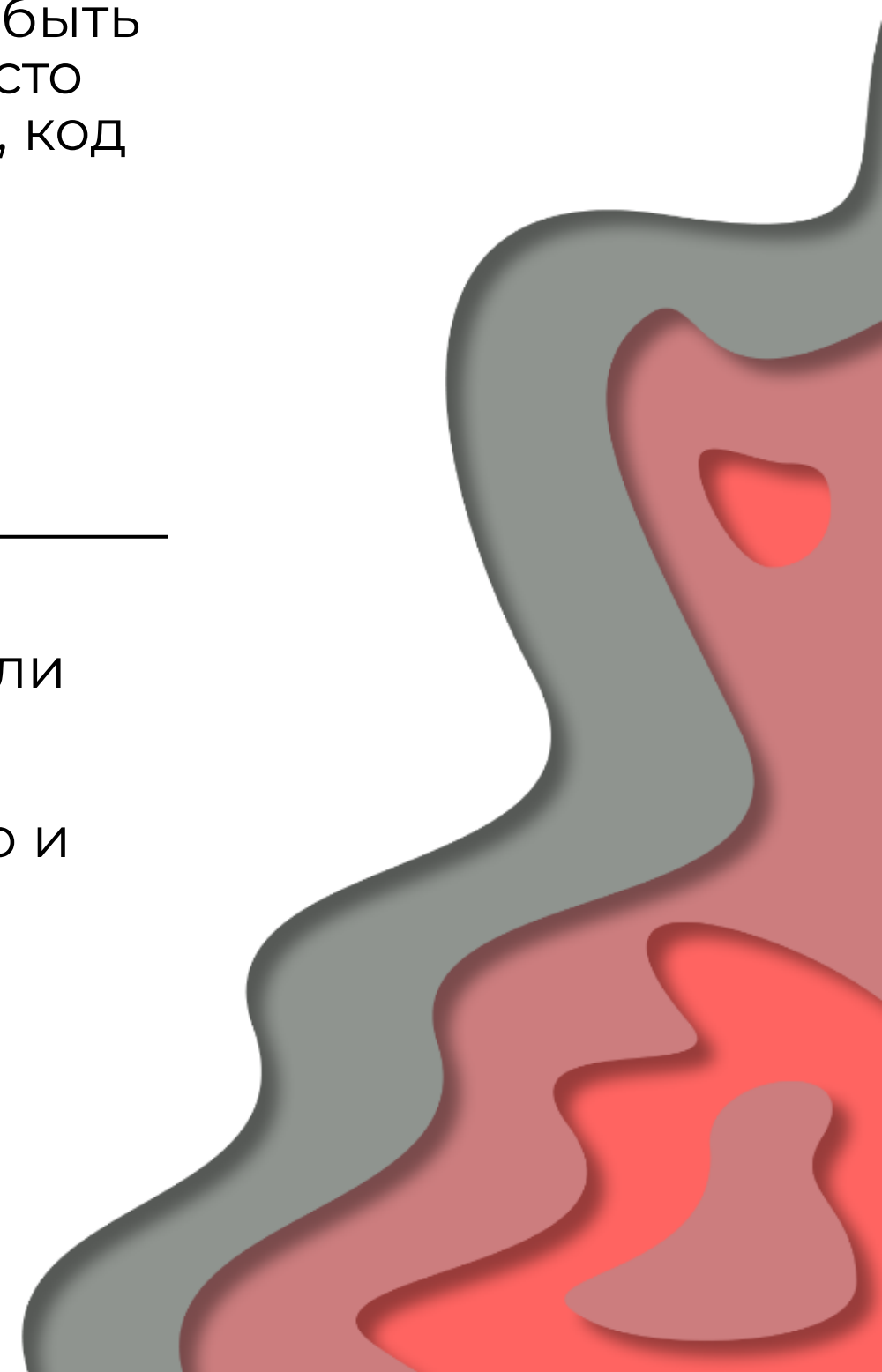
Первичный ключ: уникальный идентификатор записи

Первичный ключ однозначно определяет каждую строку таблицы. Его значения не должны повторяться, иначе база не сможет отличить две разные записи и корректно выполнить поиск или связь.

Первичный ключ также не бывает пустым: для каждой строки должно быть понятное значение. На практике часто используют поле ID, номер ученика, код предмета или номер заказа.

Благодаря первичному ключу запросы выполняются быстрее и точнее. Он помогает находить нужные записи, не полагаясь на «сходные» значения, которые могут меняться.

Ещё одна важная роль — предотвращение дублирования. Если объект уже существует в базе, первичный ключ позволяет не создавать повторные записи одного и того же экземпляра.

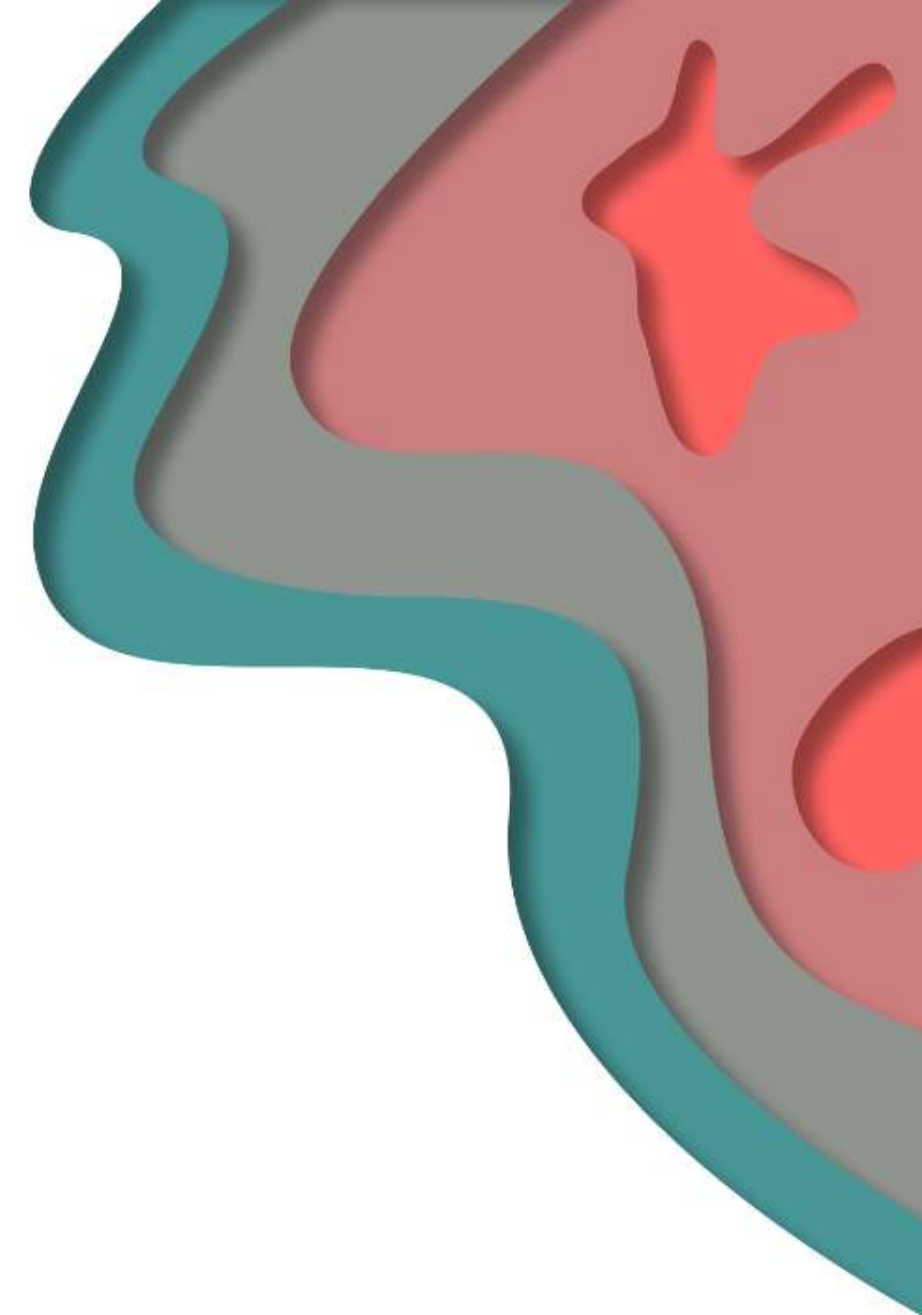


Внешний ключ и связь между таблицами

Внешний ключ хранит значение первичного ключа другой таблицы. Он показывает, к какой записи относится текущая строка, тем самым связывая данные без ручного «склеивания» строк.

Например, в таблице оценок поле `student_id` указывает на ученика, а `subject_id` — на предмет. В таблице заказов `customer_id` связывает запись заказа с клиентом, к которому она относится.

Так формируется ссылочная целостность: нельзя добавить строку, которая ссылается на несуществующий объект. База не позволит создать «оценку несуществующему ученику» или заказ без клиента.

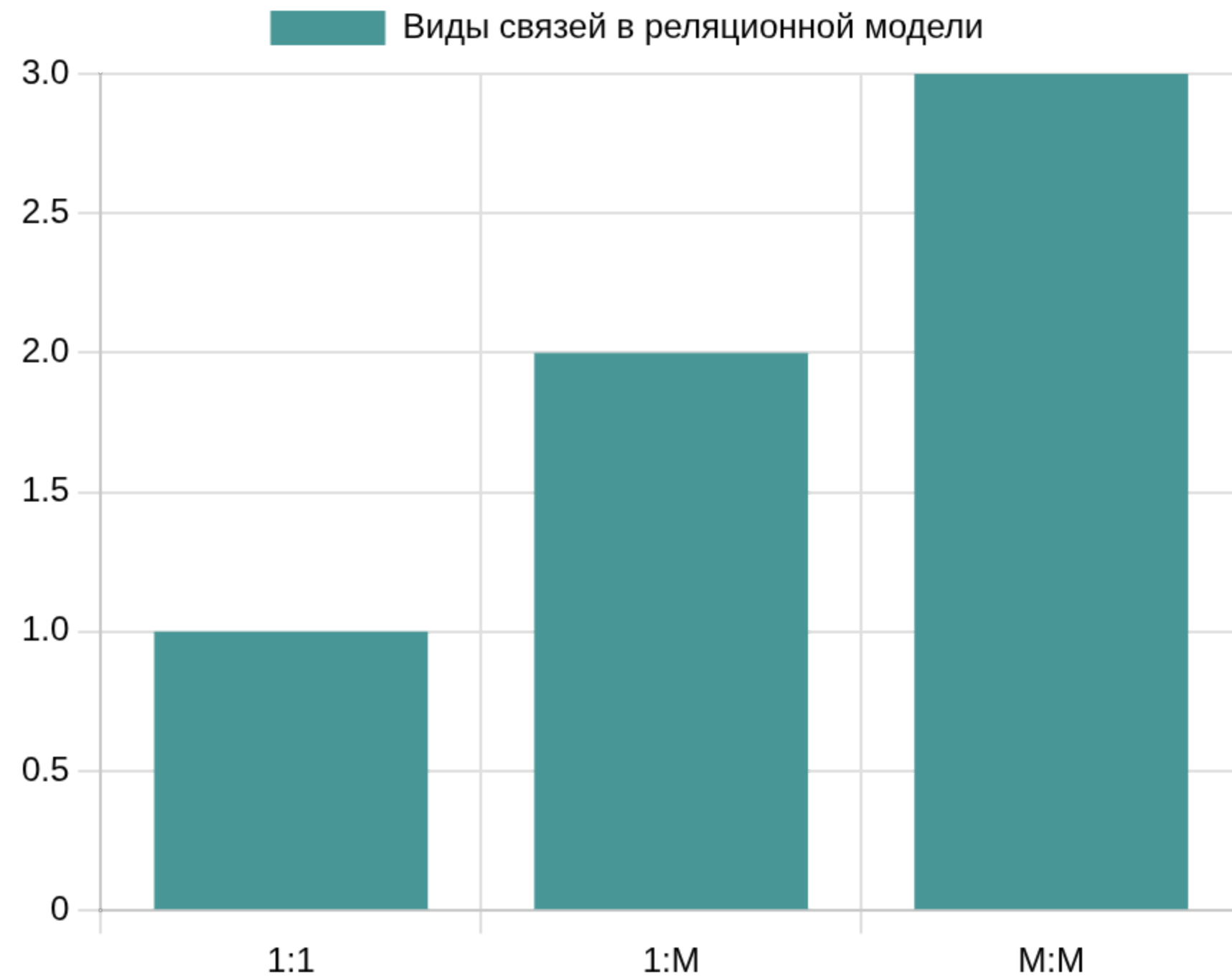


Основные типы связей между таблицами

Чаще всего в школьных данных встречается связь один ко многим: один ученик получает много оценок, один класс включает много учеников.

Разные типы связей задают разную структуру таблиц и определяют, нужен ли отдельный механизм для «связей между многими».

Теория реляционных баз данных



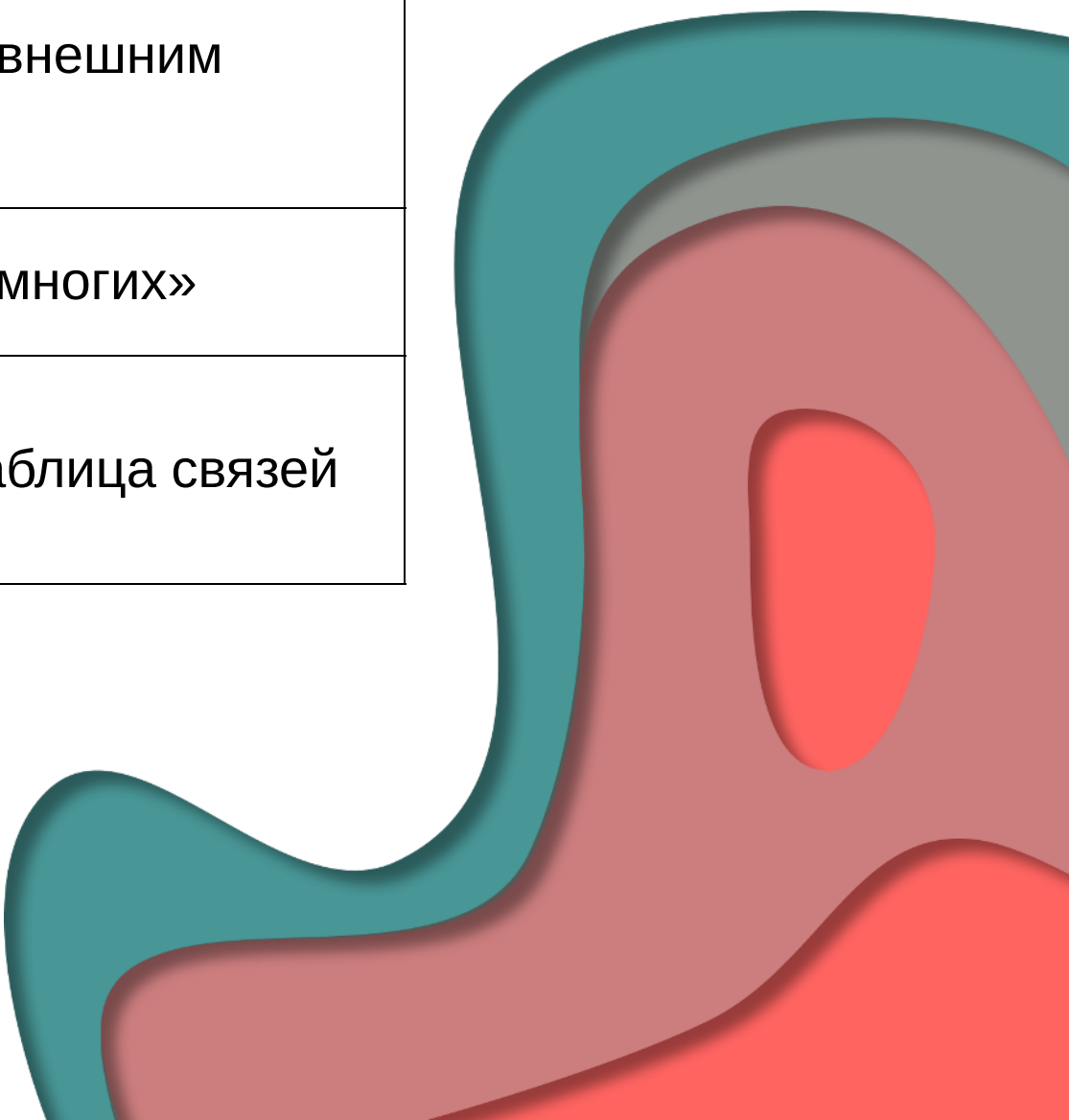
Сравнение типов связей и способов реализации

Тип связи задаёт, как должны быть устроены таблицы и как строятся запросы для получения нужных данных.

Если связь многие-ко-многим, обычно появляется отдельная таблица, иначе невозможно корректно хранить множество ссылок.

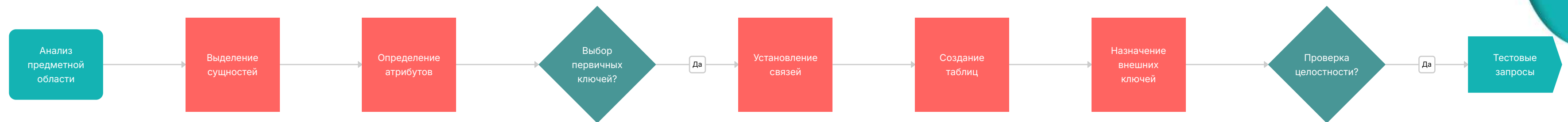
Теория реляционных баз данных

Тип связи	Пример из жизни	Как реализуется в БД
1:1	человек—паспорт	вторая таблица с внешним ключом
1:M	класс—ученики	внешний ключ у «многих»
M:M	ученики—предметы	промежуточная таблица связей



Как превратить задачу в многотабличную схему

Последовательность проектирования структуры базы данных для предметной области



Нормализация уменьшает дублирование

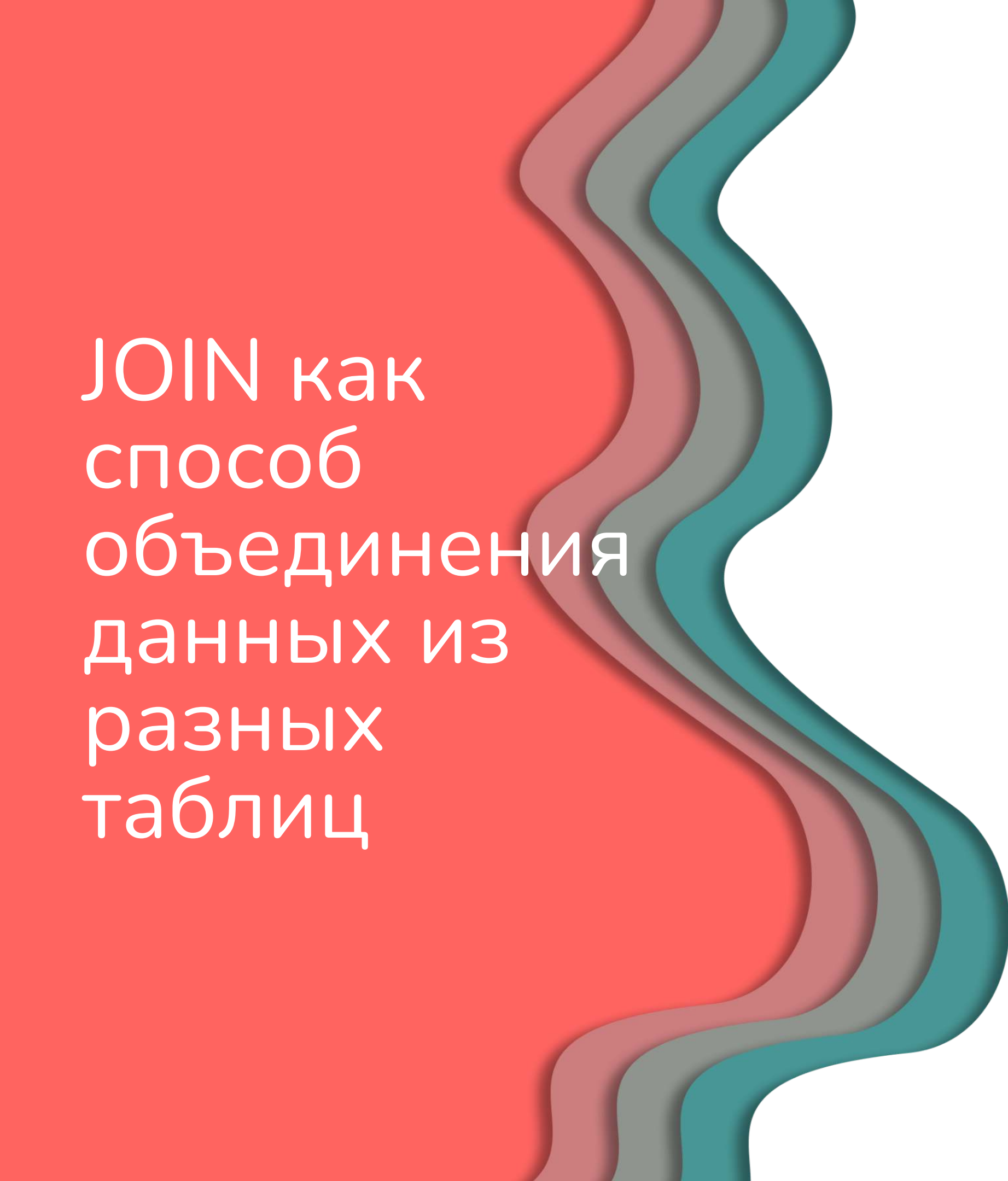
Идея нормализации

Нормализация распределяет данные так, чтобы каждый факт хранился в одном месте. Когда информация меняется, например название предмета или параметр класса, база меньше рискует стать противоречивой. Это делает обновления безопаснее.

Первые нормальные формы для школьного уровня

Для понимания на школьном уровне обычно рассматривают первые три нормальные формы: устранение повторяющихся групп, устранение зависимости от части ключа и устранение транзитивных зависимостей. На схеме часто видно переход от одной большой таблицы к нескольким связанным.





JOIN как способ объединения данных из разных таблиц

JOIN соединяет строки из нескольких таблиц по связанным полям, не нарушая структуру данных. При этом результирующая выборка становится «сводной», хотя исходные сведения хранятся отдельно и относятся к разным объектам.

INNER JOIN возвращает только совпавшие записи, где связь действительно существует. LEFT JOIN сохраняет все строки левой таблицы, даже если соответствий справа нет, что удобно для анализа «какие записи отсутствуют».

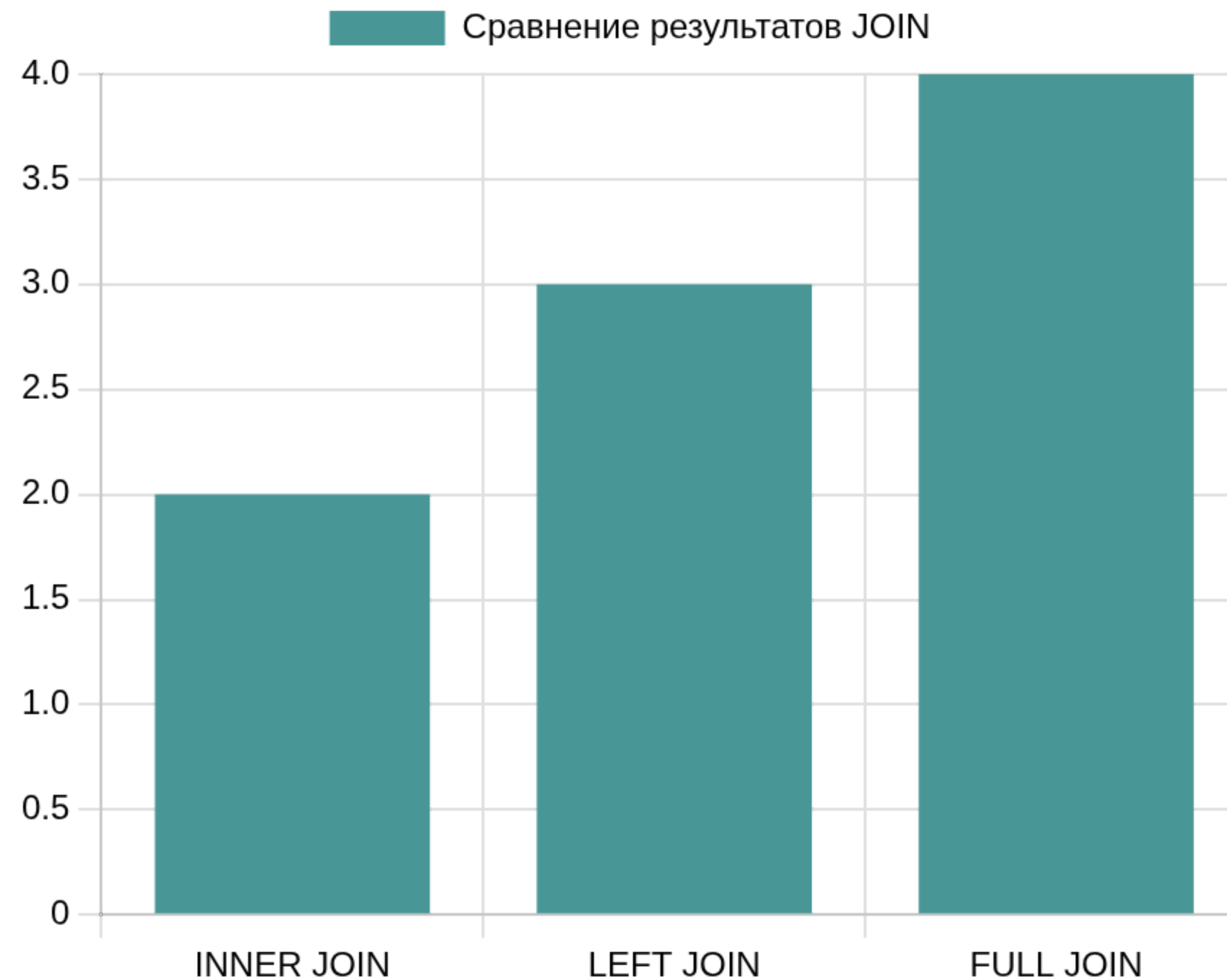
RIGHT JOIN аналогичен LEFT JOIN, но опирается на правую таблицу. Благодаря JOIN можно получить, например, список учеников вместе с предметами и их оценками, используя ключи и связи вместо ручного сопоставления.

Когда используется каждый вид JOIN

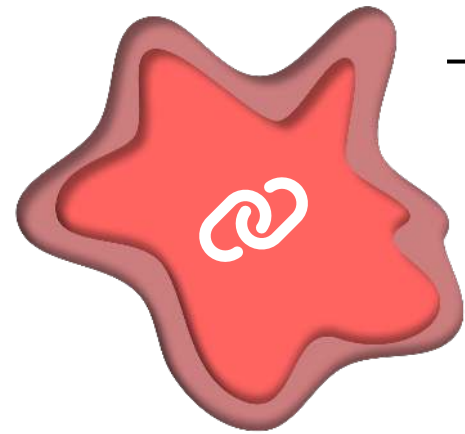
INNER JOIN полезен, когда важны только пары «совпали в обеих таблицах». Он отсеивает строки без соответствия и тем самым делает анализ более «чистым».

Выбор типа JOIN определяется тем, что вы хотите проверить: только пересечения, сохранение базовой выборки или полное сравнение совпадений и расхождений.

Теория реляционных баз данных (обобщённое учебное сравнение результатов JOIN)



Промежуточная таблица для связи многие ко многим



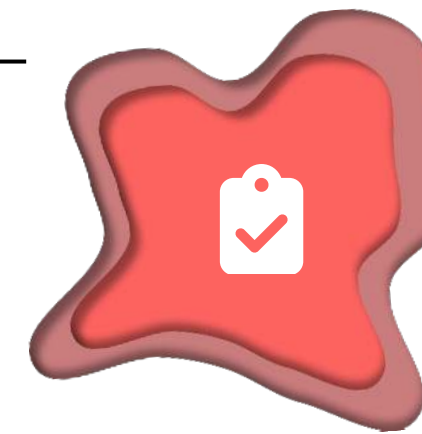
Разрыв прямой ссылки

При связи «многие ко многим» нельзя записать в одной строке сразу множество ссылок так, чтобы структура не ломалась. Поэтому создают промежуточную таблицу: она разносит участие на отдельные строки.



Два внешних ключа

В промежуточной таблице как минимум два поля — внешние ключи на обе связываемые таблицы. Эти ключи задают, какие именно объекты связаны, и удерживают ссылочную целостность при любых изменениях.



Дополнительные атрибуты связи

Часто связь имеет смысл «в моменте»: дата записи, статус участия, оценка, результат работы. Эти характеристики хранятся в промежуточной таблице, чтобы не смешивать их с справочниками и результатами.

Пример структуры учебной базы данных

Таблица	Поля	Назначение
Students	ID, ФИО, класс	Ученики
Classes	ID, класс, руководитель	Справочник классов
Subjects	ID, предмет, кабинет	Справочник предметов
Marks	ID, student_id, subject_id, дата, балл	Оценки/факты

Базовая схема разделяет справочники и результаты, снижает повторение сведений и ускоряет поиск: например, всех оценок конкретного ученика или всех учеников класса.

Такое разбиение поддерживает аккуратные связи: справочные данные обновляются отдельно, а реальные результаты собираются через связку ключей.

Учебный пример реляционной модели данных

Агрегатные функции в запросах по нескольким таблицам

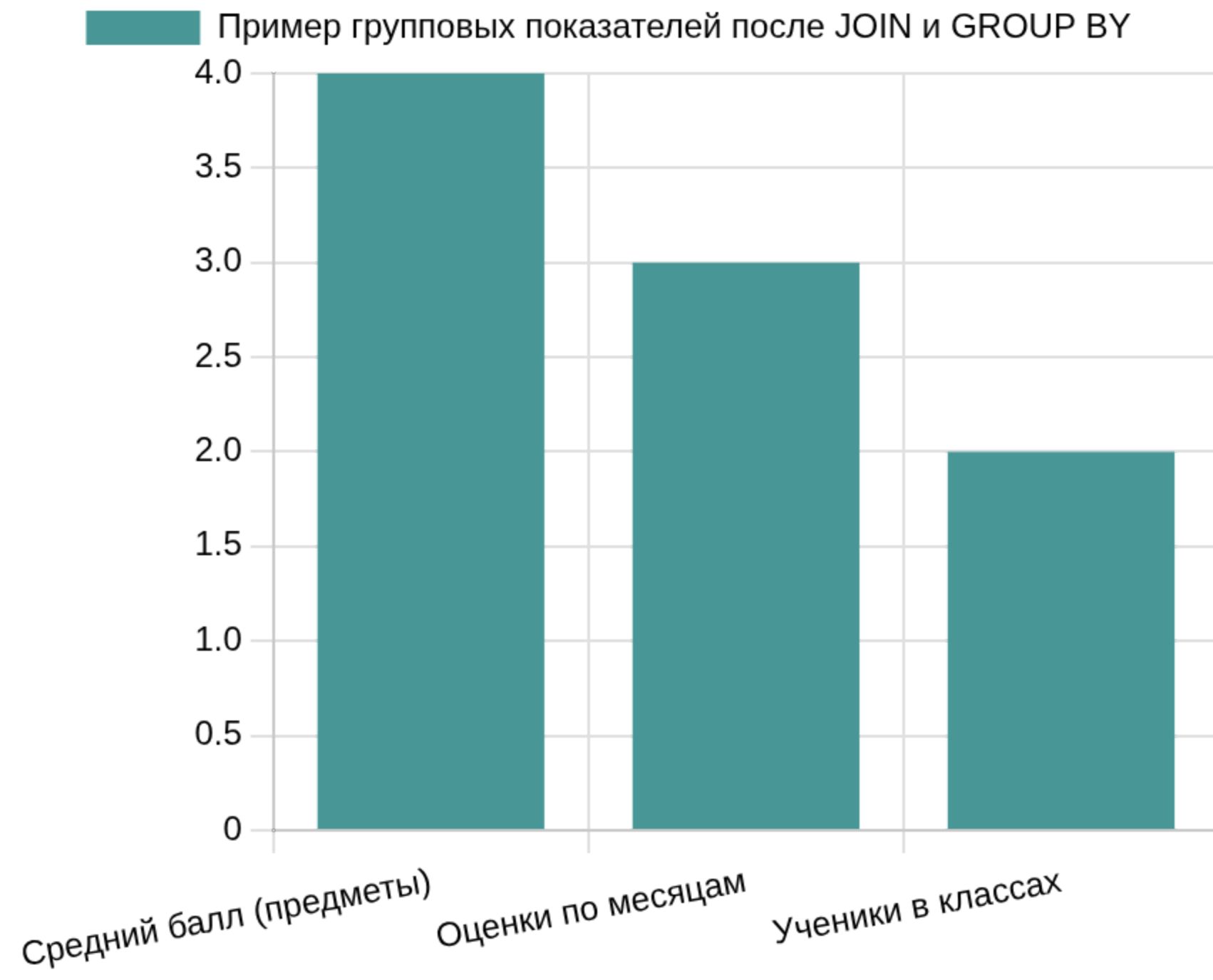
- После JOIN часто требуется не «список строк», а сводка. COUNT подсчитывает записи, SUM суммирует значения, AVG находит среднее, а MIN и MAX — границы для числовых показателей.
- Чтобы получить корректный отчёт, обычно добавляют GROUP BY по ключевому полю. Тогда каждая группа становится отдельным «контейнером» для подсчётов и вычислений, а не перемешивается с другими.
- Для школьных данных это напрямую прикладывается к задачам: средний балл по предмету, количество оценок у конкретного ученика, число учеников в классе и максимальный балл за контрольную работу.
- Агрегаты помогают перейти от хранения фактов к интерпретации. Вы видите тенденции и сравнения между группами, не переписывая вручную длинные списки результатов.

Сводные результаты после соединения таблиц

Диаграмма создаётся по уже объединённым данным и показывает групповые метрики. Это помогает сопоставлять группы, а не ограничиваться проверкой отдельных строк в таблицах.

Если сравнение по группам демонстрирует заметные пики или провалы, стоит уточнить состав групп и корректность связей в JOIN перед дальнейшими выводами.

Учебные сценарии анализа данных (сводка по результату GROUP BY)

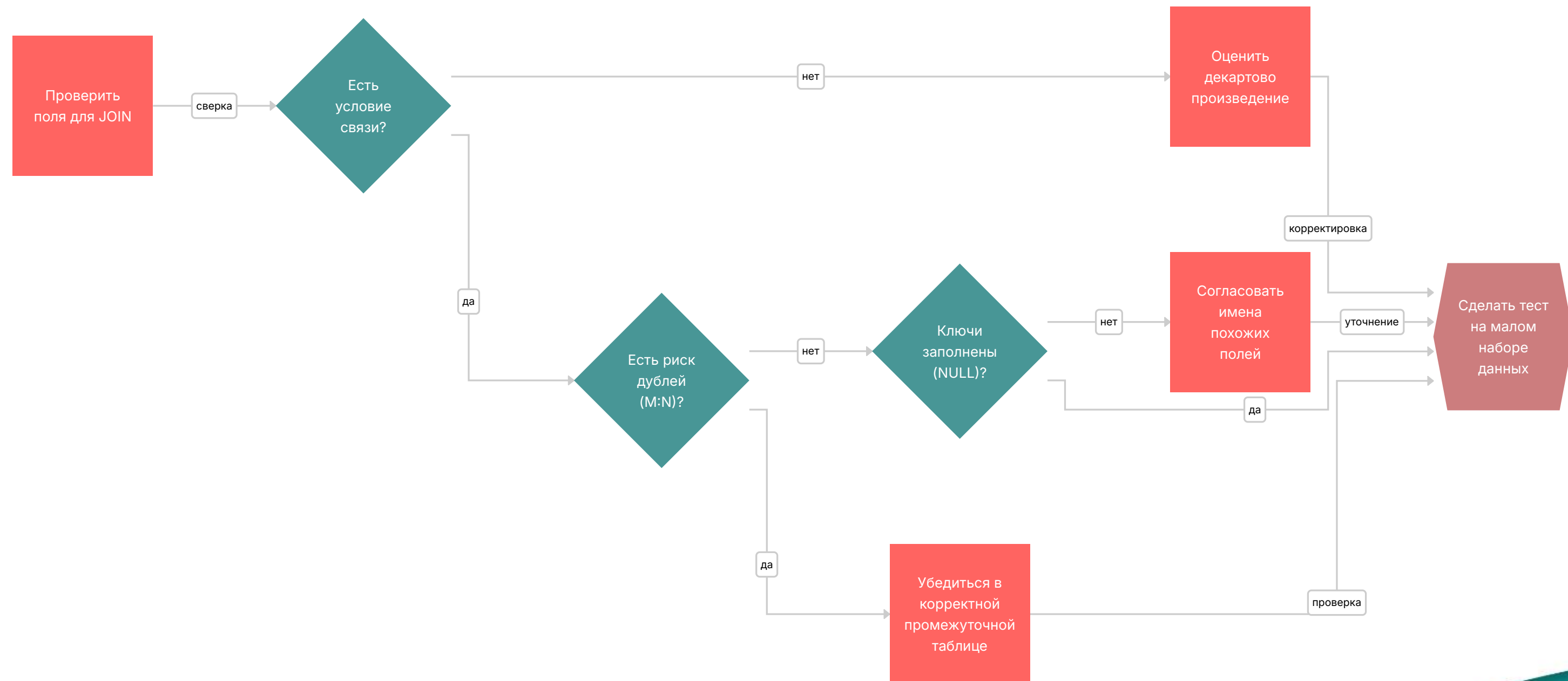


Ограничения целостности данных

- NOT NULL не даёт оставить обязательное поле пустым, а UNIQUE предотвращает повторения. Это важно, когда один и тот же объект (например, ученик или предмет) должен однозначно идентифицироваться в базе.
- CHECK задаёт допустимые значения: например, балл в разумном диапазоне. Благодаря этому система сразу «отклоняет» некорректные записи, не позволяя испортить статистику и отчёты.
- FOREIGN KEY поддерживает ссылочную целостность: запись с внешним ключом должна ссылаться на существующий объект. В многотабличной схеме одна неправильная ссылка может нарушить целый отчёт.

Типичные ошибки при работе с соединением таблиц

Схема проверки запроса перед выполнением JOIN и формированием результата

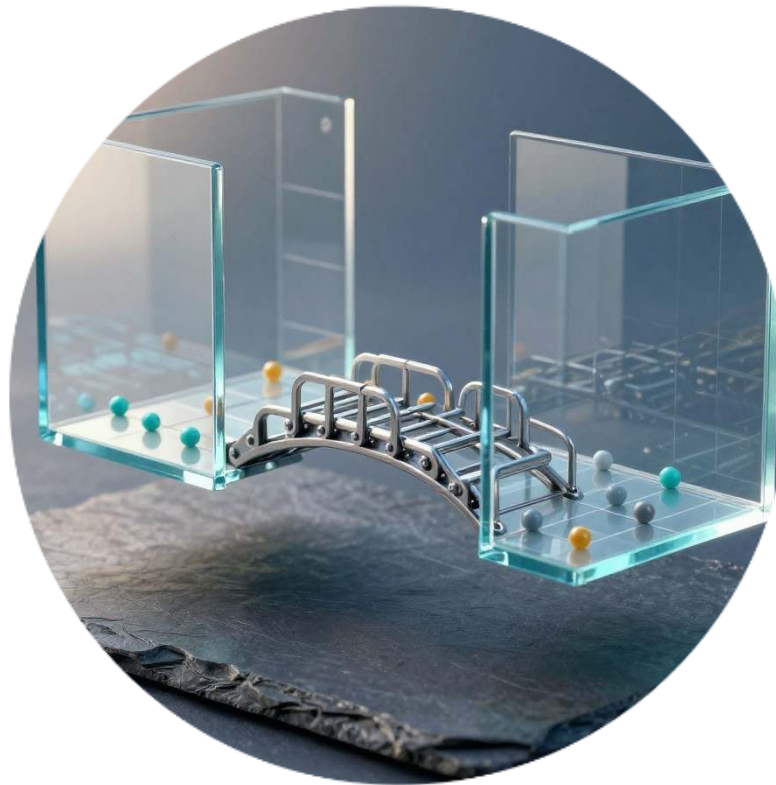


SQL-запрос как практический инструмент работы с данными



Сначала SELECT: выбираем только нужное

Представьте, что вы хотите посмотреть не всю базу, а конкретную информацию. SELECT позволяет перечислить столбцы, которые действительно нужны, и тем самым сделать результат компактным. Когда таблиц много, такой шаг уменьшает «информационный шум».



JOIN собирает факты в один «разрез»

Когда данные разнесены по разным таблицам, JOIN соединяет их по ключам. Вы получаете цельный ответ: ученик — предмет — оценка, хотя каждое звено хранится отдельно. Важно контролировать, какие строки попадут в результат.



WHERE и ORDER BY превращают выборку в отчёт

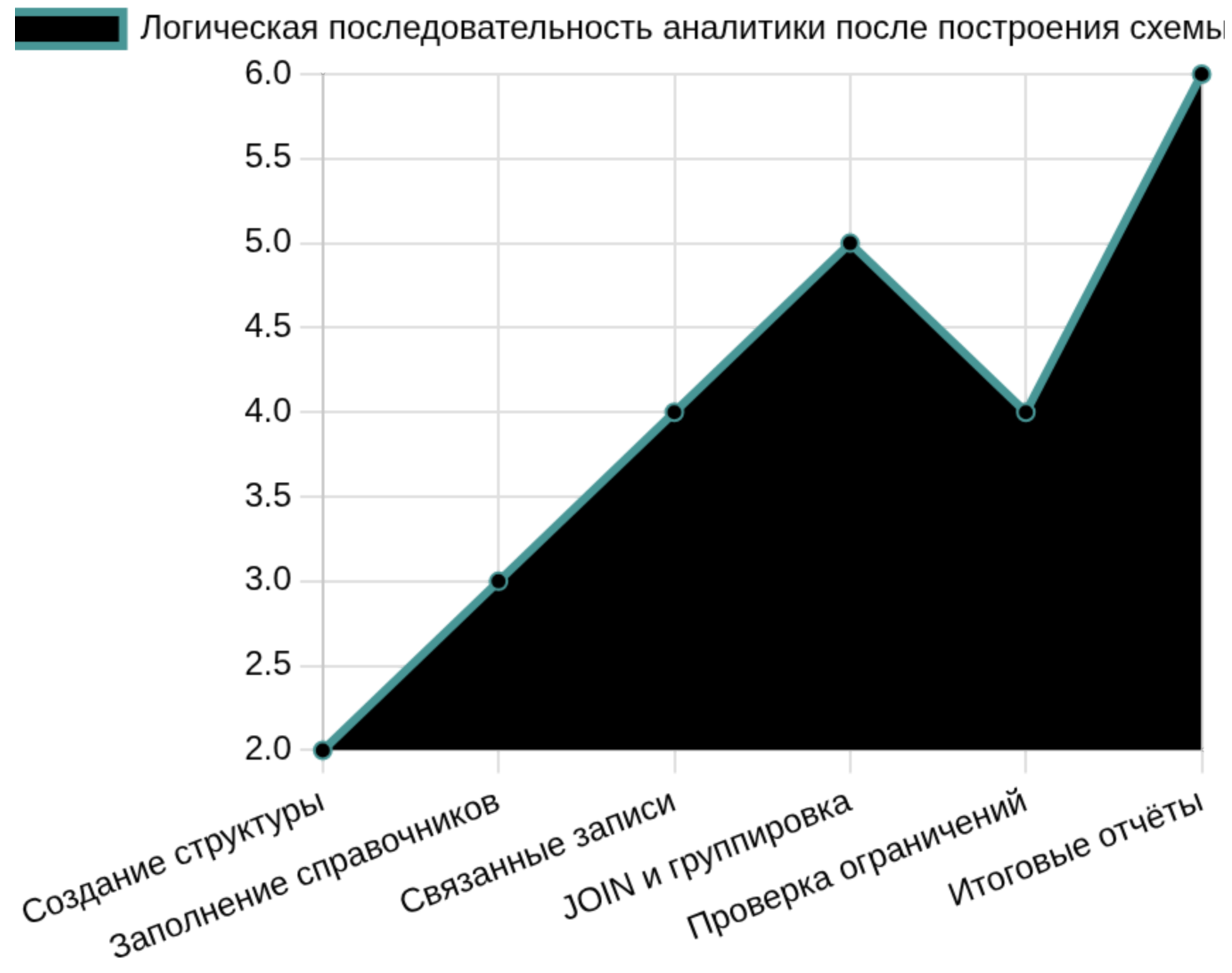
Чтобы найти конкретный период или отобрать учеников по условию, добавляют WHERE. А чтобы результат читался как отчёт, применяют ORDER BY — сортировку по фамилии, дате или баллу. В учебных задачах это помогает быстро проверить корректность запроса.

Логика работы с данными в многотабличной БД

Анализ строится не «сразу по таблицам», а как цепочка шагов: от структуры и связей к корректным выборкам и суммарным отчётам.

Если пропустить проверку ключей и ограничений, JOIN может собрать неправильную статистику. Последовательность снижает вероятность логических ошибок в данных.

Обобщённая учебная методика работы с реляционными базами данных



Многотабличная БД: данные становятся точнее и удобнее

Многотабличная структура делает данные точнее и компактнее для анализа. Ключи, JOIN, нормализация и ограничения целостности сокращают ошибки и дублирование, повышая надёжность запросов и точность школьной статистики.

