Lab 1

Part 1

Task 1.1:

a)

1.Example of superkeys:

EmpID,SSN,Email,Phone,EmpID name,SSN Department;

2.Candidate keys:

EmpID,SSN,Email,Phone;

3.Primary key: EmpID

This is an artificial identifier that is easiest to use in a database.

SSN and Email may change or contain errors.

An EmpID is usually created specifically for a database.

4.Theoretically yes, if is a work number shared by several employees. But in the example table, all phones are unique. Therefore the answer depends on the business rules:

If the phone is unique – it is a candidate key.

If not – the phone does not fit as a key.

b)

1.Primary Key:

The minimal set of attributes needed for the primary key is:
(StudentID, CourseCode, Section, Semester, Year)

2.

- **StudentID** → identifies the student.
- **CourseCode** → identifies the course.
- **Section** → distinguishes between multiple sections of the same course in the same semester.
- **Semester + Year** → allow the same student to retake the same course in different terms.

3. Additional candidate keys:
There are no obvious alternative candidate keys because the business rules require all of the above attributes to uniquely identify a registration record.

Task 1.2:

- **Student(AdvisorID) → Professor(ProfID)**

(Each student has an advisor who is a professor.)

- **Course(DepartmentCode) → Department(DeptCode)**

(Each course belongs to a department.)

- **Department(ChairID) → Professor(ProfID)**

(Each department has a chair who is a professor.)

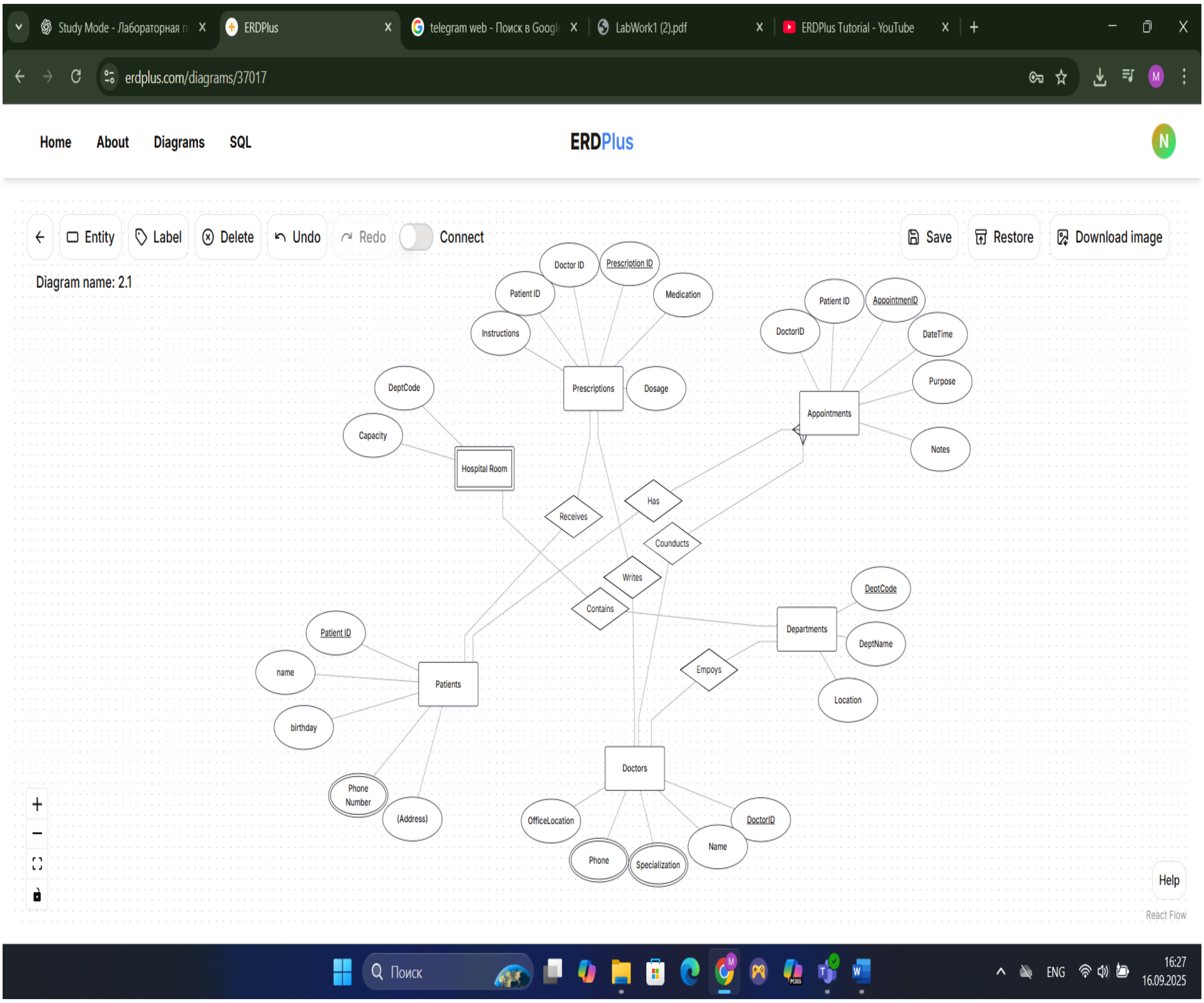
- **Enrollment(StudentID) → Student(StudentID)**

(Each enrollment record belongs to a student.)

- **Enrollment(CourseID) → Course(CourseID)**

(Each enrollment record belongs to a course.)

Task 2.1:


Study Mode - Лабораторная
ERDPlus
telegram web - Поиск в Google
LabWork1 (2).pdf
ERDPlus Tutorial - YouTube
erdplus.com/diagrams/37017
Home
About
Diagrams
SQL
ERDPlus
Entity
Label
Delete
Undo
Redo
Connect
Save
Restore
Download image
Diagram name: 2.1
Doctor ID
Prescription ID
Patient ID
Medication
Instructions
Prescriptions
Dosage
Patient ID
AppointmentID
DoctorID
DateTime
Purpose
Appointments
Notes
DeptCode
Capacity
Hospital Room
Receives
Has
Counducts
Writes
Contains
DeptCode
Departments
DeptName
Location
Empoys
Patient ID
name
Patients
birthday
Phone Number
(Address)
Doctors
OfficeLocation
DoctorID
Name
Phone
Specialization
Help
React Flow
Поиск
ENG
16:27
16.09.2025

Task 2.2:

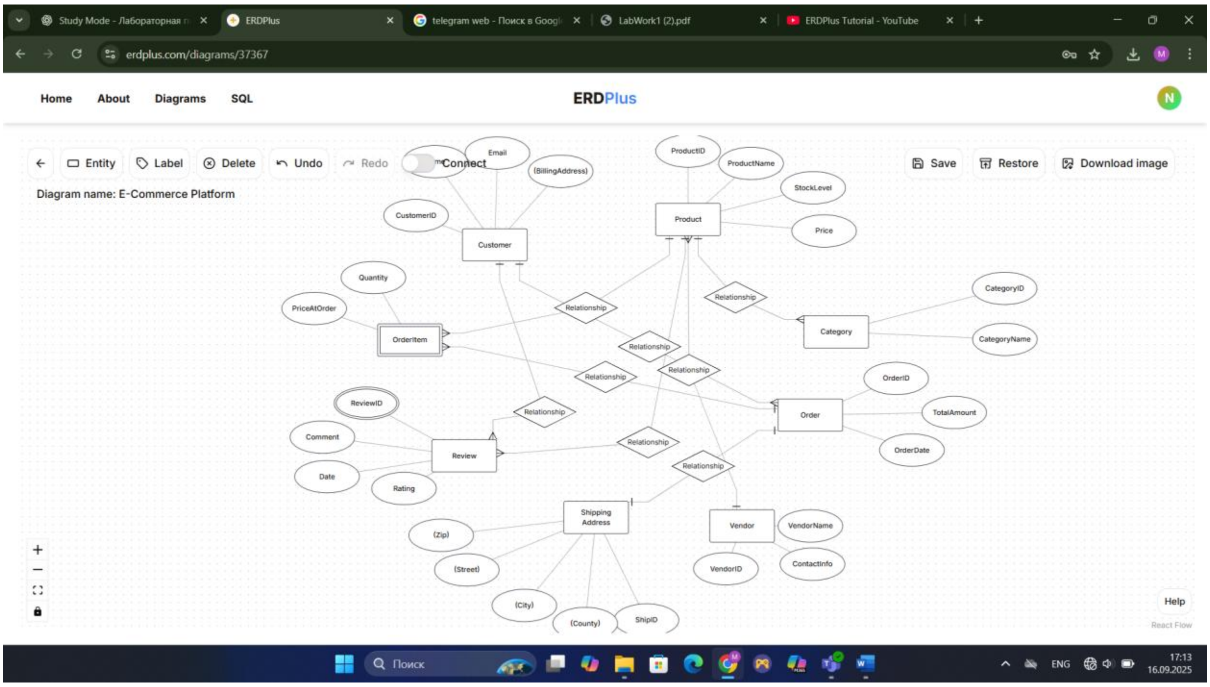


Diagram name: E-Commerce Platform

Task 4.1:

1. Functional Dependencies (FDs):

- StudentID → StudentName, StudentMajor
- ProjectID → ProjectTitle, ProjectType
- SupervisorID → SupervisorName, SupervisorDept
- (StudentID, ProjectID) → Role, HoursWorked, StartDate, EndDate

2. Problems (Redundancy & Anomalies):

- Redundancy: StudentName/Major and Supervisor data repeat in many rows.
- Update anomaly: Changing a student’s major requires multiple updates.
- Insert anomaly: Cannot add a student without a project.

- Delete anomaly: Deleting a student’s last project removes all info about that student.

3. 1NF:

- All attributes are atomic → Table is in 1NF.

4. 2NF:

- Primary key = (StudentID, ProjectID)
- Partial dependencies removed.
- **Decomposition:**
    1. Student(StudentID, StudentName, StudentMajor)
    2. Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)
    3. Supervisor(SupervisorID, SupervisorName, SupervisorDept)
    4. StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

5. 3NF:

- Transitive dependency: SupervisorID → SupervisorName, SupervisorDept
- Already separated into its own table (Supervisor).
- Final 3NF schema is:
    - Student(StudentID, StudentName, StudentMajor)
    - Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

  - Supervisor(SupervisorID, SupervisorName, SupervisorDept)
  - StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

Task 4.2:

1.Functional Dependencies:

Student ID ---Student Major

Course ID --- Course Name, Instructor ID

Instructor ID --- Instructor Name

Room --- Building

(Course ID, TimeSlot) --- Room

2. Candidate Keys

- The minimal candidate key is:
  (StudentID, CourseID, TimeSlot)
- Other attributes (InstructorName, Room, Building, etc.) are functionally dependent on this key, so they do not belong to the key.

---

3. Problems in Current Table

- Redundancy:

    - StudentMajor repeats for every student in many rows.
    - InstructorName repeats for every course taught by the same instructor.
    - Building repeats for every room.
- Update anomaly: changing a student’s major or instructor’s name requires updates in multiple rows.
- Insert anomaly: cannot add a new student without assigning a course and time.
- Delete anomaly: deleting the last row of a student removes their information entirely.

---

4. Decomposition to BCNF

Final schema in BCNF:

1. Student(StudentID, StudentMajor)
2. Instructor(InstructorID, InstructorName)
3. Course(CourseID, CourseName, InstructorID)
4. Room(Room, Building)
5. Schedule(CourseID, TimeSlot, Room)

6. Enrollment(StudentID, CourseID, TimeSlot)

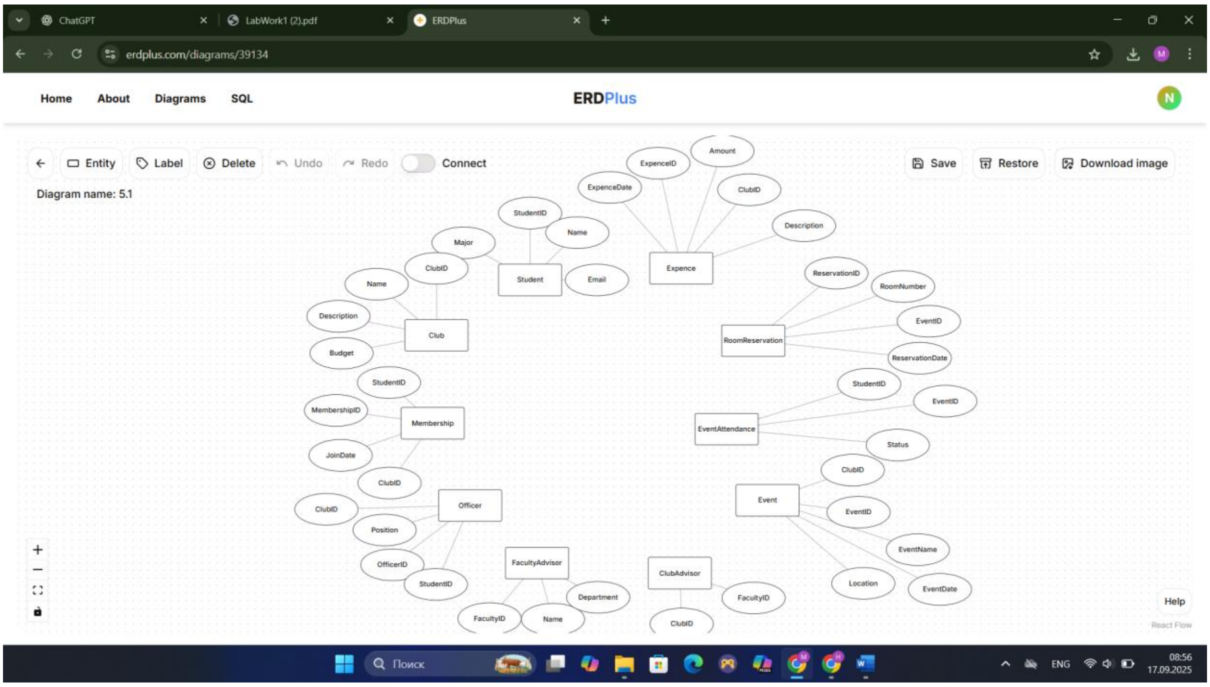
---

5. Result

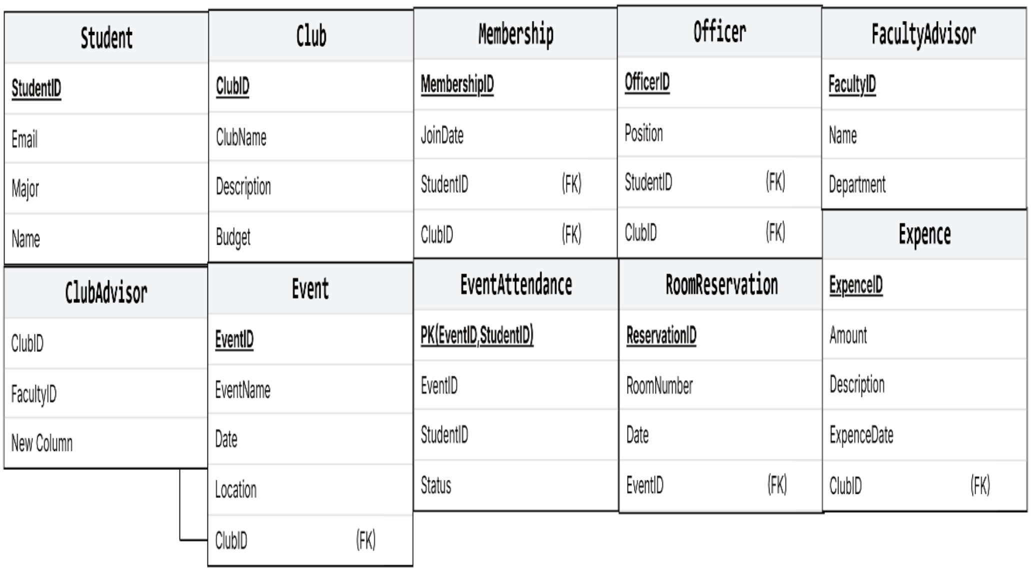
- All functional dependencies are preserved.
- Redundancy and anomalies are eliminated.
- The database schema is now in BCNF.

Task 5.1:

1.ER Diagram



2.Relational Schema:



Student
StudentID
Email
Major
Name

Club
ClubID
ClubName
Description
Budget

Membership
MembershipID
JoinDate
StudentID (FK)
ClubID (FK)

Officer
OfficerID
Position
StudentID (FK)
ClubID (FK)

FacultyAdvisor
FacultyID
Name
Department

ClubAdvisor
ClubID
FacultyID
New Column

Event
EventID
EventName
Date
Location
ClubID (FK)

EventAttendance
PK(EventID,StudentID)
EventID
StudentID
Status

RoomReservation
ReservationID
RoomNumber
Date
EventID (FK)

Expence
ExpenceID
Amount
Description
ExpenceDate
ClubID (FK)

3.

I created a separate Membership table instead of storing students directly inside the Club table or clubs inside the Student table.

The reason is that the relationship between students and clubs is many-to-many:

- One student can join several clubs.
- One club can have many students.

If this information were stored in one table, it would lead to data duplication and poor normalization. With a separate Membership table, the database remains normalized, and we can also store additional details such as the join date or membership status.

4.

1)Find all students who are officers in the Computer Science Club.

2)List all events scheduled for next week, including their room reservations.

3)Show each club's budget and the total expenses spent so far.