SQL – Data Definition Language

Carla Teixeira Lopes

Bases de Dados
Mestrado Integrado em Engenharia Informática e Computação, FEUP

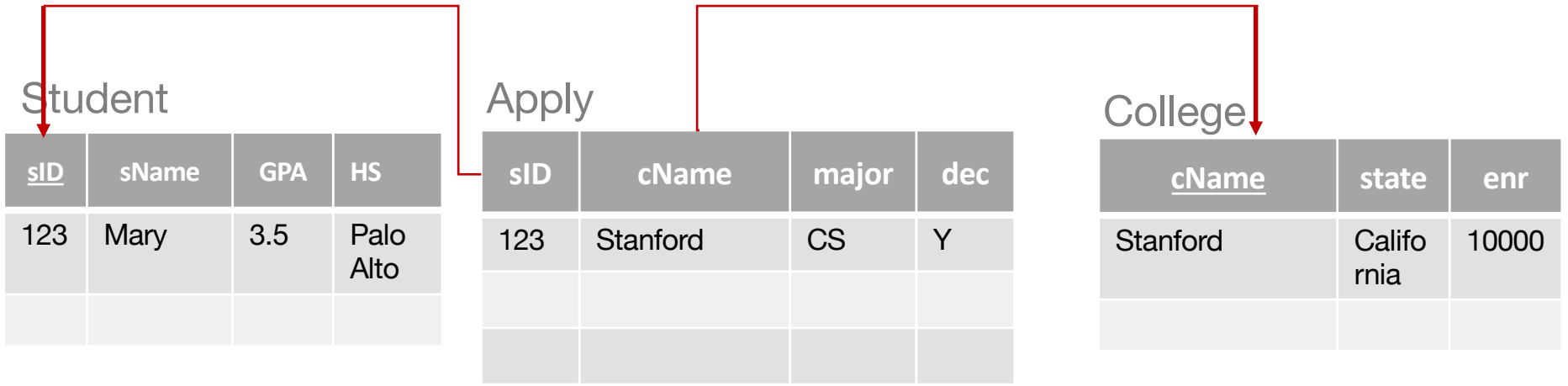
Referential Integrity

Integrity of references

No “dangling pointers”

Referential integrity from R.A to S.B

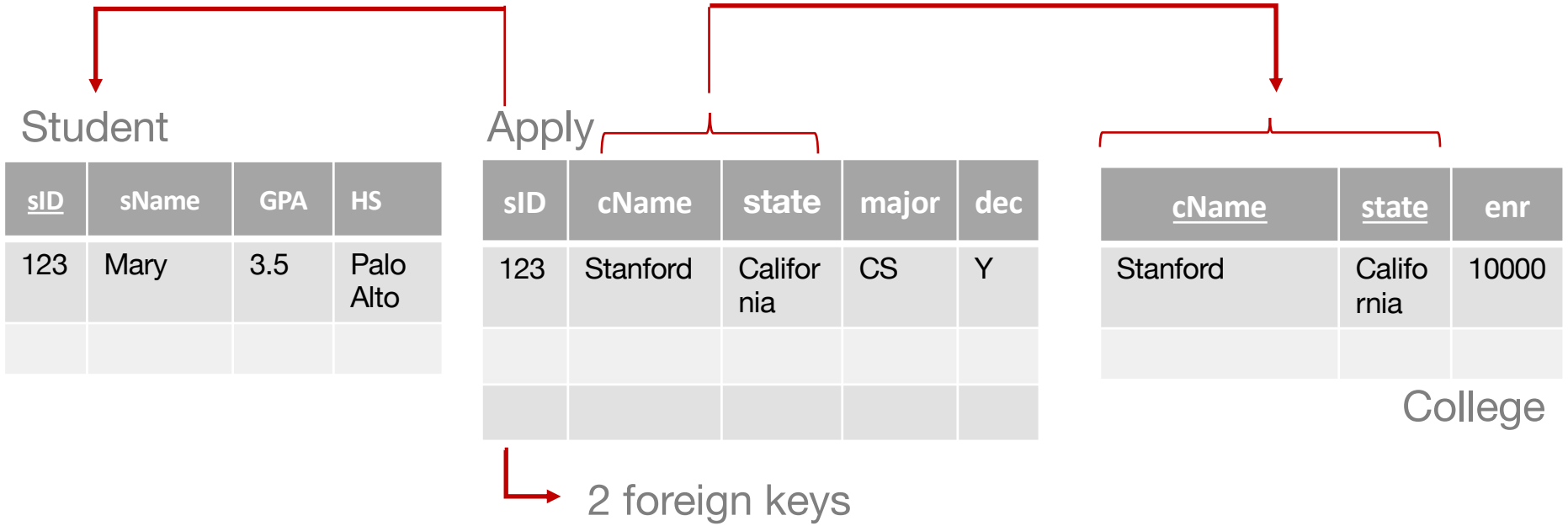
Each value in column A of table R must appear in column B of table S



Referential Integrity

Referential integrity from R.A to S.B

- A is called the “foreign key”
- B is usually required to be the primary key for table S or at least unique
- Multi-attribute foreign keys are allowed



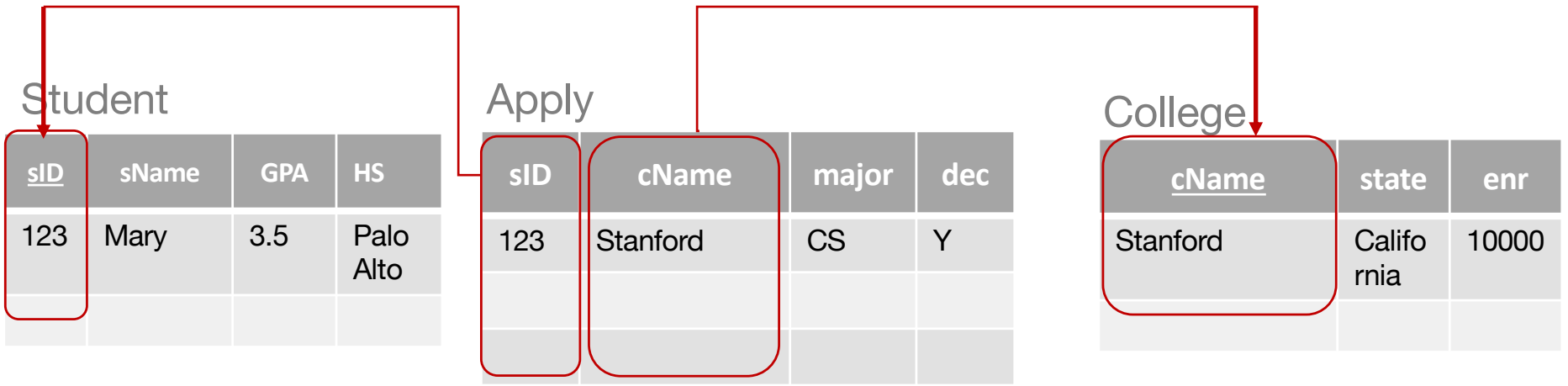
Referential Integrity Enforcement (R.A to S.B)

Potentially violating modifications

- Insert into R
- Delete from S
- Update R.A
- Update S.B

If violation -> error

Depends on Foreign key definition



Referential Integrity Enforcement (R.A to S.B)

Delete from S

Restrict (default)

Generate an error, modification disallowed

Set Null

Replace R.A by NULL

Cascade

Delete tuples having a referencing value

Student

sID	sName	GPA	HS
~~123~~	~~Mary~~	~~3.5~~	~~Palo Alto~~
234	Louis	3.8	Palo Alto

Apply

sID	cName	major	dec
NULL	Stanford	CS	Y
~~234~~	~~MIT~~	~~CS~~	~~Y~~

College

cName	state	enr
Stanford	California	10000
MIT	Massachusetts	15000

Referential Integrity Enforcement (R.A to S.B)

Update S.B

- Restrict (default)
  - Generate an error, modification disallowed
- Set Null
  - Replace R.A by NULL
- Cascade
  - Do the same update to R.A

Student

sID	sName	GPA	HS
123	Mary	3.5	Palo Alto

456

Apply

sID	cName	major	dec
~~123~~ NULL	~~Stanford~~ Standford	CS	Y
234	MIT	CS	Y

College

cName	state	enr
~~Stanford~~ Standford	California	10000

Foreign Key Declaration

```
CREATE TABLE <table_A> (
    <column_A> <data_type> PRIMARY KEY,
    <column_B> <data_type>,
     ...
    <column_C> <data_type>
);

CREATE TABLE <table_B> (
    <column_X> <data_type> PRIMARY KEY,
    <column_Y> <data_type>,
    ...
    <column_Z> <data_type> REFERENCES <table_A>(<column_A>)
);
```

Example

```
CREATE TABLE College (cName text PRIMARY KEY, state text,
enrollment int);

CREATE TABLE Student (sID int PRIMARY KEY, sName text, GPA
real, sizeHS int);

CREATE TABLE Apply (
        sID REFERENCES Student(sID),
        cName text REFERENCES College (cName),
        major text,
        decision text,
        PRIMARY KEY(sID, cName)
);
```

Foreign Key to Primary Key

If the referenced column is the primary key of the other table, we can omit the name of the column

```
CREATE TABLE <table_A> (
    <column_A> <data_type> PRIMARY KEY,
    <column_B> <data_type>,
     ...
    <column_C> <data_type>
);

CREATE TABLE <table_B> (
    <column_X> <data_type> PRIMARY KEY,
    <column_Y> <data_type>,
    ...
    <column_Z> <data_type> REFERENCES <table_A>
);
```

Example

CREATE TABLE College (cName text **PRIMARY KEY**, state text, enrollment int);

CREATE TABLE Student (sID int **PRIMARY KEY**, sName text, GPA real, sizeHS int);

CREATE TABLE Apply (
    sID int **REFERENCES** Student,
    cName text **REFERENCES** College,
    major text,
    decision text,
    PRIMARY KEY(sID, cName)
);

Multiple Column Foreign Key Declaration

```
CREATE TABLE <table_A> (
    <column_A> <data_type>,
    <column_B> <data_type>,
     ...
    <column_C> <data_type>,
    PRIMARY KEY (<column_A>, <column_B>)
);

CREATE TABLE <table_B> (
  <column_X> <data_type> PRIMARY KEY,
  <column_Y> <data_type>,
  ...
  <column_Z> <data_type>,
  FOREIGN KEY (<column_X>, <column_Y> ) REFERENCES <table_A>(<column_A>, <column_B>)
);
```

Example

CREATE TABLE College (cName text, state text, enrollment int, **PRIMARY KEY** (cName, state));

CREATE TABLE Student (sID int **PRIMARY KEY**, sName text, GPA real, sizeHS int);

CREATE TABLE Apply (
    sID **REFERENCES** Student,
    collegeName text,
    collegeState text,
    major text,
    decision text,
    FOREIGN KEY (collegeName, collegeState) REFERENCES College(cName, state),
    PRIMARY KEY(sID, collegeName, collegeState)
);

We can omit the referenced columns if they are primary keys

On Delete and On Update Actions

Define actions that take place when deleting or modifying parent key values.

Use the ON DELETE and ON UPDATE clauses with one of three possible values

- RESTRICT - prohibit operation on a parent key when there are child keys mapped to it
- SET NULL – child key columns are set to NULL
- CASCADE – propagates the operation on the parent key to each dependent child key

Example

```
CREATE TABLE College (cName text PRIMARY KEY, state text, enrollment int);

CREATE TABLE Student (sID int PRIMARY KEY, sName text, GPA real, sizeHS int);

CREATE TABLE Apply (
        sID REFERENCES Student(ID),
        cName text REFERENCES College (cName) ON DELETE SET NULL ON
        UPDATE CASCADE,
        major text,
        decision text,
        PRIMARY KEY(sID, cName)
);
```

Enabling Foreign Key Support in SQLite

Foreign key constraints are disabled by default

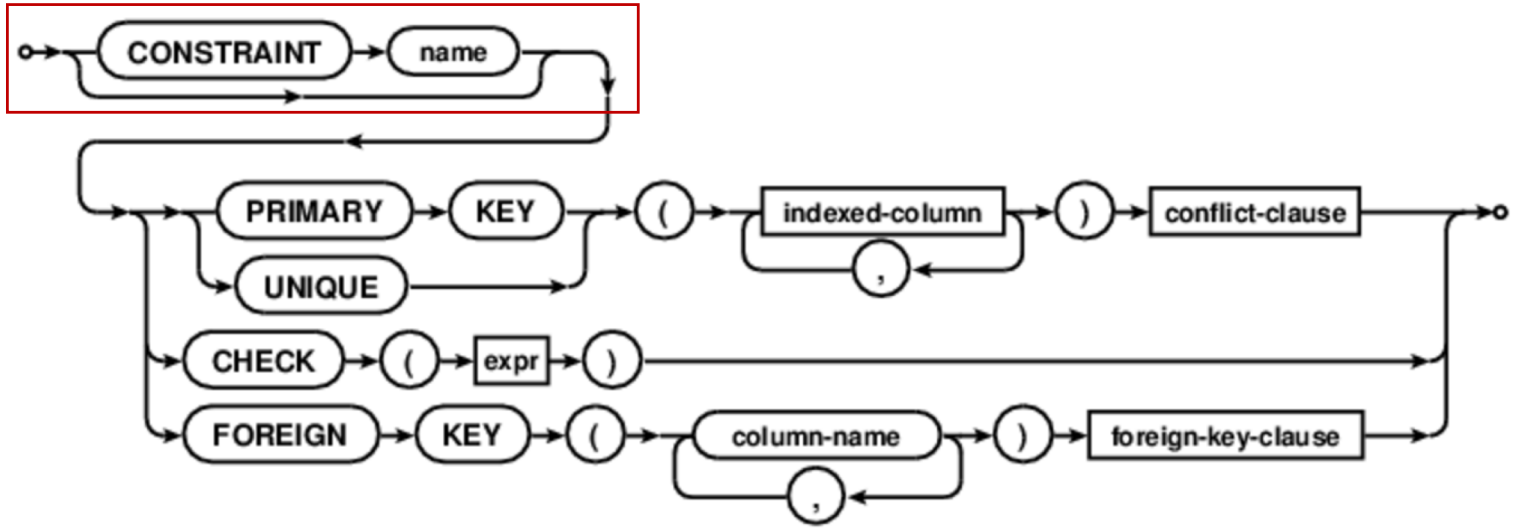
Must be enabled separately for each database connection

PRAGMA foreign_keys = ON;

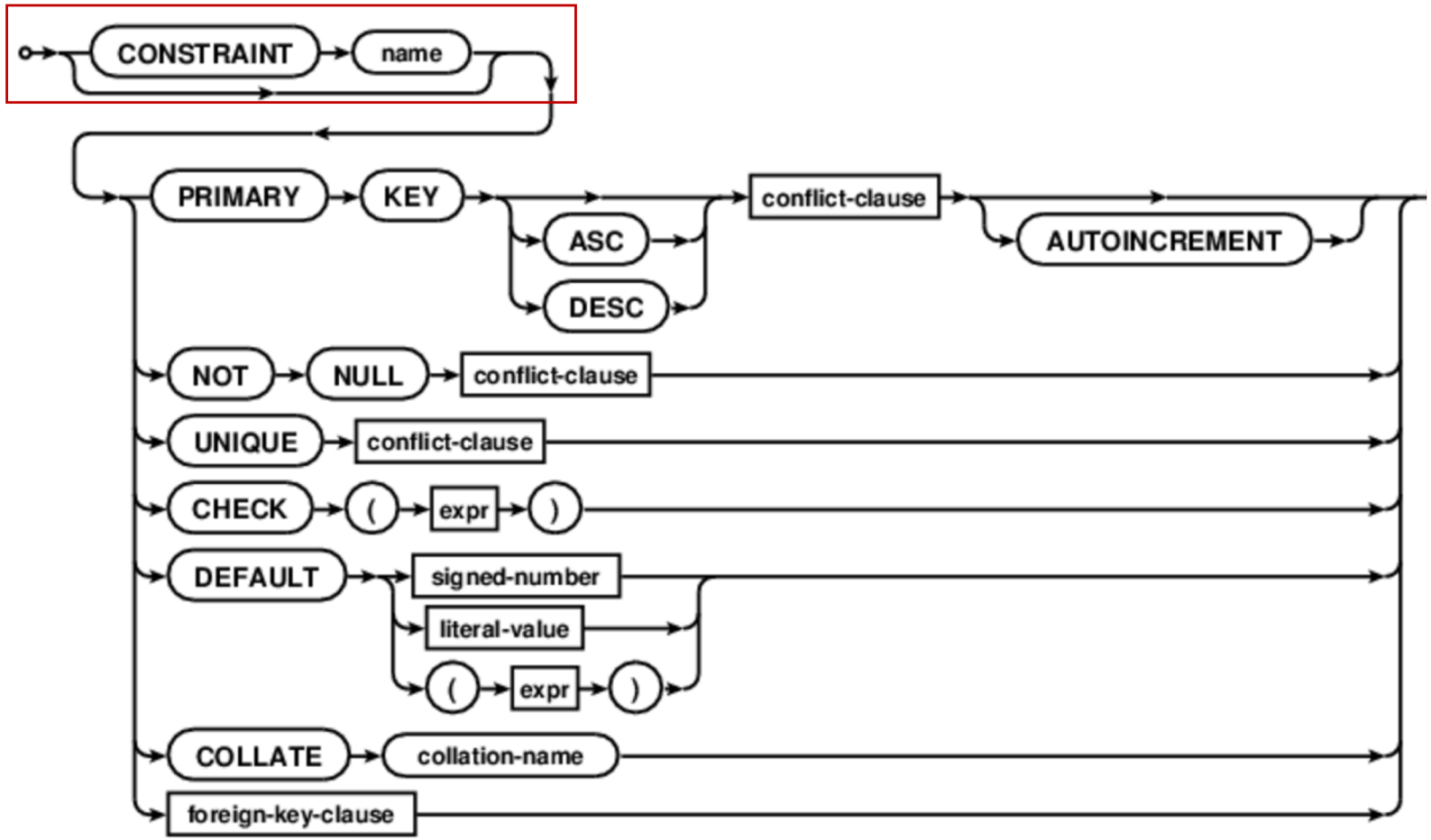
Constraint Naming

Naming constraints is optional but is a good practice

It makes it easier to identify the constraints when errors occur and to refer to them



Constraint Naming



Example

```
CREATE TABLE Student (
        sID         INTEGER,
        sName       TEXT,
        GPA         REAL  CONSTRAINT GPARange CHECK (GPA<=4.0),
        sizeHS      INTEGER  CONSTRAINT maxSizeHS CHECK (sizeHS < 5000),
        CONSTRAINT StudentPK PRIMARY KEY (sID)
);
```

Assertions

Constraints on entire relation or entire database

Are in the SQL standard but are not supported by any database system

CREATE ASSERTION <assertion_name> **CHECK** (<condition>);

Example

```
CREATE ASSERTION Key CHECK(
(select count(distinct A) from T) = (select count(*) from T)));

CREATE ASSERTION ReferentialIntegrity CHECK(
    not exists (SELECT * from Apply
                    where sID not in (select sID from Student)));

CREATE ASSERTION AvgAccept CHECK(
      3.0 < (select avg(GPA) from Student
            where sID in
              (select sID from Apply where decision = 'Y')));
```

Assertion checking

```
CREATE ASSERTION AvgAccept CHECK(
        3.0 < (select avg(GPA) from Student
                where sID in
                  (select sID from Apply where decision = 'Y')));
```

Determine every possible change that could violate the assertion

- What changes are these in the above assertion?
- Modifying a GPA, student ID and decision
- Inserting or deleting from students or apply

After those modifications

- check the constraint,
- make sure they it's still satisfied and, if not, generate an error and disallow the database change

Kahoot time!

Any doubts?

Readings

Jeffrey Ullman, Jennifer Widom, A first course in Database Systems 3rd Edition

- Section 2.3 – Defining a Relation Schema in SQL
- Section 2.5 – Constraints on Relations
- Section 7.1 – Keys and Foreign Keys
- Section 7.2 – Constraints on Attributes and Tuples
- Section 7.3 – Modification if Constraints
- Section 7.4 - Assertions