Parul® University

Relational Database Management System

Prof. Manish Kumar Joshi, Assistant Professor
Prof. Nishant Khatri, Assistant Professor
Parul Institute Of Computer Application.

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-4

Relational Database Design

Relational Data Structure

A type of DS (Data Structure) in which data are represented as tables in which no entry contains more than one value.

Features of relational Data structure

All data stored in the tables are provided by an Relational Database Management System.

Ensures that all data stored are in the form of rows and columns

Facilitates primary key, which helps in uniquely identification of the rows

Index creation for retrieving data at a higher speed.

Facilitates a similar or common column to be shared amid two or more tables.

Multi-users accessibility is facilitated to be controlled by individual users

A virtual table creation is enabled to store sensitive data and simplifies queries.

Relational data manipulation

One of the primary functions of a database management system (DBMS) is to be able to manipulate data. This means adding new data, changing the values of existing data and reorganizing the data. Other basic form of data manipulation is to retrieve specific information from the database.

For example, for a database of employees within an organization, you may want to find just the employees hired within the last year or those holding a certain position. In database terminology, this is called a query. The term 'query' means 'to search, to question or to find.' So, a database query is like asking a question of the database.

Integrity constraint

Integrity constraints are a set of rules. It is used to maintain the quality of information.

Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.

Since, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity constraint

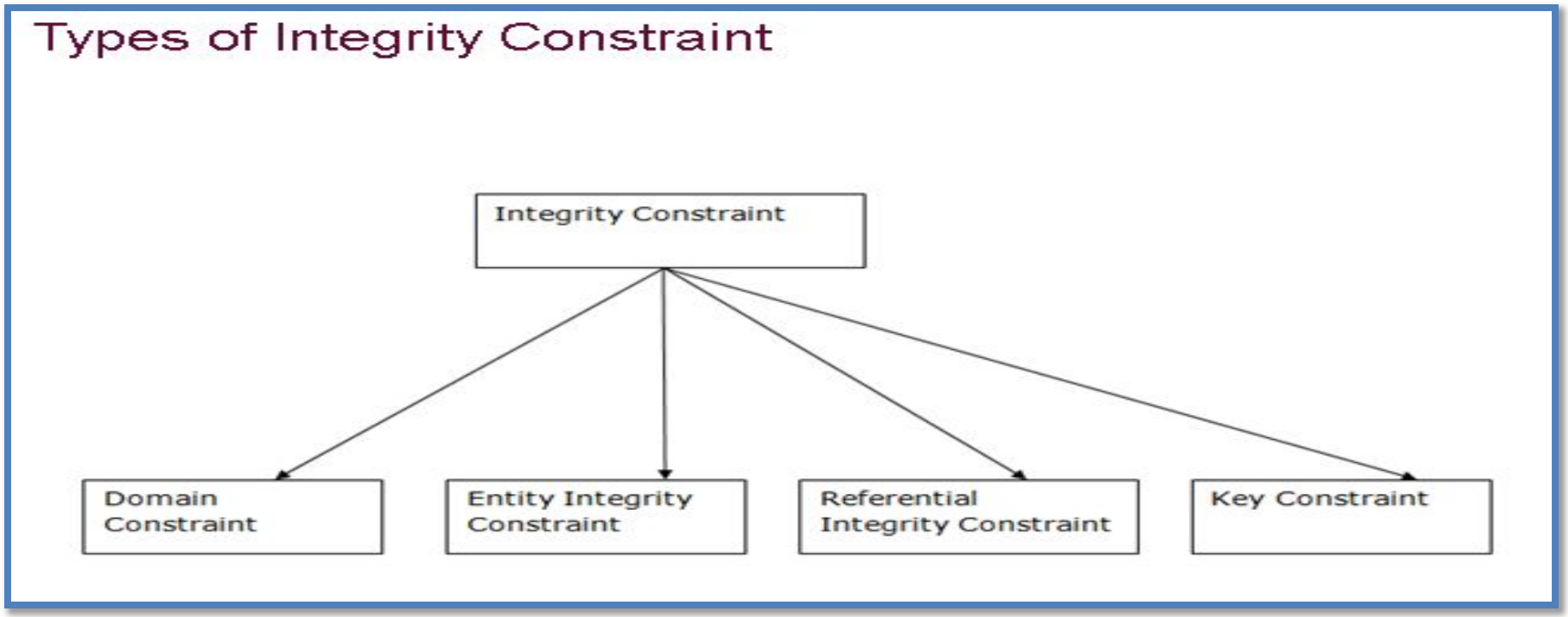


Image From Self

Domain Constraints

Domain Constraints : Domain constraints can be define as the definition of a valid set of values for an attribute.

The data type of domain which includes string, character, integer, time, date, currency, etc. The values of the attribute must be available in the corresponding domain**.**

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

Image From Self

Referential Integrity Constraints

Referential Integrity Constraints: A referential integrity constraint is specialised between two tables.

In this type of Referential Integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table1 must be null() or be available in Table2.

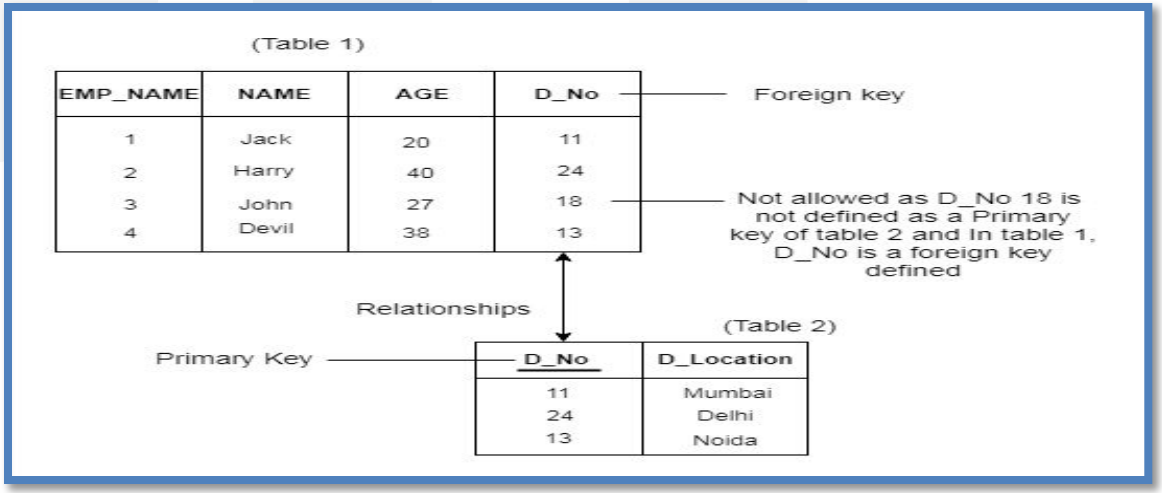


Image From Self

Entity Integrity Constraints

Entity integrity constraints : The Entity integrity constraint define that primary key value can't be null.

This is because the primary key value is used to identify or uniquely individual rows in relation and if the primary key has a null value, then we can't identify those rows.

A tables can contain a null value other than the primary key field.

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Image From Self

Key Constraints

- **Key constraints:** Keys are the entity set that is used to identify an entity within its entity set unique.
- An entity set can have many keys, but out of which one key will be the primary key. A primary key contain a unique and null value in the relational table.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Image From Self

Function Dependency

Functional Dependency is a relationship that occurs between multiple attributes of a relation.
This concept is given by E.F.Codd (Edgar Freak Codd).
Functional dependency represents a formalism on the infrastructures of relation.
Its a type of constraint existing between various attributes of a relation.
It is used to describe various normal forms.
These dependencies are restrictions imposed on the data in database.
If P is a relation with A and B attributes, a functional dependency between these two attributes is represented as {A → B}. It specifies that,
A It is a determinant set.

Function Dependency

B It is a dependent attribute. {A → B}A functionally determines B. B is a functionally dependent on A.

Each value of A is associated precisely with other one B value. A function dependency is trivial if B is a subset of A.

'A' Functionality determines 'B' {A → B} (Left hand side attributes determine the values of Right hand side attributes).

For example: <Employee> Table

EmpId	EmpName

Image From Self

Function Dependency

In the above <Employee> table, EmpName (employee name) is functionally dependent on EmpId (employee id) because the EmpId is unique for individual names.

The EmpId identifies the employee specifically, but EmpName cannot distinguish the EmpId because more than one employee could have the same name.

The functional dependency between attribute eliminates the repetition of informations.

It is related to a candidate key, which uniquely identifies a tuple and determines the value of all other attributes in the relation.

Advantages of Functional Dependency

- Functional Dependency avoid data redundancy (Repetitions) where same data should not be repeated at multiple locations in same database.
- It maintains the quality of data in database.
- It allows clearly defined meanings and constraints of databases.
- It helps in identifying bad designs.
- It expresses the facts about the database design.

Normalization

Database normalizations is a database schema design techniques, by which an existing schema is modified to reduce the redundancy and dependency of data.
Normalization split a long table into smaller table and defines relationships between them to increases the clarity in organizing data.

Some facts About database Normalization

- The word normalization and normal form refer to the structure of a database.
- Normalization was developed by IBM researcher E.F. Codd In the 1970s.
- Normalization increases clarity in organizing data in Databases.
- Normalization of a Database is achieved by following a set of rules called 'forms' in creating the database

First Normal Form (1NF)

- First normal form enforces these criteria:
- Eliminate repeating groups in individual tables.
- Creates a different table for each set of related data.
- Identify every set of related data with a primary key.

Sample Employee table, it displays employees are working with multiple departments.

Employee	Age	Department
Melvin	32	Marketing, Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

Image From Self

Employee table following 1NF:

Employee	Age	Department
Melvin	32	Marketing
Melvin	32	Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

Image From Self

Second normal form[2nf]

The entity should be considered already in 1NF, and all attributes within the entity should only depend solely on the unique identifier of the entity.

eg:-

Sample Products table:

productID	product	Brand
1	Monitor	Apple
2	Monitor	Samsung
3	Scanner	HP
4	Head phone	JBL

Second normal form[2nf]

Product table following 2NF:

Products Category table:

productID	product
1	Monitor
2	Scanner
3	Head phone

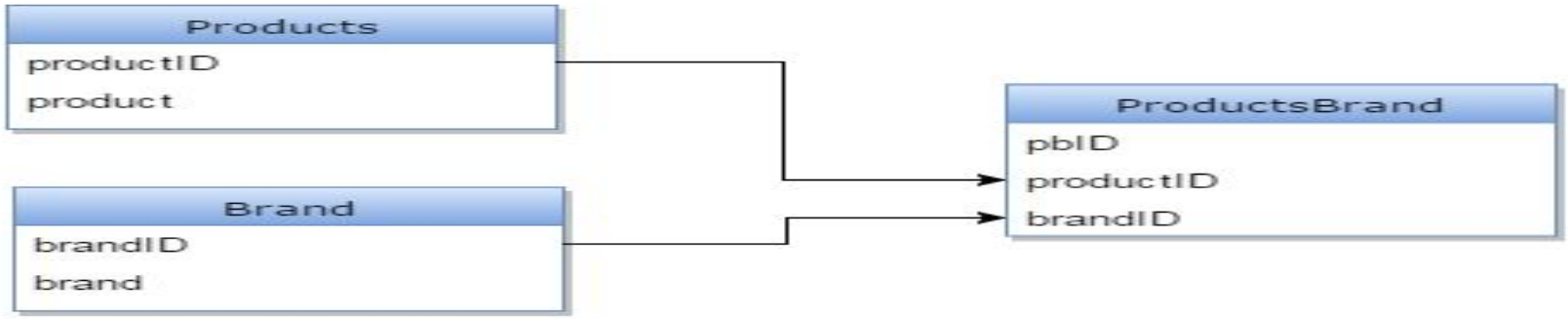
Brand table:

brandID	brand
1	Apple
2	Samsung
3	HP
4	JBL

Second normal form[2nf]

Products Brand table:

pbID	productID	brandID
1	1	1
2	1	2
3	2	3
4	3	4



Second Normal Form (2NF)

Third normal form(3nf)

The entity should be considered within 2NF, and no columns entry should be dependent on any other entry (value) other than the key for the table.
If such an entity exists, move it outside into a new tables.
3NF(Third Normal Form) is achieved, considered as the database is normalized.

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

Third normal form(3nf)

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

X is a super key.

Y is a prime attribute, example:- each element of Y is part of some candidate key

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Image from self

Third normal form(3nf)

Super key in the table above:

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

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on 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.

Third normal form(3nf)

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

BCNF[Boyce codd normal form]

3NF(Third Normal Form) and all tables in the database should be only one primary key.

BCNF is the advance or new 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 tables.

For BCNF, the table should be in 3NF, and for every FD, LHS is super keys.

BCNF[Boyce codd normal form]

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	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

BCNF[Boyce codd normal form]

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

BCNF[Boyce codd normal form]

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

BCNF[Boyce codd normal form]

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

```
EMP_ID      →    EMP_COUNTRY
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}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

Fourth normal form[4nf]

Table cannot have multi-valued dependency on a Primary Key(Null).
A relation will be in 4NF if it is in BoyceCodd normal form and has no multi-valued dependency.
For dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Fourth normal form[4nf]

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_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.

Fourth normal form[4nf]

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_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.

Fourth normal form[4nf]

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Fifth normal form [5nf]

A relations is in 5NF if it is in 4NF and not contains any join dependency and joinings should be lossless.

5NF (Fifth Normal Form)is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.

5NF is also called as Project-join normal form (PJ/NF).

Fifth normal form [5nf]

Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

Parul® University

DIGITAL LEARNING CONTENT

Fifth normal form [5nf]

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

Fifth normal form [5nf]

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Domain-Key Normal Form (DCNF)

There is no rule to define normal form up to 5NF. Historically the process of normalization and the processes of discovering undesirable dependencies were carried through 5NF, but it has been possible to define the stricter normal form that takes into accounts additional type of dependency and constraint.

The basic idea behind the *Domain Key Normal Form* is to specify the normal form that takes in the account all the possible dependencies and constraints.

In simple words, we can say that Domain Key Normal Form is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

Domain-Key Normal Form (DCNF)

In other word, a relation schema is said to be in Domain Key Normal Form only if all the constraints and dependencies that should hold on the valid relation state can be enforced simply by enforcing the domain constraints and the key constraints on the relation. For a relation in Domain Key Normal Form , it becomes very straight forward to enforce all the database constraints by simply checking that each attribute value is a tuple (Mathematical Row) is of the appropriate domain and that every key constraint is enforced.

Reason to use DKNF are follow:

To avoid general constraints in the database that are not clear key constraint.

Most database can easily test or check key constraints on attribute.

Domain-Key Normal Form (DCNF)

Since, because of the difficulty of including complex constraints in a Domain Key Normal Form relation its practical utility is limited means that they are not in practical use, since it may be quite difficult to specified general integrity constraints.

Let's understand this by taking an example:

Consider relations CAR (MAKE, vin#) and MANUFACTURE (vin#, country), Where vin# represents the vehicle identification number 'country' represents the name of the country where it is manufactured.

Domain-Key Normal Form (DCNF)

A general constraints may be of the following forms:
The Make is either ‘HONDA’ or ‘MARUTI’ then the first character of the vin # is a ‘B’ If the country of manufacture is ‘INDIA’
If the MAKE is either ‘FORD’ or ‘ACCURA’, the 2nd character of the vin# is a ‘B” if the country of manufacture is ‘INDIA’.
There is no simplified way to represents such constraint short of writing a procedure or general assertion to test them. Since such a procedure needs to enforce an appropriate integrity constraint. Hence, transforming a higher normal form into domain/key normal form is not always a dependency-preserving transformation and these are not possible always.

Denormalization

Denormalization is a database optimizations method in which we add redundant data to one or more table. This can help us avoid costly joins in a relational database. Note that denormalization does not mean that we do not have to do normalization. It is an optimizations technique that is applied after doing normalization.

In a traditional normalized database, we store data in different logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in database.

Denormalization

For eg:-, in a normalized database, we might have a Course table and a Teachers table .Each entry in Courses would store the teacherID for a Course but not the teacherName. When we need to retrieve a list of all Courses with the Teacher name, we would do a joins among these two tables.
In many ways, this is great; if a teacher changes is or her name, we only have to update the name in one place.
The disadvantage is that if tables are large, we may spend an unnecessarily long time doing joins on tables.
Denormalization, then, strikes a different compromise. Under denormalization, we decide that we're okay with some redundancy and some extra effort to update the database in order to get the efficiency advantages of fewer or less joins.

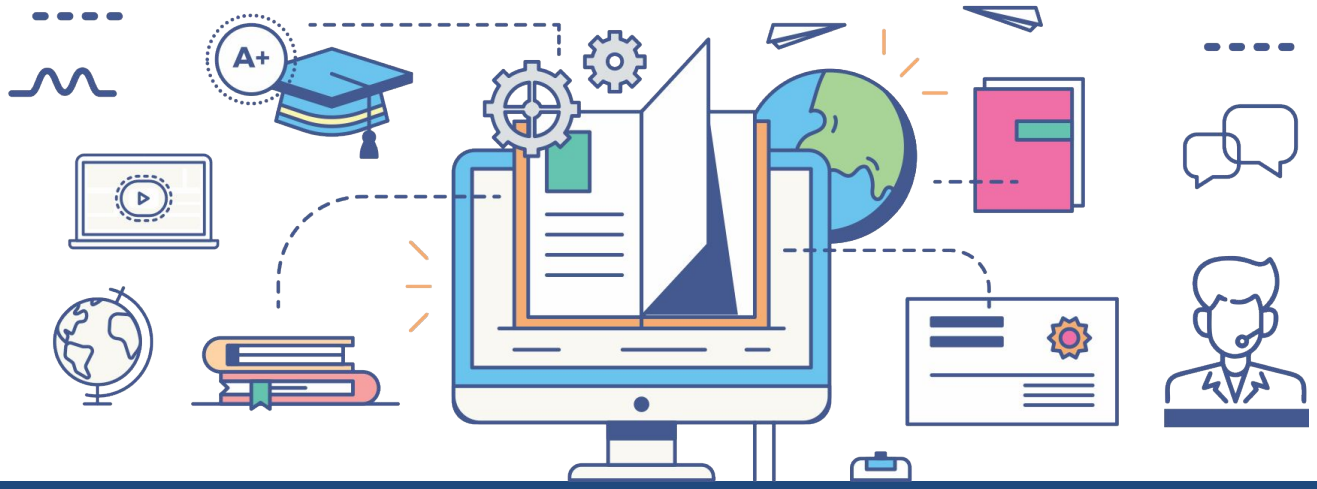
Pros Of Denormalization

- Retrieving or data fetching of data is faster hence we do fewer joins
- Queries to retrieve can be easier or simpler(and therefore less likely to have bugs), hence we need to look at fewer tables.

Cons of Denormalization

- Updates and inserts are more expensive.
- Denormalization make supdates and insert code hard to write.
- Data may be inconsistent
- Data redundancy necessitates more storage.
- In a system that demand scalability, like that of any major tech companies, we almost always use elements of both of the things normalized and denormalized database.

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in