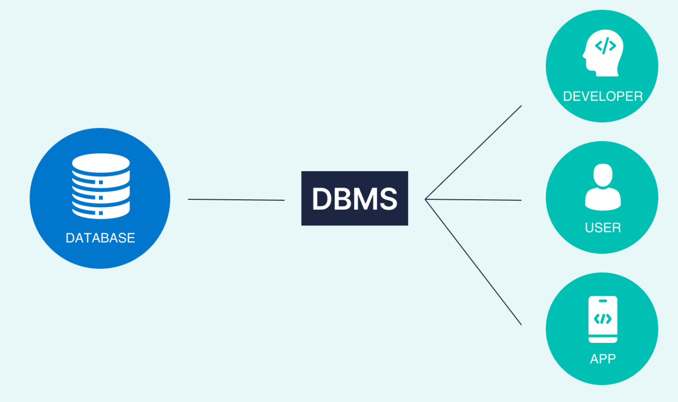
DBMS
AND
RDBMS

DBMS (Database Management System)

Database

- Database is an organized collection of structured information, or data.
- Typically stored electronically in a computer system
- It is controlled by a Database Management System (DBMS)
- Data is information that can be processed



Database Management System

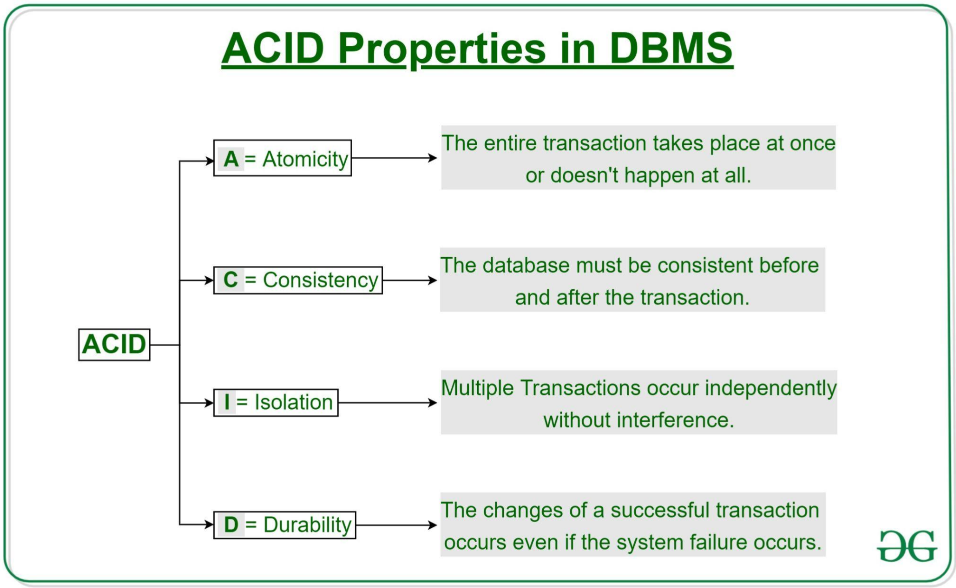
- A computerized data-keeping system.
- A software system that is designed to manage and organize data in a structured manner.
- It allows users to create,modify and query a database
- For example XML,Window registry

RDBMS(Relational Database Management System)

- A relational database (RDB) is a way of structuring information in tables, rows, and columns
- Each table represents a specific object in the database like users, products, orders, and so on.
- An RDB has the ability to establish links (relationships) between information by joining tables
- Each entity (table) in a relational database can "relate" to another to create more tables which makes the flow of data flexible
- RDBMSs include MySQL, PostgreSQL, Microsoft SQL Server, and Oracle Database.
- Relation refers to a table in Relational Database

Difference between DBMS and RDBMS:

DBMS	RDBMS
DBMS stores data as file.	stores data in tabular form
Data elements need to access individually.	Multiple data elements can be accessed at the same time.
No relationship between data.	Data is stored in the form of tables which are related to each other
Normalization is not present.	Normalization is present.
DBMS does not support distributed database.	RDBMS supports distributed database.
It stores data in either a navigational or hierarchical form.	It uses a tabular structure where the headers are the column names, and the rows contain corresponding values.
It deals with small quantity of data.	It deals with large amount of data
Data redundancy is common in this model.	Keys and indexes do not allow Data redundancy.
It is used for small organization and deal with small data	It is used to handle large amount of data.
The data in a DBMS is subject to low security levels with regards to data manipulation	There exists multiple levels of data security in a RDBMS.
Examples: XML, Window Registry, Forxpro, dbaseIIIplus etc.	Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

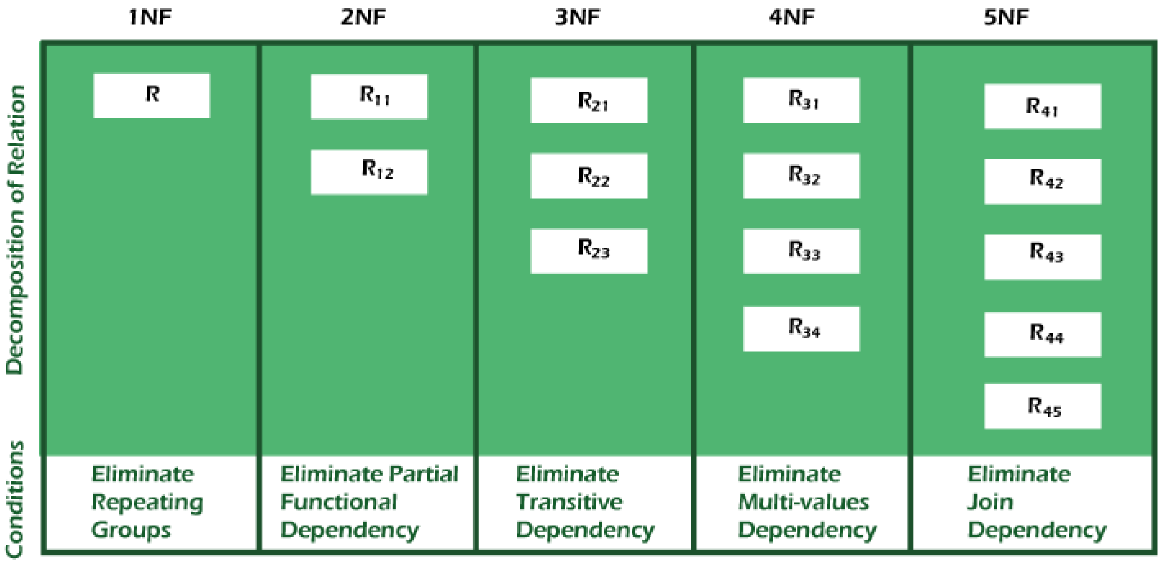

ACID Properties in DBMS
ACID
A = Atomicity
The entire transaction takes place at once or doesn't happen at all.
C = Consistency
The database must be consistent before and after the transaction.
I = Isolation
Multiple Transactions occur independently without interference.
D = Durability
The changes of a successful transaction occurs even if the system failure occurs.

NORMALIZATION AND TYPES OF NORMALIZATION

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Types of Normal Forms:


1NF
2NF
3NF
4NF
5NF
Decomposition of Relation
Conditions
R
R11
R12
R21
R22
R23
R31
R32
R33
R34
R41
R42
R43
R44
R45
Eliminate Repeating Groups
Eliminate Partial Functional Dependency
Eliminate Transitive Dependency
Eliminate Multi-values Dependency
Eliminate Join Dependency

Advantages of Normalization:

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Types of Normalization:

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
- 5NF

1NF(First Normal Form)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attributes.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE

EMPLOYEE TABLE

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMPLOYEE TABLE

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

2NF (Second Normal Form)

- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Maths	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL

TEACHER_ID	TEACHER_AGE
25	30
25	35
47	38

TEACHER_SUBJECT

TEACHER_ID	SUBJECT

25	Chemistry
25	Biology
47	English
83	Maths
83	Computer

3NF(Third Normal Form)

- A relation will be in 3NF if it is in 2NF and does not contain any transitive partial dependency.
- 3NF is used to reduce data duplication.
- It is also used to achieve data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in the third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial functional dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

EMPLOYEE_DETAIL Table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
2	Ajay	201	UP	Noida
3	Stephen	022	US	Boston
4	Femina	333	US	Chicago
5	Nimmy	453	UK	Norwich
6	John	234	MP	Bhopal

Super Key in the above Table:

{EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate Key: {EMP_ID}

Non-Prime Attributes:

Here, EMP_STATE and EMP_CITY depend on EMP_ZIP and EMP_ZIP depends on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) are transitively dependent on the super key(EMP_ID). It violates the rule of third normal form.
That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE Table:

EMP_ID	**EMP_NAME**	**EMP_ZIP**
2	Ajay	201
3	Stephen	022
4	Femina	333
5	Nimmy	453
6	John	234

EMPLOYEE ZIP Table:

EMP_ZIP	**EMP_STATE**	**EMP_CITY**
201	UP	Noida
022	US	Boston
333	US	Chicago
453	UK	Norwich
234	MP	Bhopal

Boyce Codd Normal Form(BCNF)

- BCNF is the advanced version of 3NF. It is stricter than 3NF.
- A table is in BCNF ,if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every Functional Dependency, LHS is a super key.

Example: Let's assume there is a company where employees work in more than one department

EMPLOYEE Table

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D3	283
264	India	Testing	D3	300
364	UK	Stores	D2	232
364	UK	Developing	D2	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP_ID, EMP_DEPT}

EMP_COUNTRY Table

EMP_ID	EMP_COUNTRY
264	India
364	UK

EMP_DEPT Table

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D3	283
Testing	D3	300
Stores	D2	232
Developing	D2	549

EMP_DEPT_MAPPING Table

EMP_ID	EMP_DEPT
D3	283
D3	300
D2	232
D2	549

Functional dependencies:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID

For the second table: EMP_DEPT

For the third table: {EMP_ID, EMP_DEPT}

Fourth Normal Form(4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multivalued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multivalued dependency.

STUDENT Table

STUD_ID	**COURSE**	**HOBBY**
21	Computer	Dancing
21	Maths	Singing
34	Chemistry	Dancing
74	Biology	Hockey
59	Physics	Cricket

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entities. Hence, there is no relationship between COURSE and HOBBY.In the STUDENT relation, a student with STUD_ID, 21 contains two courses, Computer and Math and two hobbies, Dancing and Singing. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STUD_ID	**COURSE**
21	Computer
21	Maths
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STUD_ID	**HOBBY**
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey