Database System Concepts

Chapter 2: Intro to Relational Model
18-Aug-22

Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages

Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages

Relational Model

- Use a collection of tables to represent both data and relationships among those data
- Terminology (basic notions of the relational model)
  - relation/table
  - tuple/row
  - attributes/column

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)

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages

Relation Schema and Instance

- Database Schema
  - Logical design of the database
  - Like type definition in programming-language
- Database Instance
  - Snapshot of the data
  - Like variable in programming-language

Relation Schema and Instance

- *Attributes*: $A_1, A_2, \ldots, A_n$
- *Relation schema*: $R = (A_1, A_2, \ldots, A_n)$

  Example:

  instructor = (*ID, name, dept_name, salary*)

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is
  - a subset of $D_1 \times D_2 \times \ldots \times D_n$
  - a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
  - the current values (**relation instance**) of a relation are specified by a table
  - an element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Relations are 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

Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages

Keys

- How to distinguish the tuples in a given relation?
  - the value of the attribute values of a tuple should be able to *uniquely identify* the tuple
- Terminology
  - Superkey
  - Candidate Key
  - Primary Key
  - Foreign Key

Keys

选修 (<u>学号, 课程号</u>, 成绩)

- Let K $\subseteq$ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is minimal
  - Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one?

Primary Key Constraint

- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*

Referential Integrity Constraint

Foreign Key

- 假设存在关系r和s：r(A, B, C), s(B, D)，则在关系r上的属性B称作参照s的**外码**，r也称为外码依赖的参照关系，s叫做外码被参照关系
  - 例 **学生**(学号,姓名,性别,专业号,年龄) - *参照关系*

    专业(专业号,专业名称) - *被参照关系 (目标关系)*

    其中属性专业号 称为关系学生的外码

选修 (学号, 课程号, 成绩)
课程 (课程号,课程名, 学分, 先修课号)

- *Instructor* (ID, name, dept_name, salary) - 参照关系

  Department (dept_name, building, budget) - 被参照关系

参照关系中外码的值必须在被参照关系中实际存在或为null

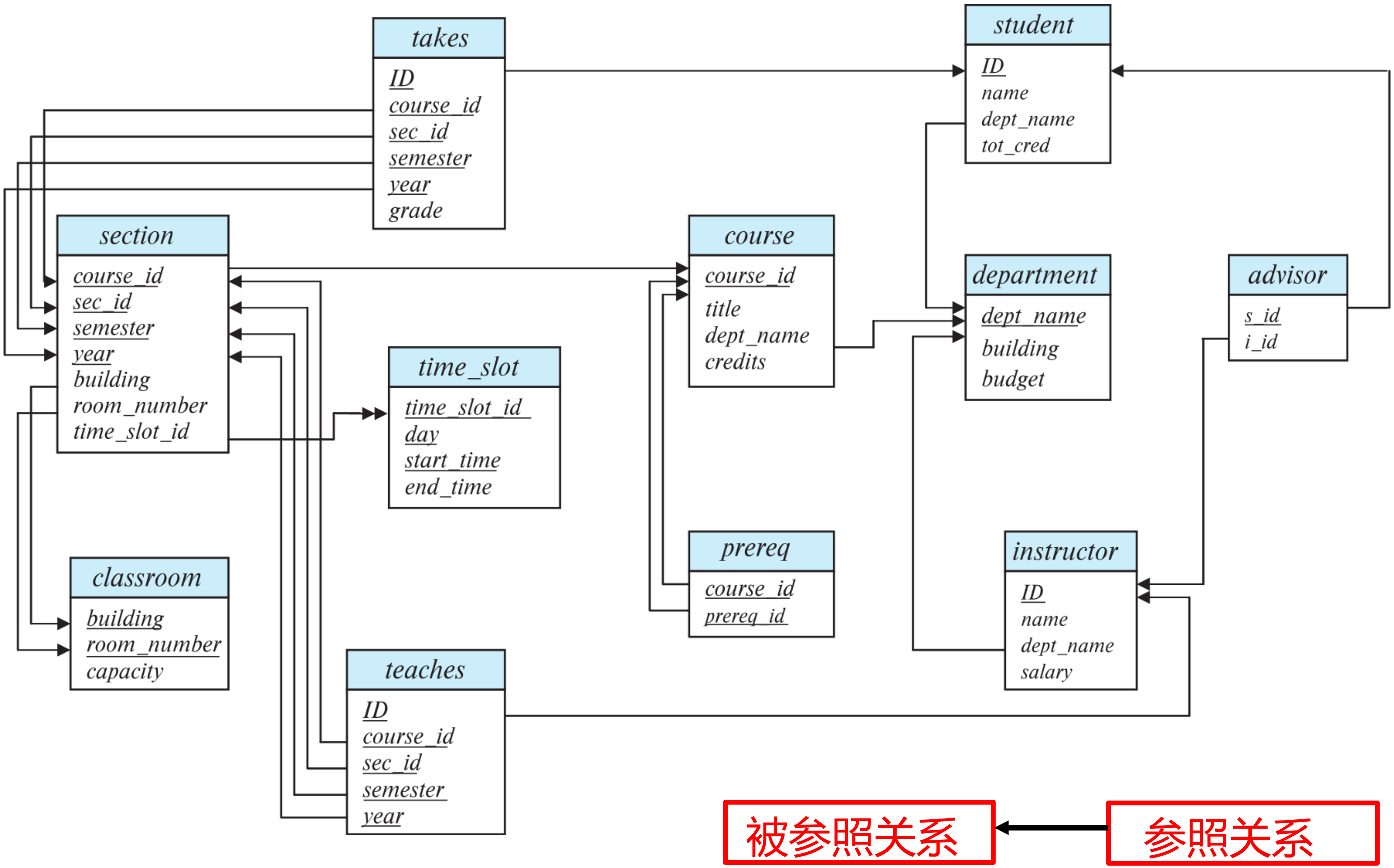
Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages

Schema Diagram for University Database

- *classroom* (building, room_number, capacity)
- *department* (dept_name, building, budget)
- *course* (course_id, title, dept_name, credits)
- *instructor* (ID, name, dept_name, salary)
- *section* (course_id, sec_id, semester, year, building, room_number, time_slot_id)
- *teaches* (ID, course_id, sec_id, semester, year)
- *student* (ID, name, dept_name, tot_cred)
- *takes* (ID, course_id, sec_id, semester, year, grade)
- *advisor* (s_ID, i_ID)
- *time_slot* (time_slot_id, day, start_time, end_time)
- *prereq* (course_id, prereq_id)

Schema Diagram for University Database

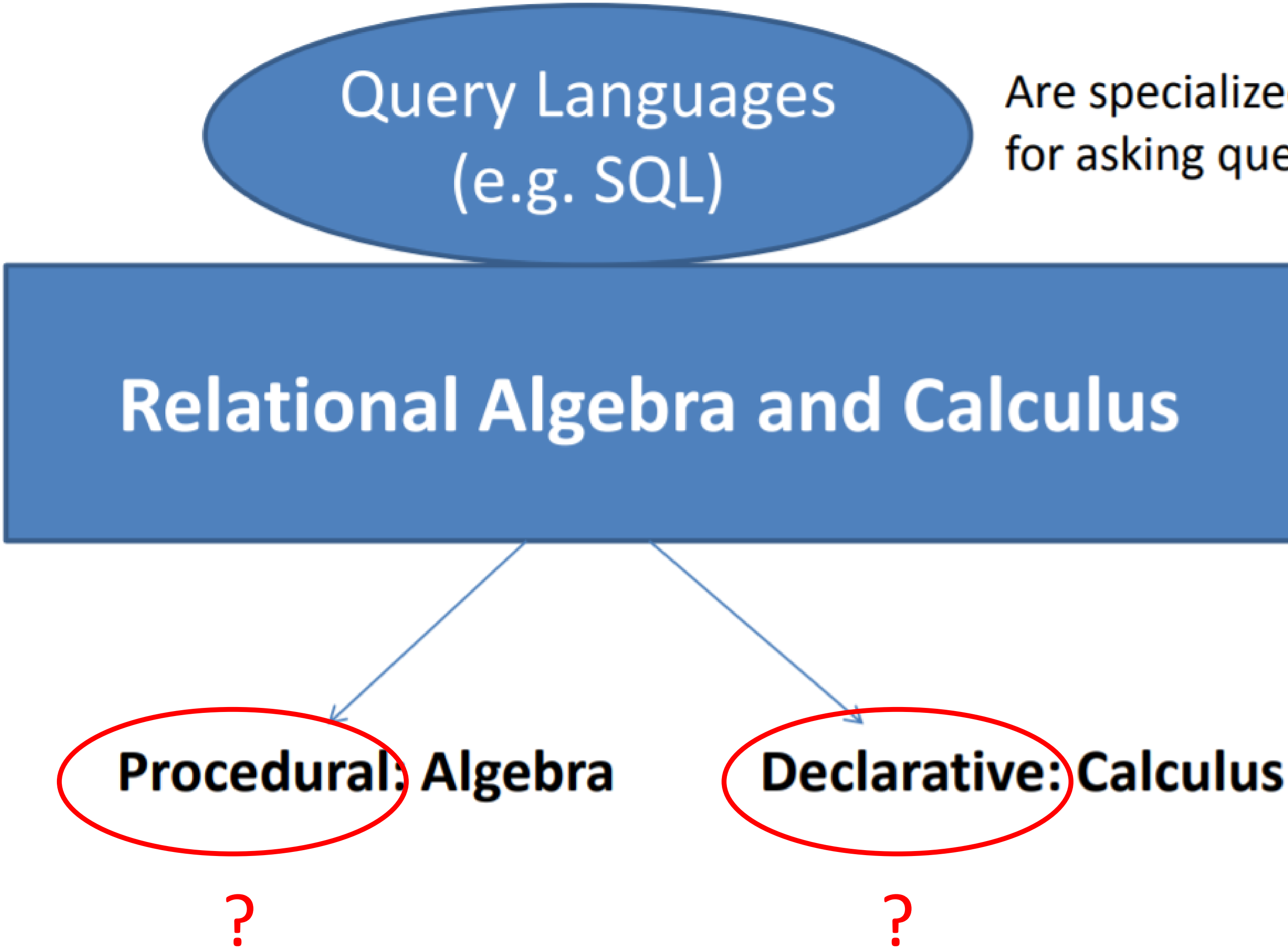


Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages

Relational Query Language

- Query Languages
  - allow manipulation (操纵) and retrieval (检索) of data from a database
- Relational Query Languages
  - query languages for Relational Database
  - “real”/ “practical” query languages
    - e.g. SQL
  - “pure”/ “mathematical” query languages
    - Relational Algebra
    - Relational Calculus



Are specialized languages for asking questions.

Formal Relational Query Languages

Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:

- ☆ *Relational Algebra*: More operational (过程化), very useful for representing execution plans.
- 🕐 *Relational Calculus*: Lets users describe what they want, rather than how to compute it. (Non-operational, declarative (陈述).)

Understanding Relational Algebra & Calculus is key to understanding SQL, query processing!

Declarative vs Procedural

- Procedural programming requires that the programmer tell the computer what to do.
  - how to get the output for the range of required inputs
  - the programmer must know an appropriate algorithm.
- Declarative programming requires a more descriptive style.
  - the programmer must know what relationships hold between various entities.

Why do we need Query Languages anyway?

- Two key advantages
  - Less work for user asking query
  - More opportunities for optimization
- Relational Algebra
  - Theoretical foundation for SQL
  - Higher level than programming language
    - but still must specify steps to get desired result
- Relational Calculus
  - Formal foundation for Query-by-Example
  - A first-order logic description of desired result
  - Only specify desired result, not how to get it

Relational Query Languages

- Procedural vs .non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate in this chapter on relational algebra
  - Not turning-machine equivalent
  - consists of 6 basic operations

Thank you!

Q&A