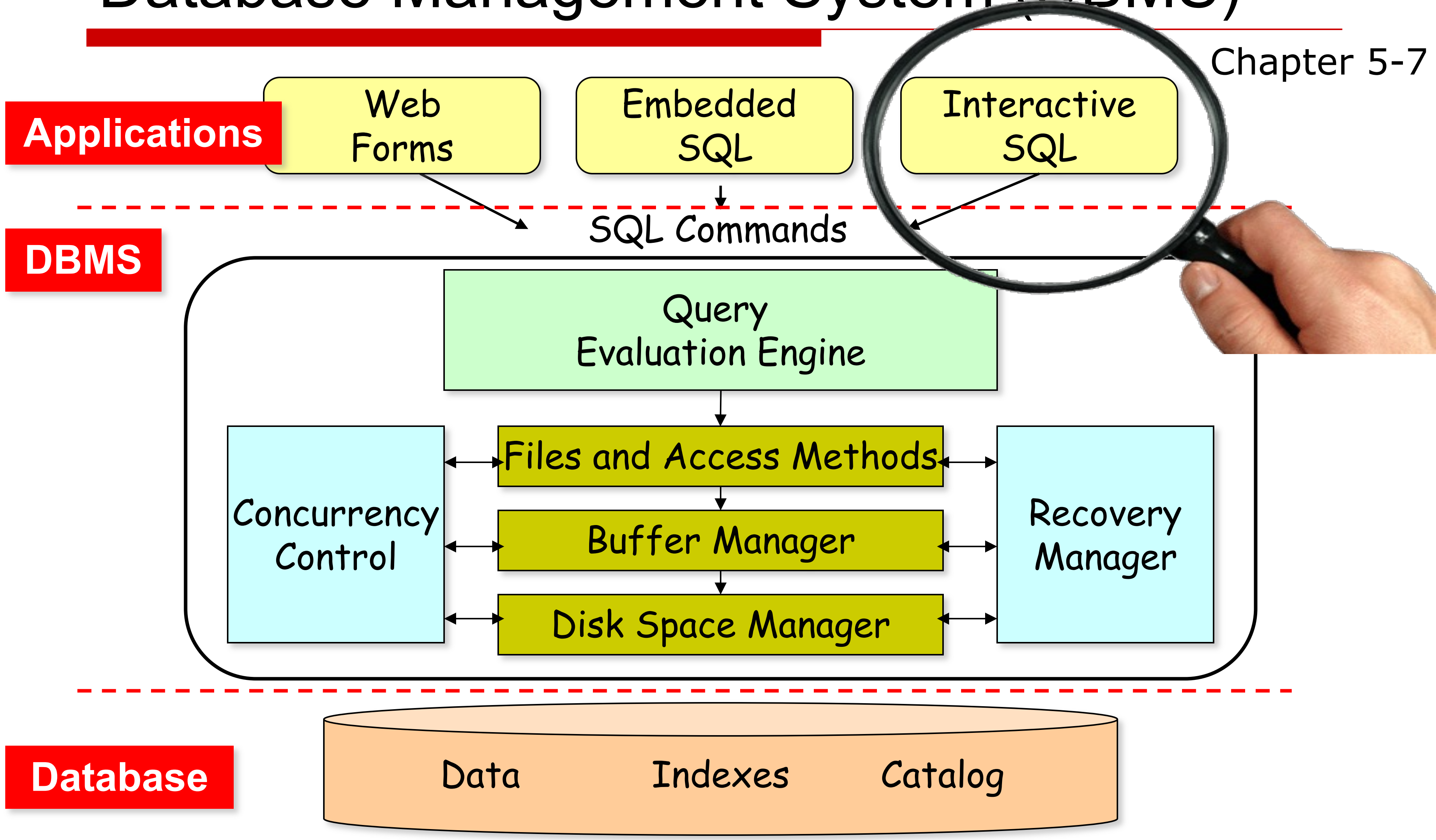
Database Management System (DBMS)

Chapter 5-7



Relational Database Management Systems

- Data & Execution Abstraction
  - Overview
  - SQL Queries
  - Views
  - Integrity Constraints
  - Complex Integrity Constraints

Database Languages

- **Data Definition Language (*DDL*):**
  - Define schemas
  - Define **Integrity Constraints**
    - Example: unique *SID*s
  - More…
- **Data Manipulation Language (*DML*):**
  - To ask questions = ***Query***
    - Example: Which students have GPA > 3.75?
  - To insert, delete and update data
- ***SQL:*** Most widely used database language

DDL -- Creating Relations in SQL

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

- Corresponding database is at an **empty** state!
- Initial state when the database is **populated** (loaded)
- Domain (type) of each field is **specified** and **enforced** by the DBMS whenever tuples are added or modified

Example: Domain Constraints

SID	*Name*	*Login*	*Age*	*GPA*
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

- Example of **IC Violation**:

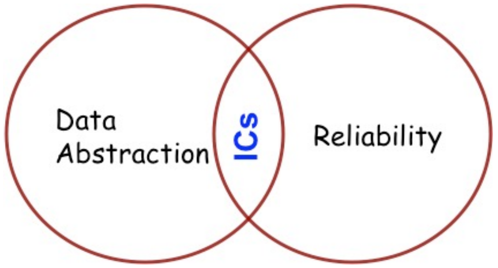
UPDATE *Students S*

SET *S.age* = ‘Eighteen’ ✘ ✘ ✘

WHERE *S.name* = ‘Jones’

Integrity Constraints (ICs)

- **IC**: condition that must be true for *any* instance of the database (e.g., domain constraints)
  - A **legal** instance of a relation is one that satisfies all specified ICs
  - ICs are <u>specified</u> when schema is **defined**
    - DBMS Data Abstraction (DDL)
  - ICs are <u>enforced</u> when tables are **modified**
    - DBMS Reliability

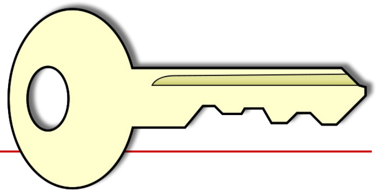


Integrity Constraints (IC)

- Primary Key IC
- Unique Value IC
- Foreign Key IC

- Complex Integrity Constraints
  - Utilize the full power of SQL queries

Primary Key Constraint



- A set of fields is a **key** for a relation if :
  - No two distinct tuples can have same values in all key fields
- If there is more than one key for a relation:
  - Each is called a **candidate key**
  - One candidate key is designated as the **primary key**
  - Other candidate key(s) are designated as **unique key(s)**

Example of Keys

SID	Name	Login	Age	GPA
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

- **Candidate Keys:** *SID*, and *Login*
- **Primary Key:** *SID*
- **Unique Key:** *Login*

Specifying Key Constraints in SQL

SID	Name	Login	Age	GPA
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

```
CREATE TABLE Students (
    sid: CHAR(20),
    name: CHAR(20),
    login: CHAR(10),
    age: INTEGER,
    gpa: REAL,
    UNIQUE (login),
    PRIMARY KEY (sid))
```

Enforcing Primary Key Constraints

SID	*Name*	*Login*	*Age*	*GPA*
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

INSERT INTO Students

VALUES (546100, 'Mike', 'mike@ee', 21, 3.9)

INSERT INTO Students

VALUES (*null*, 'Mike', 'mike@ee', 21, 3.9)

Enforcing Primary Key Constraints

SID	Name	Login	Age	GPA
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

- Examples of **IC Violations**:

INSERT INTO Students

VALUES (546100, ‘Mike’, ‘mike@ee’, 21, 3.9)

INSERT INTO Students

VALUES (null, ‘Mike’, ‘mike@ee’, 21, 3.9)

NULL and DEFAULT

SID	*Name*	*Login*	*Age*	*GPA*
546007	Jones	jones@cs	18	3.4
546100	Smith	smith@ee	18	3.2
546500	Smith	smith@math	19	3.8

CREATE TABLE Students (

sid: CHAR(20),

name: CHAR(20) **NOT NULL,**

login: CHAR(10),

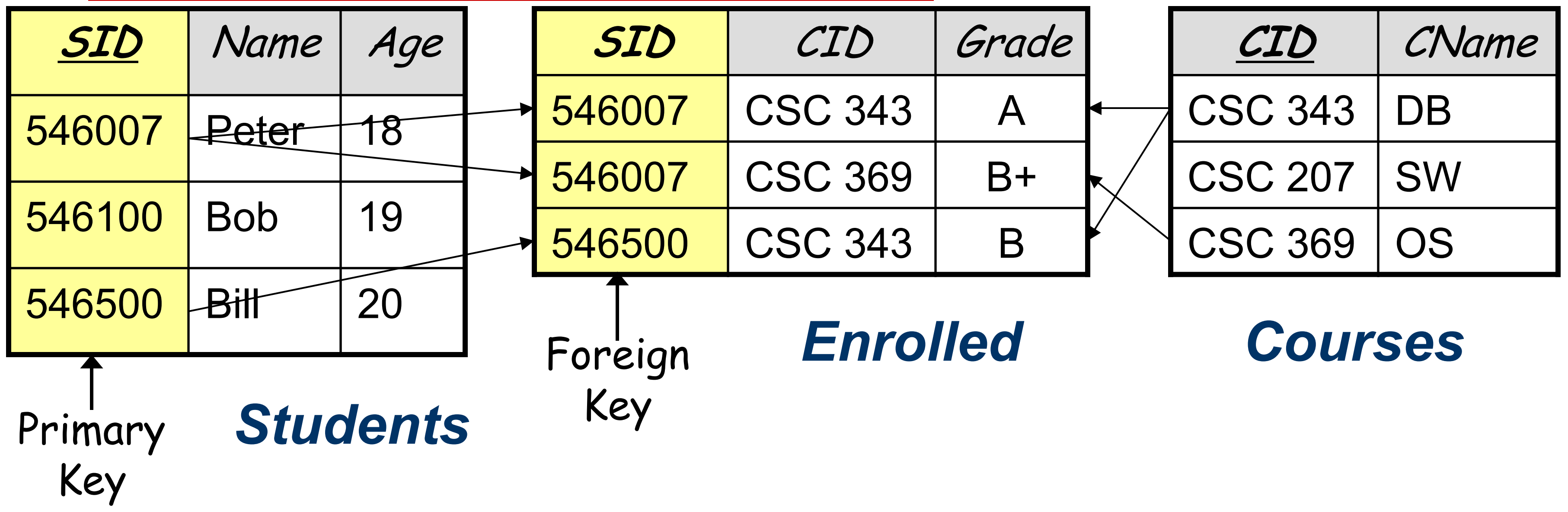
age: INTEGER,

gpa: REAL **DEFAULT** 7.0,

UNIQUE (*login*),

PRIMARY KEY (*sid*))

Foreign Key



<u>SID</u>	Name	Age
546007	Peter	18
546100	Bob	19
546500	Bill	20

SID	CID	Grade
546007	CSC 343	A
546007	CSC 369	B+
546500	CSC 343	B

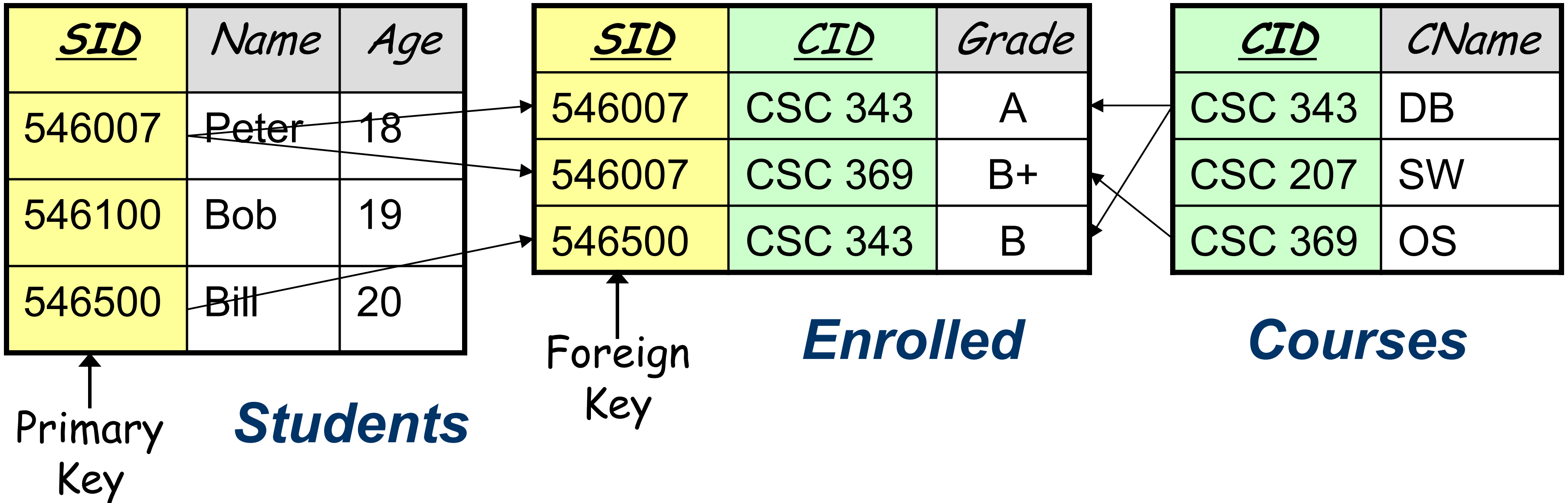
<u>CID</u>	CName
CSC 343	DB
CSC 207	SW
CSC 369	OS

- **Foreign key :** Set of fields in one relation that is used to "refer" to a tuple in another relation
  - Must correspond to <u>primary key</u> of the referred relation
  - E.g. *SID* is a foreign key referring to *Students*

Foreign Key Constraints

- If foreign key constraints are enforced, **referential integrity** is achieved
  - E.g.: Only students can enroll in a class
    - Only students listed in the "Students" relation should be allowed to enroll for courses
- Like a "logical pointer"
  - There shouldn't be dangling references

Specifying Foreign Key in SQL



Students

SID	Name	Age
546007	Peter	18
546100	Bob	19
546500	Bill	20

Enrolled

SID	CID	Grade
546007	CSC 343	A
546007	CSC 369	B+
546500	CSC 343	B

Courses

CID	CName
CSC 343	DB
CSC 207	SW
CSC 369	OS

- **CREATE TABLE Enrolled(**

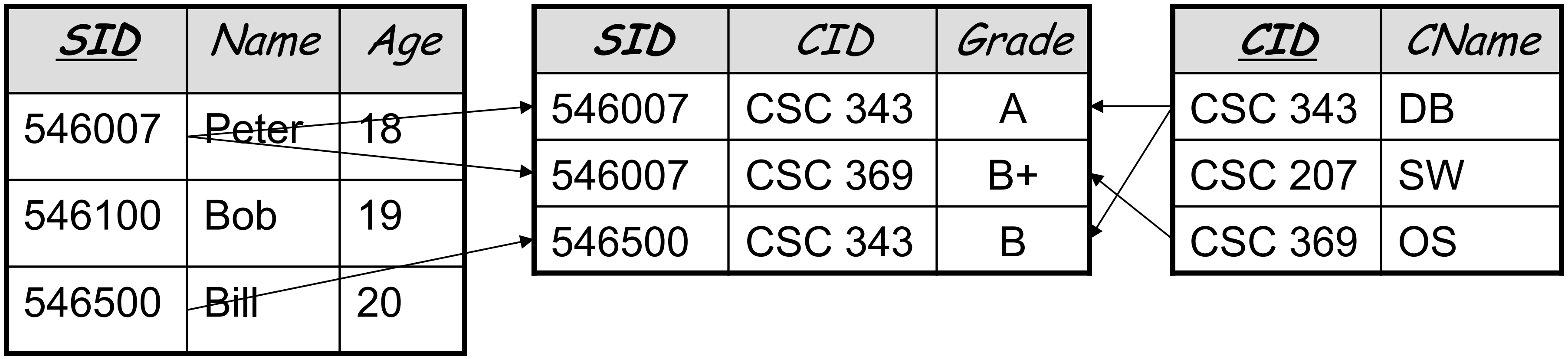
  sid `CHAR(20)`, *cid* `CHAR(20)`, *grade* `CHAR(2)`,

  PRIMARY KEY (*sid,cid*),

  FOREIGN KEY (*sid*) **REFERENCES** *Students*

  FOREIGN KEY (*cid*) **REFERENCES** *Courses*)

Referential Integrity Constraints



<u>SID</u>	Name	Age
546007	Peter	18
546100	Bob	19
546500	Bill	20

Students

SID	CID	Grade
546007	CSC 343	A
546007	CSC 369	B+
546500	CSC 343	B

Enrolled

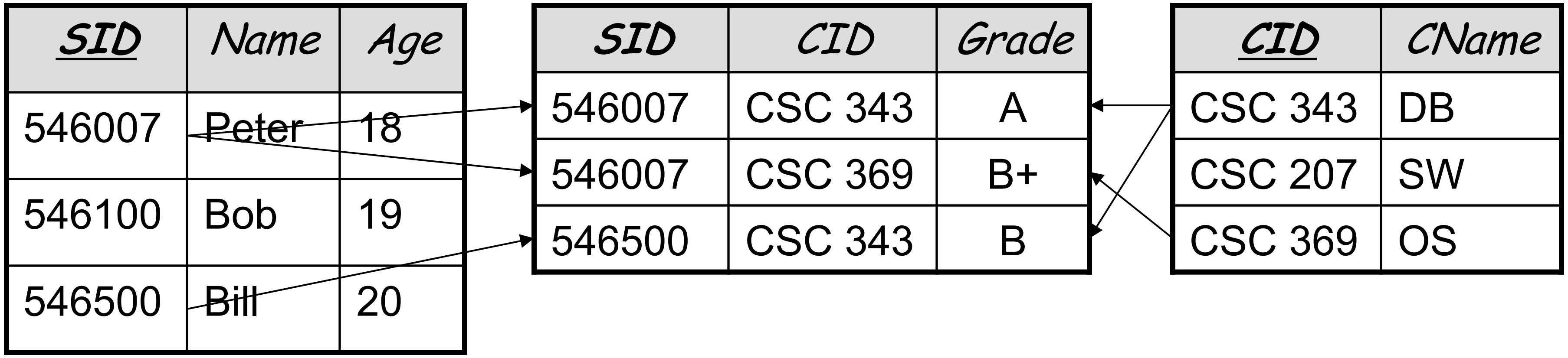
<u>CID</u>	CName
CSC 343	DB
CSC 207	SW
CSC 369	OS

Courses

```
DELETE
FROM  Enrolled E
WHERE   E.sid = 546500
```

No Violations! ✓

Referential Integrity Constraints



Students

SID	Name	Age
546007	Peter	18
546100	Bob	19
546500	Bill	20

Enrolled

SID	CID	Grade
546007	CSC 343	A
546007	CSC 369	B+
546500	CSC 343	B

Courses

CID	CName
CSC 343	DB
CSC 207	SW
CSC 369	OS

- Example of **IC Violation**:

`INSERT INTO` Enrolled

`VALUES` (546105, CSC 207, B+)

- **`Insert`** might cause a violation, but **`Delete`** is ok
- The opposite for the "Students" relation !

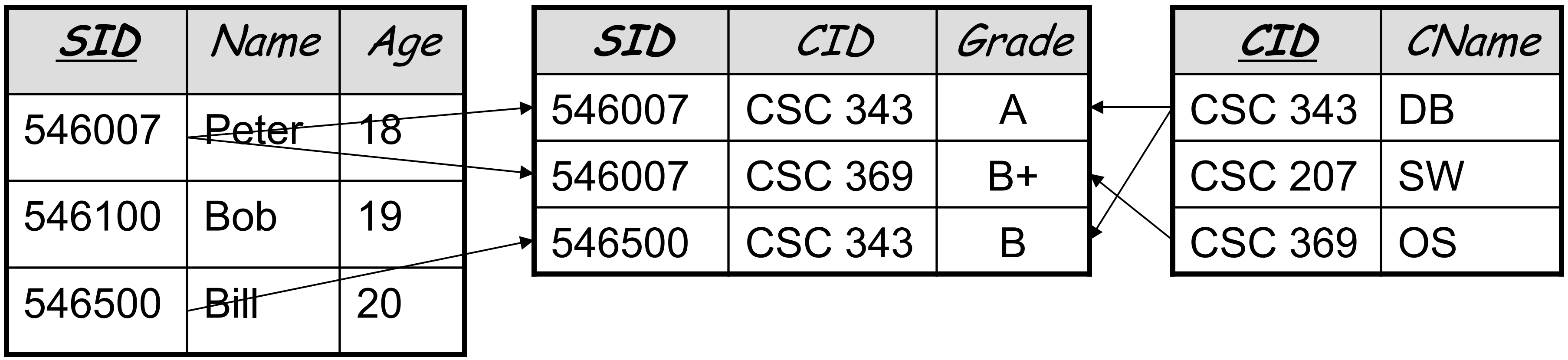
Referential Integrity Enforcement

- What are the alternatives when a "Students" tuple is **deleted**?
  1. **Delete all** Enrolled tuples that refer to it
  2. **Disallow** deletion of a Students tuple that is referred to
  3. **Set** sid in Enrolled tuples that refer to it to some "default" *sid* (e.g., 000000)
  4. **Set** sid in Enrolled tuples that refer to it to a special value "null", denoting *"unknown"* or *"inapplicable"*

Referential Integrity in SQL

- SQL/92 and SQL:1999 support all 4 options on <u>delete</u> and <u>update</u>:
  - **`NO ACTION`** (default)
    - Delete/update is rejected
  - **`CASCADE`**
    - Also delete all tuples that refer to deleted tuple
  - **`SET NULL / SET DEFAULT`**
    - Set foreign key value of referencing tuple

Referential Integrity Constraints



SID	Name	Age
546007	Peter	18
546100	Bob	19
546500	Bill	20

Students

SID	CID	Grade
546007	CSC 343	A
546007	CSC 369	B+
546500	CSC 343	B

Enrolled

CID	CName
CSC 343	DB
CSC 207	SW
CSC 369	OS

Courses

- **CREATE TABLE Enrolled(**

  sid CHAR(20), *cid* CHAR(20), *grade* CHAR(2),

  PRIMARY KEY (*sid,cid*),

  FOREIGN KEY (*sid*) **REFERENCES** *Students*

  ON UPDATE CASCADE

  ON DELETE NO ACTION)