SQL.
Резюме

Выводы

- **База данных (БД, DB) —** совокупность самостоятельных данных любого типа, систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ, компьютер);
- **Система управления базами данных (СУБД) —** специальное программное обеспечение, которое управляет созданием и использованием баз данных с помощью ЭВМ. В рамках данного курса использовалась СУБД PostgreSQL;
- **Реляционная база данных** — база данных, основанная на реляционной модели данных (кроме реляционной существуют и другие модели данных: сетевые, иерархические, объектно-ориентированные и т.д.);
- Реляционная модель определяет термины: отношение, атрибут, кортеж, но на практике для упрощения используют термины таблица, столбец (или поле), строка (или запись);
- **Таблица реляционной базы данных** — объект базы данных, который хранит структурированную информацию в виде набора строк и столбцов;
- **Нормализация —** процесс преобразования отношения (таблицы) базы данных к виду, отвечающему нормальным формам;
- Конечной целью нормализации является **уменьшение потенциальной противоречивости** хранимой в базе данных информации;

Выводы

- Нормализация **НЕ ВСЕГДА** полезна, особенно в Web приложениях, где на первую очень выходят требования производительности запросов и масштабируемости базы данных. Для таких систем получили широкое распространение NoSQL решения;
- С помощью первичных и внешних ключей СУБД контролирует целостность данных, с помощью «проверок» не позволяет добавить некорректные данные в таблицу.
- Модель сущность-связь (**ER-модель**) (англ. entity-relationship model, ERM) — модель данных, позволяющая описывать концептуальные схемы предметной области.
- Реляционная модель описывает правила хранения и модификации **ЛЮБЫХ** данных, которые хранятся в базе данных, **ER модель** описывает модель предметной области КОНКРЕТНОГО проекта;
- ER модель чаще всего преобразуют в реляционную модель для использования реляционных СУБД для хранения данных, но ER модель может быть также преобразована в любую другую модель данных: объектно-ориентированную, сетевую, иерархическую и т.д;
- Между таблицами базы данных могут быть следующие связи, создаваемые с помощью внешних ключей: один к одному, один ко многим, многие ко многим. При проектировании базы данных если следовать рекомендациям по созданию связей, то правильность связей будет контролироваться СУБД;

Выводы

- SQL (structured query language — язык структурированных запросов) — формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных (СУБД);
- Группы операторов SQL:
  - Операторы определения данных (Data Definition Language, **DDL**): **CREATE, ALTER, DROP**;
  - Операторы манипуляции данными (Data Manipulation Language, **DML**): **SELECT, INSERT, UPDATE, DELETE**;
  - Операторы управления транзакциями (Transaction Control Language, **TCL**): **COMMIT, ROLLBACK, SAVEPOINT**;
  - Операторы определения доступа к данным (Data Control Language, **DCL**): **GRANT, REVOKE, DENY**.
- Для переноса базы данных с одной СУБД на другую создается дамп, который хранит SQL команды для восстановления базы данных;
- **SELECT** — оператор DML языка SQL, возвращающий набор данных (выборку) из базы данных, удовлетворяющих заданному условию;
- Оператор SELECT позволяет реализовать операции реляционной алгебры: Объединение, Пересечение, Вычитание, **Декартово произведение, Выборка (ограничение), Проекция, Соединение.**
- Порядок следования в операторе SELECT важен! Именно в таком порядке выполняются обработка команд:

Выводы

```
SELECT
  cat.name as category, c.age, count(*) as cnt
  FROM contact c, contact_value cv, category cat, contact_type ct
  WHERE c.id_category=cat.id AND cv.id_contact = c.id
  GROUP BY category, age
  HAVING count(*) > 4
  ORDER BY cnt ASC, category DESC
  LIMIT 1 OFFSET 1;
```

- JOIN, декартовое произведение и подзапросы **соединяют** несколько таблиц в одну по установленным правилам;
- Операция **соединения** в отличии от операции **объединения** соединяет таблицы с разной структурой и является комбинацией декартового произведения, выборки и проекции именно поэтому ОЧЕНЬ часто используется в запросах к реляционной базе данных;
- **Типы JOIN:** INNER JOIN, CROSS JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN. Для удобства понимания работы каждого типа JOIN используются диаграммы Венна;
- Если возникает выбор использовать подзапрос или JOIN, следует выбирать JOIN, так как запросы с JOIN выполняются быстрее. Для более аргументированного выбора следует воспользоваться утилитой EXPLAIN.

Выводы

- **Представление** (view) — виртуальная (логическая) таблица, представляющая собой **поименованный запрос** (синоним к запросу), который будет подставлен как подзапрос при использовании представления.
- Различают обычное представление и материализированное;
- Индекс — объект базы данных, создаваемый с целью повышения скорости поиска данных;
- Использование индексов увеличивает производительность запросов к таблице базы данных при этом синтаксис запроса не зависит от наличия индекса, что позволяет создавать индексы уже после проектирования и написания системы;
- При использовании индексов следует помнить о том, что наличие индексов увеличивает время выполнения INSERT, UPDATE, DELETE – ведь при изменении данных нужно обязательно перестроить индексы по всем полям для которых они созданы;
- С помощью утилиты EXPLAIN можно получить план выполнения конкретного запроса, чтобы проанализировать используются ли индексы для выполнения запроса или нет и при необходимости его улучшить. Данная утилита присутствует в любой мощной СУБД;
- Все команды выполняемые через SQL клиент транслируются в SQL;

Выводы

- **Транзакция** (Transaction) — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными.
- Транзакционная система, чтобы обеспечить надёжную работу должна выполнять требования ACID (Atomicity, Consistency, Isolation, Durability);
- Новую транзакцию начинает сервер базы данных АВТОМАТИЧЕСКИ выполняя любую команду из подмножества DML: **INSERT, UPDATE, DELETE**;
- Для того чтобы подтвердить изменения используется команда **COMMIT**;
- Для того чтобы откатить изменения используется команда **ROLLBACK**;
- Для того чтобы создать промежуточную точку сохранения используется команда **SAVEPOINT**.
- Если в настройках текущего соединения к базе данных стоит режим autocommit, то фиксирование изменений происходит немедленно и автоматически;
- Для того чтобы назначить привилегию на выбранное действие пользователю используется команда **GRANT**;
- Для того чтобы отозвать ранее выделенную привилегию используется команда **REVOKE**;
- Назначение привилегий на отдельные таблицы (столбцы) позволяет контролировать доступ к данных на уровне СУБД.