



THE RELATIONAL MODEL

Introduction to Database Systems

*Mahdi Akhi
Sharif University of Technology*

IN THIS LECTURE

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

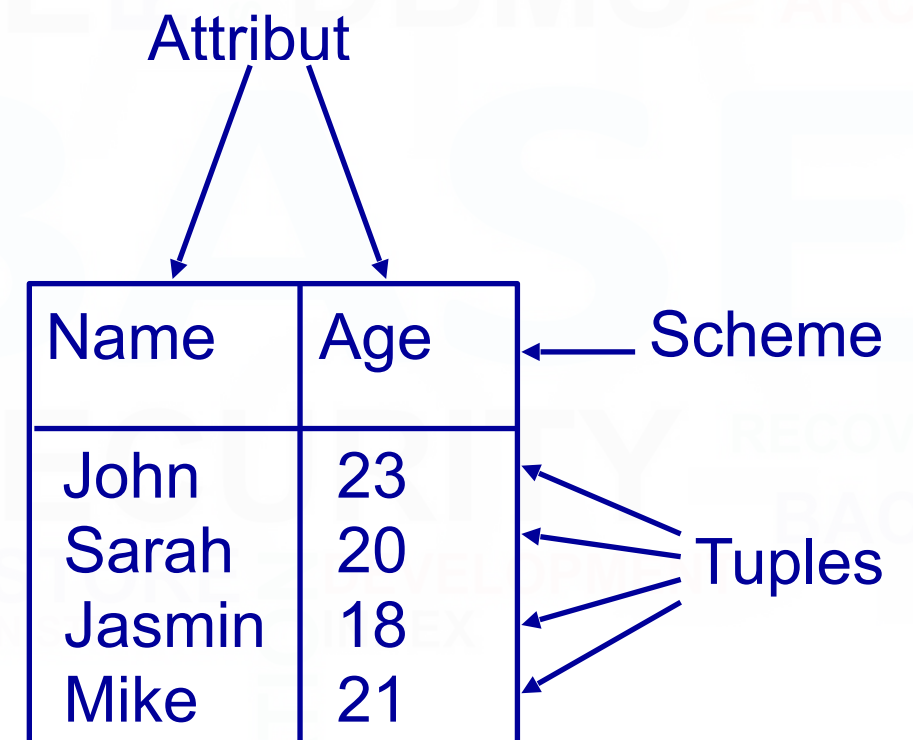
`A Relational Model of Data for Large Shared Data Banks'

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...

1	2
John	23
Sarah	20
Jasmin	18
Mike	21

← Just numbers of columns

← Tuples

After

Attributes

Name	Age
John	23
Sarah	20
Jasmin	18
Mike	21

← Scheme

← Tuples

UNNAMED AND NAMED TUPLES

A tuple is

<Ahmad, 23>

A tuple is

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

1	2
John	23
Sarah	20
Jasmin	18
Mike	21

← Just numbers of columns

← Tuples

Attributes

Name	Age
John	23
Sarah	20
Jasmin	18
Mike	21

← Scheme

← Tuples

NOT A BIG DIFFERENCE!

- There is no fundamental difference between **named** and **unnamed** perspectives on relations
- We could have written tuples $\langle a, b, c \rangle$ 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
Sarah	20
Jasmin	18
Mike	21

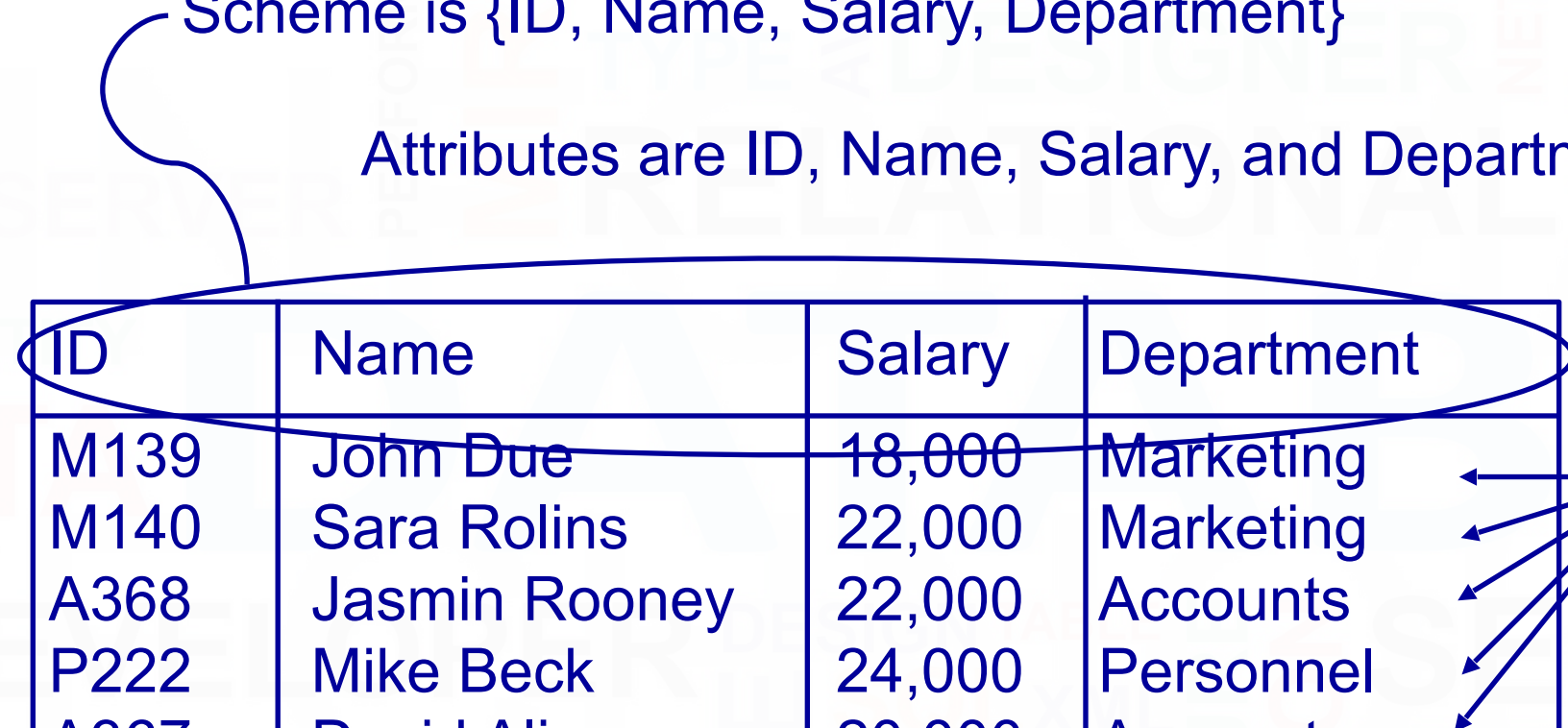
{ { (Name, Ahmad), (Age, 23) },
{ (Name, Nemat), (Age, 20) },
{ (Name, Saba), (Age, 18) },
{ (Name, Javad), (Age, 21) } }

RELATIONS

Scheme is {ID, Name, Salary, Department}

Attributes are ID, Name, Salary, and Department

Degree is 4



ID	Name	Salary	Department
M139	John Due	18,000	Marketing
M140	Sara Rolins	22,000	Marketing
A368	Jasmin Rooney	22,000	Accounts
P222	Mike Beck	24,000	Personnel
A367	David Alison	20,000	Accounts

Tuples, e.g.
{ (ID, A368),
(Name, Javad Azad),
(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	Due
S140	Sarah	Rolins
S141	Jasmin	Rooney
S142	Mike	Beck

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 be true!
- 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

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 - 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 Due	13
16	Sarah Rolins	14
17	Jasmin Rooney	13
18	Mike Beck	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

FOREIGN KEYS - EXAMPLE

Employee

ID	Name	Manager
E1496	John Due	E1499
E1497	Sarah Rolins	E1498
E1498	Jasmin Rooney	E1499
E1499	Mike Beck	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

ID	First	Last
S139	John	Due
S140	Sarah	Rolins
S141	Jasmin	Rooney
S142	Mike	Beck

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 Due	13
16	Sara Rolins	14
17	Jasmin Rooney	13
18	Mike Beck	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 Due	13
16	Sara Rolins	14
17	Jasmin Rooney	13
18	Mike Beck	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 Due and Jasmin Rooney also change
 - If Accounts is deleted then so is Sara Rollins

Department

DID	DName
13 16	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Due	13 16
16	Sara Rollins	14
17	Jasmin Rooney	13 16
18	Mike Beck	NULL

NULLIFY

- **NULLIFY** sets problem values to NULL
 - If Marketing's DID changes then John Due's and Jasmin Rooney's DIDs are set to NULL
 - If Accounts is deleted, Sara Rolins's DID becomes NULL

Department

DID	DName
13 16	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Due	13 NULL
16	Sara Rolins	14 NULL
17	Jasmin Rooney	13 NULL
18	Mike Beck	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 lecturer name lecName
- 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**

END

Thanks to Mohammad Tanhaei, Assistant Prof. At Ilam University