Chapter 2: Intro to Relational Model

Database System Concepts, 6th Ed.

Example of a Relation

Attributes (or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Tuples (or rows)

Shows a *relationship among a set of values*

Table/Relation

A collection of such relationships

Attribute Values

Relation Schema

- Relation
  - Introduced by Dr. E.F. (“Ted”) Codd in 1970… based on set theory
- Collection of relation schemas prepares a whole database
- Relation Schema describes
  - A blueprint of a database that outlines the way data is organized into tables
  - Structure and constraints of data representing in a particular domain
- Does not contain any type of data
- Each tuple is divided into fields called **Domains**

Relation Schema and Instance

- **Relation Schema**
  - $A_1, A_2, \ldots, A_n$ are *attributes*
  - $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

    Example:

    instructor = (*ID, name, dept_name, salary*)
  - Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a subset of

    $D_1 \times D_2 \times \ldots \times D_n$

    Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
- **Relation Instance**
  - The current values of a relation are specified by a table
  - An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Attribute Types

- **Domain of the attribute**
  - The set of allowed values for each attribute
  - Attribute values are (normally) required to be **atomic**; that is, indivisible
  - e.g. 1) For the instructor phone field is used, If the instructor
    - Has one phone, single value in a field
      - » Atomic
    - Have series of phones, multiple values in a field separated by delimiter (comma/space)
      - » Nonatomic (as sub part is useful)
  - e.g. 2) Phone is having country + state code + number
    - Atomic / NonAtomic
    - Application dependent
      - » If all together as a one number, then Atomic
      - » If subparts to treat separately, then NonAtomic

Attribute Types

- **Domain of the attribute**
  - The special value ***null*** is a member of every domain
    - Can be utilized if the value is unknown or does not exist
      - e.g. If the instructor does not have a phone

Relations are Ordered or Unordered?

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Until explicitly asked….

Database

- A database consists of multiple relations
- Information about an University is broken up into parts

  instructor
  student
  advisor

- Design:

  univ (*instructor Id, name, dept_name, salary, student_Id*, ..)

- Results in Bad Database design
  - Repetition of information (e.g., two students have the same instructor)
  - The need for null values (e.g., represent an student with no advisor)
- **Normalization** theory deals with how to design “good” relational schemas

Keys

- *Used to locate data uniquely*
- *Can be of*
  - *Single Attribute*
  - *More than one attribute*
- **Types**
  - **Superkey**
  - **Candidate key**
  - **Primary key**
  - **Foreign key**

Keys

- **Super key**
  - *Set of one or more attributes*
  - *K* is a **superkey** of *R* if values for *K* are sufficient to **identify a unique tuple** of each possible relation *r(R)*, where K $\subseteq$ R
  - Example: {*ID*} and {ID, name} are both superkeys of *instructor*

- **Candidate key**
  - Superkey *K* is a **candidate key** if *K* is **minimal subset of super key**
  - Example: {*ID*} is a candidate key for *Instructor*
  - *{ID, name} is NOT a candidate key as subset of ID can uniquely identify all tuples*

Keys

- **Primary key**
  - **Helps to uniquely identify the tuple**
  - **Only one primary key per table is permitted**
  - **One of the candidate keys** is selected as the primary key
  - **Care 1:** The name of person is not to considered as primary key as many **people with the same name** may exist
  - **Care 2:** Choose such that its attribute **values are never or very rarely changed**, address cannot be considered as primary key as it can be changed
  - It is a super key also. BUT, NOT ALL THE SUPER KEYS ARE PRIMARY KEY
  - All the candidate keys except primary key are called alternate keys

Keys

- Example: Employee (EmpID, EmpName, DOB, Gender, Dept_No)
- Some Super keys:
  - EmpID
  - EmpID + EmpName (Redundant attribute (EmpName) permitted)
  - EmpID + DOB, EmpID + DOB + Gender, EmpID + Dept_No, EmpName + DOB, EmpName + DOB + Gender,……
- Some Candidate keys:
  - EmpID
  - EmpName + DOB
  - EmpName +Dept_No
  - DOB + Gender
    - Are minimal super keys
    - If you divide them further, they lose the property of a key.Dept_No alone cannot uniquely identify all the tuples

Keys

- **Foreign key** constraint
  - Value in one relation must appear in another
  - The relation r1 includes among its attributes the primary key of an another relation r2 (attribute x)
  - This attribute x is called as foreign key from r1, referencing r2
  - **Referencing** relation
    - The relation r1 is called referencing relation of the foreign key dependency
  - **Referenced** relation
    - The relation r2 is called referenced relation of the foreign key
  - Example: Dept_name is foreign key from instructor
    - In instructor is a foreign key from instructor
    - In department is the primary key of department
  - Ensures consistency, parent child relationship (To have a parent record for every child)
  - Need not always refer to primary keys, it can be attributes with unique value

Relax

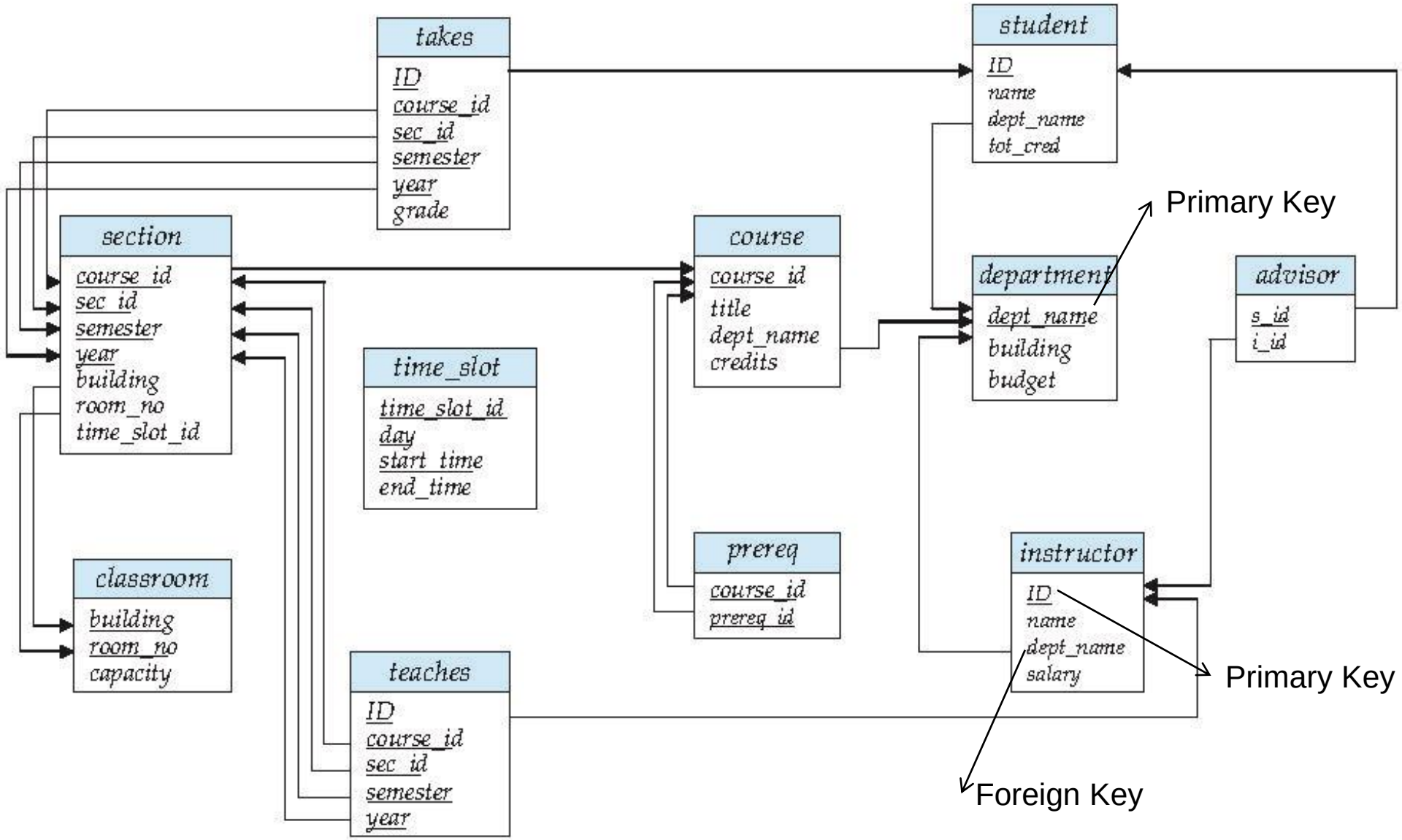
1. A _____ key is a minimal super key.
2. A key which is a set of one or more columns that can identify a record uniquely is called _____ key.
3. Student (SID, FNAME, LNAME, COURSEID)
    1. Super Keys ?
    2. Candidate Keys ?
    3. Primary Key ?
    4. Foreign Key ?
4. Staff (StfID, StfNAME, StfPhone, StfState, StfAge)
    1. Super Keys ?
    2. Candidate Keys ?
    3. Primary Key ?
    4. Foreign Key ?

Relax

1. A _____ key is a minimal super key.
2. A key which is a set of one or more columns that can identify a record uniquely is called _____ key.
3. Student (SID, FNAME, LNAME, COURSEID)
    1. Super Keys ?, 2. Primary Key ? 3. Foreign Key ?
4. Staff (StfID, StfNAME, StfPhone, StfState, StfAge)
    1. Super Keys ?, 2. Primary Key ? 3. Foreign Key ?
5. Differentiate among Candidate Key and Primary Key

- 1: Candidate key
- 2: Super key
- 3: 1: a) SID, b) FNAME + LNAME
- 3: 3: SID
- 3: 4: COURSEID
- 4. 1: a) StfID, b) StfPhone
- 4. 3: StfID (StfPhone may change)
- 4. 4: No foreign key

Schema Diagram for University Database



Relational Query Languages

- These query languages are concise and formal
- Lack of the “syntactic sugar” of commercial languages
- But they illustrate the fundamental techniques for extracting data from the database
- Procedural vs. non-procedural
  - Relational algebra
    - **Procedural language** consists of a set of operations that take one or two relations as input and produce a new relation as their result
  - Tuple relational calculus
    - **Non Procedural language** uses predicate logic to define the result desired without giving any specific algebraic procedure for obtaining that result
  - Domain relational calculus
    - **Non Procedural language** uses *domain variables that take on values from an attributes domain, rather than values* for an entire tuple

Relational Operations

- All procedural query languages provide a set of operations that can be applied to either a single relation or a pair of relations
- These operations result in a single relation
- e.g. Selection of tuples, selection of attributes, join, cartesian product, union, intersection, difference
- Detail in Ch 6. Relational Algebra

Selection of tuples

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- Select tuples with A=B and D > 5
  - $\sigma_{A=B \text{ and } D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Selection of Columns (Attributes)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select A and C
  - Projection
  - $\Pi_{A, C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Joining two relations – Cartesian Product

- Relations *r, s*:

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Union of two relations

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

Set difference of two relations

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r* – *s*:

A	B
α	1
β	1

Set Intersection of two relations

- Relation *r, s*:

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

- $r \cap s$

A	*B*
α	2

Joining two relations – Natural Join

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, the "natural join" of relations *R* and *S* is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*

Natural Join Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

- Natural Join
  - r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Figure 2.1

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{salary>=85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID, salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$ (Natural Join)	$instructor \bowtie department$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian Product)	$instructor \times department$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.

Tutorial Questions

1. Give example of natural join.
2. Write the maximum number of columns allowed per table in oracle.
3. Write the maximum number of rows allowed per table in oracle.
4. Write the maximum number of tables allowed per database in oracle.

End of Chapter 2

Chapter 3 SQL

Database System Concepts, 6th Ed.

Item	Type of Limit	Limit Value
Columns	Per table	1000 columns maximum
Constraints	Maximum per column	Unlimited
Rows	Maximum number per table	Unlimited
Tables	Maximum per database	Unlimited