CS W4111.001
Introduction to Databases
Fall 2022
Computer Science Department
Columbia University
The Relational Model
Why Study the Relational Model?

• Most widely used model, by far:
  IBM, Microsoft, Oracle, MySQL, PostgreSQL, ...
• Older models (hierarchical, network) still around in legacy systems
• Other alternative models: object-oriented, object-relational, NoSQL, NewSQL, ...
Relational Database: A Set of Relations

- A relation consists of 2 parts:
  - **Schema**: name of relation, name and type (or domain) of each column of the relation
    
    Students(sid: string, name: string, login: string, age: integer, gpa: real)

  - **Instance**: a table, with rows (tuples) and columns
    
    # rows = cardinality
    
    # fields (or attributes) = degree/arity

- A relation can be regarded (for now) as a set of rows, without repetition (i.e., all rows are distinct)
Example Instance of Students Relation

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

- Cardinality = 4
- Degree = 5
- All rows (or tuples) distinct
Creating Relations in SQL

- Creates **Students** relation; type (domain) of each attribute is specified, and enforced by DBMS when tuples added or modified; attribute order matters

  `CREATE TABLE Students`  
  `(sid CHAR(20),`  
  `name CHAR(30),`  
  `login CHAR(20),`  
  `age INTEGER,`  
  `gpa REAL)`

- Creates **Courses** relation, with information about available courses

  `CREATE TABLE Courses`  
  `(cid CHAR(10),`  
  `cname CHAR(10),`  
  `credits INTEGER)`
Creating Relations in SQL

- Creates **Enrolled** relation, with information about courses that students take, and the corresponding grades

  CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(10),
   grade CHAR(2))
Adding and Deleting Tuples

Inserting a single tuple:

```
INSERT INTO Students(sid, name, login, age, gpa)
VALUES (8, 'Nico', 'nico@cs', 24, 3.9)
```

Can delete all tuples satisfying some condition (e.g., name="Nico"):

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

Powerful variants of these commands are available; more later!
Integrity Constraints

- Integrity constraints are conditions (e.g., domain constraints, etc.) that must be true for any instance of the database
  - Specified when schema is defined
  - Checked when relations are modified

- A legal instance of a relation is one that satisfies all specified integrity constraints
  - DBMS will not allow illegal instances, so that stored data preserves intended real-world meaning
Primary and Candidate Key Constraints

A set of attributes of a relation is a key for the relation if:

1. No two distinct tuples can have the same values in all key attributes
2. Property 1 is not true for any strict subset of the key (i.e., we need all attributes in the set to satisfy the previous point: minimality)
Primary and Candidate Key Constraints

A set of attributes of a relation is a key for the relation if:
1. No two distinct tuples can have the same values in all key attributes
2. Property 1 is not true for any strict subset of the key (i.e., we need all attributes in the set to satisfy the previous point: minimality)

- A set of attributes that satisfies Property 1 is a superkey
- All keys are superkeys but the converse is not true
  {sid} is a key for Students, and {sid, gpa} is a superkey but not a key for Students
- If a relation has multiple keys, one is designated as the primary key and the rest are candidate keys
Primary and Candidate Keys in SQL

Each relation has one PRIMARY KEY, and zero or more candidate keys specified with the UNIQUE keyword.

Two different students cannot share the same value of sid or login; {sid} is the primary key, {login} is a candidate key.

Two different courses cannot share the same value of cid; {cid} is the primary key.

```
CREATE TABLE Students
    (sid CHAR(20),
     name CHAR(30),
     login CHAR(20),
     age INTEGER,
     gpa REAL,
     PRIMARY KEY(sid),
     UNIQUE(login))

CREATE TABLE Courses
    (cid CHAR(10),
     cname CHAR(10),
     credits INTEGER,
     PRIMARY KEY(cid))
```
Primary and Candidate Keys in SQL

“Each student can only take a given course once”

CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(10),
     grade CHAR(2),
     PRIMARY KEY( )
     )
CREATE TABLE Students
(sid CHAR(20),
name CHAR(30),
login CHAR(20),
age INTEGER,
gpa REAL,
PRIMARY KEY(sid),
UNIQUE(login))

CREATE TABLE Courses
(cid CHAR(10),
cname CHAR(10),
credits INTEGER,
PRIMARY KEY(cid))

CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(10),
grade CHAR(2),
PRIMARY KEY(sid, cid))

Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

Courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>
Primary and Candidate Keys in SQL: One Important Difference

- Primary key attributes cannot be “NULL” (much more on this later in the course)
- Candidate key attributes can be “NULL”

```sql
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(30),
 login CHAR(20),
 age INTEGER,
 gpa REAL,
 PRIMARY KEY(sid),
 UNIQUE(login))
```
Foreign Keys, Referential Integrity

• Attributes of one relation can reference tuples of another relation
• To avoid “dangling” references and guarantee referential integrity, we use “foreign keys”
• A foreign key is a set of attributes in one relation that is used to refer to tuples in another relation
• The foreign key attributes in a relation correspond to the primary key attributes in the referred to relation
Foreign Keys in SQL

Only students listed in the Students relation should be allowed to enroll for courses; analogous constraint for Courses

```
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(30),
 login CHAR(20),
 age INTEGER,
 gpa REAL,
 PRIMARY KEY(sid),
 UNIQUE(login))
```

```
CREATE TABLE Courses
(cid CHAR(10),
 cname CHAR(10),
 credits INTEGER,
 PRIMARY KEY(cid))
```

```
CREATE TABLE Enrolled
(sid CHAR(20),
 cid CHAR(10),
 grade CHAR(2),
 PRIMARY KEY (sid, cid),
 FOREIGN KEY (sid) REFERENCES Students,
 FOREIGN KEY (cid) REFERENCES Courses)
```
### Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

### Courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

### Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>
CREATE TABLE Students (sid CHAR(20), name CHAR(30), login CHAR(20), age INTEGER, gpa REAL, PRIMARY KEY(sid), UNIQUE(login))

CREATE TABLE Courses (cid CHAR(10), cname CHAR(10), credits INTEGER, PRIMARY KEY(cid))

CREATE TABLE Enrolled (sid CHAR(20), cid CHAR(10), grade CHAR(2), PRIMARY KEY(sid, cid), FOREIGN KEY (sid) REFERENCES Students, FOREIGN KEY (cid) REFERENCES Courses)

INSERT INTO Students (sid, name, login, age, gpa) VALUES (NULL, 'Wisam', 'w@cs', 33, 4.0)
### Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

### Courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

### Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>

---

**Insertion Example**

```sql
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (NULL, 'Wisam', 'w@cs', 33, 4.0);
```

FAILS (PRIMARY KEY attributes cannot be NULL)

---

**Create Table Examples**

```sql
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(30),
 login CHAR(20),
 age INTEGER,
gpa REAL,
 PRIMARY KEY(sid),
 UNIQUE(login))

CREATE TABLE Courses
(cid CHAR(10),
cname CHAR(10),
 credits INTEGER,
 PRIMARY KEY(cid))

CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(10),
 grade CHAR(2),
 PRIMARY KEY(sid, cid),
 FOREIGN KEY (sid) REFERENCES Students,
 FOREIGN KEY (cid) REFERENCES Courses)
```
CREATE TABLE Students (sid CHAR(20), name CHAR(30), login CHAR(20), age INTEGER, gpa REAL, PRIMARY KEY(sid), UNIQUE(login))

CREATE TABLE Courses (cid CHAR(10), cname CHAR(10), credits INTEGER, PRIMARY KEY(cid))

CREATE TABLE Enrolled (sid CHAR(20), cid CHAR(10), grade CHAR(2), PRIMARY KEY(sid, cid), FOREIGN KEY (sid) REFERENCES Students, FOREIGN KEY (cid) REFERENCES Courses)

Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

Courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>

UPDATE Students S
SET S.sid = 6
WHERE S.sid = 5?
Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>6</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>

CREATE TABLE Students
(sid CHAR(20),
name CHAR(30),
login CHAR(20),
age INTEGER,
gpa REAL,
PRIMARY KEY(sid),
UNIQUE(login))

CREATE TABLE Courses
(cid CHAR(10),
cname CHAR(10),
credits INTEGER,
PRIMARY KEY(cid))

CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(10),
grade CHAR(2),
FOREIGN KEY (sid) REFERENCES Students,
FOREIGN KEY (cid) REFERENCES Courses)

UPDATE Students S
SET S.sid = 6
WHERE S.sid = 5? OK!
CREATE TABLE Students
(sid CHAR(20),
  name CHAR(30),
  login CHAR(20),
  age INTEGER,
  gpa REAL,
  PRIMARY KEY(sid),
  UNIQUE(login))

CREATE TABLE Courses
(cid CHAR(10),
  cname CHAR(10),
  credits INTEGER,
  PRIMARY KEY(cid))

CREATE TABLE Enrolled
(sid CHAR(20),
  cid CHAR(10),
  grade CHAR(2),
  PRIMARY KEY(sid, cid),
  FOREIGN KEY (sid) REFERENCES Students,
  FOREIGN KEY (cid) REFERENCES Courses)

INSERT INTO Enrolled
VALUES (5, 1, A)?
### Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>6</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

### Courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

### Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>

```
CREATE TABLE Students (
sid CHAR(20),
name CHAR(30),
login CHAR(20),
age INTEGER,
gpa REAL,
PRIMARY KEY(sid),
UNIQUE(login))

CREATE TABLE Courses (
cid CHAR(10),
cname CHAR(10),
credits INTEGER,
PRIMARY KEY(cid))

CREATE TABLE Enrolled (
sid CHAR(20),
cid CHAR(10),
grade CHAR(2),
FOREIGN KEY (sid) REFERENCES Students,
FOREIGN KEY (cid) REFERENCES Courses)

INSERT INTO Enrolled
VALUES (5, 1, A)?
FAILS (no Students tuple with sid == 5 any more)
```
### Students

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Luis</td>
<td>l@cs</td>
<td>85</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Panos</td>
<td>p@cs</td>
<td>30</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>Amelie</td>
<td>a@cs</td>
<td>30</td>
<td>4.1</td>
</tr>
<tr>
<td>6</td>
<td>Hila</td>
<td>h@cs</td>
<td>29</td>
<td>4.1</td>
</tr>
</tbody>
</table>

### Courses

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>OS</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>PLT</td>
<td>4</td>
</tr>
</tbody>
</table>

### Enrolled

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>C-</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A+</td>
</tr>
</tbody>
</table>

### SQL Commands

- **CREATE TABLE Students**
  
  ```sql
  CREATE TABLE Students
  (sid CHAR(20),
   name CHAR(30),
   login CHAR(20),
   age INTEGER,
   gpa REAL,
   PRIMARY KEY(sid),
   UNIQUE(login))
  ```

- **CREATE TABLE Courses**
  
  ```sql
  CREATE TABLE Courses
  (cid CHAR(10),
   cname CHAR(10),
   credits INTEGER,
   PRIMARY KEY(cid))
  ```

- **CREATE TABLE Enrolled**
  
  ```sql
  CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(10),
   grade CHAR(2),
   PRIMARY KEY(sid, cid),
   FOREIGN KEY (sid) REFERENCES Students,
   FOREIGN KEY (cid) REFERENCES Courses)
  ```

- **DELETE FROM Students S WHERE S.sid = 1?**

  Options?
Enforcing Referential Integrity

CREATE TABLE Students
    (sid CHAR(20),
     name CHAR(30),
     login CHAR(20),
     age INTEGER,
     gpa REAL,
     PRIMARY KEY(sid),
     UNIQUE(login))

CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(10),
     grade CHAR(2),
     PRIMARY KEY(sid, cid),
     FOREIGN KEY (sid) REFERENCES Students,
     FOREIGN KEY (cid) REFERENCES Courses)

● What should be done if an Enrolled tuple with a nonexistent student id is inserted?
Enforcing Referential Integrity

CREATE TABLE Students
(sid CHAR(20),
 name CHAR(30),
 login CHAR(20),
 age INTEGER,
 gpa REAL,
 PRIMARY KEY(sid),
 UNIQUE(login))

CREATE TABLE Enrolled
(sid CHAR(20),
 cid CHAR(10),
 grade CHAR(2),
 PRIMARY KEY(sid, cid),
 FOREIGN KEY (sid) REFERENCES Students,
 FOREIGN KEY (cid) REFERENCES Courses)

● What should be done if an Enrolled tuple with a nonexistent student id is inserted? **Reject it!**
● What should be done if a Students tuple is deleted?
Enforcing Referential Integrity

- What should be done if an Enrolled tuple with a nonexistent student id is inserted? Reject it!
- What should be done if a Students tuple is deleted? Options:
  1. **Reject** deletion of the Students tuple if any Enrolled tuples refer to it
  2. **Delete also** all Enrolled tuples that refer to it
  3. Set sid in Enrolled tuples that refer to it to a “default” sid value, which should be present in Students
  4. Set sid in Enrolled tuples that refer to it to a special value “NULL,” which denotes “unknown” or “inapplicable”; NULL values don’t violate referential integrity

```
CREATE TABLE Students
    (sid CHAR(20),
     name CHAR(30),
     login CHAR(20),
     age INTEGER,
     gpa REAL,
     PRIMARY KEY(sid),
     UNIQUE(login))

CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(10),
     grade CHAR(2),
     PRIMARY KEY(sid, cid),
     FOREIGN KEY (sid) REFERENCES Students,
     FOREIGN KEY (cid) REFERENCES Courses)
```
Enforcing Referential Integrity

- What should be done if an Enrolled tuple with a nonexistent student id is inserted? Reject it!
- What should be done if a Students tuple is deleted? Options:
  1. **Reject** deletion of the Students tuple if any Enrolled tuples refer to it
  2. **Delete also** all Enrolled tuples that refer to it
  3. Set sid in Enrolled tuples that refer to it to a “default” sid value, which should be present in Students
  4. Set sid in Enrolled tuples that refer to it to a special value “NULL,” which denotes “unknown” or “inapplicable”; NULL values don’t violate referential integrity
- What if the primary key attribute of a Students tuple is **updated**? Analogous options …

```sql
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(30),
 login CHAR(20),
 age INTEGER,
 gpa REAL,
 PRIMARY KEY(sid),
 UNIQUE(login))

CREATE TABLE Enrolled
(sid CHAR(20),
 cid CHAR(10),
 grade CHAR(2),
 PRIMARY KEY(sid, cid),
 FOREIGN KEY (sid) REFERENCES Students,
 FOREIGN KEY (cid) REFERENCES Courses)
```
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(10),
grade CHAR(2),
PRIMARY KEY (sid, cid),
FOREIGN KEY (sid) REFERENCES Students
  ON DELETE NO ACTION
    choose one;
    default: NO ACTION
    CASCADE
    SET DEFAULT
    SET NULL
  ON UPDATE NO ACTION
    choose one;
    default: NO ACTION
    CASCADE
    SET DEFAULT
    SET NULL,
FOREIGN KEY (cid) REFERENCES Courses
...
Where Do Integrity Constraints Come From?

- Constraints originate in semantics of real-world organization described in relations.
- We can check a database instance to see if a constraint is violated, but we can never infer that an integrity constraint must be added/declared by looking at a database instance.
- Primary key, candidate key, and foreign key constraints are most common; more general constraints are supported too, as we will see later.