The Relational Model

Database Systems Lecture 3

Natasha Alechina

In This Lecture

- Relational data integrity
- For more information
  - Connolly and Begg chapter 3
  - E.F. Codd's paper

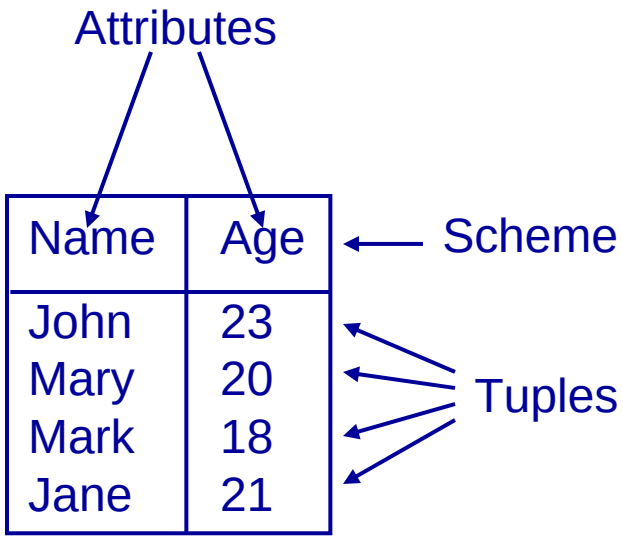
  `A Relational Model of Data for Large Shared Data Banks' – a link from the module web page, ~nza/G51DBS.

The Relational Model

- Introduced in E.F. Codd's 1970 paper "A Relational Model of Data for Large Shared Databanks"
- The foundation for most (but not all) current database systems
- Concerned with 3 main things
  - Data structure (how data is represented)
  - Data integrity (what data is allowed)
  - Data manipulation (what you can do with the data)

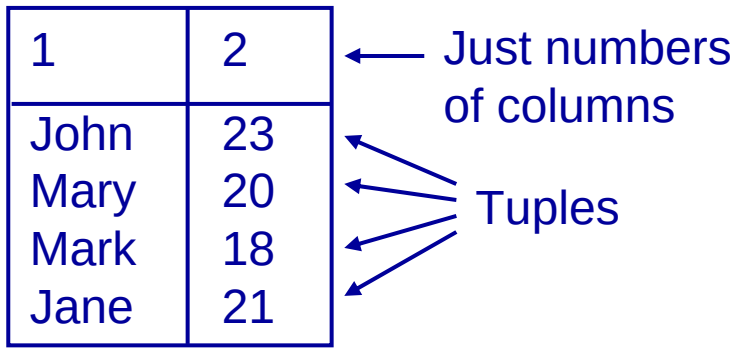
Relational Data Structure

- Data is stored in *relations* (tables)
- Each relation has a *scheme* (heading)
- The scheme defines the relation's *attributes* (columns)
- Data takes the form of *tuples* (rows)

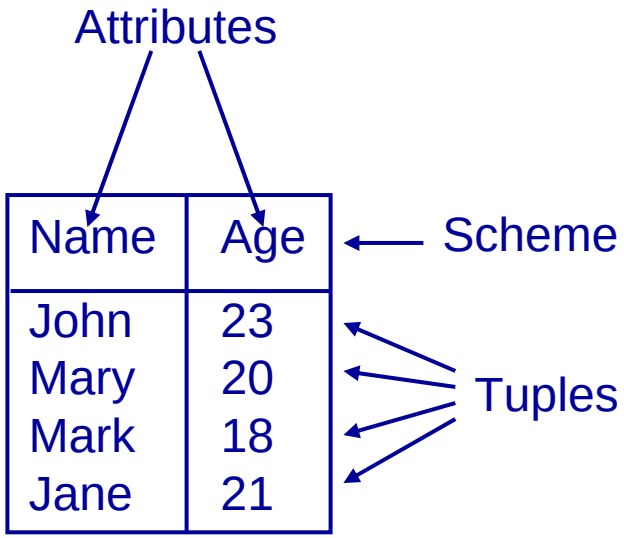


New thing:scheme (and attributes)

Before...



After

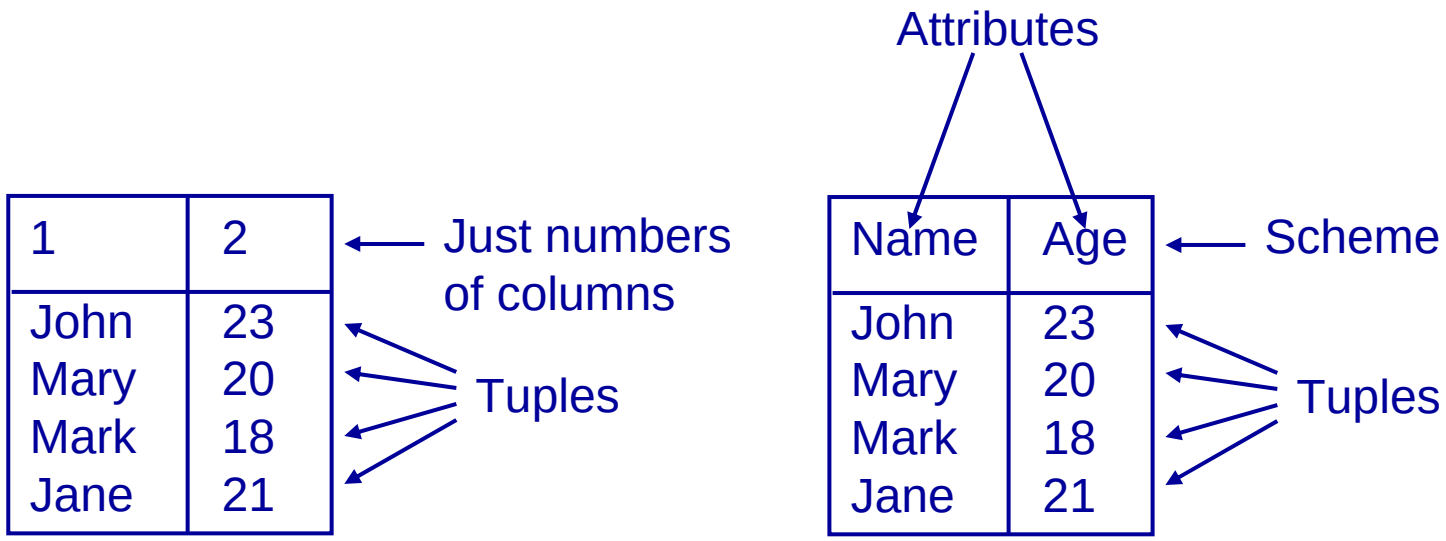


Unnamed and named tuples

A tuple is
<John, 23>

A tuple is

{(Name,John), (Age,23)}



Not a big difference!

- There is no fundamental difference between named and unnamed perspectives on relations
- We could have written tuples <a,b,c> as sets of pairs {(1,a), (2,b),(3,c)}, only we know anyway in which order 1,2,3 go, so we can skip the numbers.
- Written as sets of pairs (partial functions), tuples can be written in any order, e.g. {(3,c),(2,b),(1,a)}.

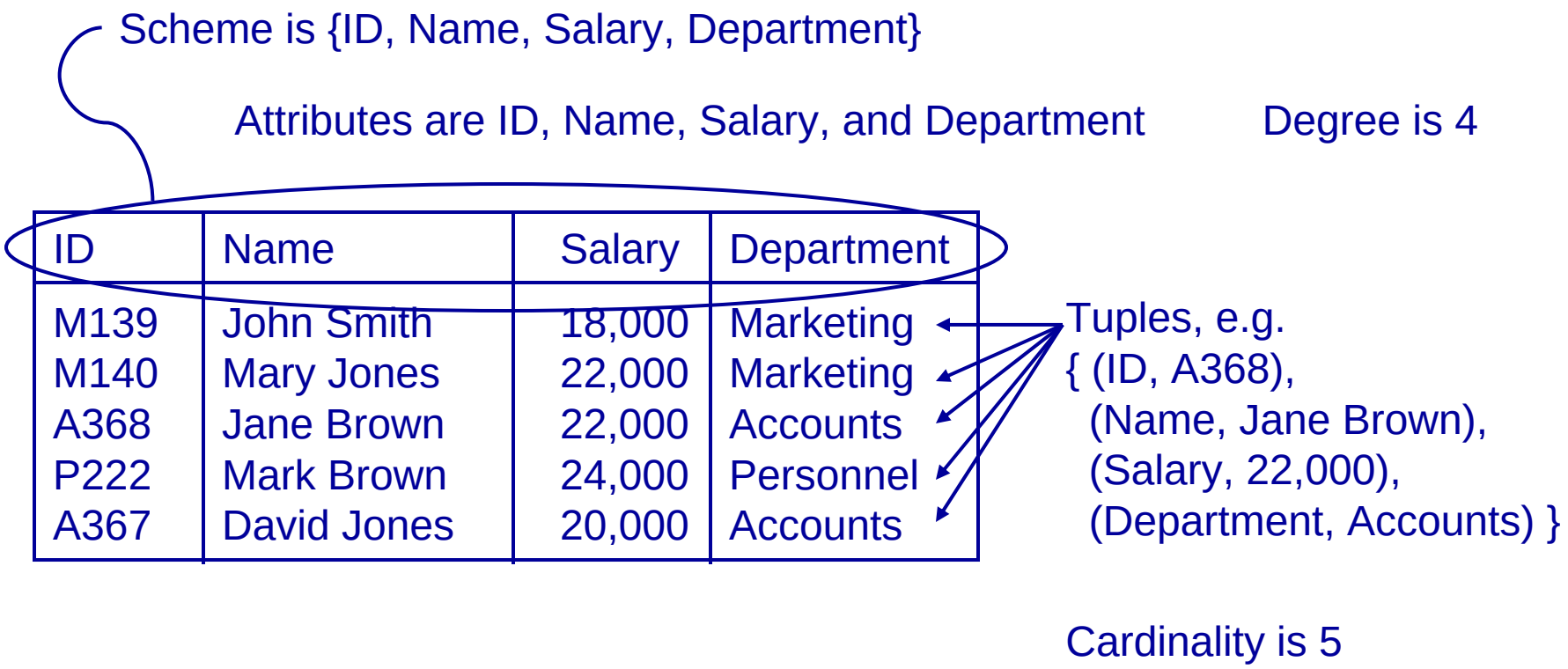
Relational Data Structure

- More formally -
  - A scheme is a set of attributes
  - A tuple assigns a value to each attribute in its scheme
  - A relation is a set of tuples with the same scheme

Name	Age
John	23
Mary	20
Mark	18
Jane	21

{ { (Name, John), (Age, 23) },
{ (Name, Mary), (Age, 20) },
{ (Name, Mark), (Age, 18) },
{ (Name, Jane), (Age, 21) } }

Relations


Scheme is {ID, Name, Salary, Department}
Attributes are ID, Name, Salary, and Department
Degree is 4
ID
Name
Salary
Department
M139
John Smith
18,000
Marketing
M140
Mary Jones
22,000
Marketing
A368
Jane Brown
22,000
Accounts
P222
Mark Brown
24,000
Personnel
A367
David Jones
20,000
Accounts
Tuples, e.g.
{ (ID, A368),
(Name, Jane Brown),
(Salary, 22,000),
(Department, Accounts) }
Cardinality is 5

Relational Data Integrity

- Data integrity controls what data can be in a relation
  - *Domains* restrict the possible values a tuple can assign to each attribute
  - *Candidate* and *Primary Keys* identify tuples within a relation
  - *Foreign Keys* link relations to each other

Attributes and Domains

- A *domain* is given for each attribute
- The domain lists the possible values for that attribute
- Each tuple assigns a value to each attribute *from its domain*

- Examples
  - An ‘age’ might have to come from the set of integers between 0 and 150
  - A ‘department’ might come from a given list of strings
  - A ‘notes’ field might allow any string at all

Candidate Keys

- A set of attributes in a relation is called a candidate key if, and only if,
  - Every tuple has a unique value for the set of attributes (*uniqueness*)
  - No proper subset of the set has the uniqueness property (*minimality*)

ID	First	Last
S139	John	Smith
S140	Mary	Jones
S141	John	Brown
S142	Jane	Smith

Candidate key: {ID}; {First,Last} looks plausible but we may get people with the same name

{ID, First}, {ID, Last} and {ID, First, Last} satisfy uniqueness, but are not minimal

{First} and {Last} do not give a unique identifier for each row

Choosing Candidate Keys

- Important: don't look just on the data in the table to determine what is a candidate key
- The table may contain just one tuple, so anything would do!
- Use knowledge of the real world – what is going to stay unique!

Primary Keys

- One Candidate Key is usually chosen to be used to identify tuples in a relation
- This is called the *Primary Key*
- Often a special ID attribute is used as the Primary Key

NULLs and Primary Keys

- Missing information can be represented using NULLs
- A NULL indicates a missing or unknown value
- More on this later…

- *Entity Integrity*: Primary Keys cannot contain NULL values
- Why: if primary key has NULLs then will not uniquely identify the tuple/entity

Foreign Keys

- *Foreign Keys* are used to link data in two relations. A set of attributes in the first (*referencing*) relation is a Foreign Key if its value always either
  - matches a Candidate Key value in the second (*referenced*) relation, or
  - is wholly NULL
- This is called *Referential Integrity*

Foreign Keys

- a Foreign Key
  - matches a Candidate Key value in the second (*referenced*) relation, or
  - is wholly NULL
- This is called *Referential Integrity*
- Why: either we know precisely what (which entity) we refer to, or we don't refer to any entity.

Foreign Keys - Example

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

{DID} is a Candidate Key for Department - Each entry has a unique value for DID

{DID} is a Foreign Key in Employee - each Employee's DID value is either NULL, or matches an entry in the Department relation. This links each Employee to (at most) one Department

Foreign Keys - Example

Employee

ID	Name	Manager
E1496	John Smith	E1499
E1497	Mary Brown	E1498
E1498	Mark Jones	E1499
E1499	Jane Smith	NULL

{ID} is a Candidate Key for Employee, and {Manager} is a Foreign Key, which refers to the same relation - every tuple's Manager value is either NULL or matches an ID value

Referential Integrity

- When relations are updated, referential integrity can be violated
- This usually occurs when a referenced tuple is updated or deleted
- There are a number of options:
  - RESTRICT - stop the user from doing it
  - CASCADE - let the changes flow on
  - NULLIFY - make values NULL

Referential Integrity - Example

- What happens if
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department?

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

RESTRICT

- RESTRICT stops any action that violates integrity
  - You cannot update or delete Marketing or Accounts
  - You *can* change Personnel as it is not referenced

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

CASCADE

- CASCADE allows the changes made to flow through
  - If Marketing's DID is changed to 16 in Department, then the DIDs for John Smith and Mark Jones also change
  - If Accounts is deleted then so is Mary Brown

Department

DID	DName
~~13~~ 16	Marketing
~~14~~	~~Accounts~~
15	Personnel

Employee

EID	EName	DID
15	John Smith	~~13~~ 16
~~16~~	~~Mary Brown~~	~~14~~
17	Mark Jones	~~13~~ 16
18	Jane Smith	NULL

NULLIFY

- NULLIFY sets problem values to NULL
  - If Marketing's DID changes then John Smith's and Mark Jones' DIDs are set to NULL
  - If Accounts is deleted, Mary Brown's DID becomes NULL

Department

DID	DName
~~13~~ 16	Marketing
~~14~~	~~Accounts~~
15	Personnel

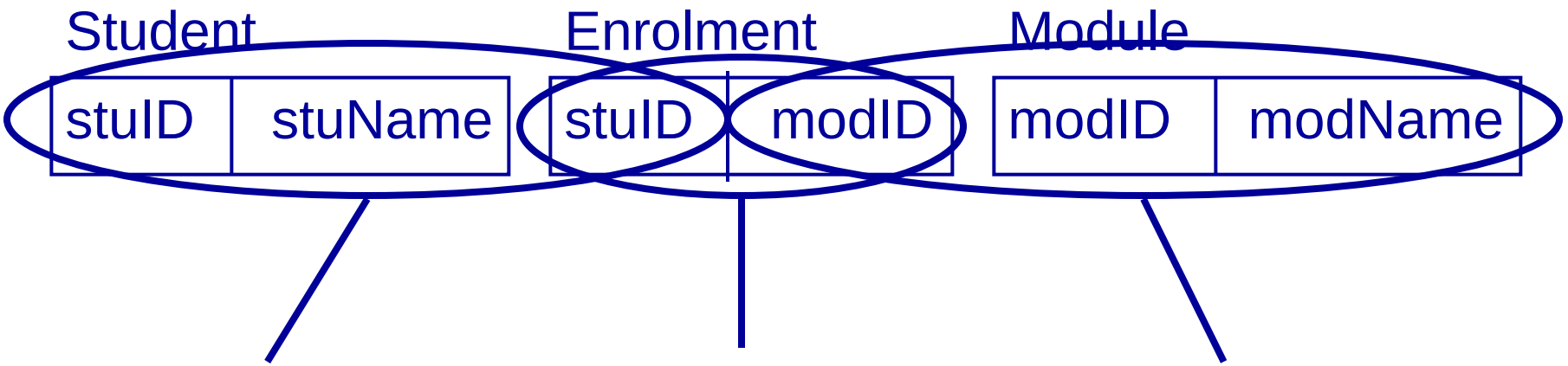
Employee

EID	EName	DID
15	John Smith	~~13~~ NULL
16	Mary Brown	~~14~~ NULL
17	Mark Jones	~~13~~ NULL
18	Jane Smith	NULL

Naming Conventions

- Naming conventions
  - A consistent naming convention can help to remind you of the structure
  - Assign each table a unique prefix, so a student name may be stuName, and a module name modName
- Naming keys
  - Having a unique number as the primary key can be useful
  - If the table prefix is abc, call this abcID
  - A foreign key to this table is then also called abcID

Example



These entries are clearly related to the Student table

These entries are clearly related to tables other than Enrolment

These entries are clearly related to the Module table

Next Lecture

- Entity/Relationship models
  - Entities and Attributes
  - Relationships and Cardinality Ratios
  - E/R Diagrams
- For more information
  - Connolly and Begg chapter 11.
  - Ullman and Widom chapter 2.