CHAPTER – 4

Advantages of The Relational Model

* replaced hierarchical/network database systems because it offers more flexibility to users in terms of ~~file systems~~, physical data independence

hierarchical/network databases → used file systems
↓
if The database changed The application needed to be re-written

Codd → mathematical abstraction → powerful DB structure
↓

* Theorem and proof → produce results
* basic structure and easy to understand, intuitive
* allows for The separation of The conceptual from physical
* mapping from ER/EER is straightforward
* DATA OPERATIONS ⇒ EASY TO XPRESS
* No required knowledge of storage structure

Properties of Tables

* Relations become Tables with tuples (rows) and columns (attributes)
* ~~Domain~~ Domain = set of allowable values (atomic) → character, number, alphanumeric
* Tables are related to each other
* Tables hold information about objects to be represented in the database
* Each cell contains at most one value
* Each column has a distinct name which is the name of the attribute it represents
* Each tuple/row is distinct ⟶ no duplicate tuples
* Order of tuples is immaterial

Properties of Relations

degree = # number of attributes

each row is therefore a n-tuple containing n-values

property of the extension does not change unless the database design changes

cardinality → number of rows/tuples

↓

changes as more tuples are added

property of the extension

RL → defines keys and integrity constraints.

* super key → uniquely identifies rows/tuples

↓

candidate key = minimum set of attributes which uniquely identifies a tuples/rows

↓

primary key = candidate key chosen to identify tuples/rows

↓

foreign key = attribute or combination of attributes that is the primary key of some relation

Integrity Constraints

Integrity = correctness and internal consistency of the data in the database

Integrity constraints = rules/restrictions applied everywhere in the database.

⇓

data entered/inserted creates a legal instance of the database

- Domain constraints: limited set of values for attributes
- Entity integrity: primary keys can't have null values
- Referential integrity = foreign key value must matches primary key of some tuple.
- General constraints = enforced by database when changes are made to the data

Relational Model → underline primary key
→ indicate foreign keys in italics
→ underline and italicize primary keys of relation tables

Manipulation Languages

* procedural or prescriptive → user tells system how to manipulate data
  → Relational algebra

* non-procedural/declarative
  user tells system what data is needed, not how to get it: Relational calc, SQL

* Graphical = Illustration of what data is needed
  Query By Example (QBE)

* Fourth-gen (4GL) = user friendly environment to generate custom applications

* Natural Language (SGL) = restricted vers of English

Relational Algebra = Theoretical language
↓ SELECT ↓ PROJECT → JOIN

SELECT - operation

Select command returns rows that meet a specified predicate, copies them to a new table

SELECT tableName WHERE condition [Giving ~~newTable~~ newTableName]

σ predicate = Theta condition

Result Table is horizontal subset of operand

Predicates can have operators such as:

$<, <=, >, >=, =, <>, \wedge$ (AND), $\vee$ (OR), $\neg$ (NOT)

PROJECT - operation

Project returns unique values in a column or combination of columns

Ex: Project tableName OVER (colName, ---, colName)

Product, A x B

* often binary operation
* Cartesian product = cross-product of A and B
  - concatenate all rows of A w/ all rows of B
* First, all columns of A, Then all columns of B
* degree = deg of A + deg of B
* cardinality = (card of A) * (card of B)
* can be formed by nested loops / algorithms

Theta join = merge two tables based on the condition represented by theta

Equi join = can be formed when two tables have common columns

We Then compare each tuple and only retrieve where the values of those columns are equal

Sudent and Enroll have matching stuid

Natural join = can be formed when two tables have common columns, but we drop repeated columns from result

Left Semi Join $A|\times B$

↳ natural join of A and B and then project the result onto the attributes of A

↓

result is just those tuples of A that participate in the natural join

Ex: Student LEFT SEMIJOIN Enroll

↳ only shows the rows of student where stuid matches with that of Enroll

Right Semi Join $A\times|B$

//

↳ project results into B-Table based on matching tuples between the two

Left Outer Join of A and B (NO MATCH)

↓ Add in A however many rows in B and ~~natural joins of A and B but with no matches in B fill in the B attributes w/ null values. The~~

Fill B's unmatched tuples with null values

Right Outer Join of A and B (NO MATCH)

↳ Fill A's unmatched tuples w/ null values

//

Full Outer Join of A and B

Add unmatched tuples for both A and B, filling in null values for unmatched tuples on both sides

SET OPERATIONS

A ∪ B = A union B

↳ shows the results of both tables, removing any duplicates

A ∩ B = ~~only~~ A intersection B

↳ shows tables where the attribute(s) exist(s) in both

A MINUS B = set of tuples in A but not in B

↓

only show the Tuples that are not in the second table.

---

strong entity ——> table

non-composite / single-valued att's ——> cols

multi-valued attributes ——> new Table w/ primary key of original table

* Keep key in the original table

Weak Entity ——> table ——> dependent on owner-entity

| physicians ——> availability

↓

has no candidate Key(s) <—— primary key of owner

For 1:M, place 1's primary key unto M-table

For 1:1, either key unto other table as foreign key

For M:M, create relationship table w/ primary keys of related entities