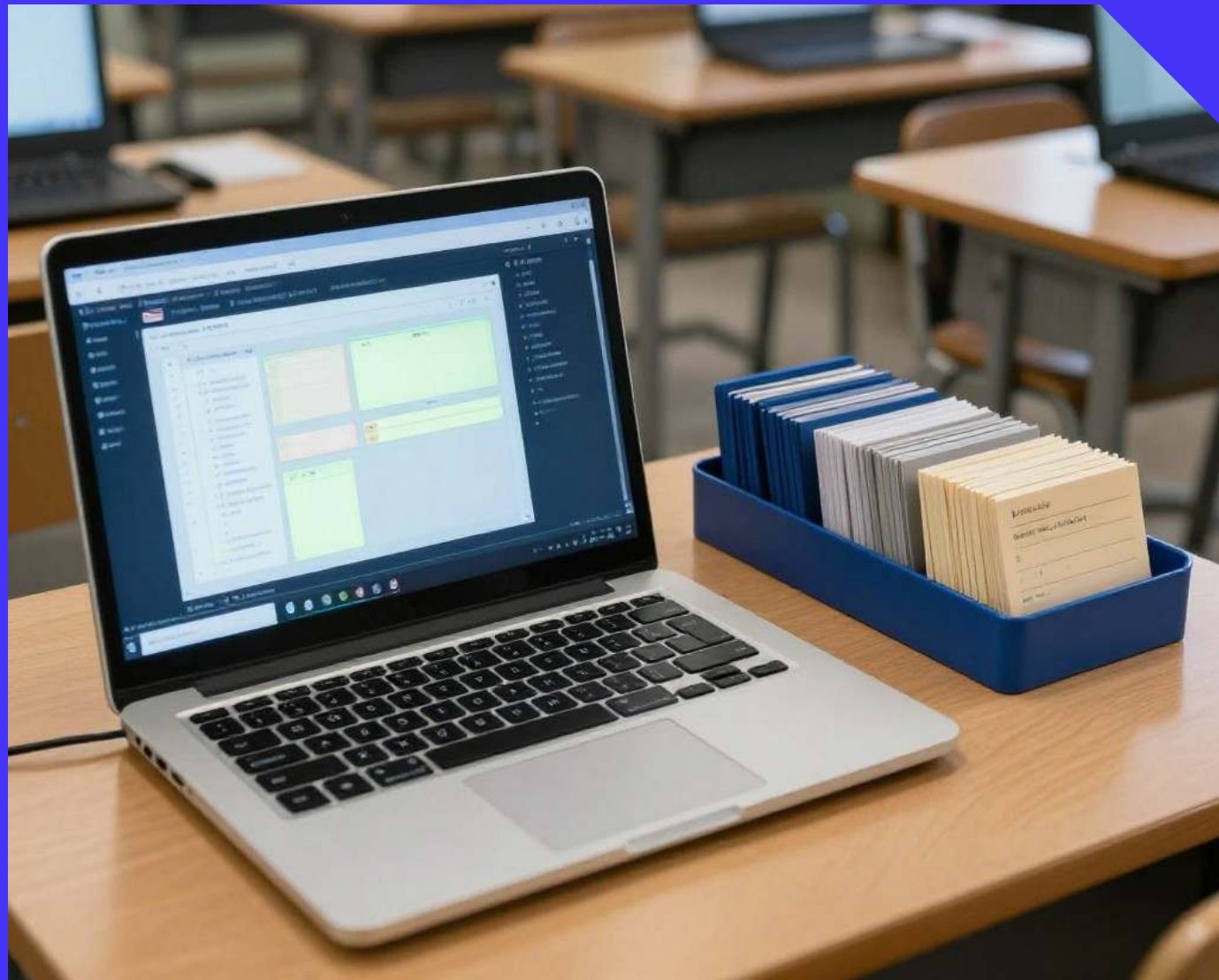


Реляционные базы данных и СУБД: основа хранения информации в цифровом мире

Реляционные базы хранят данные в таблицах и связывают их ключами, а СУБД обеспечивает доступ и защиту.





Базы данных окружают вас: электронные журналы, магазины, банки, библиотеки и соцсервисы. Реляционная модель предложена Эдгаром Коддом в 1970 году, а SQL стал стандартным языком работы. Важны целостность, транзакции и права доступа.

Почему эта тема важна для 11 класса информатики

Что такое реляционная база данных



Данные в таблицах

Реляционная база организует информацию в виде таблиц, где строки соответствуют объектам, а столбцы — их свойствам. Это помогает ясно описывать структуру данных и понимать, что именно хранится.



Связи между таблицами

Связи дают возможность объединять связанные сведения без постоянного дублирования. Благодаря этому поиск по нужным критериям становится логичнее, а обновления легче контролировать, сохраняя согласованность данных.



Ключи для точной идентификации

Чтобы однозначно находить записи, используют ключи. Первичный ключ определяет каждую строку без двусмысленности, а внешние ключи указывают, как связаны таблицы. Так уменьшаются ошибки и путаница.

Основные элементы реляционной модели

Таблица задаёт структуру данных: строки и столбцы. Ключи и ограничения обеспечивают точность, а индексы ускоряют поиск по часто используемым полям.

Реляционная модель держится на чёткой структуре и контроле ошибок: ключи связывают сведения, ограничения предотвращают некорректные данные, индексы ускоряют запросы.

Элемент	Назначение	Пример
Таблица	Хранит данные одного типа	Ученики
Первичный ключ	Однозначно определяет запись	ID_ученика
Внешний ключ	Связывает с другой таблицей	ID_класса
Индекс	Ускоряет поиск	по фамилии
Ограничения	Контролируют допустимые значения	целостность

Классическая реляционная модель данных и основные понятия SQL

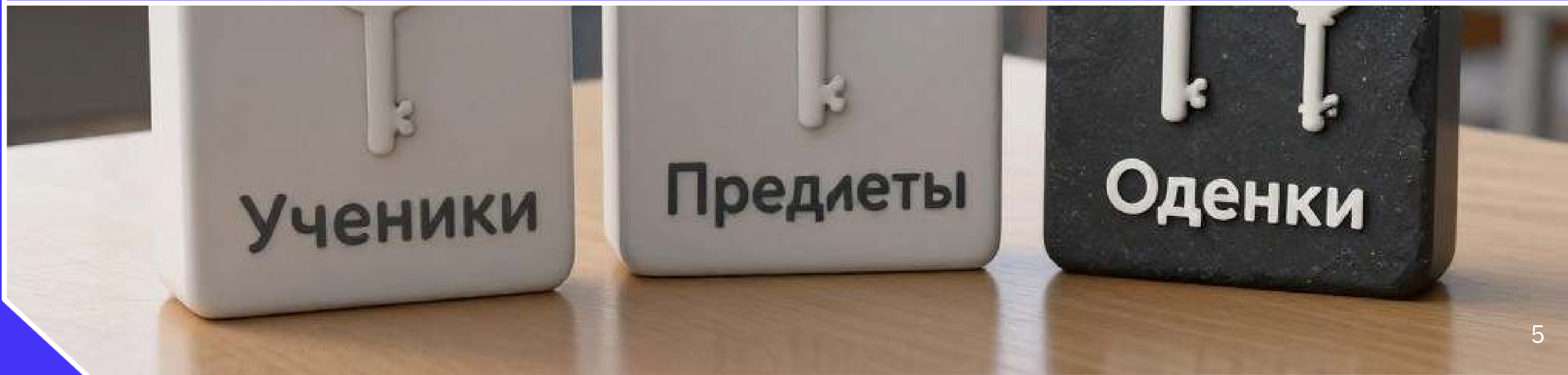
Пример схемы данных: ученики, предметы, оценки

Три таблицы и типовые отношения

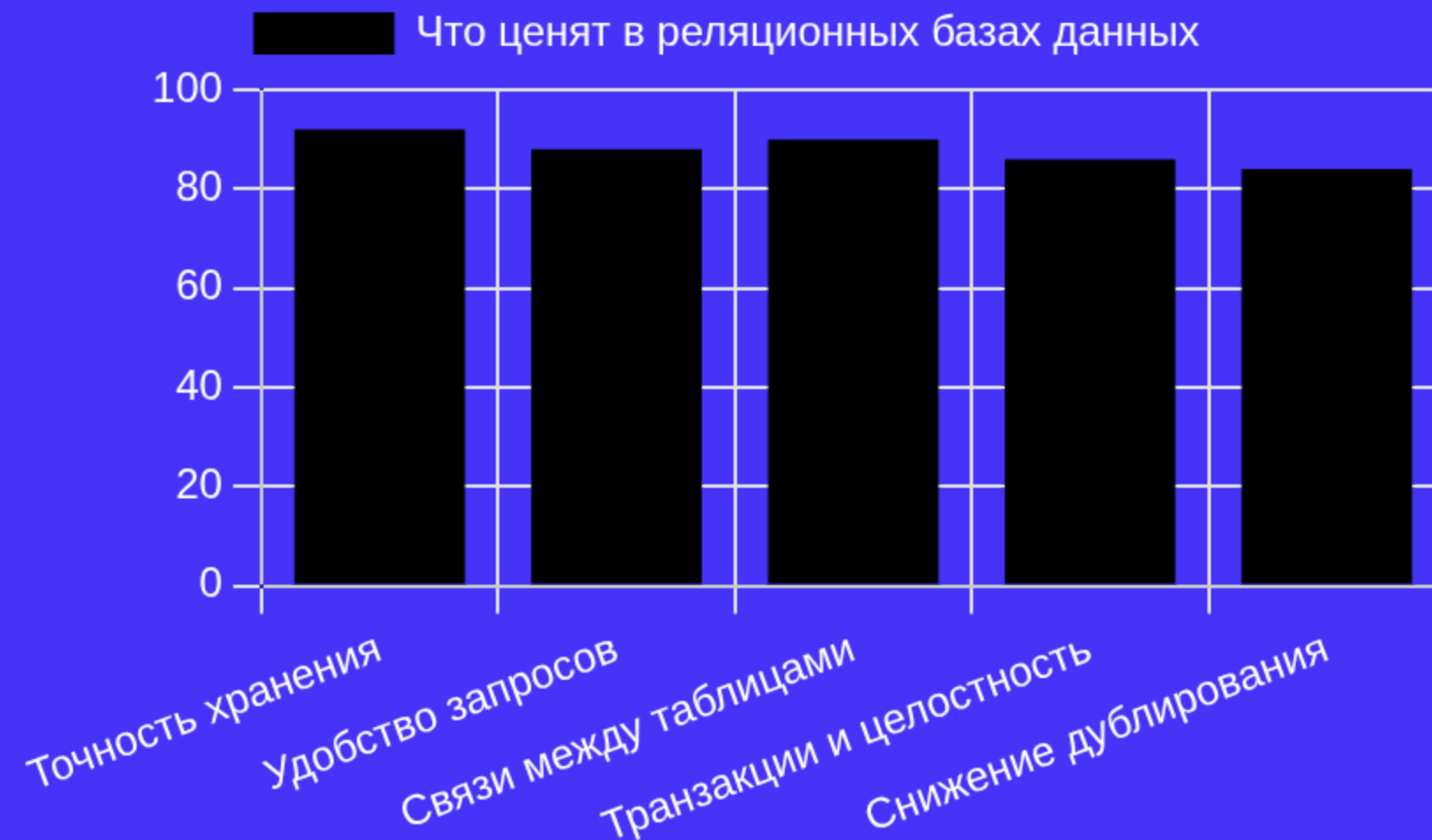
В учебной схеме обычно есть таблицы «Ученики», «Предметы» и «Оценки». Один ученик может получать много оценок, а один предмет встречается во множестве оценочных записей. Это удобно моделировать связями по ключам.

Почему почти не нужно дублировать информацию

На схеме у каждой таблицы есть первичный ключ, а в «Оценках» — внешние ключи, указывающие на конкретного ученика и предмет. Благодаря этому сведения о ученике не повторяются в каждой строке журнала, а исправления делаются в одном месте.



Преимущества реляционных баз данных



Обобщённые характеристики реляционной модели
(без привязки к конкретным числам в
источниках)

Когда данные описаны строго и связаны ключами, запросы дают стабильные результаты, а изменения легче контролировать и проверять. Это особенно заметно в учебных и учётных системах.

Диаграмма показывает: сила реляционной модели — в сочетании структуры и надёжности. Уменьшение дублирования и поддержка связей делают данные управляемыми при росте объёма.

Что делает система управления базами данных

01

СУБД принимает таблицы и хранит их на диске, организуя доступ к данным. При этом она обеспечивает быстрый поиск по запросам и ведёт работу с физическим хранением, не скрывая логику от пользователя.

02

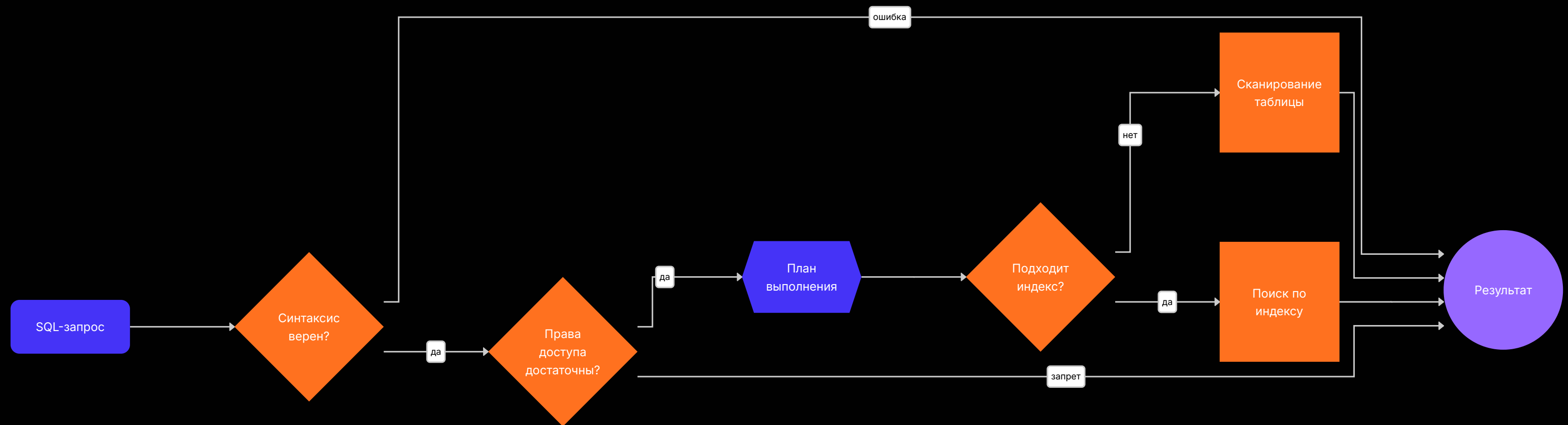
Любой запрос на изменение проходит через проверки: корректность команд, права доступа и соблюдение правил целостности. СУБД гарантирует, что изменения не приведут к противоречиям в структуре данных и связях.

03

Популярные СУБД — PostgreSQL, MySQL, Microsoft SQL Server, Oracle Database и SQLite. Они различаются масштабом и режимами работы, но объединены общей задачей: надёжно управлять таблицами и их связями.

Как запрос превращается в результат

Пользовательский SQL-запрос в реляционной СУБД: проверка, план выполнения и поиск по таблице или индексу.



SQL как язык работы с данными

01

SQL используют для чтения и изменения данных: выборки, добавления, обновления и удаления. Команды SELECT, INSERT, UPDATE и DELETE позволяют управлять содержимым таблиц без ручной обработки файлов.

02

Чтобы объединять информацию из разных таблиц, применяют JOIN. Это позволяет получать, например, записи об оценках вместе с именами учеников и названиями предметов, сохраняя связи между объектами.

03

Когда нужно подытожить множество строк, используют GROUP BY. Вместе с агрегатными функциями он помогает строить отчёты: средние баллы, количества оценок по предметам, итоги по классам и периодам.

04

Для упорядочивания результатов применяют ORDER BY. SQL также упрощает формирование отчётов и повторяемых запросов, делая работу с данными стандартной и воспроизводимой.

Нормализация: как уменьшить дублирование данных

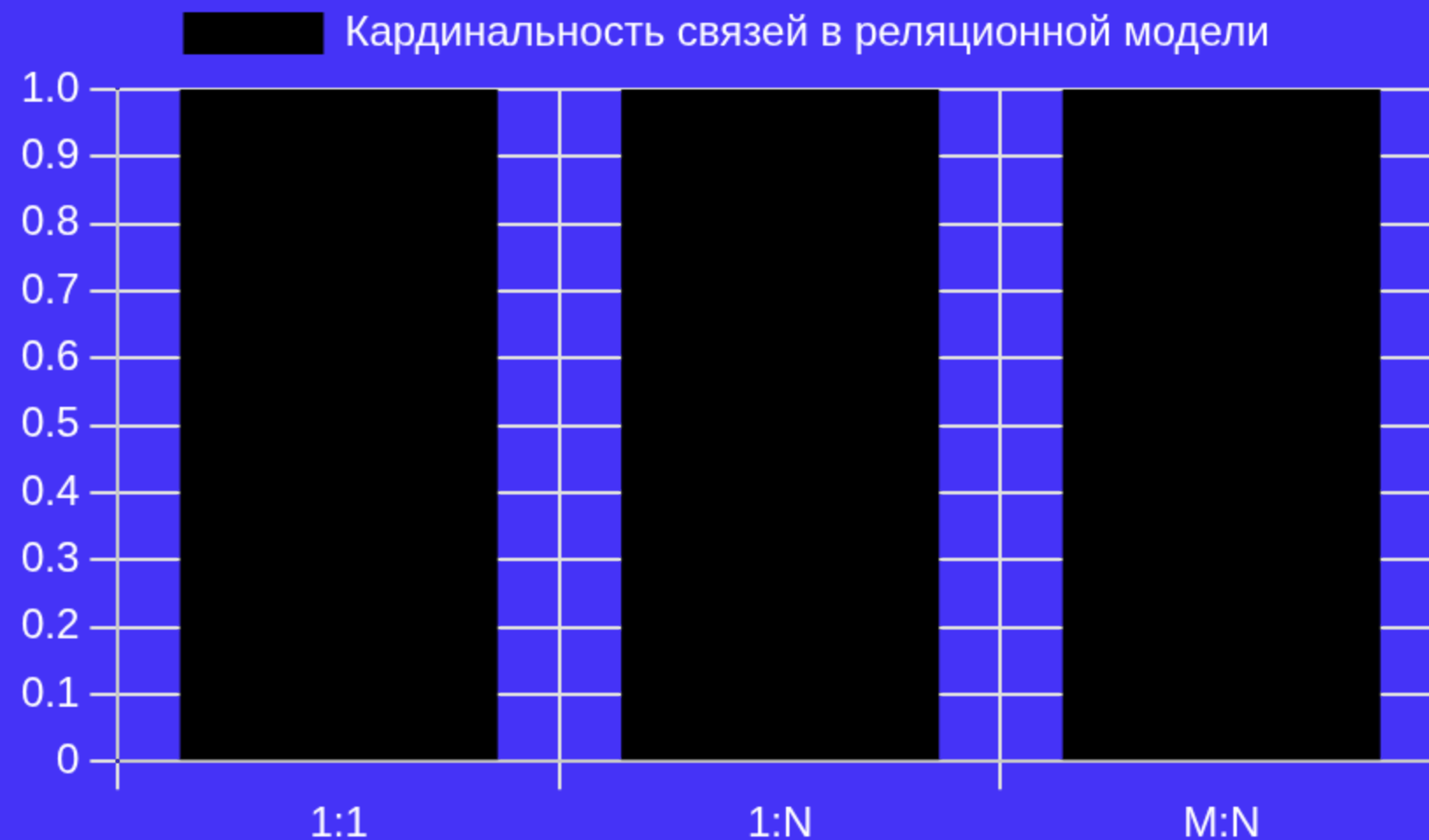
Нормализация приводит схему к более логичным формам: значения становятся атомарными, ключевые зависимости уточняются, а лишние переносы между столбцами устраняются.

Чем меньше дублирования и ложных зависимостей, тем проще поддерживать базу: обновления становятся безопаснее, а логика данных — понятнее.

Форма	Что исправляет	Главный результат
1НФ	Атомарность значений	нет «смешанных» данных
2НФ	Зависимости от части ключа	даты зависят только от ключа
3НФ	Транзитивные зависимости	неключевые данные не зависят друг от друга
Схема	Меньше повторов	меньше ошибок при обновлении

Теория нормализации реляционных баз данных

Типы связей между таблицами



Классическая теория реляционных баз данных; модель кардинальностей (без привязки к конкретным измерениям).

Кардинальность определяет, сколько строк одной таблицы связано с одной строкой другой. При проектировании учитывают и ограничения на отсутствие значений, чтобы связи оставались корректными.

Большинство учебных и прикладных схем сводится к связям 1:N, а сложные отношения реализуют через промежуточные таблицы, сохраняя целостность и избегая повторов.



Индексы и скорость поиска

- Индекс ускоряет выборку по полю, потому что СУБД быстрее находит нужные строки, не просматривая всю таблицу. Чаще всего индексируют идентификаторы, даты, фамилии и параметры фильтра.
- Цена индекса — увеличение времени на запись: при добавлении, удалении или обновлении данных обновляется и индекс. Поэтому индексы создают обдуманно, а не «на всё подряд» для экономии ресурсов.

Транзакции и надёжность данных

- Транзакция объединяет несколько операций в одно логическое действие: либо выполняется всё целиком, либо не выполняется ничего. Это критично, когда данные меняются в нескольких таблицах сразу.
- Свойства ACID описывают надёжность: атомарность (всё или ничего), согласованность (данные остаются корректными), изолированность (эффекты не мешают параллельной работе), долговечность (результат сохраняется после фиксации).
- Благодаря транзакциям база защищена от частичных изменений при сбоях программы или компьютера. Это помогает предотвратить разрывы в данных при переводе денег, оформлении заказов и массовых обновлениях.

Как СУБД завершает транзакцию

Поток действий при работе транзакции: контроль ошибок и итоговая фиксация.



Популярные СУБД и их особенности

Набор распространённых систем управления базами данных и их характерные особенности для учебных и реальных проектов.

Разные СУБД оптимизируются под разные сценарии: от небольших локальных приложений (SQLite) до больших корпоративных систем (Oracle, SQL Server), при этом принципы реляционной модели сохраняются.

СУБД	Ключевая особенность
PostgreSQL	Открытая, расширяемая; сильна в сложных запросах
MySQL	Широко применяется для веб-приложений и сайтов
SQLite	Лёгкая встроенная СУБД для небольших программ и мобильных задач
Oracle Database	Часто выбирают крупные организации и корпоративные контуры
Microsoft SQL Server	Корпоративная система для задач предприятий

Официальная документация PostgreSQL, MySQL, SQLite, Oracle и Microsoft.

Безопасность данных и права доступа



Роли и права

Доступ к данным обычно разделяют по ролям: чтение, добавление, изменение и администрирование. Это помогает ограничить «лишние» действия и снижает ущерб в случае ошибки пользователя или настройки.



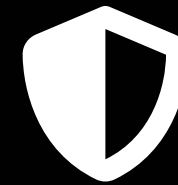
Журналирование действий

Журналы фиксируют, кто и что делал с данными. При инцидентах журнал помогает разобраться в последовательности событий и быстрее восстановить корректное состояние, соблюдая требования учителя или организации.



Аутентификация и пароли

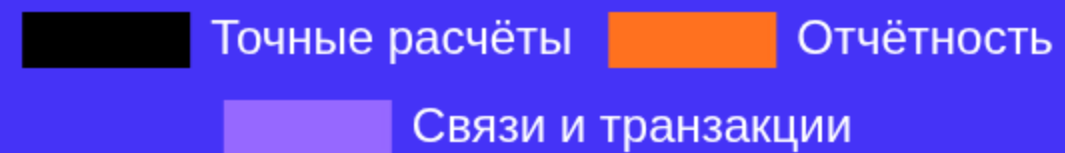
СУБД используют учётные данные для входа и проверки подлинности. Правильно настроенные учётные записи и политика паролей уменьшают риск несанкционированного доступа к таблицам и запросам.



Защита от SQL-инъекции

Одна из угроз — SQL-инъекция, когда вредоносный ввод меняет смысл запроса. Параметризованные запросы и принцип минимальных привилегий заметно уменьшают риск утечки и повреждения данных.

Когда реляционные базы данных особенно удобны



Обобщение практических применений реляционных БД (без числовых метрик); базируется на принципах модели и транзакций.

В таких задачах важны строгая структура, предсказуемые правила и проверяемая целостность. Реляционная модель удобна, когда данные нужно не только хранить, но и надёжно изменять.

Судя по набору сценариев, реляционные БД выигрывают там, где критичны корректность и связи между сущностями. Для них естественны транзакции и отчётные выборки.



Где реляционные базы данных встречаются ежедневно

- Реляционные базы используют в банковских приложениях, интернет-магазинах, школьных электронных дневниках, кассовых системах и библиотеках. Там хранятся учётные записи, заказы, платежи, книги и расписания.
- Практически в каждом таком сервисе есть формы поиска и отчёты: выборка по фамилии, фильтрация по дате, получение сводок по услугам. Это напрямую опирается на таблицы и SQL-запросы.

Типичные ошибки при проектировании базы данных

- Одна из частых проблем — отсутствие первичных ключей. Тогда строки сложнее отличать друг от друга, возрастает риск противоречий при обновлениях и удалениях, а связывать таблицы становится неудобно и ненадёжно.
- Ещё одна ошибка — неверно подобранные типы данных. Например, хранение дат как произвольного текста усложняет сортировку и сравнения, а некорректные ограничения целостности допускают «битые» значения в таблицах.
- Если не продумать индексы, запросы к большим таблицам будут выполняться медленно: СУБД вынуждена сканировать много строк вместо точного поиска. Это заметно даже в простых фильтрах и отчётах.
- Сложная структура связей и избыточное дублирование тоже мешают. Слишком большое число соединений и неправильная нормализация усложняют сопровождение, повышают вероятность ошибок при изменении требований и данных.



Реляционные базы данных соединяют таблицы, ключи, SQL, индексы и транзакции в единую систему работы с информацией. Их ценят за структурированность, точность, контроль ошибок и предсказуемую скорость поиска.

Итог: почему реляционные БД
остаются базовой технологией