Chapter 3

Introducation to Relational Database

Relational Model

- Structure aspect
  - Data in the database is perceived by the user as tables and nothing but tables
- Integrity aspect
  - Those tables satisfy certain integrity constraints
- Manipulative aspect
  - Operators available to the user for manipulating those tables are operators that derive tables from tables
    - Restrict: extract specified rows from a table
    - Project: extract specified columns from a table
    - Join: join two tables together on the basis of common values in a common column

A Database for a Company

DEPT

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M
D3	Research	5M

EMP

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

Department and Employee tables

Restrict
DEPTs where BUDGET > 8M

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M

Project
DEPTs over DEPT#, BUDGET

DEPT#	BUDGET
D1	10M
D2	12M
D3	5M

Join
DEPTs and EMP over DEPT#

DEPT#	DNAME	BUDGET	EMP#	ENAME	SALARY
D1	Marketing	10M	E1	Lopez	40K
D1	Marketing	10M	E2	Cheng	42K
D2	Development	12M	E3	Finzi	30K
D2	Development	12M	E4	Saito	35K

Relational Properties

- *Closure*
  - The output from any relational operation is the same kind of object as the input -- they are all relations
  - The output from one operation can become input to another
  - Closure implies that we can write *nested (relation-valued) expressions*.
- *Set-at-a-time*
  - An operation that operates on entire sets as operands and returns an entire set as a result.
  - Relational operations are all set-level, since they operate on and return entire relations, and relations contain *sets* of rows, as opposed to a single row.

Relational Properties

- *Information principle*
  - The entire information content of the database is represented in one and only one way, namely as explicit values in column positions in rows in tables
  - There are no pointers
- *Primary key*
  - A column or combination of columns in a table whose values can be used to identify rows within that table uniquely
- *Foreign key*
  - A column or combination of columns in one table whose values are required to match those of the *primary* key in some other tables

Relations and ”RELVARS”

- *Relation*
  - Just is a mathematical term for a table
  - First introduced by Codd in 1970
- Relation variable
  - Known as “RELVAR” whose values are relation values
  - Examples: EMP, DEPT …
- Relation value
  - Different relation values at different times
  - Analogy:
    - int a; // a is an integer variable
    - CREATE Table EMP; // EMP relation variable
    - a = 3;
    - EMP := EMP MINUS (EMP WHERE EMP# = ‘E4’)

What Relations Mean

- Data types
  - Domains
  - TYPE EMP#: the set of all possible employee numbers
- Heading & body
  - heading is a set of column-names : type-name pairs
  - body is a set of rows that conform to that heading
  - Ignore the type-name

EMP#:EMP#	ENAME:ENAME	DEPT#:DEPT#	SALARY:MONEY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

What Relations Mean

- Proposition
  - Something that evaluates to either *true* or *false,* unequivocally.
- Predicate
  - A truth-valued function
  - Every relation has a corresponding predicate that is (loosely) "what the relation means."
  - Each row in a given relation denotes a certain true proposition, obtained from the predicate by substituting certain argument values of the appropriate type for the parameters of the predicate ("instantiating the predicate")

What Relations Mean

- Predicate
  - Examples:
  - Employee EMP # is named ENAME, works in department DEPT#, and earns salary SALARY (predicate)
  - Employee E1 is named Lopez, works in department D1, and earns salary 40K (proposition)
- Types are things we can talk about
- Relations are things we say about the things we can talk about

What Relations Mean

- Types and relations are
  - both necessary (without types we have nothing to talk about; without relations we cannot say any thing)
  - sufficient as well as necessary, that is, we do not need anything else
  - not the same thing

Catalog

- Catalog
  - a set of system relvars whose purpose is to contain *descriptors* regarding the various objects that are of interest to the system itself, such as base relvars, views, indexes, users, integrity constraints, security constraints, and so on.
- Self-described
  - Describe catalog itself
  - Example: (ORACLE)

```
SELECT TABLE_NAME, COMMENTS
FROM DICT

SELECT TABLE_NAME
FROM USER_TABLES WHERE TABLE_NAME LIKE 'L%'
```

Base Relvars and Views

- Base relvars
  - Original relvar
  - CREATE TABLE EMP …;
- Derived relvars
  - Obtained from those base relations by some relational expressions such as restrict, project, and join.
- View
  - A relvar whose value at any given time is a derived relation
  - Example:

    CREATE VIEW TOPEMP AS

    (EMP WHERE SALARY >33K) {EMP#, ENAME, SALARY}

View

- View definition expression is not evaluated and saved in the catalog with the name TOPEMP
- Regarded as if there really were a relvar in the database called TOPEMP
- Changes to TOPEMP will automatically and instantaneously be applied to relvar EMP
- Base relvar "really exist" in the sense that they represent data that is actually stored in database
- Views do not "really exist" but merely provide different ways of looking at "real data"
- View is a "stored query"

Transaction

- A logic unit of work involving several database operations
- Atomic
  - Transactions are guaranteed to either to execute in their entirety or not to execute at all.
- Durable
  - Once a transaction successfully executes COMMIT, its updates are guaranteed to be applied to the database
- Isolated
  - Database updates made by T1 are not visible to any distinct transaction T2 until and unless T1 successfully executes COMMIT
- Serializable
  - Interleaved execution == one at a time execution