Lecture Notes for Lecture 5 of CS 5200 (Database Management System) for the Summer 1, 2019 session at the Northeastern University Silicon Valley Campus.

Database Design and Normalization

Philip Gust,
Clinical Instructor
Department of Computer Science

Lecture 4 Review

- Codd’s database rules are twelve rules any database must obey to be regarded as truly relational.
- Key concepts of relational model include: *table*, *tuple*, *instance*, *schema*, *key*, and *attribute domain*
- Integrity constraints are conditions that must hold to be valid: key, domain, and referential integrity constraints
- Relational algebra is procedural language that takes instances and relations as input, outputs instances of relations.
- SQL is a natural-language-oriented language comprising data definition (DDL), data manipulation (DML), data query (DQL) and data control (DCL) sub-languages.
- Studying relational algebra expressions corresponding to SQL statements can help illuminate meaning of SQL statements.

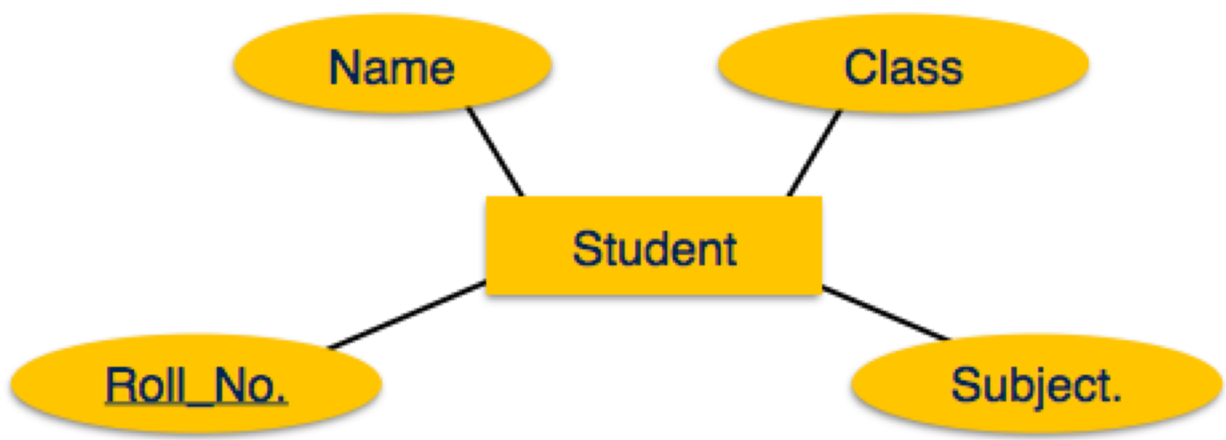
ER Model to Relational Model

- ER model gives a good overview of entity-relationship, which is easier to understand.
- ER diagrams can be mapped to relational schema. That is, it is possible to create relational schema using ER diagram.
- Not all ER constraints can be imported into relational model, but an approximate schema can be generated.
- Several processes available to convert ER diagrams into relational schema, some automated, some manual.
- ER diagrams mainly comprise of
  - Entity and its attributes
  - Relationship, which is association among entities.

ER Model to Relational Model

Mapping Entity

- An entity is a real-world object with some attributes



ER Model to Relational Model

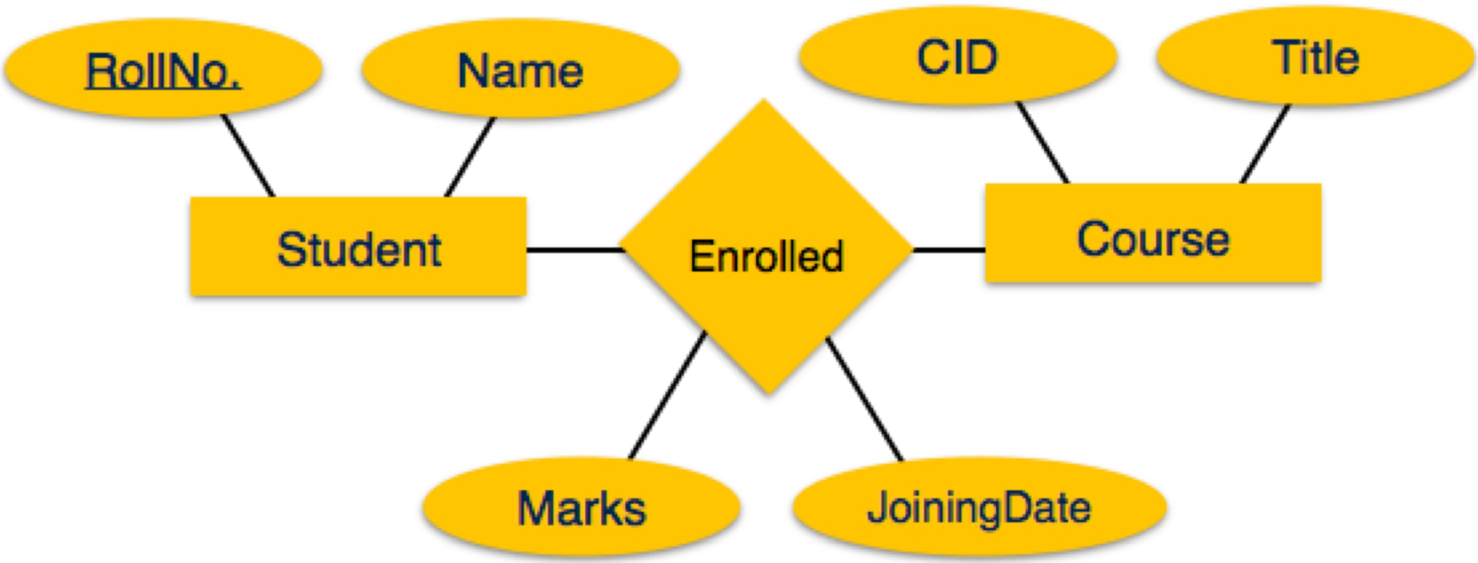
Mapping Entity

- Mapping Process (algorithm)
  - Create table for each entity
  - Entity's attributes should become fields of tables with their respective data types.
  - Declare primary key.

ER Model to Relational Model

Mapping Relationship

- A relationship is an association among entities.



ER Model to Relational Model

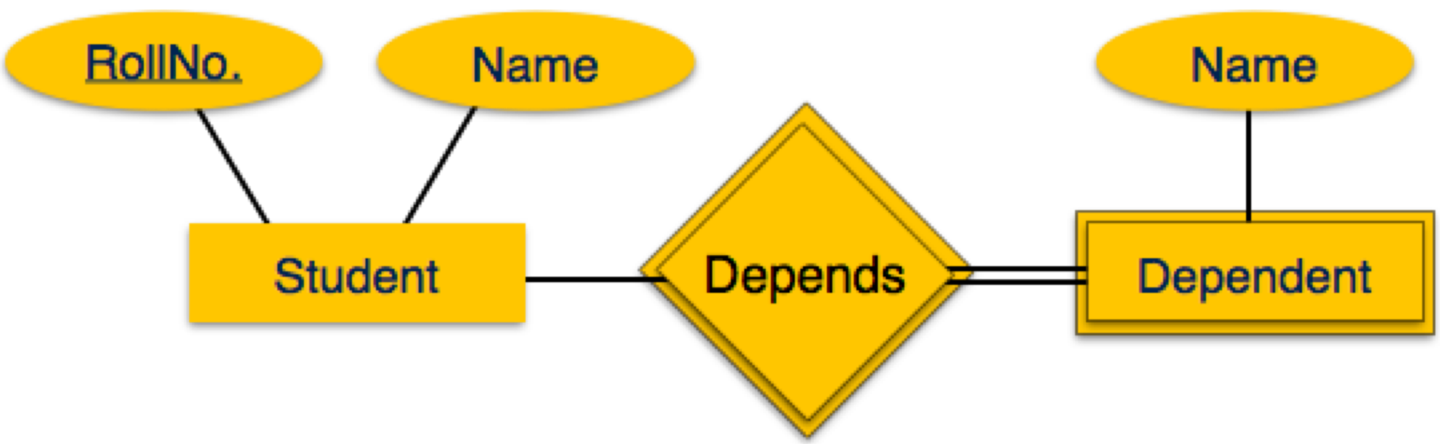
Mapping Relationship

- Mapping Process (algorithm)
  - Create table for a relationship.
  - Add the primary keys of all participating Entities as fields of table with their respective data types.
  - If relationship has any attribute, add each attribute as field of table.
  - Declare a primary key composing all the primary keys of participating entities.
  - Declare all foreign key constraints.

ER Model to Relational Model

Mapping Weak Entity Sets

- A weak entity set is one which does not have any primary key associated with it.



ER Model to Relational Model

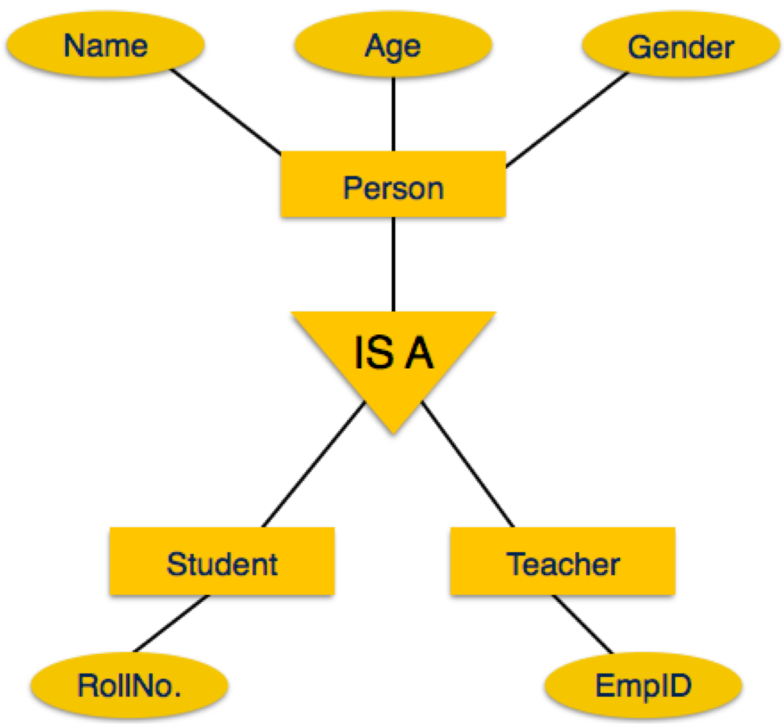
Mapping Weak Entity Sets

- Mapping Process (algorithm)
  - Create table for weak entity set.
  - Add all its attributes to table as field.
  - Add the primary key of identifying entity set.
  - Declare all foreign key constraints.

ER Model to Relational Model

Mapping Hierarchical Entities

- ER specialization or generalization comes in the form of hierarchical entity sets.



ER Model to Relational Model

Mapping Hierarchical Entities

- Mapping Process (algorithm)
  - Create tables for all higher-level entities.
  - Create tables for lower-level entities.
  - Add primary keys of higher-level entities in the table of lower-level entities.
  - In lower-level tables, add all other attributes of lower-level entities.
  - Declare primary key of higher-level table and the primary key for lower-level table.
  - Declare foreign key constraints.

Functional Dependency

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y.
- The left-hand side attributes determine the values of attributes on the right-hand side.

Functional Dependency

Armstrong's Axioms

- If F is a set of functional dependencies then the closure of F, denoted as F^+, is the set of all functional dependencies logically implied by F.
- Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

Functional Dependency

Armstrong's Axioms

- **Reflexive rule** − If alpha is a set of attributes and beta is_subset_of alpha, then alpha holds beta.
- **Augmentation rule** − If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.
- **Transitivity rule** − Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

Functional Dependency

Trivial Functional Dependencies

- **Trivial** − If a functional dependency (FD) X → Y holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.
- **Non-trivial** − If an FD X → Y holds, where Y is not a subset of X, then it is called a non-trivial FD.
- **Completely non-trivial** − If an FD X → Y holds, where X intersect Y = Φ, it is said to be a completely non-trivial FD.

Normalization

- If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.
- Normalization is a method to remove these anomalies and bring the database to a consistent state.
    - **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
    - **Insert anomalies** − We tried to insert data in a record that does not exist at all.

Normalization

- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations.
  - Example: when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.

Normalization

First Normal Form

- First Normal Form is defined in the definition of relations (tables) itself.
- This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

Normalization

- Re-arrange the relation (table) as below, to convert it to First Normal Form.
- Each attribute must contain only a single value from its pre-defined domain.

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

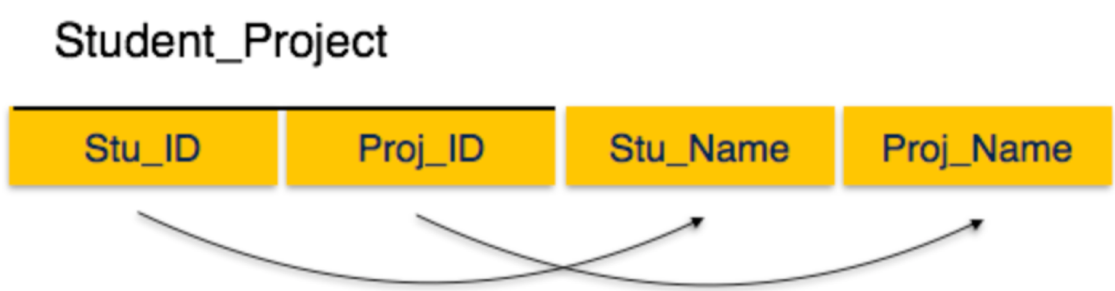
Normalization

Second Normal Form

- Before we learn about the second normal form, we need to understand the following:
  - **Prime attribute** − An attribute, which is a part of the prime-key, is known as a prime attribute.
  - **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

Normalization

- If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.
- That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



Normalization

- In Student_Project relation that the prime key attributes are Stu_ID and Proj_ID.
- According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually.
- But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently.
- This is called **partial dependency**, which is not allowed in Second Normal Form.

Normalization

- Brake relation in two so there exists no partial dependency.

Student

Stu_ID	Stu_Name	Proj_ID

Project

Proj_ID	Proj_Name

Normalization

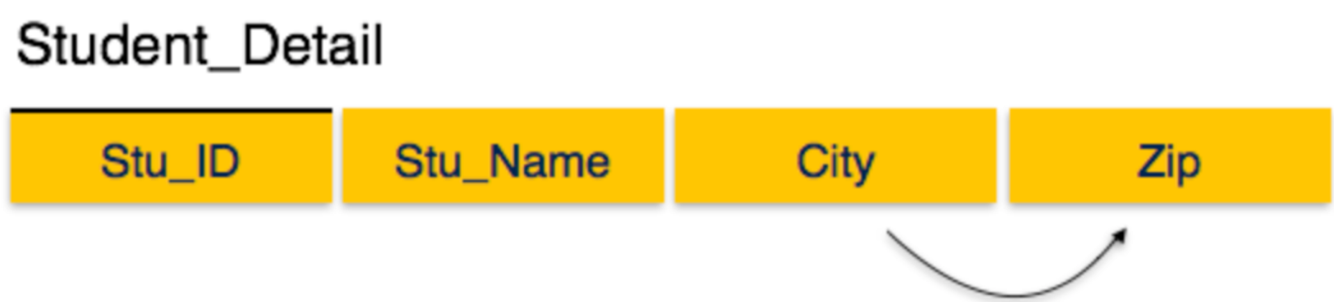
Third Normal Form

- For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy:
  - No non-prime attribute is transitively dependent on prime key attribute.
  - For any non-trivial functional dependency, X → A, then either –
    - X is a superkey or,
    - A is prime attribute.

Normalization

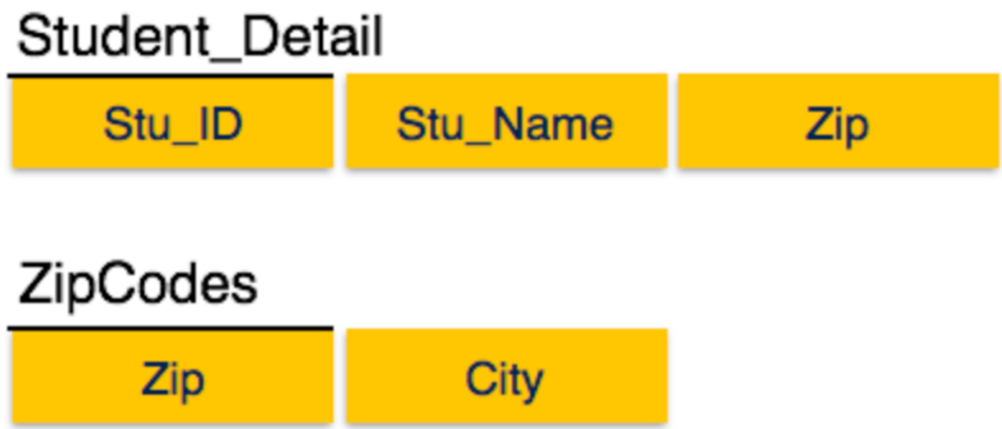
- In the following Student_detail relation, Stu_ID is the key and only prime key attribute. City can be identified by Stu_ID as well as Zip itself.

- Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.



Normalization

- To bring this relation into third normal form, we break the relation into two relations:



Normalization

Boyce-Codd Normal Form

- Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms.
  - For any non-trivial functional dependency, X → A, X must be a super-key.
  - In the earlier image , Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,
    - Stu_ID → Stu_Name, Zip
    - and
    - Zip → City
  - Which confirms that both the relations are in BCNF.

JDBC

In-Class Demonstrations Using Wikipedia Example

- Create and connect to database
- Create tables
- Drop tables
- Insert records
- Delete records
- Select records