Introduction to the Relational Model

Data Basics RCE@Pitt

Why do we care about databases?

- Your data are spread across multiple tables, you don’t have to deal with a pile of disconnected CSV files
- You have too much data for your machine
- You want want persistence of data AND structure
- You need a robust, standardized, and transferrable query language like SQL
- You need to collaborate with others on the same data store
- You want a career in data science outside of academia

Database Systems

- MySQL
- PostgreSQL
- Oracle
- SQLite
  - File-based
  - Different data types
  - Not necessarily multi-user
  - No server process
- NoSQL

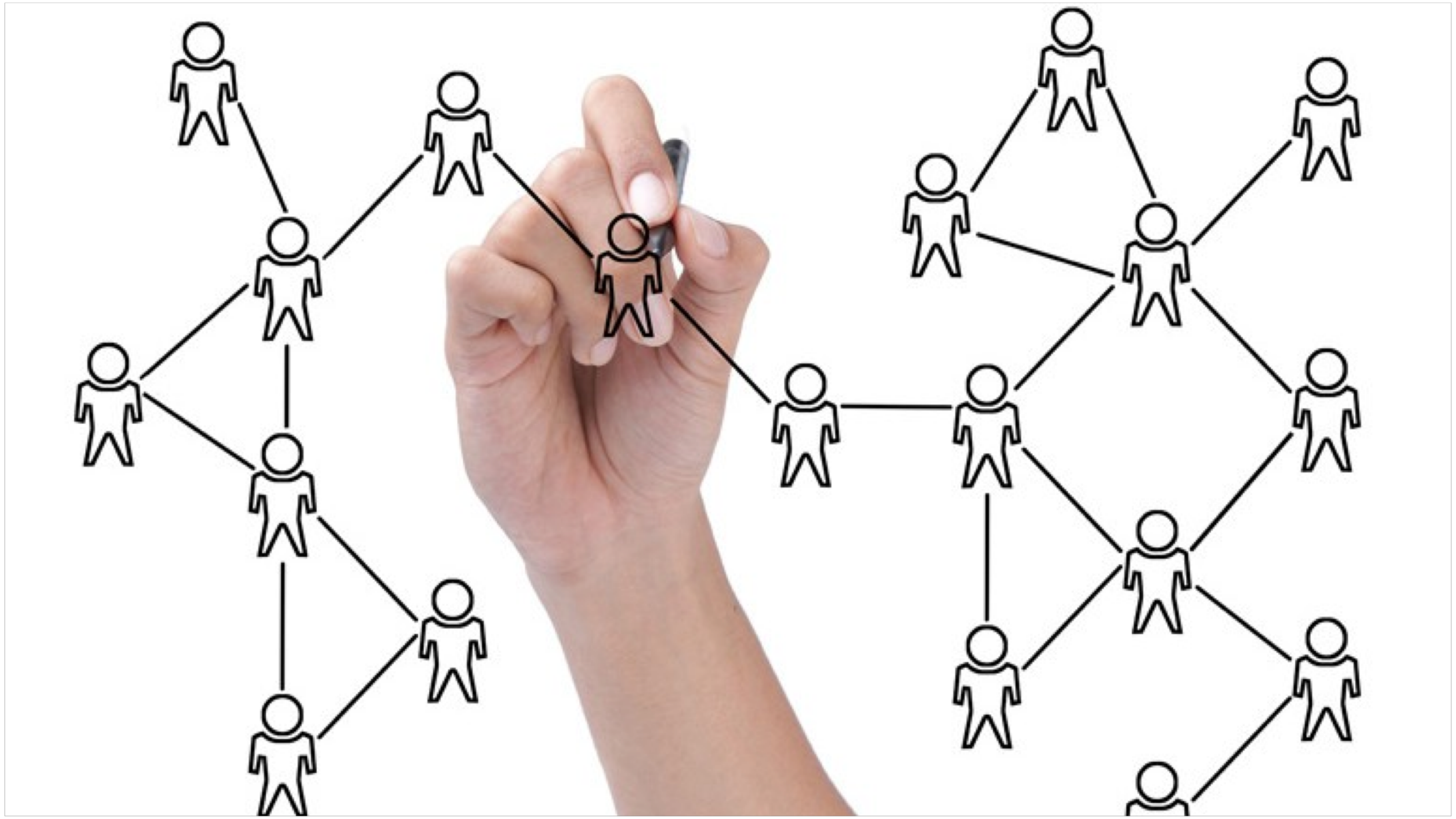
Entities

- Something that can exist *independently* and that can be identified *uniquely*.
- A real world object such as a car or an employee.
- Can be thought of as nouns that come up during the problem description.

Attributes

- Properties of entities are called attributes.
- Attributes represent a subgroup of information of the object represented by the entity.
- Attributes define the individual instances and help to differentiate between each instance by describing their characteristic.

Relations



https://thornleyfallis.com/public-relations/

Relational Model

SID	Name	Major	GPA
546007	Susan	CS	3.80
546100	Bob	CoE	3.65
546500	Bill	CS	3.70

Students

CID	Name
CS 1555	DB
CS 1530	SW
CS 1550	OS

Courses

SID	CID	Grade
546007	CS 1550	A
546007	CS 1530	B+
546100	CS 1550	B

Enrollment

- It is the most popular implementation model
  - Simplest, most uniform data structures, and is the most formal of all data model
- Both entity types and relationship types are represented by ***relations***, i.e., **tables**

Relation Schema

STUDENT

SID	*LName*	*Name*	*Class*	*Major*
123	Smith	John	3	CS
395	Aiken	Mary	4	CS

Schema

◆ ***What is the meaning***?

- A relation schema R specifies
  - The name of the relation
  - the attribute names A_i of R
  - the domain D_i (data type + format) for each attribute A_i

- data type is a set of **atomic data** values:
  - no attribute is a set-valued (1st Normal Form, 1-NF)
  - no attribute is composite
- format specifies the representation of a data value

Example Table Schema

Schema of STUDENT(SID, Name, Major, GPA)

```
 CREATE TABLE  STUDENT
 (     SID      INTEGER,
       Name  CHAR(20),
       Major   CHAR(4),
       GPA     DEC(3, 2)
);
```

SQLite Data Types

Each value stored in an SQLite database (or manipulated by the database engine) has one of the following storage classes:

NULL. The value is a NULL value.

INTEGER. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.

REAL. The value is a floating point value, stored as an 8-byte IEEE floating point number.

TEXT. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).

BLOB. The value is a blob of data, stored exactly as it was input.

Date and Time in SQLite

SQLite does not have a storage class set aside for storing dates and/or times. Instead, the built-in date and time functions of SQLite are capable of storing dates and times as TEXT, REAL, or INTEGER values:

TEXT as ISO8601 strings ("YYYY-MM-DD HH:MM:SS.SSS").

REAL as Julian day numbers, the number of days since noon in Greenwich on November 24, 4714 B.C. according to the proleptic Gregorian calendar.

INTEGER as Unix Time, the number of seconds since 1970-01-01 00:00:00 UTC.

Applications can chose to store dates and times in any of these formats and freely convert between formats using the built-in functions.

NULL in SQLite

SQLite **NULL** is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank.

A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

We’ll come back to this in our demos.

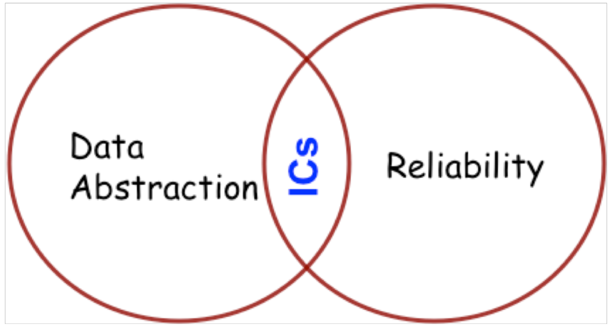
Relational Database Schema

- ❑ A *database schema* is a set of relation schemas and a set of ***integrity constraints***

- ❑ *Structural* Integrity Constraints
  - key constraints: uniqueness of keys
  - entity integrity constraint: no primary key value can be **NULL**
  - referential integrity constraint

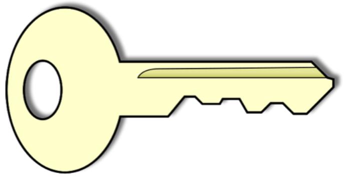
Integrity Constraints (ICs)

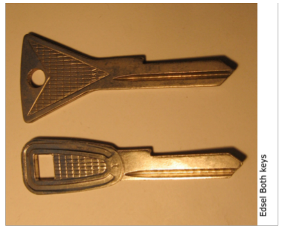
- **IC**: condition that must be true for *any* instance of the database (e.g., domain constraints)
  - A **legal** instance of a relation is one that satisfies all specified ICs
  - ICs are specified when schema is **defined**
  - ICs are enforced when tables are **modified**



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





Edsel Both keys

- ❑ 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 **alternative** or ***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*

Example Table Schema in SQL

Schema of STUDENT(SID, Login, Name, Major, GPA)

```
 CREATE TABLE  STUDENT
(    SID        INTEGER   NOT NULL,
     Login      CHAR(15),
     Name       CHAR(20),
     Major      CHAR(4),
     GPA        DEC(3,2),
      CONSTRAINT STUDENT_PK
          PRIMARY KEY (SID),
      CONSTRAINT STUDENT_UN
          UNIQUE (Login)              -- UNIQUE  can take NULL values
 );
```

Identifying the Key

- What is the key in relation GRADUATE=(SID, Degree, Major, Year) ?

SID	*Degree*	*Major*	*Year*
123	BS	CS	1992
123	MS	CS	1993
064	BA	History	1991
445	PhD	CS	1999

Identifying the Key

- What is the key in relation GRADUATE=(SID, Degree, Major, Year) ?

SID	*Degree*	*Major*	*Year*
123	BS	CS	1992
123	MS	CS	1993
064	BA	History	1991
445	PhD	CS	1999
123	BS	Math	1992

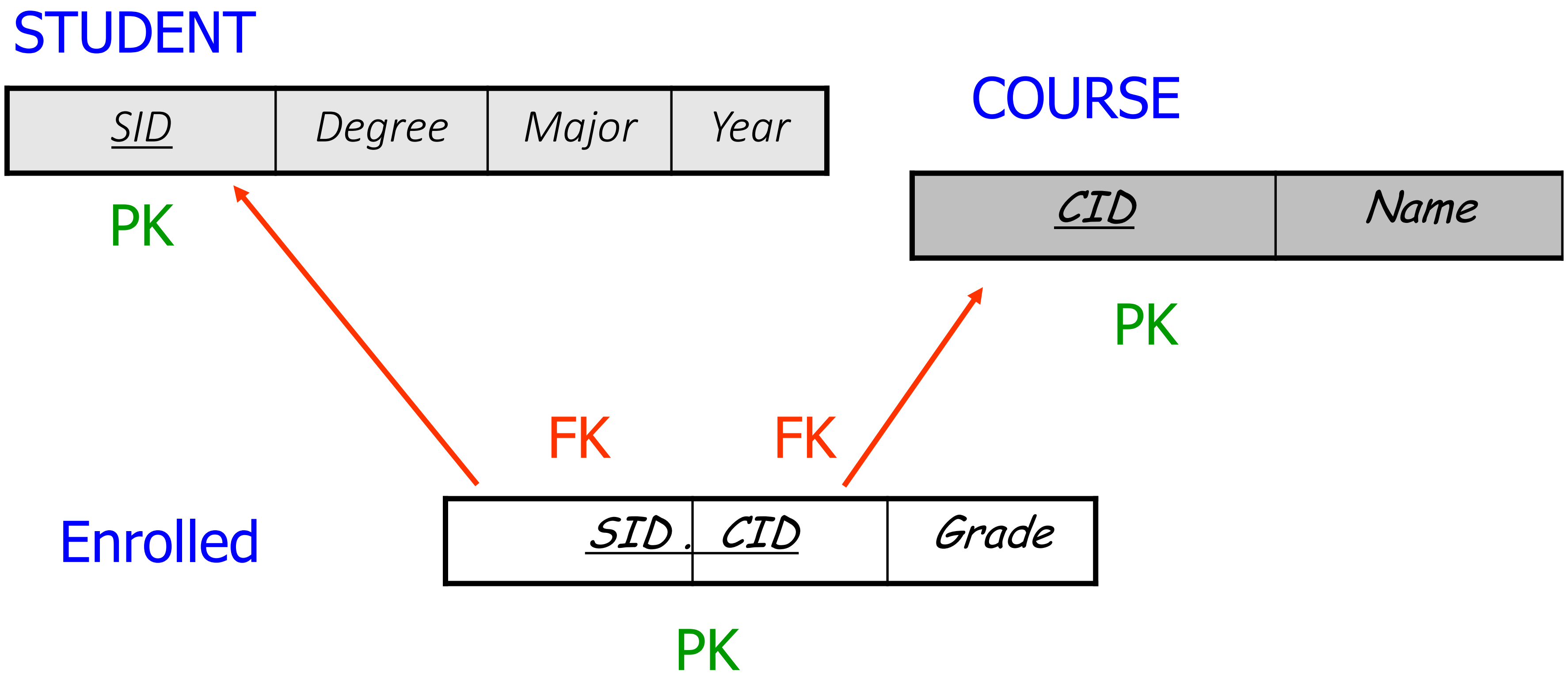
Identifying the Key

- What is the key in relation GRADUATE=(SID, Degree, Major, Year) ?

SID	*Degree*	*Major*	*Year*
123	BS	CS	1992
123	MS	CS	1993
064	BA	History	1991
445	PhD	CS	1999
123	BS	Math	1992
123	MS	Math	1992

Foreign Keys

- *Foreign key* (FK) in relation R_2 is a set of attributes of R_2 that forms a primary key (PK) of another relation R_1.
  - Attributes in FK and PK have the **same domain**



STUDENT

<u>SID</u>	*Degree*	*Major*	*Year*

PK

COURSE

<u>CID</u>	*Name*

PK

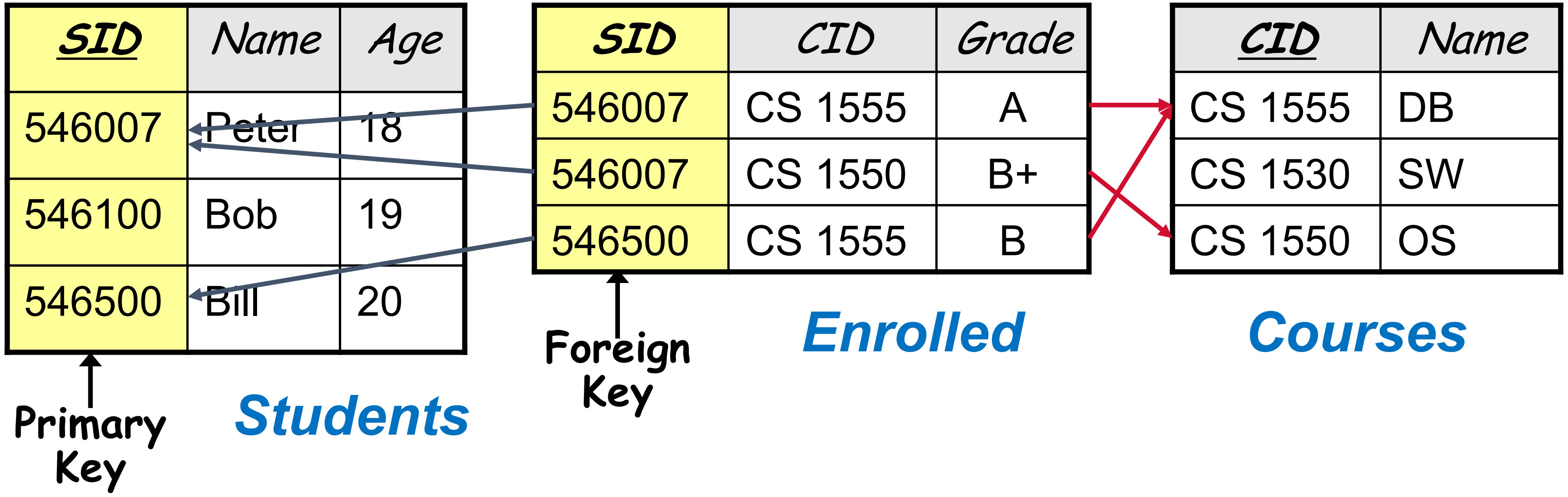
FK FK

Enrolled

<u>SID</u>	*<u>CID</u>*	*Grade*

PK

Foreign Key & Primary Key



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

Primary Key

Students

SID	CID	Grade
546007	CS 1555	A
546007	CS 1550	B+
546500	CS 1555	B

Foreign Key

Enrolled

CID	Name
CS 1555	DB
CS 1530	SW
CS 1550	OS

Courses

- **Foreign key:** Set of fields in one relation that is used to "refer" to a tuple in another relation
  - Must correspond to primary key 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
        - Either valid PK or NULL

Any Attribute can be a Foreign Key

Faculty

FID	*Name*	*Area*
007	Panos	DB
100	Daniel	OS
500	Adriana	AI

Primary Key

Courses

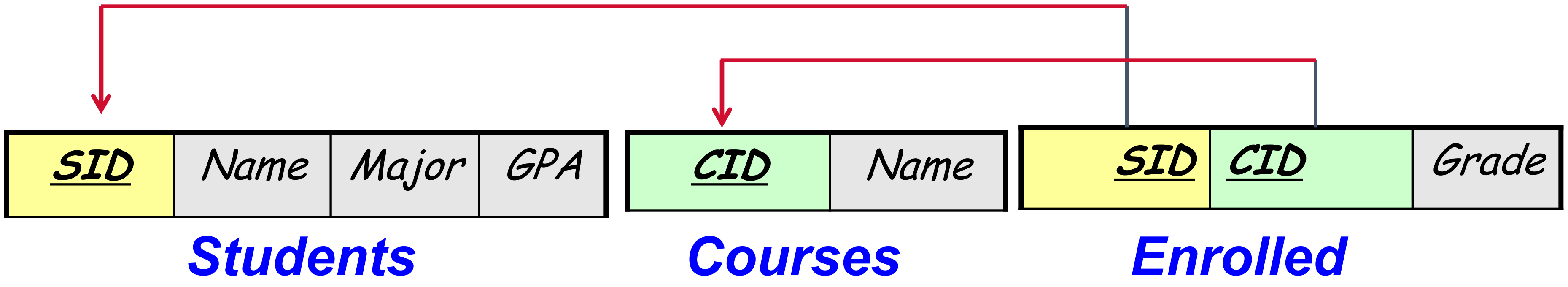
CID	*Name*	*Instructor*
CS 1555	DB	007
CS 1530	SW	NULL
CS 1550	OS	100

Primary Key

Foreign Key

- **Foreign key:** Set of fields in one relation that is used to “refer” to a tuple in another relation
  - Must correspond to primary key of the referred relation
  - If not part of a key, it could be NULL

Foreign Keys in SQL



- □
```
CREATE TABLE Enrolled (
  SID CHAR(20), CID CHAR(20), Grade CHAR(2),
  CONSTRAINT Enrolled_PK PRIMARY KEY (SID, CID) ,
  CONSTRAINT Enrolled_FK_sid
     FOREIGN KEY (SID) REFERENCES Students (SID) ,
  CONSTRAINT Enrolled_FK_cid
     FOREIGN KEY (CID) REFERENCES Courses
  );
```

Now let’s move on to using the relational model with SQLite.

Questions?