CSE370:Database Systems

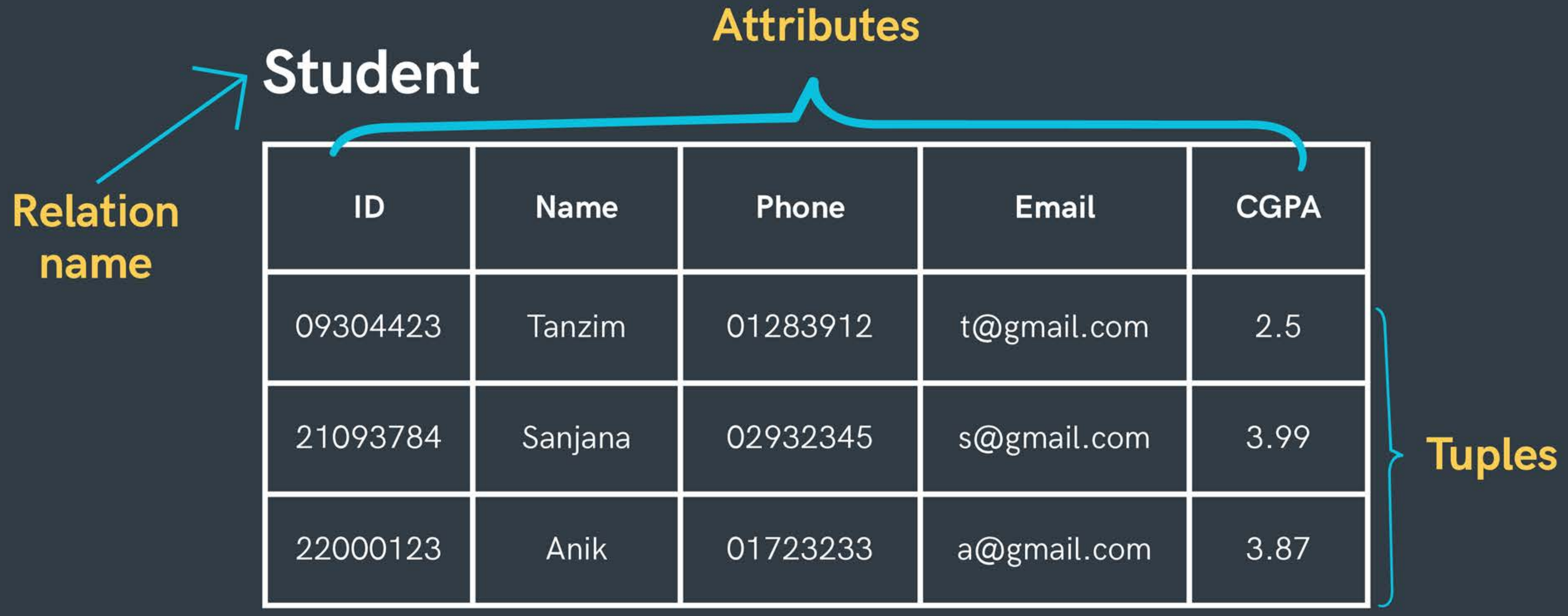
LECTURE 4: RELATIONAL DATA MODEL & DATABASE CONSTRAINTS

Relational Model Concepts

The relational Model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper: "A Relational Model for Large Shared Data Banks". This paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award.

The relational model is based on the concept of sets.

In a relational model, data is organized in tables (formally called a relation). The relation/tables have columns with a column header. The column headers are formally called attributes. The data is stored in rows (formally called tuples) which represent real world facts that correspond to entities or relationships.



Student

ID	Name	Phone	Email	CGPA
09304423	Tanzim	01283912	t@gmail.com	2.5
21093784	Sanjana	02932345	s@gmail.com	3.99
22000123	Anik	01723233	a@gmail.com	3.87

Relation Schema & State

The description of the table is formally called the **relation schema.** Formally the relation schema is denoted in the format **R(A1, A2, A3....An)** where R is the name of the relation and A1, A2 ....An are the attributes.

For example, the schema for the given relation is denoted by Student(ID, Name, Phone, Email, CGPA). Each attribute has a set of possible values that it can take, known as the **domain** of the attribute.

The **relation state** is the data stored in the relation at a given moment in time.

The tuples of a relation are not ordered, state remains same after changing ordering of tuples. But the attributes of a relation are ordered, as changing the order, changes the schema of the relation.

Student

ID	Name	Phone	Email	CGPA
09304423	Tanzim	01283912	t@gmail.com	2.5
21093784	Sanjana	02932345	s@gmail.com	3.99
22000123	Anik	01723233	a@gmail.com	3.87

"Key" Concepts

Superkey of a relation R is an attribute or set of attributes of R which will have a unique value for each tuple.

Key of R is a minimal superkey, such that every attribute in the superkey MUST be necessary to make it unique, if even after removal of an attribute from the superkey, it retains the uniqueness, then it is a superkey, but not a key.

For example, K1 = {ID, Name} or K2= {ID, Email} in Student is a superkey, but not a key. In Grades table, K3 = {SID, Ccode} is both a superkey and a key.

Student

ID	Name	Email
09304423	Tanzim	t@gmail.com
21093784	Sanjana	s@gmail.com
22000123	Anik	a@gmail.com

Course

CCode	Title
CSE110	PL1
CSE221	Algo
CSE370	Anik

Grades

SID	Ccode	Grade
09304423	CSE110	A
21093784	CSE110	B+
09304423	CSE370	A

"Key" Concepts

A relation R may have several keys, they are all **Candidate Keys** (e.g. {ID} or {Email} in student). Only one of the candidate key is chosen to become the **Primary Key**. The primary key is used to uniquely identify each tuple in a relation, also for referencing, searching etc. Standard practice is to choose the one with smallest value e.g. {ID}. Each relation will have only one primary key which can be composed of several attributes, e.g. {SID, Ccode} in Grades table.

Foreign Keys are references to primary keys of a different relation (or table). for example, {SID} in Grade references {ID} from Student.

Student

<u>ID</u>	Name	Email
09304423	Tanzim	t@gmail.com
21093784	Sanjana	s@gmail.com
22000123	Anik	a@gmail.com

Course

<u>CCode</u>	Title
CSE110	PL1
CSE221	Algo
CSE370	Anik

Grades

<u>SID</u>	<u>Ccode</u>	Grade
09304423	CSE110	A
21093784	CSE110	B+
09304423	CSE370	A

Relational Integrity Constraints

Relational Integrity Constraints are conditions that must hold on all valid relation states. By applying these conditions, databases uphold their structural integrity, preventing invalid data from entering the system and enhancing overall reliability. There are four constraints in a relational model:

- Domain Constraint
- Key Constraint
- Entity Integrity Constraint
- Referential Integrity Constraint

Domain Constraint

Every value in a tuple **must** be from the domain of its attribute (or it could be null, if allowed for that attribute). Domain is the set of all possible values for each attribute/column.

The domain includes the data type (such as integer or String) and optionally may include formats and ranges. For example, the "Age" of student has to be a positive integer below 100 or the domain of "CCode" is a String where the first few characters are letters while the last few are digits. So, if a value such as "370CSE" was inserted in "CCode" column, or -5 was inserted in the "Age" column, then it will violate the domain constraint and the data will become invalid.

Student

ID	Name	Age
09304423	Tanzim	20
21093784	Sanjana	18
22000123	Anik	-5

Course

CCode	Title
CSE110	PL1
CSE221	Algo
370CSE	Anik

Key Constraint

Every value for a key **must** be unique, thus no two tuples in a a relation R can have the same value for the keys.

Therefore, ID or Email of Student cannot have duplicate entries. In the given examples, the SID, Ccode pair (09304423, CSE110) in Grades table is inserted twice, thus it violated the key constraint. A similar violation can be observed in the Student table.

Student

ID	Name	Email
09304423	Tanzim	s1@g.com
21093784	Sanjana	s2@g.com
21093784	Anik	s3@g.com

Grades

SID	Ccode	Grade
09304423	CSE110	A
21093784	CSE110	B+
09304423	CSE110	A

Entity Integrity Constraint

The primary key attributes of each relation R **cannot** have **null values** in any tuple. If the primary key has several attributes, then none of the attributes can have null values.

In the given examples, "ID" values in the student table cannot be null. Also in the Grades table "Ccode" is part of the primary key {SID, Ccode}, therefore, the null vlaue is violating the key constraint.

Note: Other attributes of a relation may be constrained to disallow null values, even though they are not members of the primary key.

Student

ID	Name	Email
09304423	Tanzim	s1@g.com
21093784	Sanjana	s2@g.com
null	Anik	s3@g.com

Grades

SID	Ccode	Grade
09304423	CSE110	A
21093784	null	B+
09304423	CSE370	A

Referential Integrity Constraint

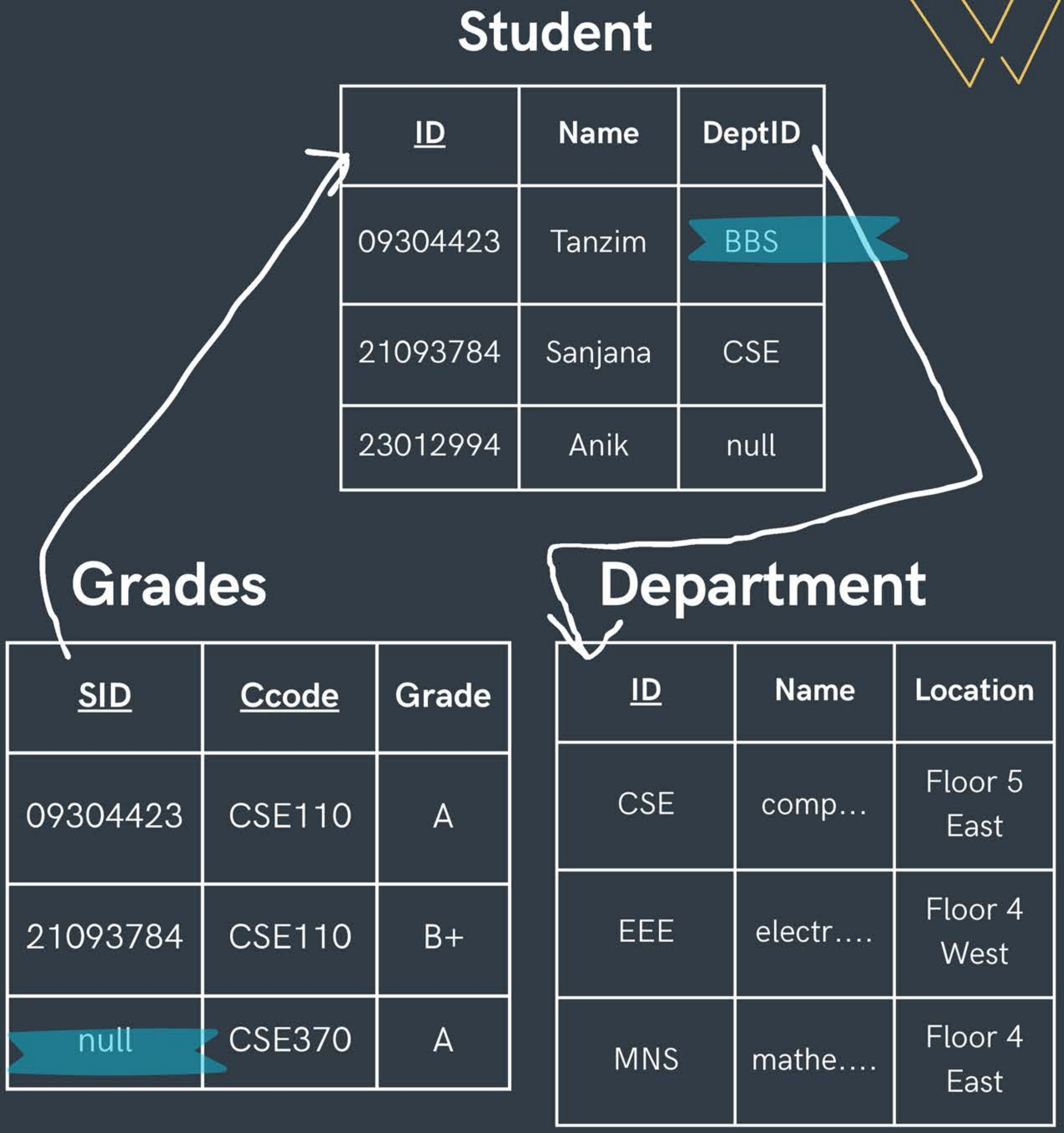
The value in the foreign key column (or columns) of the referencing table can be either:

(1) **an existing value** in the primary key column (or columns) of the referenced table.

OR

(2) **a null** (but only if the foreign key is not part of the primary key in the referencing table)

E.g. DeptID of Student can have one of the three values existing in Department table or it can be null. The value 'BBS' is a violation as it does not exist in the Department table. SID in Grades, can only have values that exists in ID column of Student, but not null, as SID is also a primary key of Grades.



Student

<u>ID</u>	Name	DeptID
09304423	Tanzim	BBS
21093784	Sanjana	CSE
23012994	Anik	null

Grades

<u>SID</u>	<u>Ccode</u>	Grade
09304423	CSE110	A
21093784	CSE110	B+
null	CSE370	A

Department

<u>ID</u>	Name	Location
CSE	comp…	Floor 5 East
EEE	electr….	Floor 4 West
MNS	mathe….	Floor 4 East

Enforcing Relational Integrity Constraints

Relational Integrity Constraints should not be violated when performing operations to manipulate the state of a relation (or table). In order to prevent violation, all constraints must be enforced. The three data manipulation operations are

- Insert (A new row is inserted)
- Delete (An existing row is deleted)
- Update (A specific value is modified)

Data Insertion

An **INSERT** operation may violate any of the four constraints:

Domain Constraint: Inserting values outside of the domain by mistake. To enforce the constraint, the data type must be specified. Additionally the check constraint, not null constraint, regular expressions and application side validation can be used depending on the requirement.

Key Constraint: Inserting duplicate values by mistake is possible. Declaring the primary key and using the unique keyword (for other candidate keys) will enforce the key constraint.

Entity Integrity Constraint: Inserting null values for primary keys by mistake. Declaring the primary key is sufficient to prevent violation.

Referential Integrity Constraint: Inserting non-existent values for foreign key columns. Declaring foreign keys with proper references will enforce the constraint

```
CREATE TABLE Student (
  ID INT,
  Name VARCHAR(100) NOT NULL,
  EmailAdd VARCHAR(100) UNIQUE,
  CGPA DECIMAL(3,2),
  Dept CHAR(3),
  PRIMARY KEY (ID),
  FOREIGN KEY (Dept) REFERENCES
Department(DepID),
  CHECK (CGPA >= 0.00 AND CGPA <=
4.00)
);
```

Data Deletion

A **DELETE** operation is removing a row from the table, it can only violate one contraint:

Referential Integrity Constraint: If the primary key value of a deleted row is referenced by another table then referential integrity will be violated. E.g. if a student record is deleted and that student is referenced from the Grades table, then it will violate referential integrity constraint. To enforce this constraint, first of all foreign key must be declared, then, the “on delete” clause can be used to specify one of three options:

Restrict → Does not allow deletion from the referenced table.
Cascade → Deletes all rows from the referencing the deleted primary key.
Set Null → Sets the foreign key of the referencing row to null (only possible if the foreign key is not part of the primary key in the referencing table).

```
CREATE TABLE Grades (
  ID INT,
  Ccode CHAR(6),
  Grade VARCHAR(2),
  PRIMARY KEY (ID, Ccode),
  FOREIGN KEY (ID) REFERENCES
Student(ID) ON DELETE Cascade,
  FOREIGN KEY (Ccode) REFERENCES
Course(Ccode) ON DELETE Restrict,
);
```

Data Modification

An **UPDATE** operation modifies a stored value in a table. The constraint violation will depend on type of attribute being modified:

Domain Constraint: May be violated when updating any value. Should be enforced in the same way as the "insert" operation.

Key and Entity Integrity Constraint: May be violated when updating primary key/candidate key values. Should be enforced by declaring the primary key and unique constraints.

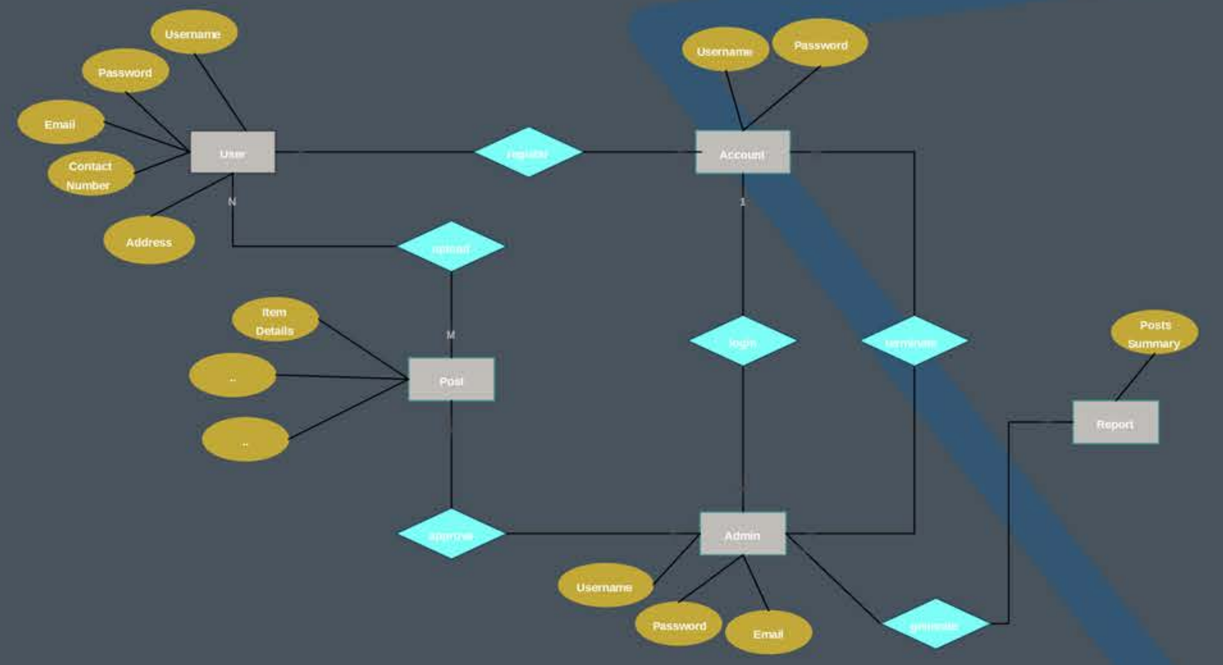
Referential Integrity Constraint: May be violated if the primary key value is updated and it is referenced by another table. To enforce this constraint, first of all foreign key must be declared, then, the "on update" clause can be used to specify one of three options: Restrict → Does not allow updates. Cascade →Updates all rows referencing the modified value. Set Null → Sets the foreign key of the referencing row to null (only possible if the foreign key is not part of the primary key in the referencing table).

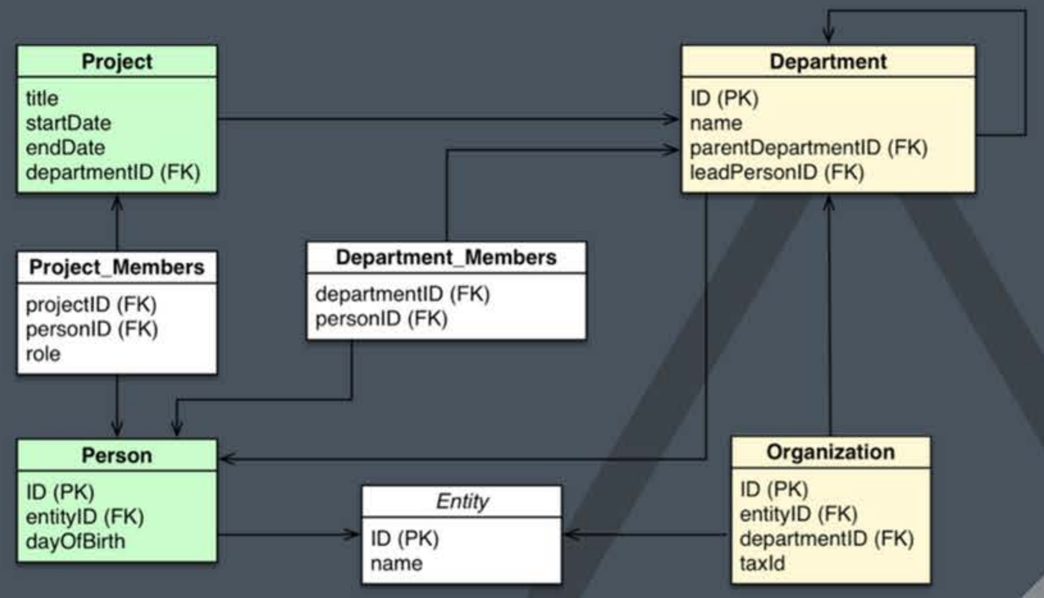
```
CREATE TABLE Grades (
  ID INT,
  Ccode CHAR(6),
  Grade VARCHAR(2),
  PRIMARY KEY (ID, Ccode),
  FOREIGN KEY (ID) REFERENCES
Student(ID) ON UPDATE Cascade,
  FOREIGN KEY (Ccode) REFERENCES
Course(Ccode) ON DELETE Cascade
ON UPDATE Cascade,
);
```

What Next?

LECTURE 5: RELATIONAL SCHEMA MAPPING

LOADING........


Username
Password
Email
Contact Number
Address
User
register
Account
Username
Password
N
upload
1
M
Item Details
..
..
Post
login
terminate
Posts Summary
Report
approve
Admin
Username
Password
Email
generate


Project
title
startDate
endDate
departmentID (FK)
Department
ID (PK)
name
parentDepartmentID (FK)
leadPersonID (FK)
Project_Members
projectID (FK)
personID (FK)
role
Department_Members
departmentID (FK)
personID (FK)
Person
ID (PK)
entityID (FK)
dayOfBirth
Entity
ID (PK)
name
Organization
ID (PK)
entityID (FK)
departmentID (FK)
taxId