



Introduction to Database System

ER to Relational

Why Study the Relational Model?



- Most widely used model.
 - Vendors: IBM, Informix, Microsoft, Oracle, Sybase, etc.

Relational Database: Definitions



- **Relational database**: a set of **relations**
- **Relation**: made up of 2 parts:
 - **Instance**: a **table**, with rows and columns.
#Rows = cardinality, #fields = degree / arity.
 - **Schema**: specifies name of relation, plus name and type of each column.
 - E.G. Students(sid: string, name: string, login: string, age: integer, gpa: real).
- Can think of a relation as a **set** of rows or **tuples** (i.e., all rows are distinct).

Example Instance of Students Relation

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

- ❖ Cardinality = 3, degree = 5, all rows distinct
- ❖ Do all columns in a relation instance have to be distinct?

Creating Relations in SQL

- Creates the Students relation. Observe that the type (**domain**) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.
- As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Students  
(sid: CHAR(20),  
name: CHAR(20),  
login: CHAR(20),  
age: INTEGER,  
gpa: REAL)
```

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

Destroying and Altering Relations

DROP TABLE Students

- Destroys the relation Students. The schema information and the tuples are deleted.

ALTER TABLE Students

ADD COLUMN first Year: integer

- ❖ The schema Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)  
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- ❖ Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE  
FROM Students S  
WHERE S.name = 'Smith'
```

Integrity Constraints (ICs)



- **IC:** condition that must be true for *any* instance of the database
 - ICs are specified when schema is defined.
 - ICs are checked when relations are modified.
- A *legal* instance of a relation is one that satisfied all specified ICs.
 - DBMS should not allow illegal instances.
- If the DBMS check 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

- Possibly many candidate keys (specified using **UNIQUE**), one of which is chosen as the primary key.

❖ “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!

```
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.

Foreign Keys in SQL

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

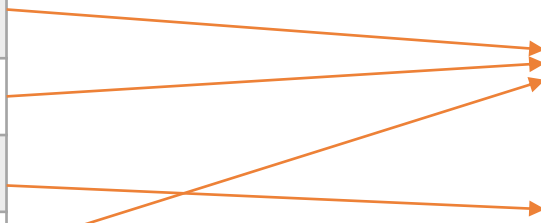
```
CREATE TABLE Enrolled  
(sid CHAR(20), cid CHAR(20), grade CHAR(2),  
PRIMARY KEY (sid, cid),  
FOREIGN KEY (sid) REFERENCES Students)
```

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 Referencial Integrity (1/2)



- 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 is inserted?

Enforcing Referencial Integrity (2/2)



- **What should be done if a Student's tuple is deleted?**
 - Also delete all Enrolled tuples that refer to it.
 - Disallow deletion of a Student's 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'.)

Referential Integrity in SQL

- SQL 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)
```