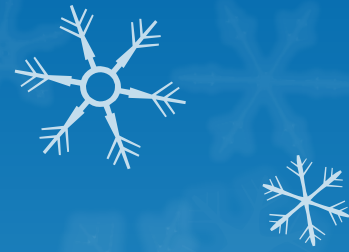


**Course Name:**  
**Database Management**  
**Systems**



# Lecture 6

## Topics to be covered

- ❑ Integrity Constraints



# Integrity Constraints (ICs)


- **IC:** condition that must be true for *any* instance of the database; e.g., *domain constraints*.
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!

# Primary Key Constraints


- A set of fields is a key for a relation if :
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.
- Part 2 false? A *superkey*.
- If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the *primary key*.
- E.g., *sid* is a key for Students. (What about *name*?)  
The set {*sid*, *gpa*} is a superkey.

# Primary and Candidate Keys in SQL


- ❖ “For a given student and course, there is a single grade.” vs. “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”
- ❖ Used carelessly, an IC can prevent the storage of database instances that arise in practice!
- Possibly many *candidate keys* (specified using **UNIQUE**), one of which is chosen as the *primary key*.



```
CREATE TABLE Enrolled
(sid CHAR(20)
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid) )
```



```
CREATE TABLE Enrolled
(sid CHAR(20)
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid),
UNIQUE (cid, grade))
```



# Foreign Keys, Referential Integrity

- Foreign key : Set of fields in one relation that is used to 'refer' to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a 'logical pointer'.

- E.g. *sid* is a foreign key referring to **Students**:

- Enrolled(*sid*: string, *cid*: string, *grade*: string)
- If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references.
- Can you name a data model w/o referential integrity?
  - Links in HTML!

# Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses.

```
CREATE TABLE Enrolled
(sid CHAR(20), cid CHAR(20), grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid) REFERENCES Students )
```

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8



# Enforcing Referential Integrity

- Consider Students and Enrolled; *sid* in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted? (*Reject it!*)
- What should be done if a Students tuple is deleted?
  - Also delete all Enrolled tuples that refer to it.
  - Disallow deletion of a Students tuple that is referred to.
  - Set *sid* in Enrolled tuples that refer to it to a *default sid*.
  - (In SQL, also: Set *sid* in Enrolled tuples that refer to it to a special value *null*, denoting *'unknown'* or *'inapplicable'*.)
- Similar if primary key of Students tuple is updated.

# Referential Integrity in SQL



- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
  - Default is **NO ACTION** (*delete/update is rejected*)
  - CASCADE** (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT** (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET
DEFAULT )
```

# Where do ICs Come From?



- ICs are based upon the semantics of the real-world enterprise that is being described in the database relations.
- We can check a database instance to see if an IC is violated, but we can **NEVER** infer that an IC is true by looking at an instance.
  - An IC is a statement about *all possible* instances!
  - From example, we know *name* is not a key, but the assertion that *sid* is a key is given to us.
- Key and foreign key ICs are the most common; more general ICs supported too.

