Lecture 5: Manipulating Tabular Data

COSC 480 Data Science, Spring 2017
Michael Hay

Goals for today

- Wrap up data processing from last time

- Present relational data model: a way of representing data and constraints on that data

- SQL: a *declarative* language for creating, manipulating relational data

    - Declarative = say ***what*** you want, not ***how***

Motivation: why DBs and SQL?

- **Bigger data**: databases are optimized for processing data that is too large to fit in memory

- **More complex data**: CSV file is great for describing a single collection of entities. What about multiple types of entities? Relations between entities?

- **Common data processing patterns**:

  - Typical operations: *filter* rows/columns, *group* related rows, *aggregate* statistics, *join* different data sources

  - Codified into formal language (SQL)

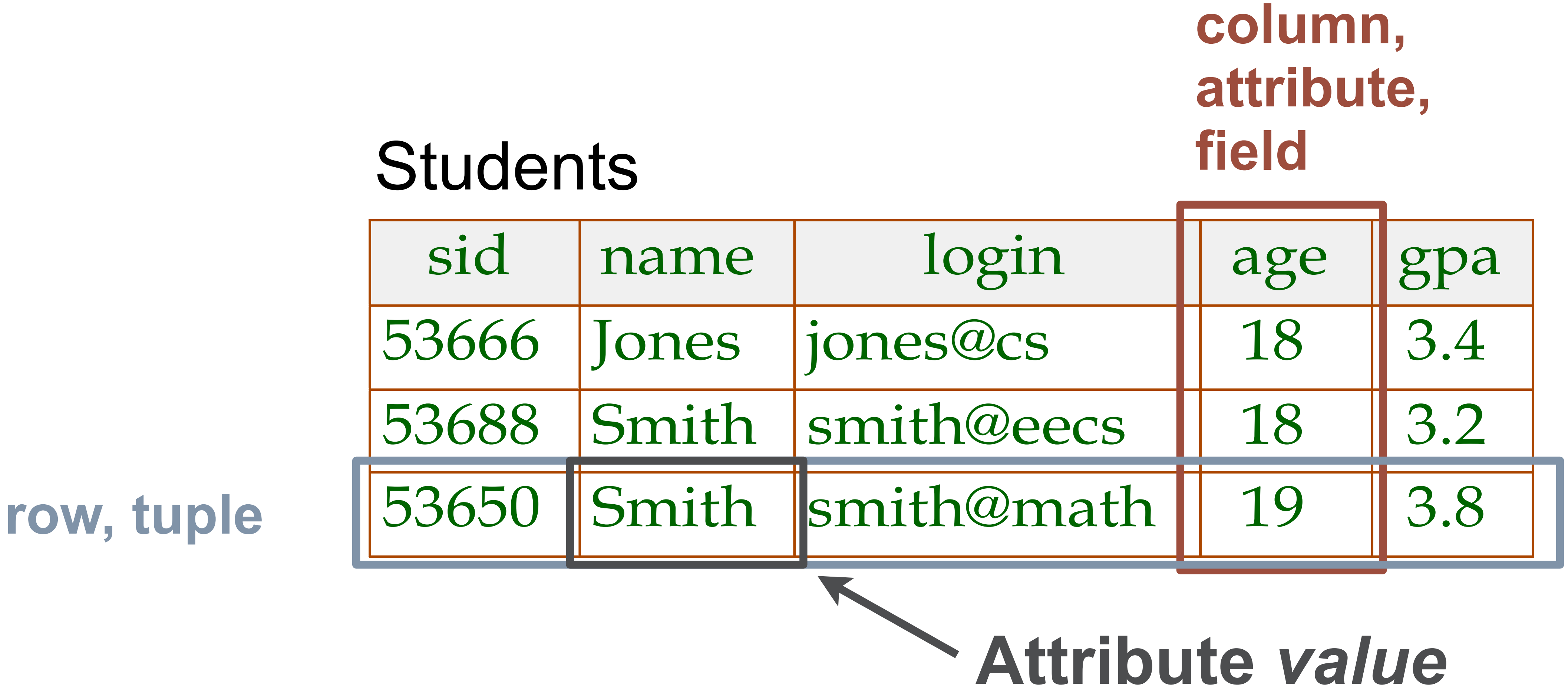
Relational Database: Definitions

- *Relational database:* a set of *relations*
- *Relation:* made up of 2 parts:
  - *Schema* : name of relation, plus name and type/domain of each column.

Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).

  - *Instance* : the actual data at a given time

Relational instance: a table



column, attribute, field

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

row, tuple

Attribute *value*

- #rows = cardinality
- #fields = degree / arity

Synonyms

Formal	Not-so-formal-1	Not-so-formal 2
Relation	Table	
Tuple	Row	Record
Attribute	Column	Field
Domain	Type	

Example Database Instance

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

Takes

sid	cid
1	460
1	301
3	301

COURSE

cid	title	sem
460	DB	F16
480	DS	S17
301	OS	F16

PROFESSOR

fid	name
1	Hay
2	Sommers
8	Mulry

Teaches

fid	cid
1	460
1	480
2	301

Integrity constraints

- An **integrity constraint** (IC) is a condition that must be true for *any* instance of the DB
  - ICs specified when schema defined
  - ICs checked when relations are modified
- If the DBMS checks ICs, stored data is more faithful to real-world meaning (helps avoid data entry errors too!)

Keys

- **Keys** are a kind of integrity constraint
  - **Key**: field(s) such that no two distinct tuples can have same value
  - **Primary key**: if more than one key for relation, one key is chosen (by DBA) to be **primary key**
  - **Foreign key**: field(s) in one relation that is used to “refer" to tuple in another relation. (Must correspond to primary key of the second relation.) Like a “logical pointer.”
- Examples of key constraints in student courses database?

Example Database Instance

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

Takes

sid	cid
1	460
1	301
3	301

COURSE

cid	title	sem
460	DB	F14
480	AI	S14
301	OS	F14

What are the keys of these relations?

Are there any foreign keys?

PROFESSOR

fid	name
1	Hay
2	Sommers
8	Mulry

Teaches

fid	cid
1	460
1	480
2	301

Question: can a student apply to the same college twice?

CREATE TABLE

```
create table Student(
  sID int,
  sName text,
  GPA real CHECK (gpa>0 AND gpa<=4.0),
  sizeHS int,
  PRIMARY KEY(sID)
);
```

Integrity constraint: GPA value must fall within this range

Key constraint: no two rows can have same key value

```
create table Apply(
  sID int,
  cName text,
  major text,
  decision text,
  PRIMARY KEY (sID, cName, major),
  FOREIGN KEY(sID) REFERENCES Student(sID),
  FOREIGN KEY(cName) REFERENCES College(cName)
);
```

Foreign key constraint: every sID value must match some student sID value found in Student;
i.e., no “dangling” references

SQL Overview

See handout.