# CS245: Databases

## SQL

Vijaya saradhi

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

# Creation

## Elements to be created

- Database
- Table
- Constraint
- Function
- Procedure
- Trigger
- Event
- And other elements

All these elements use **CREATE** statement

# Database Creation

## Database

- Database is a collection of tables (and programs that manipulate tables)
- Tables cannot exist independently. They must be under a specified database
- In order to create tables, we needs to create a database

# Database Creation - SQL statement

## Invoking MySQL client

```
mysql -uroot -p
```

```
CREATE DATABASE cs245;
```

```
CREATE DATABASE IF NOT EXISTS cs245;
```

# Database Creation - SQL statement

## Using a specific database

- DBMS system may hold several databases. We need to specify which database we intend to use
- Only after this step, tables can be created

# Specifying Database to Use

```
USE cs245;
```

# Creating a table

## Syntax

```
CREATE TABLE table_name(
                column_1 data_type,
                column_2 data_type,
              ... column_n data_type);
```

# Creating a table

```
CREATE TABLE student(
        sid int,
        name char(50),
        login char(10),
        age int, spi float);
```

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |

## CREATE statement

- `sid` is column name `int` is its data type
- `name` is column name `char(50)` is its data type
- `login` is column name `char(10)` is its data type
- `age` is column name `int` is its data type
- `spi` is column name `float` is its data type
- There are no constraints on this table

# Creating a table

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| 190101000 | Atul Kumar | atul | 18 | 8.0 |
| 190101000 | Atul Gupta | atul | 18 | 8.2 |
| 190101000 | Atul M | atul | 18 | 8.2 |
| 190101000 | Atul Gupta | atul | 19 | 7.2 |

- Same roll number is assigned to several students
- Same login is assigned to several students
- It is not possible to distinguish between two Atul Gupta's (row 2 & 4)
- In case you have to update the spi of Atul Gupta which row will you update? 2 or 4?

# Creating another table

```
CREATE TABLE register(sid int, grade char(2), cid char(6));
```

| register | | |
|---|---|---|
| sid | grade | cid |
| 190101000 | AB | CS101 |
| 190101000 | BB | CS101 |
| 190109001 | AA | CS101 |
| 190109001 | BB | CS101 |

# Creating a table with primary key

```
CREATE TABLE student(
        sid int primary key,
        name char(50),
        login char(10),
        age int,
        spi float);
```

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| 190101000 | Atul Kumar | atul | 18 | 8.0 |
| 190101001 | Atul Gupta | atul | 18 | 8.2 |

# Creating a table with primary key

```
CREATE TABLE student(
        sid int primary key,
        name char(50),
        login char(10),
        age int,
        spi float);
```

Inserting two identical values of primary key is not allowed

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| 190101000 | Atul Kumar | atul | 18 | 8.0 |
| 190101001 | Atul Gupta | atul | 18 | 8.2 |
| 190101000 | Atul Kumar | atul | 18 | 8.0 |

# Creating a table with two constraints

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(10) unique,
                     age int,
                     spi float);
```

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| 190101000 | Atul Kumar | atul | 18 | 8.0 |
| 190101000 | Atul Gupta | atul01 | 18 | 7.2 |
| 190101001 | Atul Gupta | atul | 18 | 6.2 |
| 190101001 | Atul Gupta | atul02 | 18 | 8.6 |

# Creating a table with unique key

## One constraint alone

- Table with NO primary key constraint
- Having UNIQUE constraint on `login` column
- Note login can take NULL values.

```
CREATE TABLE student(sid int,
                     name char(50),
                     login char(10) unique,
                     age int,
                     spi float);
```

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| 190101001 | Atul Kumar | atul | 18 | 8.0 |
| 190101002 | Atul Gupta | atul | 18 | 8.2 |
| 190101003 | Atul M | atul01 | 18 | 7.2 |
| 190101004 | Atul K | $\perp$ | 18 | 6.2 |
| 190101005 | Atul H | $\perp$ | 18 | 8.6 |

# Creating a table with primary key

## One constraint alone

- Table with NO primary key constraint
- Having UNIQUE constraint on `login` column
- Having NOT NULL constraint on `login` column
- This is identical to specifying `login` column as primary key implicitly

```
CREATE TABLE student(sid int,
                     name char(50),
                     login char(10) unique not null,
                     age int,
                     spi float);
```

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| 190101001 | Atul Kumar | atul | 18 | 8.0 |
| 190101002 | Atul Gupta | atul | 18 | 8.2 |
| 190101003 | Atul M | atul01 | 18 | 7.2 |
| 190101004 | Atul K | ⊥ | 18 | 6.2 |
| 190101005 | Atul H | atul02 | 18 | 8.6 |

# Creating a table with not null columns

```
CREATE TABLE student(sid int,
                     name char(50),
                     login char(10),
                     age int not null,
                     spi float);
```

| student | | | | |
|---|---|---|---|---|
| sid | name | login | age | spi |
| ⊥ | Atul Kumar | atul | 18 | 8.0 |
| 190101000 | Atul Gupta | atul | ⊥ | 8.2 |
| 190101000 | Atul Gupta | atul01 | 18 | ⊥ |
| 190101000 | Atul Gupta | ⊥ | 18 | 8.2 |
| 190101000 | ⊥ | atul02 | 18 | 8.2 |

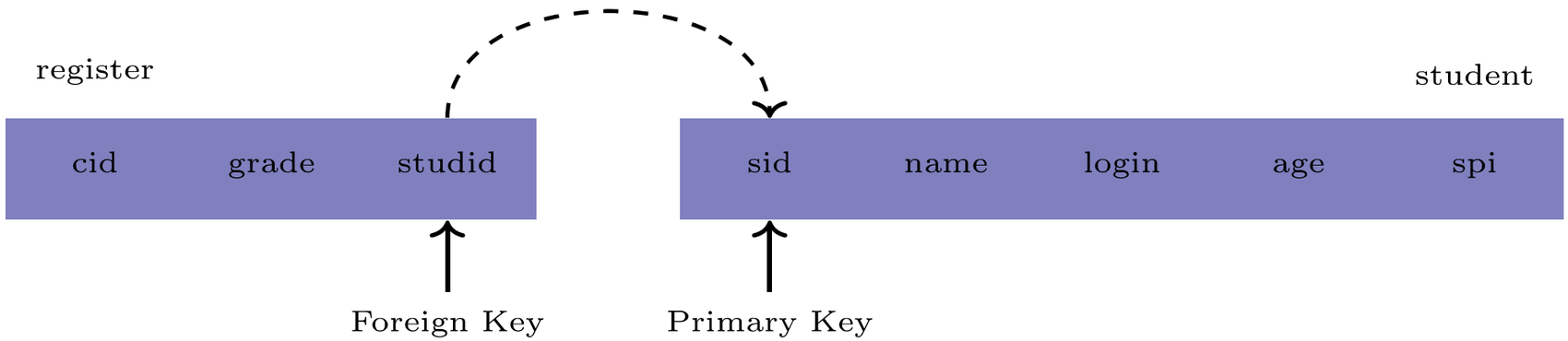
# Foreign Key

## Cases

- Needs two or more tables (Need not be distinct tables!)
- Each (refering) table must have primary key constraint
- Each (refering) table: primary key is expressed using only one column
- One (refering) table has primary key expressed on only one column. Other (refering) table(s) express primary key using two or more columns
- All (refering) tables express primary key using two or more columns

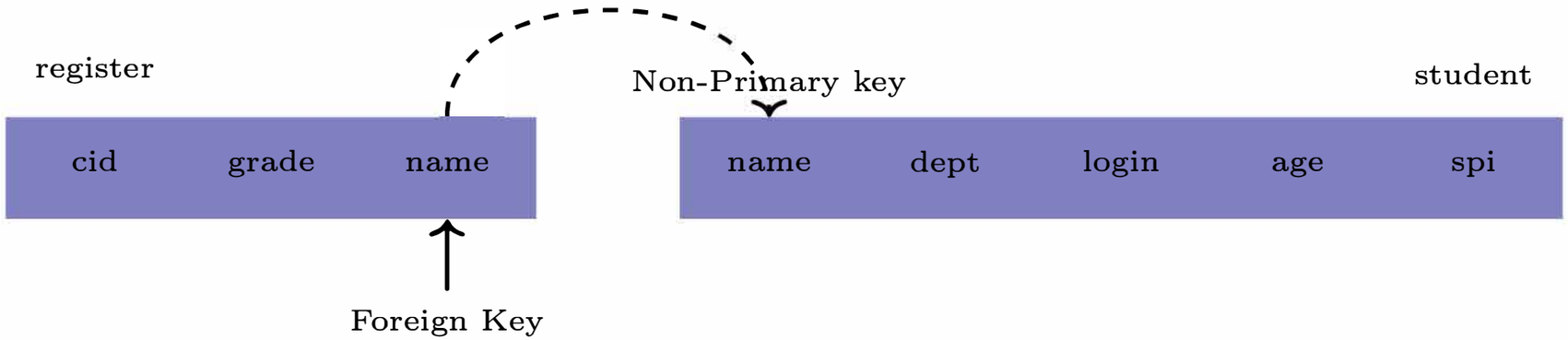
Draw figure for this explanation;

# Foreign Key - Example 1




```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid));
```

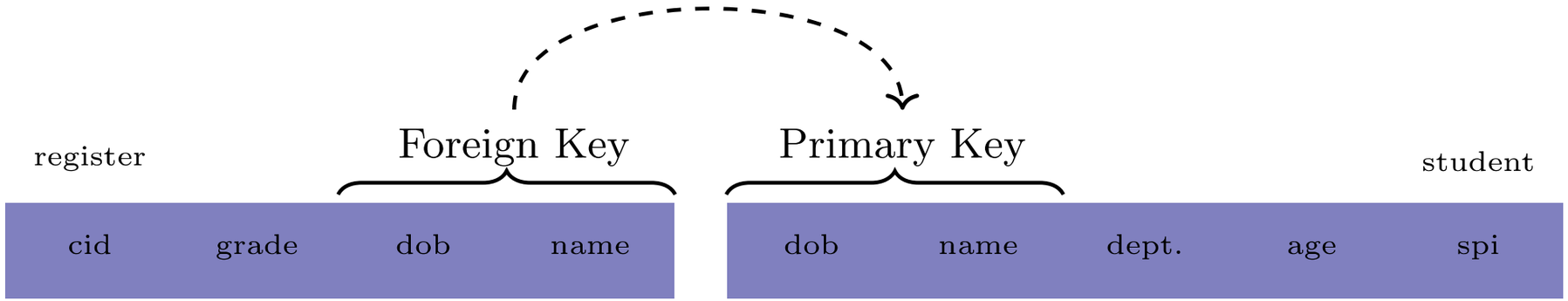
# Foreign Key - Example 2 (Error)




```
CREATE TABLE student(name char(50),
                     dept char(10),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      name char(50),
                      primary key(cid, name)
                      foreign key(name) references student(name));
```

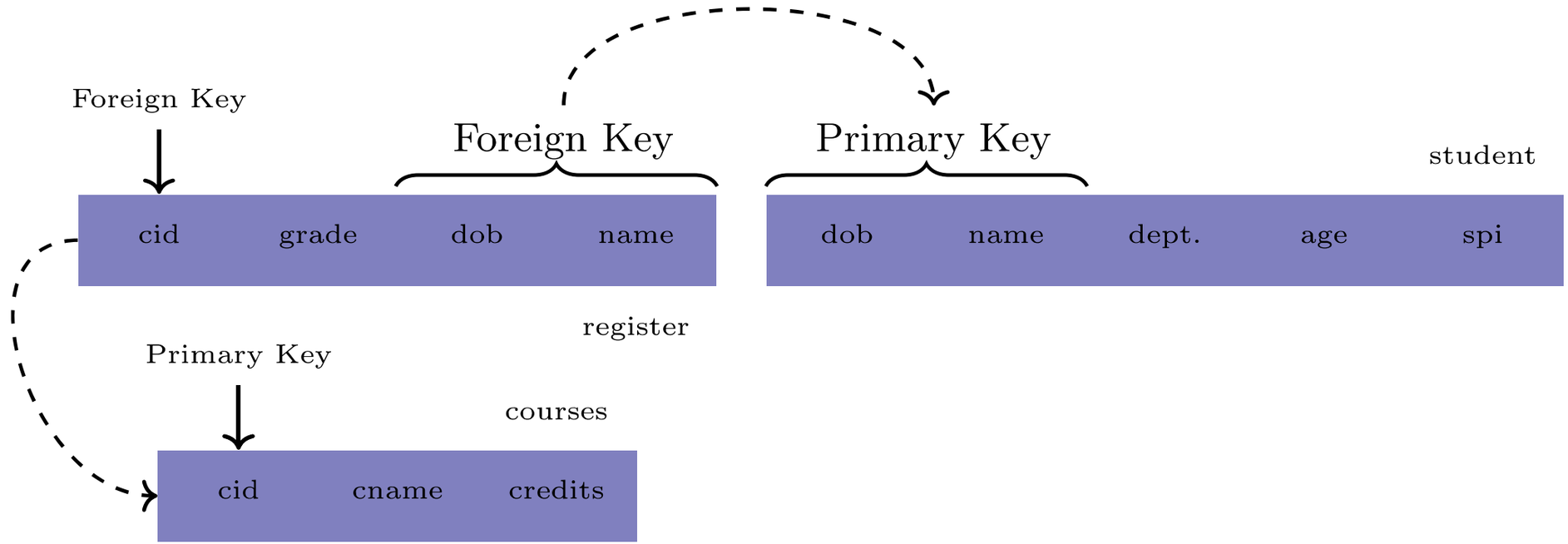
*— Error: name is not primary key in student table*

# Foreign Key - Example 3



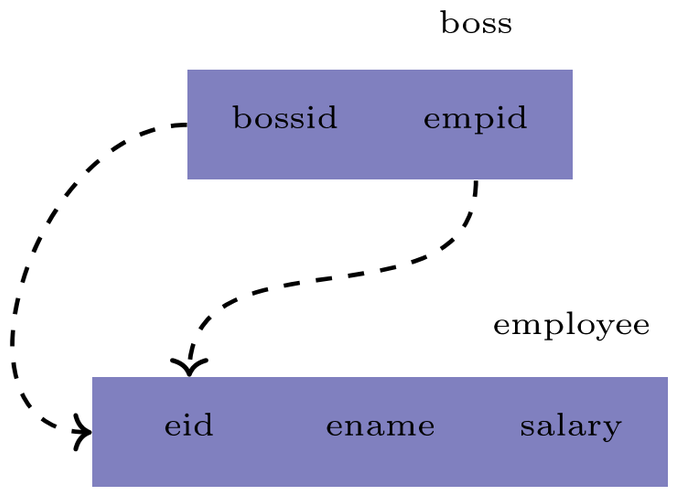

```
CREATE TABLE student(dob date,
                     name char(50),
                     dept char(10),
                     age int,
                     spi float,
                     primary key(dob, name));

CREATE TABLE register(cid int,
                      grade char(2),
                      dob date,
                      name char(50),
                      primary key(cid, dob, name),
                      foreign key(dob, name) references student(dob, name));
```

# Foreign Key - Example 4



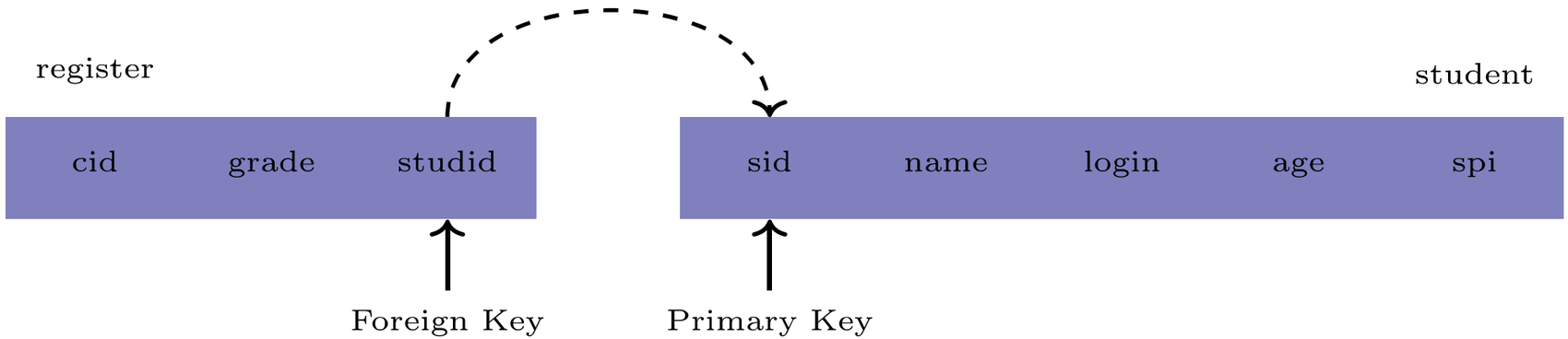

```
CREATE TABLE courses(cid char(2),
                     cname char(50),
                     credits char(6),
                     primary key(cid));

CREATE TABLE register(cid int,
                      grade char(2),
                      dob date,
                      name char(50),
                      primary key(cid, dob, name),
                      foreign key(dob, name) references student(dob, name),
                      foreign key(cid) references courses(cid));
```

# Foreign Key - Example 5



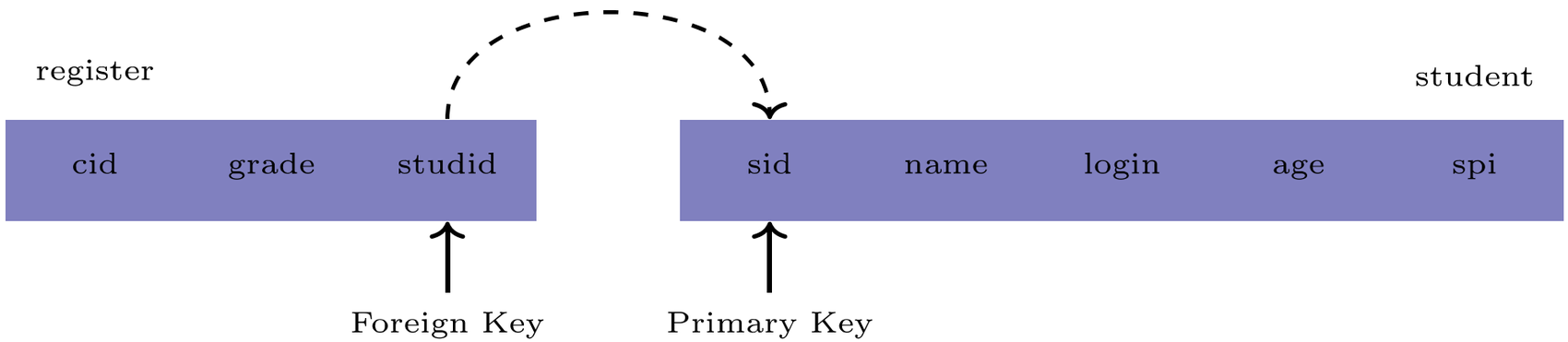

```
CREATE TABLE employee(eid char(10),
                      ename char(50),
                      salary bigint,
                      primary key(eid));

CREATE TABLE boss(bossid char(10),
                  empid char(10),
                  primary key(bossid, empid),
                  foreign key(bossid) references employee(eid),
                  foreign key(empid) references employee(eid));
```

# Foreign Key - Example 1 (Deleting paraent table row) - Action 01 (Delete)



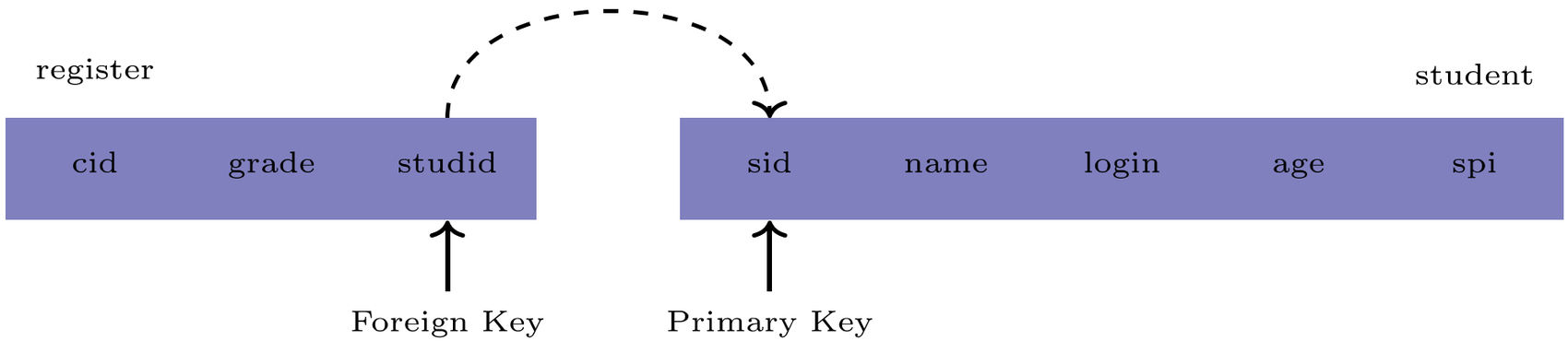

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON DELETE CASCADE);
```

# Foreign Key - Example 1 (Deleting paraent table row) - Action 02 (SET Default value)




- **SET DEFAULT:** This action is recognized by the MySQL parser, but both InnoDB and NDB reject table definitions containing ON DELETE SET DEFAULT or ON UPDATE SET DEFAULT clauses.

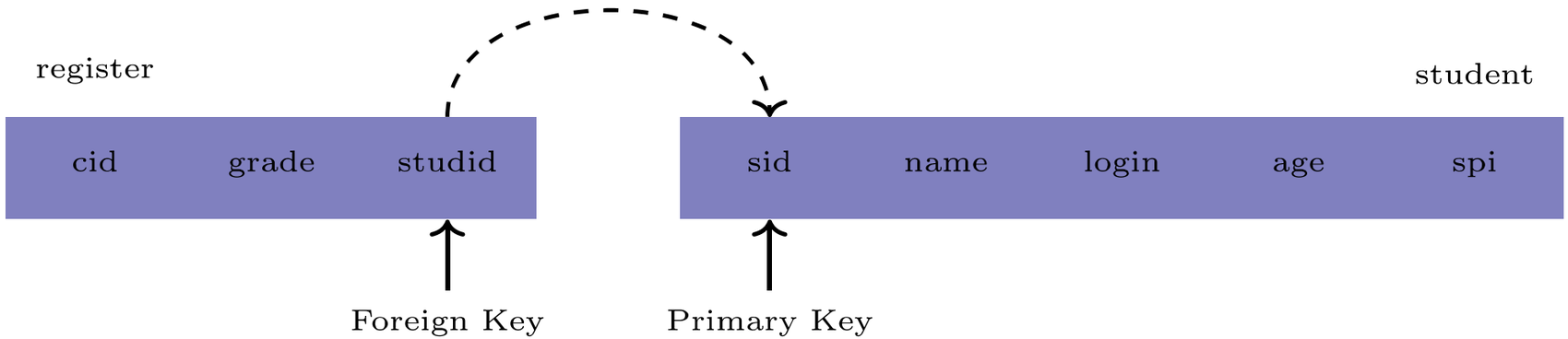
# Foreign Key - Example 1 (Deleting paraent table row) - Action 03 (set NULL)




- **SET NULL:** NOT NULL constraint should not be placed on `studid`

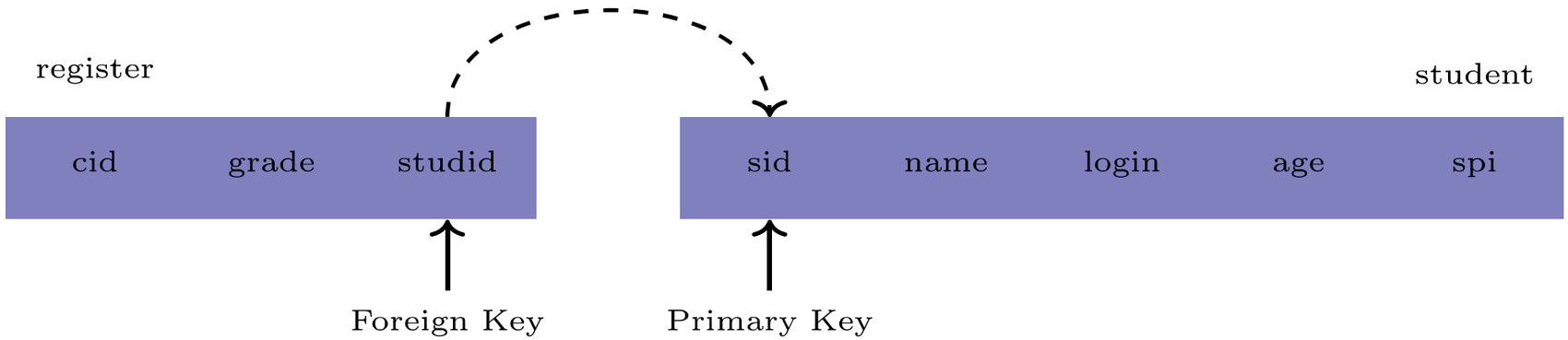
```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON DELETE SET NULL);
```

# Foreign Key - Example 1 (Deleting paraent table row) - Action 04 (Disallow)



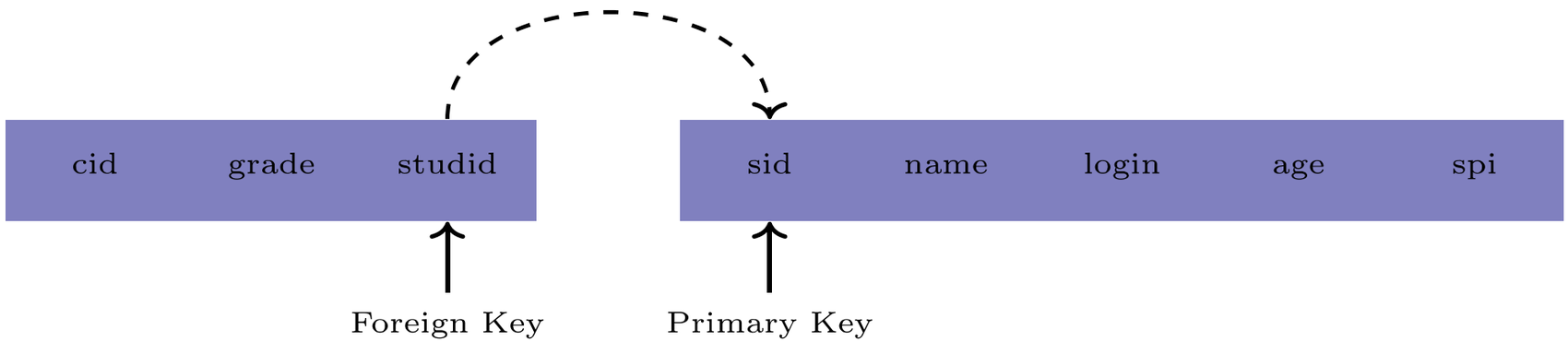

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON DELETE RESTRICT);
```

# Foreign Key - Example 1 (Updating paraent table row) - Action 01 (Update)



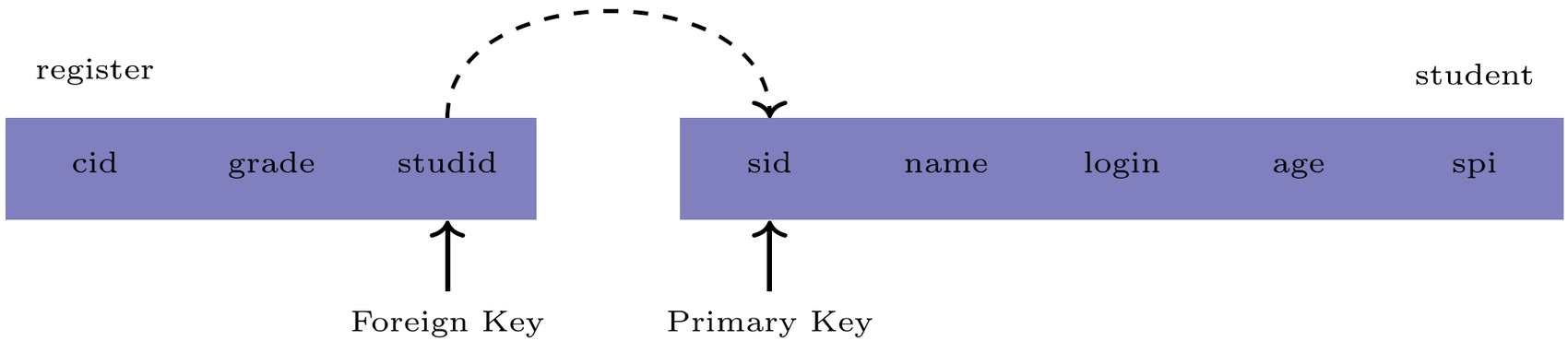

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON UPDATE CASCADE);
```

# Foreign Key - Example 1 (Updating paraent table row) - Action 02 (SET Default value)




- **SET DEFAULT:** This action is recognized by the MySQL parser, but both InnoDB and NDB reject table definitions containing ON DELETE SET DEFAULT or ON UPDATE SET DEFAULT clauses.

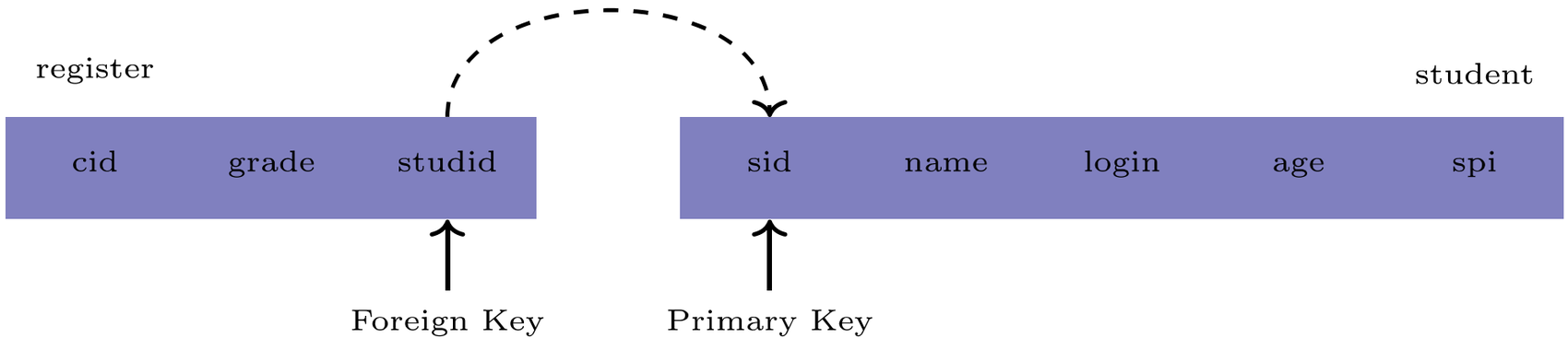
# Foreign Key - Example 1 (Updating paraent table row) - Action 03 (set NULL)




- **SET NULL:** NOT NULL constraint should not be placed on `studid`

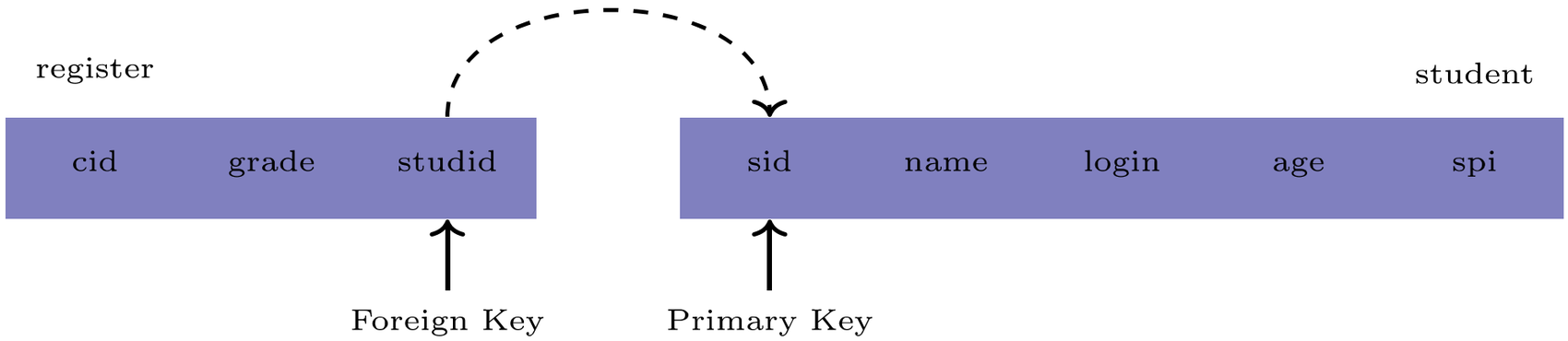
```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON UPDATE SET NULL);
```

# Foreign Key - Example 1 (Updating paraent table row) - Action 04 (Disallow)




```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON UPDATE RESTRICT);
```

# Foreign Key - Example 1 (Deleting/Updating) - Action 01 (Delete/Update)




```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid)
                     ON DELETE CASCADE
                     ON UPDATE CASCADE);
```

# DDL - Table manipulations

## Possible manipulations

- Add column at beginning
- Add column at the middle
- Add column at the end
- Delete column
- Specify data type
- Modify data type
- Add constraints
- Delete constraints

# DDL - Adding a column at the beginning

## Altering Table

| R | | | |
|---|---|---|---|
| c2 | c3 | c4 | c5 |

- Adding a column c1 at the beginning

```
ALTER TABLE R ADD COLUMN c1 INT FIRST;
```

| R | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |

# DDL - Adding a column at the beginning

| R: before adding c1 | | | |
|---|---|---|---|
| c2 | c3 | c4 | c5 |
| 1 | 2 | 3 | 4 |
| 1 | 2 | 3 | 5 |
| 1 | 2 | 4 | 6 |

| R: after adding c1 | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |
| $\perp$ | 1 | 2 | 3 | 4 |
| $\perp$ | 1 | 2 | 3 | 5 |
| $\perp$ | 1 | 2 | 4 | 6 |

# DDL - Adding a column at the beginning

- Existing rows will be unaltered
- Values for the new column for each existing rows is not specified
- $\perp$ by default is added to the existing rows

# DDL - Adding a column

## Altering Table

| R | | | |
|---|---|---|---|
| c1 | c2 | c4 | c5 |

- Adding a column between c2 and c4

```
ALTER TABLE R ADD COLUMN c3 INT AFTER c2;
```

| R | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |

# DDL - Adding a column at the end

## Altering Table

| R | | | |
|---|---|---|---|
| c1 | c2 | c3 | c4 |

- Adding a column c1 at the end

```
ALTER TABLE R ADD COLUMN c5 INT;
```

| R | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |

# DDL - Dropping a column (with no constraints)

## Altering Table

| R | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |

- Dropping the column c1

```
ALTER TABLE R DROP COLUMN c1;
```

| R | | | |
|---|---|---|---|
| c2 | c3 | c4 | c5 |

# DDL - Adding Constraints

## Primary Key

```
CREATE TABLE R(c1 INT, c2 INT, c3 INT, c4 INT);
```

| R | | | |
|---|---|---|---|
| c1 | c2 | c3 | c4 |

- Adding a primary key c1

```
ALTER TABLE R ADD CONSTRAINT my_c1 PRIMARY KEY(c1);
```

# DDL - Adding Constraints

## Foreign Key

```
CREATE TABLE R(c1 INT, c2 INT, c3 INT, c4 INT, PRIMARY KEY(c1));
CREATE TABLE S(s1 INT, s2 INT, PRIMARY KEY(s1));
```

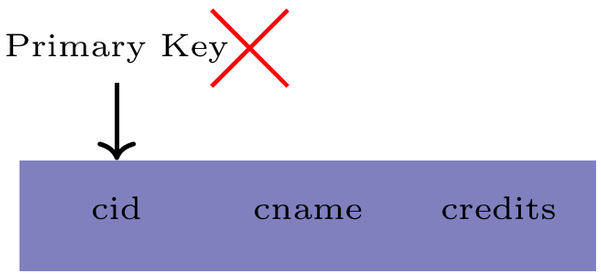
- Adding a foreign key c2 to R

```
ALTER TABLE R ADD CONSTRAINT my_c2_fkey FOREIGN KEY(c2) REFERENCES S(s1);
```

# DDL - Dropping constraints

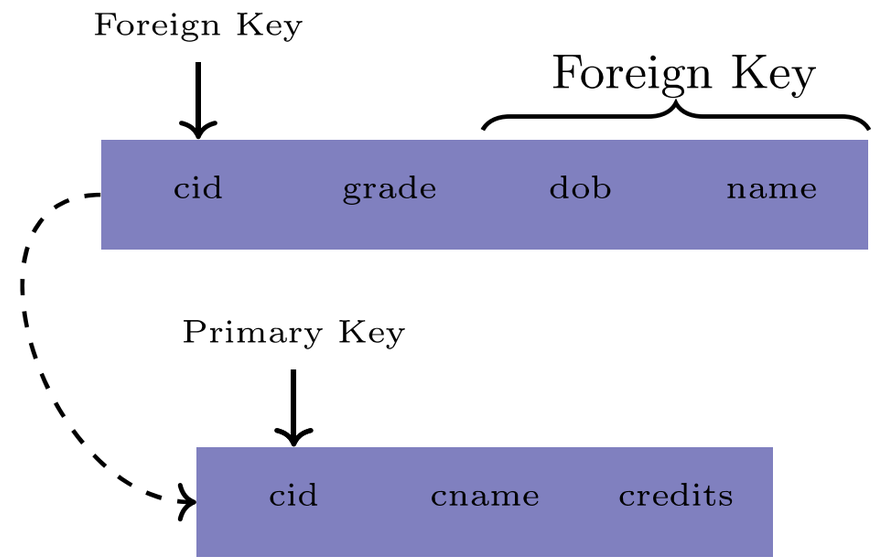
- Primary key - simple case
- Primary key - complex case (includes dropping foreign key)
- NULL
- DEFAULT

# Primary key deletion - simple case



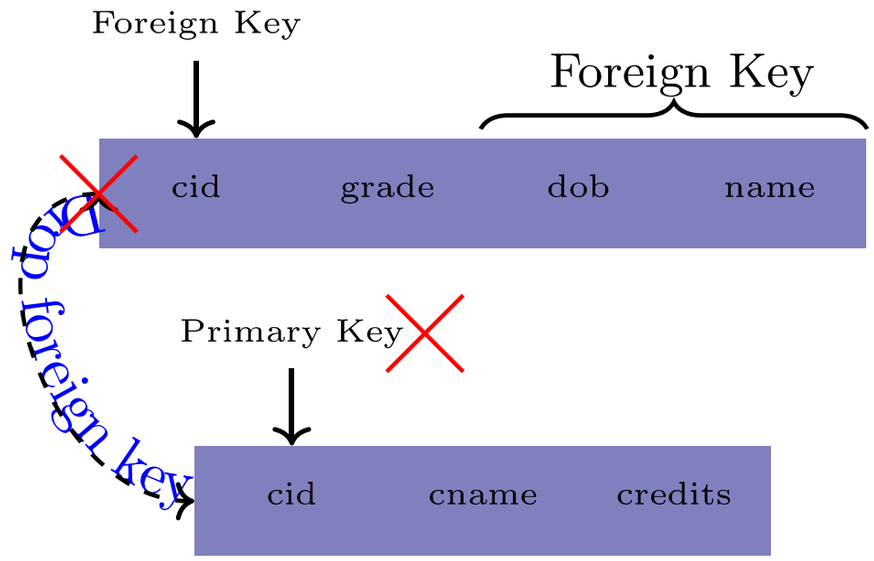

```
ALTER TABLE R DROP PRIMARY KEY;
```

# Primary key deletion - complex case



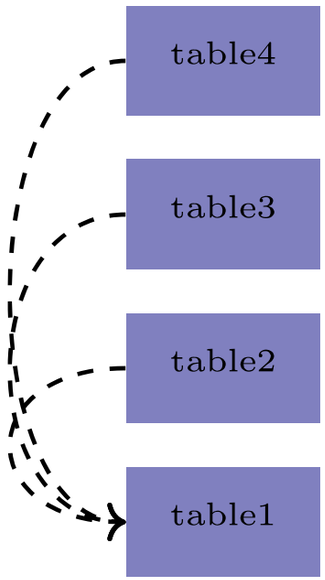

- Remove all foreign keys
- Delete the primary key

# Primary key deletion - complex case




```
ALTER TABLE R DROP FOREIGN KEY my_cid_fkey;
ALTER TABLE R DROP PRIMARY KEY;
```

# Primary key deletion - complex case




- Drop foreign key from table4
- Drop foreign key from table3
- Drop foreign key from table2
- Delete the primary key from table1

# DDL - Dropping default constraint

## DEFAULT value

| R | | | |
|---|---|---|---|
| c2 | c3 | c4 | c5 |

**ALTER TABLE** R **ADD COLUMN** C1 **INT DEFAULT** 10 **FIRST**;

| R | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |

**ALTER TABLE** R **DROP COLUMN** C1;

# DDL - Dropping NOT NULL constraint

## NOT NULL column

| R | | | |
|---|---|---|---|
| c2 | c3 | c4 | c5 |

**ALTER TABLE** R **ADD COLUMN** C1 **INT NOT NULL** 10 **FIRST**;

| R | | | | |
|---|---|---|---|---|
| c1 | c2 | c3 | c4 | c5 |

**ALTER TABLE** R **DROP COLUMN** C1;

# DDL - Changing Domains

## Altering Attribute Domains

```
ALTER TABLE R CHANGE c3 c3 CHAR(20);

ALTER TABLE R CHANGE c3 new_c3 CHAR(30);
```

One has to be carful while changing the data types when columns are either primary key or foreign key constraints.

# DDL - changing domains pit falls

## Altering Attribute Domains

| c1 (int) |
|---|
| 129 |
| 130 |
| 131 |
| 132 |

```
ALTER TABLE R CHANGE c1 c1 tinyint;
```

will result in an error and the operation is not permitted due to `Out of range value for column 'c1'`

# DDL - Changing Domains

## Altering Attribute Domains

| c1 (int) |
|---|
| 1 |
| 2 |
| 3 |
| 4 |

```
ALTER TABLE R CHANGE c1 c1 tinyint;
```

No issues. c1 is made `tinyint`.

# DDL - Default Constraint

## Expressing Default Constraint

```
CREATE TABLE R(c1 INT, c2 INT DEFAULT 245, PRIMARY KEY(c1))
```