数据库系统概念

Chapter 2

Structure of Relational Databases

> In the relational model the term **relation** is used to refer to a table, while the term **tuple** is used to refer to a row. Similarly, the term **attribute** refers to a column of a table. The term **relation instance** is used to refer to a specific instance of a relation, i.e., containing a specific set of rows.

- **Domain**: For each attribute of a relation, there is a set of permitted values, called the domain of that attribute.
- **Atomic**: A domain is atomic if elements of the domain are considered to be indivisible units. (*The important issue is not what the domain itself is, but rather how we use domain elements in our database.*)

Database Schema

> The concept of a relation corresponds to the programming-language notion of a *variable*, while the concept of a **relation schema** corresponds to the programming-language notion of *type definition*.

- In general, a relation schema consists of **a list of attributes and their corresponding domains**.

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Figure 2.5 The *department* relation.

- For instance, the schema for the department relation of Figure 2.5 is:

$$department(dept\_name,\ building,\ budget)$$

Keys

- **Superkey**:
  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).
- **Candidate key**:
  K is a ***candidate key*** if K is one of the minimal superkeys (i.e., *no proper subset of K is a superkey*).
- **Primary key**:
  The term ***primary key*** is used to denote a candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation.

  > A relation, say r_1, may include among its attributes the primary key of another relation, say r_2. This attribute is called a ***foreign key*** from r_1, referencing r_2. The relation r_1 is also called the **referencing relation** of the foreign key dependency, and r_2 is called the **referenced relation** of the foreign key.

Relational Query Languages

- A **query language** is a language in which a user requests information from the database.

Relational Operations

- The Relational Operations have the nice and desired property that their result is always a *single* relation.

RELATIONAL ALGEBRA

The relational algebra defines a set of operations on relations, paralleling the usual algebraic operations such as addition, subtraction or multiplication, which operate on numbers. Just as algebraic operations on numbers take one or more numbers as input and return a number as output, the relational algebra operations typically take one or two relations as input and return a relation as output.

Relational algebra is covered in detail in Chapter 6, but we outline a few of the operations below.

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

Chapter 5

Triggers

> A **trigger** is a statement that the system executes automatically as a side effect of a modification to the database. It is a special type of **stored procedure**.

- To design a trigger mechanism, we must meet **two requirements**:
  1. Specify when a trigger is to be executed. This is broken up into an **event** that causes the trigger to be checked and a **condition** that must be satisfied for trigger execution to proceed.
  2. Specify the **actions** to be taken when the trigger executes.
- **Usages**:
  - Compare different versions of data.
  - Read data from tables in other databases.
  - Modification or deletion of all associated tables in the database
  - Rollback invalid changes.
  - Enforce more complex restrictions than those provided by the CHECK constraint.
  - Execute local and remote stored procedures.
- Template:
  - Create or modify a trigger:

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure;
```

> **WHEN**
>
> - `BEFORE` –invoke before the event occurs
> - `AFTER` –invoke after the event occurs
>
> **EVENT**
>
> - `INSERT` –invoke for INSERT
> - `UPDATE` –invoke for UPDATE
> - `DELETE` –invoke for DELETE
>
> **TRIGGER_TYPE**
>
> - `FOR EACH ROW`
> - `FOR EACH STATEMENT`

  - Delete a trigger:

```
DROP TRIGGER trigger_name;
```