The Relational Model

Database Systems

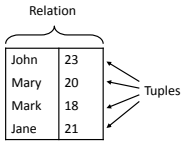
Michael Pound

This Lecture

- The Relational Model
  - More on Relations
  - Relational data integrity
- Further reading
  - Database Systems, Connolly & Begg, Chapter 4.2
  - The Manga Guide to Databases, Chapter 2

Last Lecture

- Data is stored in *relations* (tables)
- Data takes the form of *tuples* (rows)
  - The order of tuples is not important
  - There must not be duplicate tuples



Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

R

Anne	111111
Bob	222222

S

Chris	111111
Dan	222222

Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

R x S

Anne	111111	Chris	111111
Anne	111111	Dan	222222
Bob	222222	Chris	111111
Bob	222222	Dan	222222

Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

$\sigma_{col(2) = col(4)}(R \times S)$

Anne	111111	Chris	111111
Bob	222222	Dan	222222

Example from Last Lecture

What is the result of the following operation $\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$

Anne	Chris
Bob	Dan

Example from Last Lecture

What about a single table? Can we find a list of pairs of people who share a phone number?

R

Anne	111111
Chris	222222
Bob	333333
Dan	111111
Max	222222
Sam	444444
Joe	555555

Example from Last Lecture

What about a single table? Can we find a list of pairs of people who share a phone number?

A: $\pi_{1,3}(\sigma_{col(2) = col(4) \text{ and } col(1) \neq col(3)}(R \times R))$

R

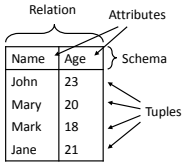
Anne	111111
Chris	222222
Bob	333333
Dan	111111
Max	222222
Sam	444444
Joe	555555

Schemas and Attributes

- Previously, we referenced specific columns in a relation using numbers
  - E.g. $\pi_{1,2}(R)$
- It is often helpful to reference columns using names, which we will have to provide
- Attributes are named columns in a relation
- A schema defines the attributes for a relation

Relational Data Structure

- Each relation has a *schema* (sometimes called a scheme or heading)
- The schema defines the relation's *attributes* (columns).



Named and Unnamed Tuples

- Tuples specify values for each attribute in a relation
- When writing tuples down, they can be named as sets of pairs, e.g.
  - { (1, John), (2, 23) } or { (2, 23), (1, John) }
  - { (Name, John), (Age, 23) }
- Or unnamed, for convenience, e.g.
  - (John, 23) (equivalent to the above)
- There is no real difference between named and unnamed tuples, but be careful with the ordering of unnamed tuples.

Relational Data Structure

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

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) } }

Example Relation

Attributes are ID, Name, Salary and Department

The degree of the relation 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

Schema is { ID, Name, Salary, Department }

Tuples, e.g.
{ (ID, A368),
(Name, Jane Brown),
(Salary, 22,000),
(Department, Accounts)}

The cardinality of the relation 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* consist of an attribute, or set of attributes, that uniquely identify all tuples
  - *Foreign Keys* link relations to each other

Attributes and Domains

- A *domain* is given for each attribute
- The domain lists possible values for the 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 list of given strings
  - A 'notes' field may allow any string at all

Candidate Keys

- A set of attributes in a relation is a candidate key if, and only if:
  - Every tuple has a unique value for that 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 is {ID}; {First, Last} looks plausible, but people might have 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

- You can't necessarily infer the candidate keys based solely on the data in your table
  - More often than not an instance of a relation will only hold a small subset of all the possible values
- You must use knowledge of the real-world to help

Choosing Candidate Keys

What are the candidate keys of the following relation?

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Choosing Candidate Keys

The candidate keys are {OfficeID}, {Phone} and {Name, Postcode/Zip}

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Note: Keys like {Name, Country, Phone} satisfy uniqueness, but not minimality

Primary Keys

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

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

We might use either {ID} or {First,Last} as the primary key. ID is more convenient as we know it will always be unique. People could have the same name

NULLs and Primary Keys

- Missing information can be represented using NULLs
- A NULL indicates a missing or unknown value
- This will be discussed in a later lecture

- *Entity integrity*

  Primary Keys cannot contain NULL values

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:
  - Matches a Candidate Key value in a second (referenced) relation
  - Is wholly NULL
- This is called *Referential Integrity*

Foreign Keys Example

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

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

Employee

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

{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

Recursive 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 that refers to the same relation. Every tuple's Manager value must match and ID value, or be NULL

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 when this occurs:
  - RESTRICT – stop the user from doing it
  - CASCADE – let the changes flow on
  - SET NULL – make referencing values null
  - SET DEFAULT – make referencing values the default for their column

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	14
17	John	13
18	Jane	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	14
17	John	13
18	Jane	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~~	~~14~~
17	John	~~13~~ 16
18	Jane	NULL

SET NULL

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

Department

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

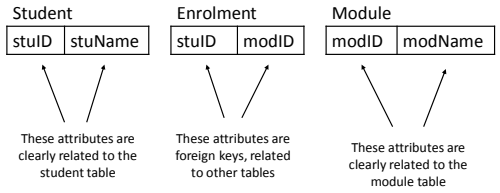
Employee

EID	EName	DID
15	John Smith	~~13~~ NULL
16	Mary	~~14~~ NULL
17	John	~~13~~ NULL
18	Jane	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
  - You may even wish to assign a project prefix to the tables you use
- 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

Naming Example



Next Lecture

- Entity/Relationship models
  - Entities and Attributes
  - Relationships
  - Attributes
  - E/R Diagrams
- Further Reading
  - Database Systems, Connolly & Begg, Chapter 12
  - The Manga Guide to Databases, Chapter 3