**Lab Document: Data Modeling Foundations and ERD Practices**

**Lab Title:** Data Modeling Fundamentals – From Requirements to Normalized ERDs

**Objective:**  
Learn how to translate real-world business scenarios into normalized ER diagrams through interviews, applying cardinality, atomicity, normalization (up to BCNF), naming standards, and version control using Git.

**Step-by-Step Lab with Sample Answers**

**Step 1: Requirements Interview Drill**

**Scenario:**  
Design a database for a university's online course registration system.

**Step 1.1: Draft Interview Questions**

* What information do we need to store about students?
* Can a student enroll in multiple courses?
* Can a course have multiple instructors?
* What data do we store for instructors?
* Do students need to provide course reviews?

**Step 1.2: Simulated Responses**

* Each student has ID, name, email, and department.
* A student can enroll in many courses.
* A course can be taught by one instructor.
* Instructors have ID, name, and specialization.
* Students can optionally submit reviews after course completion.

**Step 1.3: Identify Candidate Entities & Attributes**

* Student (student\_id, name, email, department)
* Course (course\_id, title, credits)
* Instructor (instructor\_id, name, specialization)
* Enrollment (student\_id, course\_id, enrollment\_date)
* Review (student\_id, course\_id, rating, comment)

**Step 1.4: Identify Relationships**

* Student ENROLLS IN Course
* Course IS TAUGHT BY Instructor
* Student WRITES Review for Course

**Step 2: ER Cardinality & Optionality**

**Step 2.1: Define Relationships**

* Student to Course → Many-to-Many (via Enrollment)
* Course to Instructor → Many-to-One
* Review to Student & Course → One-to-One per course per student

**Step 2.2: Optionality**

* Student must enroll in at least one course (optional: no)
* Course must have at least one student (optional: yes)
* Course must have an instructor (optional: no)
* Review is optional (students may choose to skip reviews)

**Sample Diagram (Textual)**

* Student (1) — (M) Enrollment (M) — (1) Course
* Course (1) — (1) Instructor
* Review (1) — (1) Student | Review (1) — (1) Course

**Step 3: Attribute Atomicity Check**

**Step 3.1: Original Attributes**

* student\_name → Not atomic
* address → Not atomic

**Step 3.2: Split to Atomic Attributes**

* student\_name → first\_name, last\_name
* address → street, city, state, zip

**Final Student Entity:**

* student\_id, first\_name, last\_name, email, department, street, city, state, zip

**Step 4: BCNF Walkthrough**

**Given Table:**  
| RegistrationID | StudentID | StudentName | CourseID | CourseName | InstructorName |

**Step 4.1: 1NF** – Ensure atomic values

**Step 4.2: 2NF** – Remove partial dependencies:

* Student(StudentID, StudentName)
* Course(CourseID, CourseName, InstructorName)
* Registration(RegistrationID, StudentID, CourseID)

**Step 4.3: 3NF** – Remove transitive dependencies:

* Instructor should be a separate entity.

**Step 4.4: BCNF** – All FDs have keys on the left:

* Final Tables:
  + Student(StudentID, StudentName)
  + Instructor(InstructorID, InstructorName)
  + Course(CourseID, CourseName, InstructorID)
  + Registration(RegistrationID, StudentID, CourseID)

**Step 5: Crow-foot ERD Tooling Practice**

**Instructions:**

1. Go to [dbdiagram.io](https://dbdiagram.io/)
2. Define tables in DBML or drag-n-drop.
3. Example DBML:

Table student {

student\_id int [pk]

first\_name varchar

last\_name varchar

email varchar

department varchar

}

Table instructor {

instructor\_id int [pk]

name varchar

specialization varchar

}

Table course {

course\_id int [pk]

title varchar

credits int

instructor\_id int [ref: > instructor.instructor\_id]

}

Table enrollment {

enrollment\_id int [pk]

student\_id int [ref: > student.student\_id]

course\_id int [ref: > course.course\_id]

enrollment\_date date

}

1. Export as PNG.

**Step 6: Naming Standards**

**Standard Naming Rules Applied:**

1. Singular table names (e.g., student not students)
2. snake\_case for attributes (e.g., first\_name)
3. Foreign keys use \_id suffix
4. Avoid special characters/spaces
5. All names in lowercase

**Sample Table with Standards:**

CREATE TABLE course (

course\_id INT PRIMARY KEY,

title VARCHAR(100),

instructor\_id INT

);

**Step 7: Versioning ERD in Git**

**Step 7.1: Initialize Git Repo**

git init

**Step 7.2: Add ERD & Commit**

git add erd.dbml erd.png

git commit -m "Initial ERD design"

**Step 7.3: Push to GitHub**

git remote add origin https://github.com/your-username/erd-project.git

git push -u origin main

**Bonus Challenge:**  
Add a new requirement: Students must complete a course to submit a review. Modify ERD, add constraints, update DBML, and push changes to Git.