RDBMS and SQL

Roudranil Das

RDBM Sep-Nov 2022
Roll No: MDS202227
roudranil@cmi.ac.in

September 15, 2022

Contents

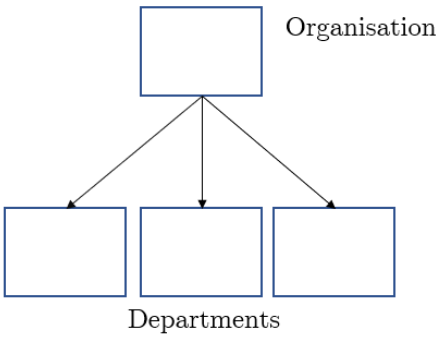
1 September 15, 2022

1.1 Introduction

- Need to organise and manipulate (large) volumes of data in a systematic way.
  - We could have customised formats for each requirement
  - Update and query
  - manipulation: requires knowledge of implementation. This brings up the argument of *imperative vs declarative* programming

What is the "best" way to declaratively work with data? The following methods were proposed:

IBM - Hierarchical model, like a tree (1960)



Godd - He came up with the idea that information should be stored in flat tables.
Rows are items
Columns are attributes

See the following

Parts:

ID-parts	...
...	...

Projects:

ID-projects	...
...	...

Parts in projects:

ID-parts	ID-projects	No.
...	...	...

1.2 Relational Database

Mathematically, what we saw above is a relational database: there are sets S_1, S_2, and $S_1 \times S_2 \to$ Cartesian product. Say $S_1 =$ All possible names, $S_2 =$ All possible dates of birth.

We define a relation $R \subseteq S_1 \times S_2$. We can also do $D \subseteq S_1 \times S_2 \times \cdots \times S_n$.

A table is a relation.

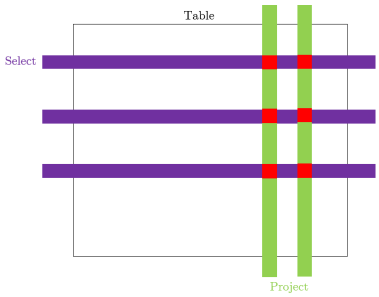
Type - Allowed values of Column j in S_j

Limitations:

- Relations are sets of tuples → there cannot be any duplicates.
- All columns must exist.
- Atomicity: every value must belong to the set, so much depends on what the set is. For example, in a library some books may have many authors. So is our set a set of all authors, or set of all subsets of all authors? That will decide the atomicity of the entries.

We need a mechanism to extract useful information from tables (relations). Operators to manipulate data in tables: - "Relational Algebra" (remember that outputs of σ and π are implicitly other tables)

1. Extract rows (items) based on some criterion: `Select` σ
   Notation: $\sigma_{\text{condition}}(\text{relation})$
   Example: $\sigma_{\text{supplier = "XYZ"}}$ (parts), $\sigma_{C_1 \bigwedge C_2}(\theta)$
2. Retain only relevant columns (attributes): `Project` π
   Notation: $\pi_{\text{column names}}(\text{relation})$
   Example: $\pi_{\text{company, salary}}(\text{Placement})$



3. `Cartesian product` $\times$
4. `Renaming values` ρ

Supppose we have courses data at CMI, say we want to have separate tables for each course. Say, we want to ask which students are registered in two courses. If we want to do this imperatively, then we can jsut run two loops across all courses and get our answer. What if we want to do it declaratively?

What we do is concatenating the two course tables, kind of like a cartesian product between their rows. For example if the first course had 2 students and the second one had 3, then our resulting table would have 6 rows. One small glitch would be that multiple columns might have the same name, which can be easily addressed by renaming them suitably. This should be done before we take the product. In this new table, we check for the rows with the two roll number columns having the same entry.