Data Management

Leonardo Maria Carrozzo

2022/2023

Contents

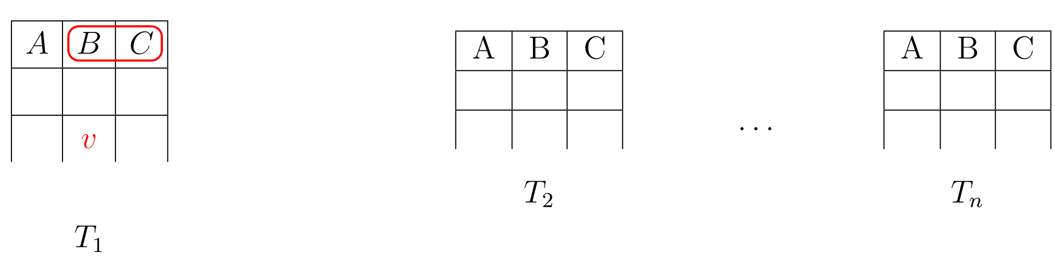
1 Introduction

The course is based on the following topics:

- **The structure of a Data Base Management System (DBMS)**: Realtional data and queries, Buffer manager;
- **Transaction management**: The concept of transaction, Concurrency management;
- **Crash management**: Classification of failures, Recovery;
- **Data Warehousing**: Data warehousing architectures and operators, Data warehousing design;
- **NoSQL databases**: Document-based databases (such as MongoDB), Graph databases OLAP vs OLTP (such as Neo4j);
- **Physical structures for data bases**: File organizations for data base management, Principles of physical database design;
- **Query processing**: Evaluation of realational algebra operatos, Fundamentals of query optimization;

1.1 The relational data model

A **database** in the Realtional Model is a **set of tables** (or **relations**). Each **table** is a **set of rows** (or **tuples**). Each one with the **same set of columns** (or **attributes**).



v is the value of the corresponding column and row. The attributes B and C form a **superkey**.

We have then:

- **Integrity constraint**: a rule at the level of the schema that all the rows must respect;
- **Superkey**: there cannot exist two or more rows that have the same value as the combination of multiple attributes;

- **Key**: attribute in a table;
- **Foreign key**: attributes in a table are a reference of another table;
- **Primary key**: special key that doesn't allow null values (a null value is a a special value that says that the value is missing).

A	B	C
		c_1
		c_2
		c_3
		c_3

T_1 Unordered set

D	E	F
c_0		
c_1		
c_2		
c_3		

T_2 Must not miss any key

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_2	c_3
$null$	b_2	c_3

T_1 Unordered set

D	E	F
c_0		
c_1		
c_2		
c_3		

T_2 Must not miss any key

We have a "no predicate" on the null value: never equal and never different, comparation is always false.

As we have said, the Relational Data Model uses the mathematical concept of a relation as the formalism for describing and representing data. A relation is a subset of a cartesian product of sets. A relation can be considered as a "table" with rows and columns.

Codd introduced two different query languages for the relational data model:

- **Relational Algebra**, which is a procedural language. It is an algebraic formalism in which queries are expressed by applying a sequence of operations to relations.
- **Relational Calculus**, which is a declarative language. It is a logical formalism in which queries are expressed as formulas of fist-order logic.

Codd's Theorem: Relational Algebra and Relational Calculus are essentially equivalent in terms of expressive power.

DBMSs are based on **SQL**, a hybrid of a procedural and a declarative language that combines features from both relational algebra and relational calculus.

1.2 Relational algebra

The operators of Relational Algebra can be divided into two groups:

- Three standard set-theoretic binary operations:
  - Union
  - Difference
  - Cartesia Product
- Two special unary operations on relations:
  - Projection
  - Selection

The Relational Algebra consists of all expressions obtained by combining these five basics operations in syntactically correct ways.

- In relational algebra, both arguments of the union and the difference must be relations of the same arity.
- In SQL, there is the additional requirement that the corresponding attributes must have the same data type.
- However, the corresponding attributes need not have the same names; the corresponding attribute in the result can be renamed arbitrarily.

1.2.1 Union

- Takes in input two k-ary relations R and S, for some k.
- Gives in output the k-ary relation R ∪ S, where:

R ∪ S={$(a_1, ..., a_k)$: $(a_1, ..., a_k)$ is in R or $(a_1, ..., a_k)$ is in S}

1.2.2 Difference

- Takes in input two k-ary relations R and S, for some k.
- Gives in output the k-ary relation R − S, where:

R − S = {$(a_1, ..., a_k)$: $(a_1, ..., a_k)$ is in R and $(a_1, ..., a_k)$ is not in S}

1.2.3 Cartesian Product

- Take in input an m-ary relation R and an n-ary relation S.
- Gives in output the (m+n)-ary relation R $\times$, where:

R $\times$ S = $\{(a_1, ..., a_m, b_1, ..., b_n)$: $(a_1, ..., a_m)$ is in R and $(b_1, ..., b_n)$ is in S$\}$

Note: $|R \times S| = |R| \times |S|$

Let's see an example:

Emp	Dept
Rossi	A
Neri	B
Bianchi	B

Employee

Dept	Char
A	Mori
B	Bruni

Dept

Emp	Dept	Code	Chair
Rossi	A	A	Mori
Rossi	A	B	Bruni
Neri	B	A	Mori
Neri	B	B	Bruni
Bianchi	B	A	Mori
Bianchi	B	B	Bruni

Employee $\times$ Dept

1.2.4 Projection Operation

Given a table R, we want to rearrange the order of the columns and/or suppress/rename some columns.

Projection is a family of unary operations of the form:

$$\pi_{\text{<attribute list>}}(<\text{ relation name }>) \text{ or } \text{PROJ}_{\text{<attribute list>}}(<\text{ relation name }>)$$

When the projection is applied to a relation R, it removes all columns whose attributes do <u>not</u> appear in the $<$ attribute list $>$ (we assume that an attribute can appear only once in the list).

The remaining columns may be re-arranged (and also renamed by means of the notation a $\leftarrow$ b) according to the order and name in the $<$ attribute list $>$.

Any duplicate rows are eliminated.

1.2.5 Selection Operation

Selection is a family of unary operations of the form:

$$\sigma_\theta(\text{R}) \text{ or } \text{SEL}_\theta(\text{R})$$

where R is a relation and θ is a condition that can be applied as a test to each row of R.

When a selection operation is applied to R, it returns the subset of R consisting of all rows that satisfy the condition θ.

Here are some examples: $\sigma_{\text{A=10}}\text{(T)=}$ or $\sigma_{\text{(A=10 or B>20) and C is not null}}\text{(T)=}$

Where A = 10 is a boolean expression. T might be an expression or a table.

We have two special predicates:

- is null
- is not null

A condition in the selection operation is an expression built up from:

- Comparison operators $=, <, >, \neq, \leq, \geq$ applied to operands that are constants or attribute names or component numbers. (These are the basic (atomic) clauses) of the conditions).
- The boolean login operators $\wedge, \vee,$: applied to basic clauses.

1.2.6 Relational Algebra Expression

A relational algebra expression is an expression obtained from relation schemas using union, difference, cartesian produtct, projection, and selection.