

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	John	Ki#Bu!GK.\$@q	19	2.1
301	Elaine	301 Wilshire	263	3.9
401	James	183 Westwood	17	-1.0
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	Carlo Zaniolo
EE	143	01	134	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
999	CS	143	01
401	AT	000	00
303	EE	143	01
303	CS	112	01

◀ ▶ ⏪ ⏩ 🔍 ↻

Key constraint

- ▶ Course(dept, cnum, sec, unit, instructor, title)
- Course(dept, cnum, sec, unit, instructor, title)
- Course(dept, cnum, sec, unit, instructor, title)

```
▶ CREATE TABLE Course (  
    dept CHAR(2) NOT NULL,  
    cnum INT NOT NULL,  
    sec INT NOT NULL,  
    unit INT,  
    instructor VARCHAR(30),  
    title VARCHAR(30),  
    PRIMARY KEY(dept, cnum, sec),  
    UNIQUE(dept, cnum, instructor),  
    UNIQUE(dept, sec, title) )
```

◀ ▶ ⏪ ⏩ 🔍 ↻

Referential Integrity Constraint

- ▶ CREATE TABLE Enroll (
 sid INT REFERENCES Student(sid),
 dept CHAR(2),
 cnum INT,
 sec INT,
 FOREIGN KEY (dept, cnum, sec)
 REFERENCES Class(dept, cnum, sec)
 ON DELETE CASCADE
 ON UPDATE SET NULL)

▶ Notes:

- ▶ Referencing attributes called FOREIGN KEY
- ▶ Referenced attributes must be PRIMARY KEY or UNIQUE

◀ ▶ ⏪ ⏩ 🔍 ↻

CHECK constraint

- ▶ CHECK(<condition>) in CREATE TABLE
- ▶ Example:
 - ▶ All CS classes are of 3 units or more.
 - ▶ CRATE TABLE Class(
 - dept CHAR(2),
 - cnum INT,
 - unit INT,
 - title VARCHAR(50),
 - instructor VARCHAR(30))
CHECK (dept <> 'CS' OR unit >= 3)
 - ▶ (dept='CS' → unit>3) ≡ (¬(dept='CS') ∨ unit>3)
- ▶ Constraint is checked whenever a tuple is inserted/updated.
- ▶ In SQL92, conditions can be more complex, e.g., with subqueries
 - ▶ In practice, complex CHECK conditions are not supported

◀ ▶ ⏪ ⏩ 🔍 ↻

General Assertions

- ▶ Constraint on the entire relation or database
- ▶ CREATE ASSERTION <assertion_name> CHECK (<condition>)
- ▶ Example: Average GPA > 3.0
 - ▶ CREATE ASSERTION HighGPA CHECK(3.0 < (SELECT AVG(GPA) FROM Student))

◀ ▶ ⏪ ⏩ 🔍 ↻

Trigger: Event-Condition-Action rule

- ▶ CREATE TRIGGER <name>
 - <event>
 - <referencing clause> // optional
 - FOR EACH ROW // optional
 - WHEN (<condition>) // optional
 - <action>
- ▶ <event> can be:
 - ▶ BEFORE | AFTER INSERT ON R
 - ▶ BEFORE | AFTER DELETE ON R
 - ▶ BEFORE | AFTER UPDATE [OF A1, A2, ..., An] ON R
- ▶ <condition>: any condition
- ▶ <action>: sequence of SQL statements

◀ ▶ ⏪ ⏩ 🔍 ↻

Trigger: Event-Condition-Action rule

- ▶ FOR EACH ROW
 - ▶ If present, "row-level" trigger
 - ▶ Execute trigger once for each tuple changed.
 - ▶ If absent, "statement-level" trigger
 - ▶ Execute trigger once for each relevant statement.
- ▶ <referencing clause>:
REFERENCING OLD|NEW TABLE|ROW AS <var>, ...
 - ▶ OLD ROW: previous value of deleted or updated tuple, row-level only
 - ▶ NEW ROW: current value of inserted or updated tuple, row-level only
 - ▶ OLD TABLE: previous values of deleted or updated tuples, row-level or statement-level
 - ▶ NEW TABLE: current values of inserted or updated tuples, row-level or statement-level

◀ ▶ ⏪ ⏩ 🔍 ↻

Trigger: Example 1

- ▶ After UPDATE of GPA, if $\text{AVG}(\text{GPA}) < 3.0$, revert to old GPA
- ▶ CREATE TRIGGER HighGPA1
AFTER UPDATE OF GPA ON Student
REFERENCING OLD ROW AS old, NEW ROW AS new
FOR EACH ROW
WHEN ((SELECT AVG(GPA) FROM Student) < 3.0)
BEGIN
UPDATE Student SET GPA = old.GPA
WHERE sid = new.sid;
END

◀ ▶ ⏪ ⏩ 🔍 ↻

Trigger: Example 2

- ▶ After INSERT, if $\text{AVG}(\text{GPA}) < 3.0$, drop all inserted tuples
- ▶ CREATE TRIGGER HighGPA2
AFTER INSERT ON Student
REFERENCING NEW TABLE AS new
WHEN ((SELECT AVG(GPA) FROM Student) < 3.0)
BEGIN
DELETE FROM Student
WHERE sid IN (SELECT sid FROM new);
END

◀ ▶ ⏪ ⏩ 🔍 ↻