



**CHANDIGARH  
UNIVERSITY**  
Discover. Learn. Empower.

# **Relational Database Management System**

**Kundan Kumar Jha – 22MSM40052**

**Hritayan Devnath - 22MSM40081**

**22SDT-604: DATABASE MANAGEMENT SYSTEM**

**Branch - MSc Data Science**

**Department of Mathematics**

**Chandigarh University, Mohali Punjab**

**December 2022**

## **Abstract**

Applications that are more complicated are less maintainable and extensible. Maintainability and complexity can be balanced with the use of flexible and dynamic principles. In order to support dynamic objects, programming languages and software systems both use the role idea. Since there are no database systems that support dynamic data objects, complicated mappers must be used if data must be stored relationally. In this work, we present research on a relational database system that supports roles. We explore open research problems connected to our definition and provide a description of this notion based on current findings. We also cover the architecture as well as query language extensions because the role notion cannot be handled by current RDBMSs on an intrinsically basis.

## **Introduction**

Database management systems are really ubiquitous in the age of e-commerce. The development of relational database systems in theory and practice can be traced back to E. F. Codd's influential 1970 paper Relational Models of Data for Large Shared Data Banks [1]. With the advent of the Internet. Databases not only provide critical infrastructure for computer applications; they also process the transactions and exchanges that power much of the global economy. An important and growing segment of the software industry known as the database industry includes IBM Corporation, Oracle Corporation, Informix Corporation, Sybase Incorporated, Teradata Corporation, and Microsoft Corporation.

What Codd called the "relational model" was based on his two key points:

1. It provided the ability to describe data using only natural structures. H. No aspect of format, i. H. Additional structural overlays for machine rendering.
2. Between an application program on one side and a database system on the other. It thus provided a natural and consistent foundation for a high-level query language that allows for maximum independence.

## Relational Database

- ✚ Since data is organized into one or greater tables (or "relationships") of columns and rows in relational databases, it's miles critical to understand and recognize how diverse data systems relate to at least one another. will be Social databases that be given predetermined relationships are constituted of data.
- ✚ A connection is a everyday affiliation shaped because of interplay among diverse tables.
- ✚ All cutting-edge database systems, consisting of MySQL, Microsoft SQL Server, Oracle, and Microsoft Access, are constructed on RDBMS.RDBMS accesses records withinside the database the usage of SQL queries.
- ✚ You can essentially create, remove, and replace relational databases with this application.
- ✚ A relational database is a sort of database that makes use of rows and columns to organize records into tabular shape for storing and retrieving.

## Features

- ✚ **Entity Integrity:** No two records of the database table can be completely duplicate.
- ✚ **Referential Integrity:** Only the rows of those tables can be deleted which are not used by other tables. Otherwise, it may lead to data inconsistency.
- ✚ **User-defined Integrity:** Rules defined by the users based on confidentiality and access.
- ✚ **Domain integrity:** The columns of the database tables are enclosed within some structured limits, based on default values, type of data or ranges.



## Characteristics

- ✚ Data must be stored in tabular form in DB file, that is, it should be organized in the form of rows and columns.
- ✚ Each row of table is called record/tuple. Collection of such records is known as the cardinality of the table
- ✚ Each column of the table is called an attribute/field. Collection of such columns is called the arity of the table.
- ✚ No two records of the DB table can be same. Data duplicity is therefore avoided by using a candidate key. Candidate Key is a minimum set of attributes required to identify each record uniquely.
- ✚ Tables are related to each other with the help for foreign keys.
- ✚ Database tables also allow NULL values, that is if the values of any of the element of the table are not filled or are missing, it becomes a NULL value, which is not equivalent to zero. (NOTE: Primary key cannot have a NULL value).

## Difference Between DBMS and RDBMS

No.	DBMS	RDBMS
1)	DBMS applications store data as file.	RDBMS applications store data in a tabular form.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	Normalization is not present in DBMS.	Normalization is present in RDBMS.
4)	DBMS does not apply any security with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Atomicity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be no relation between the tables.	In RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS does not support distributed database.	RDBMS supports distributed database.
8)	DBMS is meant to be for small organization and deal with small data. it supports single user.	RDBMS is designed to handle large amount of data. it supports multiple users.
9)	Examples of DBMS are file systems, xml etc.	Example of RDBMS are MySQL, Postgre, SQL server, oracle etc.

## Advantages

-  Easy to manage: Each table can be independently manipulated without affecting others.
-  Security: It is more secure consisting of multiple levels of security. Access of data shared can be limited.

- ✚ Flexible: Updating of data can be done at a single point without making amendments at multiple files. Databases can easily be extended to incorporate more records, thus providing greater scalability. Also, facilitates easy application of SQL queries.

- ✚ Users: RDBMS supports client-side architecture storing multiple users together.
- ✚ Facilitates storage and retrieval of large amount of data.
- ✚ Easy Data Handling:
  - ✚ Data fetching is faster because of relational architecture.
  - ✚ Data redundancy or duplicity is avoided due to keys, indexes, and normalization principles.
  - ✚ Data consistency is ensured because RDBMS is based on ACID properties for data transactions (Atomicity Consistency Isolation Durability).
- ✚ Fault Tolerance: Replication of databases provides simultaneous access and helps the system recover in case of disasters, such as power failures or sudden shutdowns

## Disadvantages

- ✚ **High Cost and Extensive Hardware and Software Support:** Huge costs and setups are required to make these systems functional.
- ✚ **Scalability:** In case of addition of more data, servers along with additional power, and memory are required.
- ✚ **Complexity:** Voluminous data creates complexity in understanding of relations and may lower down the performance.
- ✚ **Structured Limits:** The fields or columns of a relational database system is enclosed within various limits, which may lead to loss of data.

## Database Table/ Relation

- ✚ Everything in a relational database is stored in the form of relations.
- ✚ A table is a collection of related data entries, and it consists of columns and rows.
- ✚ A column holds specific information about every record in the table.
- ✚ A record (or row) is each individual entry that exists in a table.
- ✚ Each table represents some real-world objects such as person, place, or event about which information is collected.
- ✚ The organized collection of data into a relational table is known as the logical view of the database.

## Properties of a Relation

- ✚ Each relation has a unique name by which it is identified in the database.
- ✚ Name of the relation is distinct from all other relations.
- ✚ Each relation cell contains exactly one atomic (single) value
- ✚ Each attribute contains a distinct name
- ✚ Attribute domain has no significance

- ✚ Tuple has no duplicate value
- ✚ Order of tuple can have a different sequence.
- ✚ Relation does not contain duplicate tuples. The tuples of a relation have no specific order.
- ✚ All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

## Row/ Record

A row of a table is also called a record or tuple. It contains the specific information of each entry in the table. It is a horizontal entity in the table.

### Properties of a row:

- ✚ No two tuples are identical to each other in all their entries.
- ✚ All tuples of the relation have the same format and the same number of entries.
- ✚ The order of the tuple is irrelevant. They are identified by their content, not by their position.

## Column/attribute

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example, "name" is a column in the above table which contains all information about a student's name.

### Properties of an Attribute:

- ✚ Every attribute of a relation must have a name.
- ✚ Null values are permitted for the attributes.
- ✚ Default values can be specified for an attribute automatically inserted if no other value is specified for an attribute.
- ✚ Attributes that uniquely identify each tuple of a relation are the primary key.

## Data item/Cells

The smallest unit of data in the table is the individual data item. It is stored at the intersection of tuples and attributes.

**Properties of data items:**

- ✚ Data items are atomic.
- ✚ The data items for an attribute should be drawn from the same domain.

## RDBMS constraints

**Primary keys** -- this identifies each row in the table. One table can only contain one primary key. The key must be unique and without null values.

**Foreign keys** -- this is used to link two tables. The foreign key is kept in one table and refers to the primary key associated with another table.

**Not null** -- this ensures that every column does not have a null value, such as an empty cell.

**Check** -- this confirms that each entry in a column or row satisfies a precise condition and that every column holds unique data.

**Data integrity** -- the integrity of the data must be confirmed before the data is created.

## Relational Model concept

Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

**Domain:** It contains a set of atomic values that an attribute can take.

**Attribute:** It contains the name of a column in a particular table. Each attribute  $A_i$  must have a domain,  $dom(A_i)$

**Relational instance:** In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

**Relational schema:** A relational schema contains the name of the relation and name of all columns or attributes.

**Relational key:** In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

## Real Life Example

## Hospital Management System

**Aim:** XYZ hospital is a multi-specialty hospital that includes a number of departments, rooms, doctors, nurses, compounders, and other staff working in the hospital. Patients having different kinds of ailments come to the hospital and get checkup done from the concerned doctors. If required they are admitted in the hospital and discharged after treatment.

The aim of this case study is to design and develop a database for the hospital to maintain the records of various departments, rooms, and doctors in the hospital. It also maintains records of the regular patients, patients admitted in the hospital, the checkup of patients done by the doctors, the patients that have been operated, and patients discharged from the hospital.

**Description:** In hospital, there are many departments like Orthopedic, Pathology, Emergency, Dental, Gynecology, Anesthetics, I.C.U., Blood Bank, Operation Theater, Laboratory, M.R.I., Neurology, Cardiology, Cancer Department, Corpse, etc. There is an OPD where patients come and get a card (that is, entry card of the patient) for check up from the concerned doctor. After making entry in the card, they go to the concerned doctor's room and the doctor checks up their ailments. According to the ailments, the doctor either prescribes medicine or admits the patient in the concerned department. The patient may choose either private or general room according to his/her need. But before getting admission in the hospital, the patient has to fulfill certain formalities of the hospital like room charges, etc. After the treatment is completed, the doctor discharges the patient. Before discharging from the hospital, the patient again has to complete certain formalities of the hospital like balance charges, test charges, operation charges (if any), blood charges, doctors' charges, etc.

Next, we talk about the doctors of the hospital. There are two types of the doctors in the hospital, namely, regular doctors and call on doctors. Regular doctors are those doctors who come to the hospital daily. Calls on doctors are those doctors who are called by the hospital if the concerned doctor is not available.

### Table Description:

Following are the tables along with constraints used in Hospital Management database.

**1. DEPARTMENT:** This table consists of details about the various departments in the hospital. The information stored in this table includes department name, department location, and facilities available in that department.

Constraint: Department name will be unique for each department.

**2. ALL\_DOCTORS:** This table stores information about all the doctors working for the hospital and the departments they are associated with. Each doctor is given an identity number starting with DR or DC prefixes only.

Constraint: Identity number is unique for each doctor and the corresponding department should exist in DEPARTMENT table.

**3. DOC\_REG:** This table stores details of regular doctors working in the hospital. Doctors are referred to by their doctor number. This table also stores personal details of doctors like name, qualification, address, phone number, salary, date of joining, etc.

Constraint: Doctor's number entered should contain DR only as a prefix and must exist in ALL\_DOCTORS table.

**4. DOC\_ON\_CALL:** This table stores details of doctors called by hospital when additional doctors are required. Doctors are referred to by their doctor number. Other personal



details like name, qualification, fees per call, payment due, address, phone number, etc., are also stored.

Constraint: Doctor's number entered should contain DC only as a prefix and must exist in ALL\_DOCTORS table.

**5. PAT\_ENTRY:** The record in this table is created when any patient arrives in the hospital for a checkup. When patient arrives, a patient number is generated which acts as a primary key. Other details like name, age, sex, address, city, phone number, entry date, name of the doctor referred to, diagnosis, and department name are also stored. After storing the necessary details patient is sent to the doctor for checkup.

Constraint: Patient number should begin with prefix PT. Sex should be M or F only.

Doctor's name and department referred must exist.

**6. PAT\_CHKUP:** This table stores the details about the patients who get treatment from the doctor referred to. Details like patient number from patient entry table, doctor number, date of checkup, diagnosis, and treatment are stored. One more field status is used to indicate whether patient is admitted, referred for operation or is a regular patient to the hospital. If patient is admitted, further details are stored in PAT\_ADMIT table. If patient is referred for operation, the further details are stored in PAT\_OPR table and if patient is a regular patient to the hospital, the further details are stored in PAT\_REG table.

Constraint: Patient number should exist in PAT\_ENTRY table and it should be unique.

**7. PAT\_ADMIT:** When patient is admitted, his/her related details are stored in this table. Information stored includes patient number, advance payment, mode of payment, room number, department, date of admission, initial condition, diagnosis, treatment, number of the doctor under whom treatment is done, attendant name, etc.

Constraint: Patient number should exist in PAT\_ENTRY table. Department, doctor number, room number must be valid.

**8. PAT\_DIS:** An entry is made in this table whenever a patient gets discharged from the hospital. Each entry includes details like patient number, treatment given, treatment advice, payment made, mode of payment, date of discharge, etc.

Constraint: Patient number should exist in PAT\_ENTRY table.

**9. PAT\_REG:** Details of regular patients are stored in this table. Information stored includes date of visit, diagnosis, treatment, medicine recommended, status of treatment, etc.

Constraint: Patient number should exist in patient entry table. There can be multiple entries of one patient as patient might be visiting hospital repeatedly for check up and there will be entry for patient's each visit.

**10. PAT\_OPR:** If patient is operated in the hospital, his/her details are stored in this table. Information stored includes patient number, date of admission, date of operation, number of the doctor who conducted the operation, number of the operation theater in which operation was carried out, type of operation, patient's condition before and after operation, treatment advice, etc.

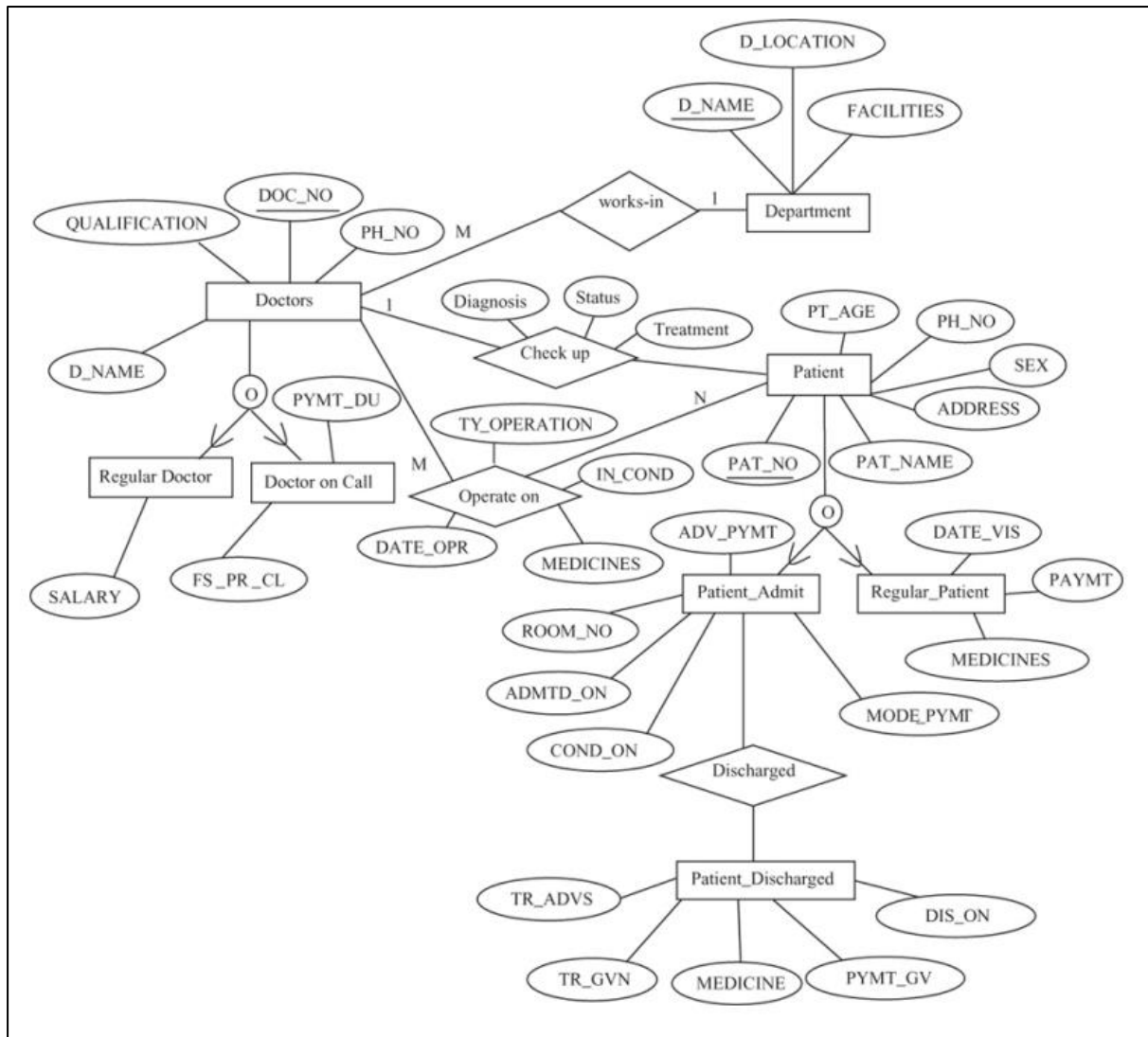
Constraint: Patient number should exist in PAT\_ENTRY table. Department, doctor number should exist or should be valid.

**11. ROOM\_DETAILS:** It contains details of all rooms in the hospital. The details stored in this

table include room number, room type (general or private), status (whether occupied or not), if occupied, then patient number, patient name, charges per day, etc.

Constraint: Room number should be unique. Room type can only be G or P and status can only be Y or N

## **ER Diagram**



## References

1. Bachman, C.W., Daya, M.: The role concept in data models. In: Proceedings of the third international conference on Very large data bases - Volume 3. VLDB '77, VLDB Endowment (1977) 464–476
2. Steimann, F.: On the representation of roles in object-oriented and conceptual modelling. *Data & Knowledge Engineering* 35(1) (1999) 83 – 106
3. Guarino, N.: Concepts, attributes and arbitrary relations: Some linguistic and ontological criteria for structuring knowledge bases. *Data & Knowledge Engineering* 8(3) (1992) 249 – 261
4. Jaeschke, G., Schek, H.J.: Remarks on the algebra of non first normal form relations. In: Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems. (1982) 124–138
5. Copeland, G.P., Khoshafian, S.N.: A decomposition storage model. In: Proceedings of the 1985 ACM SIGMOD. SIGMOD '85 (1985) 268–279
6. Falkenberg, E.D.: Concepts for modelling information. In: IFIP Working Conference on Modelling in Data Base Management Systems. (1976) 95–109
7. Halpin, T.: Orm/niam object-role modeling. In Bernus, P., Mertins, K., Schmidt, G., eds.: *Handbook on Architectures of Information Systems. International Handbooks on Information Systems.* Springer Berlin Heidelberg (1998) 81–101
8. Brachman, R.J., Schmolze, J.G.: An overview of the kl-one knowledge representation system. *Cognitive Science* 9(2) (1985) 171 – 216
9. Monpratarnchai, S., Tetsuo, T.: The design and implementation of a role model based language, epsilonj. In: *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. Volume 1.* (2008) 37–40
10. Steimann, F.: *Formale Modellierung mit Rollen.* (2000)
11. Wong, R.K., Chau, H.L., Lochovsky, F.H.: A data model and semantics of objects with dynamic roles (1996)