The Relational Model

Data Model

- Integrated collection of concepts for describing data, relationships between data, and constraints on the data.
- Has three components:
  - a structural part;
  - a manipulative part;
  - a set of integrity rules.

RDBMS

- The dominant data-processing software in use today
- Based on the relational data model proposed by E. F. Codd (1970)
- All data is logically structured within relations (tables)
- A great strength of the relational model is its simple logical structure
- A sound theoretical foundation that is lacking in the first generation of DBMSs (the network and hierarchical DBMSs)

Brief History of the Relational Model

- First proposed by E. F. Codd in his seminal paper ‘A relational model of data for large shared data banks’ (Codd, 1970).
- This paper is now generally accepted as a landmark in database systems

Brief History of the Relational Model

- **1st Project**
- IBM's San José Research Laboratory in California, prepared a prototype relational DBMS System R, which was developed during the late 1970s
- This project was designed to prove the practicality of the relational model

Brief History of the Relational Model

- System R project led to two major developments:
  - The development of a structured query language called SQL which has since become the formal ISO and *de facto* standard language for relational DBMSs
  - The production of various commercial relational DBMS products during the late 1970s and the 1980s: for example, DB2 from IBM and Oracle from Oracle Corporation.

Brief History of the Relational Model

- **<u>2nd Project</u>**
- INGRES (Interactive Graphics Retrieval System) project at the University of California at Berkeley
- A prototype of RDBMS, with the research concentrating on the same overall objectives as the System R project.
- This research led to an academic version of INGRES

Brief History of the Relational Model

- **3rd Project**
- Peterlee Relational Test Vehicle at the IBM UK Scientific Centre in Peterlee (Todd, 1976)
- This project had a more theoretical orientation than the System R and INGRES projects and was principally for research into such issues as **query processing** and **optimization**

RM Terminology

- The relational model is based on the mathematical concept of a **relation**, which is physically represented as a **table**
- Codd, a trained mathematician, used terminology taken from mathematics, principally set theory and predicate logic

Relation

- A relation is a table with columns and rows
- An RDBMS requires only that the database be perceived by the user as tables
- This perception applies only to the logical structure of the database: that is, the external and conceptual levels of the ANSI-SPARC architecture
- It does not apply to the physical structure of the database

Attribute

- An attribute is a named column of a relation
- Attributes can appear in any order and the relation will still be the same relation, and therefore convey the same meaning

Domain

- The set of allowable values for an attribute
- Every attribute in a relation is defined on a **domain**
- Domains may be distinct for each attribute, or two or more attributes may be defined on the same domain

Example Attribute Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Tuple

- A tuple is a row of a relation
- Tuples can appear in any order and the relation will still be the same relation, and therefore convey the same meaning

Degree of a Relation

- The degree of a relation is the number of attributes it contains
- A relation with only one attribute would have degree one and be called a **unary** relation
- A relation with two attributes is called **binary**, with three attributes is called **ternary**, and after that the term ***n*-ary** is usually used.

Cardinality of a Relation

- The number of tuples a relation contains and this changes as tuples are added or deleted

Alternative Terminology

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Mathematical Relations

Let $D1 = \{2, 4\}$ and $D2 = \{1, 3, 5\}$ are two sets

- **Cartesian product** $D1 \times D2$, is the set of all ordered pairs such that the first element is a member of $D1$ and the second element is a member of $D2$.
- $D1 \times D2 = \{(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)\}$
- Any subset of this Cartesian product is a **relation**. For example, we could produce a relation R such that:

$$R = \{(2, 1), (4, 1)\}$$

Mathematical Relations

- We may specify which ordered pairs will be in the relation by giving some condition for their selection
- For example, if we observe that *R* includes all those ordered pairs in which the second element is 1, then we could write *R* as:

$$R = \{(x, y) \mid x \in D1, y \in D2, \text{ and } y = 1\}$$

e.g., $R = \{(2, 1), (4, 1)\}$

Mathematical Relations

- We may form another relation S in which the first element is always twice the second.

  $S = \{(x, y) \mid x \in D1, y \in D2, \text{ and } x = 2y\}$

  or, in this instance,

  $S = \{(2, 1)\}$

  since there is only one ordered pair in the Cartesian product that satisfies this condition

Mathematical Relations

- We can easily extend the notion of a relation to three sets.
- $D1 \times D2 \times D3$ of these three sets is the set of all ordered **triples** such that the first element is from $D1$, the second element is from $D2$, and the third element is from $D3$.
- Any subset of this Cartesian product is a **relation**

$$D1 = \{1, 3\} \; D2 = \{2, 4\} \; D3 = \{5, 6\}$$

$$D1 \times D2 \times D3 = \{(1, 2, 5), (1, 2, 6), (1, 4, 5), (1, 4, 6), (3, 2, 5), (3, 2, 6), (3, 4, 5), (3, 4, 6)\}$$

- Any subset of these ordered triples is a **relation**

Relation Schema

- A named relation defined by a ***set of attribute*** and ***domain-name*** pairs
- Let $A1, A2, \ldots, An$ be attributes with domains $D1, D2, \ldots, Dn$.
- The set $\{A1{:}D1, A2{:}D2, \ldots, An{:}Dn\}$ is a relation schema S
- Thus, a relation R is a set of n-tuples:

  $(A1{:}d1, A2{:}d2, \ldots, An{:}dn)$ such that $d1 \in D1$, $d2 \in D2, \ldots, dn \in Dn$
- Each element in the n-tuple consists of an attribute and a value for that attribute

Relation Schema

- Normally, when we write out a relation as a table, we list the attribute names as column headings and write out the tuples as rows having the form ($d1$, $d2$, . . . , dn), where each value is taken from the appropriate domain. In this way, we can think of a **relation in the relational model** as **any subset** of the **Cartesian product** of the **domains of the attributes**.

Examples of a Schema along with an Instance

<u>Schema</u>

Students(*sid:* string (5), *name:* string (30), *login:* string (20), *age:* integer, *gpa:* real)

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

An Instance ‘S1’ of the ‘Students’ Schema

Schemas

Reserves(*sid:* integer, *bid:* integer, *day:* date)

Sailors(*sid:* integer, *sname:* string(30), *rating*: integer, *age:* real)

An instance "R1" of "Reserves" schemas and instacnes "S1" and "S2" of "Sailors" schemas

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Relational Database Schema

- In the same way that a relation has a schema, so too does the relational database.
- A set of relation schemas, each with a distinct name.
- If *R*1, *R*2, . . . , *Rn* are a set of relation schemas, then we can write the *relational database schema*, or simply *relational schema*, *R*, as:

$$R = \{R1, R2, \ldots, Rn\}$$

Representing Relational Database Schemas

Branch (<u>branchNo: integer(5)</u>, street: string(10), city: string(15), postcode: string(10))

Staff (<u>staffNo: integer(5)</u>, fName: string(20), lName: string(20), position: string(20), sex: string(1), DOB: date, salary: money, branchNo: integer(5))

.....

....

Representing Relational Database Schemas

- The common convention for representing a relation schema is to give the name of the relation followed by the attribute names (along with data types) in parentheses.
- Normally, the primary key is underlined.
- The *conceptual model*, or *conceptual schema*, is the set of all such schemas for the database.

Relational Database: Definitions

- *Relational database:* a set of *relations*
- *Relation:* made up of 2 parts:
  - *Schema* : specifies name of relation, plus name and type of each column.
    - E.G. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)
  - *Instance* : a *table,* with rows and columns. #Rows = *cardinality*, #fields = *degree / arity.*

Properties of Relations

- The relation has a name that is distinct from all other relation names in the relational schema;
- Each cell of the relation contains exactly one atomic (single) value;
- Each attribute has a distinct name;
- Values of an attribute are all from the same domain;
- Each tuple is distinct; there are no duplicate tuples;
- The order of attributes has no significance;
- The order of tuples has no significance.

Relational Keys

- **Superkey**
- **An attribute, or set of attributes, that uniquely identifies a tuple within a relation**
- A superkey uniquely identifies each tuple within a relation. However, a superkey may contain additional attributes that are not necessary for unique identification, and we are interested in identifying superkeys that contain only the minimum number of attributes necessary for unique identification.

Relational Keys

Candidate Key

- The minimal subset of the super key is a candidate key
- A candidate key, *K*, for a relation *R* has two properties:
  - **uniqueness** – in each tuple of *R*, the values of *K* uniquely identify that tuple;
  - **irreducibility** – no proper subset of *K* has the uniqueness property.
- There may be several candidate keys for a relation
- When a key consists of more than one attribute, we call it a **composite key**

Relational Keys

- **Primary Key**
  – The candidate key that is selected to identify tuples uniquely within the relation
- **Alternate Keys**
  – The candidate keys that are not selected to be the primary key are called **alternate keys**. For the Branch relation, if we choose branchNo as the primary key, postcode would then be an alternate key. For the Viewing relation, there is only one candidate key, comprising clientNo and propertyNo, so these attributes would automatically form the primary key.

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Relational Keys

- **Foreign Key**
  - An attribute, or set of attributes, within one relation that matches the primary key of another relation
- When an attribute appears in more than one relations, its appearance usually represents a relationship between tuples of the two relations

NULL (unknown, unassigned, not applicable)

- Represents a value for an attribute that is currently unknown or is not applicable for this tuple
- Deals with incomplete or exceptional data
- Represents the absence of a value and is not the same as zero or spaces, which are values

Integrity Constraints (ICs) (Ramakrishnan)

- IC: condition that must be true for *any* instance of the database
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.

Types of Constraints

- Entity Integrity
- Referential Integrity
- Domain
  - (implemented through data types)
- General
  - (e.g., CHECK constraints)
- NOT NULL
- UNIQUE

Entity Integrity

- In a base relation, no value of a primary key column can be a null
- **Base Relation**
  - Any named relation at the conceptual schema level

Referential Integrity

- If a foreign key exists in a table, either FK value must match a primary key value of some record in its parent table or foreign key value must be wholly null.

- A valid foreign key value must always reference an existing primary key in the parent table or contain a *null*

Domain Constraints

- All of the values that appear in an attribute of a relation must be taken from the same domain

General Constraints

- Domain, entity integrity, and referential integrity constraints are considered to be a fundamental part of the relational data model and are given special attention. Sometimes, it is necessary to specify more general constraints.
- Additional rules that define or constrain some aspect of the enterprise
- For example, if an upper limit of 20 has been placed upon the number of staff that may work at a branch office, then the user must be able to specify this general constraint and expect the DBMS to enforce it. (similarly an upper limit on salary amount)

NOT NULL

- Does not allow a NULL value in that column e.g.,

  person_id NUMBER(2) NOT NULL,

  Or

  ALTER TABLE abc

  MODIFY (person_id NOT NULL);

UNIQUE

- Values of that column(s) must be unique OR it can have NULL values e.g.,

  person_id NUMBER(5) UNIQUE

  Or

  ALTER TABLE abc

  ADD CONSTRAINT uq_abc

  UNIQUE (person_id)

Views

- A **virtual relation** that does not necessarily exist by its own, but may be dynamically derived from one or more **base relations**
- **Base Relation**
  - Any named relation at the conceptual schema level

Purpose of Views

- Security
- Customization
  - Permits users to view the same data in different ways at the same time
- Removes Complexity