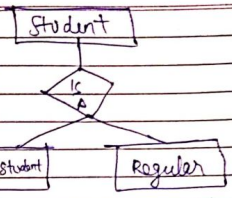
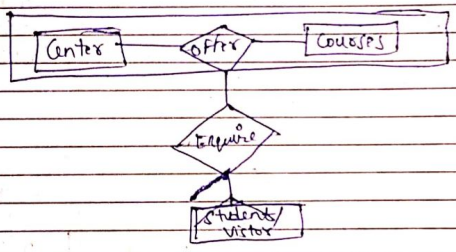


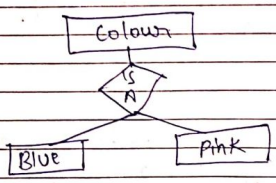
Top down approach

(iii) Aggregation

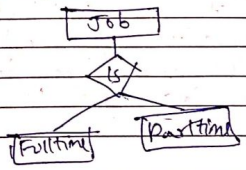
It is the process when relation blw two entities treated as a single entity.



(i)



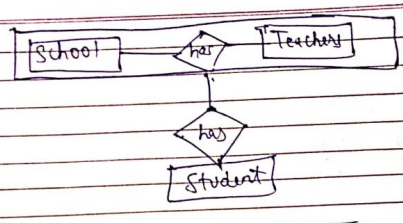
(ii)

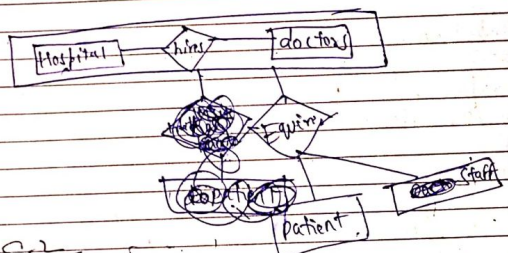


Spiral

Spiral

(iii)





S-2

* Relational Model → DBMS

It represents the database as a collection of relations which is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real world entity or relationship.

* Relational Model Concepts

1. Attribute — column in a table.

Attributes are the properties which define a relation.

Spiral

Date.....

2- Tables :- A relation are saved in the table format. It has two properties one is row and other is column. Rows represents record and column represents attributes. A single row of a table having a single record is called tuple.

3- Relation schema

It represents the name of the relation with its attributes.

4- Degree

The total no. of attributes (column) which in the relation exist is called the degree of relation.

5- Cardinality

The total no. of rows in a table.

* Relational Integrity Constraints

It is a condition that can be applied on the database schema to restrict the data according to the need. The condition is satisfied then only it can be stored in the database. It is to ensure that there should not be any loss in data consistency due to changes made to the database by the authorised users.
like - Phone no. (are always 10 digits)

Spiral

Date.....

We have different type of constraints.
1- Entity - Integrity constraint
2- Referential - Integrity constraint
3- Domain constraint
4- Key constraint

A relation schema R
(i) denoted by $R(A_1, A_2, \ldots, A_n)$ is made up of relation name R and a list of attributes $A_1, A_2, \ldots, A_n$.
(ii) Each attribute A_i is a name of role played by some domain D in the relation schema are these called a Domain of A_i. and is denoted by $dom(A_i)$.

- The data types describing the types of values that can appear in each column is called domain.
- A Domain D is a set of atomic values By atomic we mean that each value in the domain is indivisible as far as the relational model is concerned.
For ex- Phoneno, Aadharno.,

1. Entity - Integrity Constraint

It states that no primary key value can be NULL.

2. Referential Integrity Constraint

It ensures that a value that appears in one relation for a given set of attributes also

Spiral

Date

appear for a ~~cost~~ certain set of attribute in ~~of~~ another relation. It is the Foreign key. For ex.

Customers

Cust ID	Name	status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Billing

Invoice no	Cust Id	Amount
1	1	100
2	1	200
3	2	300

The total amount associated with google is 300 Here Cust_Id acts as a Foreign Key.

3. Domain Constraint

Domain - is a set of atomic values which means that each value in the domain is individual as far as the relation model is concerned. It specifies that the value of each attribute A must be an atomic value from the Domain dom(A).

For ex.

Spiral

Date

→ Aadhaar no. ~~of~~ (set of 12 digits) or (set of valid 12 digits in India) India_mobileno.

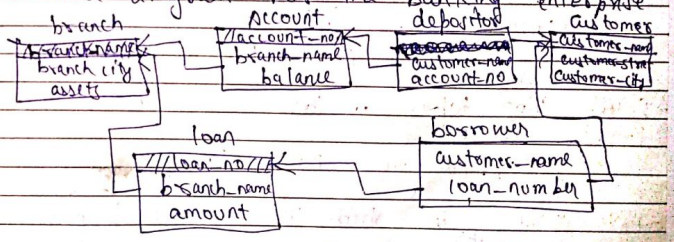
→ Employee_age (possible ages of a employee of a company each must be a value btw ≥22 and 60 yrs ~~old~~ old)

4. key constraint

A relation is defined as a set of tuples. All tuples in a relation must be distinct. This means that no two tuples can have the same combination of values for all their attributes (columns).

* Relational Algebra

→ Schema diagram for the banking enterprise



Spiral

Date

Relational Algebra is a procedural query language. It consist of set of operations that take one or two relation as input and produce the new relation as their result.
The fundamental operations are -
(i) Select
(ii) Project
(iii) Union
(iv) Set - difference
(v) Cartesian product
(vi) Set - intersection
(vii) Natural joint
(viii) Division
(ix) Assignment

(i) Select - Select tuples that satisfy the given predicate. The selection in a relational algebra is denoted by σ.

$\sigma_{predicate}(\text{Argument Relation})$

-) Select those tuples of loan relation where the branch is Noida

$\sigma_{branch\_name = \text{"Noida"}}(loan)$

-) All tuples in which the amount is more than 1200 $

$\sigma_{amount > 1200}(loan)$

Comparision = equals to, $\geq$ greater than equal to, $\neq$ not equal to

Date

We can combine several predicates into a larger predicates by using connectives
And - $\wedge$ OR - $\vee$ Not - $\neg$

Find all tuples pertaining to loans of more than 1200 $ made by the Noida branch.

Ans

$\sigma_{branchname = \text{"Noida"}}(loan) \wedge amount > 1200(loan)$

(ii) Project

It is used for columns. It is a unary operation that returns its argument relation, with certain attributes left out. denoted by Π

$\Pi_{attribute}(\text{Argument Relation})$

-) It is list all loanno. and the amount of the loan

$\Pi_{loanno, amount}(loan)$

Composition type -)
(i) Find those customer name who lives in Noida.

$\Pi_{customer\_name}(\sigma_{customername = \text{"Noida"}}(Customa))$

(iii) Union ∪

Date.....

(ii) Find the names of all bank customers who have either an account or an loan or both.

$\Pi_{customer\text{-}name}(borrower) \cup \Pi_{customer\text{-}name}(depositor)$

(iii) Set-difference (−)

It allows us to find tuples that are in one relation but are not in another.

→ Find out the customer with an account but no loan.

Ans $\Pi_{customer\_name}(depositor) - \Pi_{customer\_name}(borrower)$

(iv) Cartesian product operation (×)

It allows us to combine information from any two relations.

→ R_1 → borrower
R_2 → loan
$R_1 \times R_2$ = borrower × loan = R

Eg - Find the names of all customers who have a loan at noida branch.

Customer_name

$\Pi_{customer\_name}(\sigma_{loan.loan\_no = borrower.loan\_no}(\sigma_{branch\_name = "Noida"}(loan \times borrower)))$

Spiral

Date.....

Customer_name

1
1
1
20

10

$\Pi_{customer\_name}(\sigma_{loan.loan\_no = borrower.loan\_no}(\sigma_{branch\_name = "Noida"}(loan \times borrower)))$

(vii) Set-intersection operation (∩)

Find out customers who have a both loan and an account.

$\Pi_{customer\_name}(borrower) \cap \Pi_{customer\_name}(depositor)$

(viii) Natural join operation (⋈)

It performs a selection forcing equality on those attributes that appear in two relation schema and finally removes duplicate attributes.

→ Find the name of all customers who have a loan at the bank alongwith the loan no and the loan amount.

$\Pi_{customer\_name, loan\_no, amount}(\sigma_{loan.loan\_no = borrower.loan\_no}(loan \times borrower))$

$\Pi_{customer\_name, loan\_no, amount}(borrower \bowtie loan)$

Spiral

- Natural join is a binary operation that allows us to combine certain selection and a cartesian product into one operation

Borrower and loan have loanno. attribute in both, the natural join operation considers only pair of tuples that have the same value on loanno. It combine each such pair of tuples into a single tuple on the union of two schemas.

-) Find the names of all branches with customers who have an account in the bank. and who live in Noida.

$\Pi_{branchname}(\sigma_{customercity = "Noida"}(account \bowtie depositor \bowtie customer))$

-) Find all customers who have both a loan and a account at the bank.

$\Pi_{customer\ name}(borrower \bowtie depositor)$

(VIII) Division operator (÷)

It is suited to queries that include the phrase "for all". The result consist of the restriction of the tuple in R to the attribute name

Spiral

Date

unique to R i.e. in the header of R but not in the header of S for which it holds that all their combination with tuple in S are present are present in R

Completed		DB Project		Completed ÷ DB Project
Student	Task	Task		Student
X	Db 1	Db 1		X
X	2	Db 2		Z
X	Comp 1			
Y	Db 1			
Y	Comp 1			
Z	Db 1			
Z	Db 2			

Q Use a attribute are id, age, gender, occupation_id or city_id

Occupation: occ_id, occupation name

City : city_id, city_name

Table 1-

Id	Name	Age	Gender	Occ Id	City Id
1	John	25	Male	1	2
2	Sara	30	Female	3	4
3	Victor	31	Male	2	5
4	Jaino	27	Female	1	3

Table 2 : occupation

ID	Name
1	Software engineer
2	Accountant
3	Doctor
4	Library assistant

Spiral

Date

Table 3 : City	id	Name
	1	Helifax
	2	Calgary
	3	Boston
	4	New yorr
	5	Toronto

* Solve the following relational expression for above relation.

(i) Pname (Rage > 25 (user))

(ii) User ⋈ occupation ⋈ city

(iii) Ruser.occ.id = occupation.occupation ID (user × occupation)

(iv) Pname, gender (Rcityname = "Boston" (user ⋈ city))

(i) Sara, Victor, Jaine

(ii) ϕ

(iii) Jaine, Female

(iv)

4	Jaine	27	Female	1	3	Boston

Q Consider the DB with the following schema -

1. Person (Name, age, gender)
2. Frequents (Name, pizzeria)
3. Eats (Name, Pizza)
4. Serves (Pizzeria, Pizza, price)

Queries (1) Find all pizzerias frequented by atleast 1 person under the age of 18.

(2) Find the name of all females who eat either mushroom or plain pizza or both.

Spiral

Date

(3) Find all pizzerias that serves atleast 1 pizza that anny eats for less than 10 $.

Ans 1 - Πpizzeria (σage < 18 (Person ⋈ Frequent))

2. Πname (σgender = 'F' (σpizza = "Mushroom" () σpizza = "Plane Pizza (Persons ⋈ Eats)))

3. Πpizzeria (σname = "Anny" (σprice < 10 (Person ⋈ eats ⋈ Serves)))

* Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form

$\{t \mid P(t)\}$

- It is the set of all tuples t such that predicate P is true for t
- t is a tuple variable, t[A] denotes the value of t on attribute A
- $t \in r$ denotes that tuple t is in relation r.
- P is a formula similar to that of the predicate calculus.

* Predicate calculus formula

- Set of attributes and constants
- Set of comparison operators (eg $<, \leq, =, \neq, >, \geq$)
- Set of connectives : and ($\wedge$), or ($\vee$), not ($\neg$)
- Implication ($\Rightarrow$) : $x \Rightarrow y$, if x is true then y is true $x \Rightarrow y = \neg x \vee y$

Spiral

Date

→ Set of quantifiers :

→ $\exists t \in r (Q(t))$ = "there exists" a tuple in t in relation r such that predicate Q(t) is true

→ $\forall t \in r (Q(t))$ = Q is true "for all" tuples t in relation r.

Ex Find the ID, name, dept-name, salary for instructors whose salary is greater than $50000

Ans $\{t \mid t \in instructor \wedge t[salary] > 50000\}$

Notice that a relation on schema (ID, name, dept-name, salary) is implicitly defined by the query.

→ As in the previous query, but output only the ID attribute value

$\{t \mid \exists s \in instructor (t[ID] = s[ID] \wedge s[salary] > 50000)\}$

Notice that a relation on schema (ID) is implicitly defined by the query.

Ex Find the name of all instructors whose dept. is in the Watson building

$\{t \mid \exists s \in instructor (t[name] = s[name]) \wedge \exists u \in department (u[department-name] = s[department-name] \wedge u[building] = "Watson")\}$

Spiral

Date

* Find all employees whose salary is above 50000

Ans $\{t \mid t \in employee \wedge t[salary] > 50000\}$

* Find the branch name, loan no. and amount for loans of over 1200

Ans $\{t \mid t \in loan \wedge t[amount] > 1200\}$

Query 1. Find the loan no. for each loan of an amount greater than 1200

2. Find the name of all customers who have a loan from Noida branch.

Ans 1. $\{t \mid \exists s \in loan (t[loan-no.] = s[loan-no.] \wedge s[amount] > 1200)\}$

The set of all tuples t such that there exist a tuple s in relation loan for which the values of t and s for the loan-no. attribute are equal & the value of s for the amount attribute is greater than 1200

2. $\{t \mid \exists s \in borrower (t[customer-name] = s[customer-name] \wedge \exists u \in loan (u[branch-name] = "Noida" \wedge u[loan-no.] = s[loan-no.]))\}$

* Domain Relational Calculus

Customer

cust-name	city	Street
A	Noida	P
B	Calcutta	Q
C	Bijnor	R
D	Bombay	S

loan

Loan no.	Branch name	Amount
L01	Main	1200
L03	Main	150
L10	Sub	90
L08	Main	60

Borrower

Cust-name	Loan no.
D	L01
A	L08
C	L03

Spiral Bound

Date.....

It is a non-procedural query language equivalent in power to tuples relational calculus. It provides only the description of the query but it doesn't provide the method to solve it. A query is expressed as

$\{\langle x_1, x_2, x_3, \ldots x_n \rangle \mid P(x_1, x_2, x_3 \ldots x_n)\}$

where $\langle x_1, x_2, x_3 \ldots x_n \rangle$ represents resulting domain variables $P(x_1, x_2, x_3 \ldots x_n)$ represent the condition of formula equivalent to predicate calculus

Query: Find the loan no, branch, amount of loans of greater than or equal to 100 amount.

$\{\langle l, b, a \rangle \mid \langle l, b, a \rangle \in loan \wedge (a \geq 100)\}$

2. Find the loanno. for each loan of an amount greater or equal to 150

$\{\langle l \rangle \mid \langle l \rangle \in loan \wedge (a \geq 150)\}$

3. Find the name of all customers having a loan at the main branch and find the loan amount.

$\{\langle c, a \rangle \mid \exists l (\langle c, l \rangle \in borrower \wedge \exists b (\langle l, b, a \rangle \in loan \wedge b = \text{"main"}))\}$

Spiral

Normalization

Date.....

* Design guidelines for relations schema

(i) Reducing the redudant values in tuples
(ii) Reducing the null values in tuples
(iii) Disallows the possibility of generating spurious tuples.
(iv) Semantics of the attributes means attributes should have a meaning.

The aim to focus on redudency and consistency.

* Normalization

It is the process of decomposing a set of relations with anamolies to produce smaller and well structured relations that contain minimum or no redudency. It is the formal process of deciding which attribute should be grouped together in a relation. Using Normalization only change to the values stored in a database can be achieved with the few-est possible update operations. During normalization it is ensure that a normalized schema :-
(i) does not lose any info. present in the unnormalized schema.
(ii) does not include spurious info when the original schema is reconstructed.

Spiral

Date.....

- Preserves dependency present in the original scheme.

Normal forms

It is a state of a relation that results from applying simple rules regarding functional dependency to that relation. Relation is defined as containing two components namely first is an attribute and functional dependency b/w them

1. Attributes
2. Functional dependency b/w them

$R_1 = (\{x, y, z\}, \{x \to y, x \to z\})$

Functional dependency (FD)

It is a constraint b/w two sets of attributes from the database. FD $X \to Y$, b/w two set of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that conform a relation state or of R.

Value of the Y component of a tuple in r depends on, or are determined by the value of X component.

It can be written as value of the X component of a tuple uniquely determined the value of the Y component.

Spiral

P = Project

Date.....

SSN	P.NO	Hours	Ename	P name	P-location

FD

Ans {SSN, Project-no} → hours

SSN → Ename

P-NO → {P-location, P-name}

RollNO | SName | Address | Phno

- RollNo → Name
- Name → {Address, Phno}

Emp id | Employee | Dept | Company

- Emid → Employee
- Company → {Employee, Dept}

Spiral

1st Normal form

Date.....

Table on the relation is in 1st NF, if the values in the domain of each attribute of the relation are atomic. Only one value is associated with each attribute the value is not set of values or a list of values.
A relation is in 1st NF if it doesn't have any repeating groups.
It disallows multivalue, composite and their combination attributes.

Spiral

Date.....

* 2nd Normal form

A functional dependency $X \to Y$ is the full functional dependency
If removal of any attribute A from X means that the dependency does not hold any more.
FD $X \to Y$ is a partial dependency if some attribute $A \in X$ can be removed from X. In a dependency still holds.

* 3rd Normal form

It is based on the transitive dependency

$X \to Y$, $X \to Z$, $Z \to Y$

A FD $X \to Y$ in a relation schema R is the transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both $X \to Z$, $Z \to Y$ holds.

Ename	SSN	Birthdate	Address	DNo.	Dname	Dmgr SSN

SSN → primary key

Spiral

A-(7)

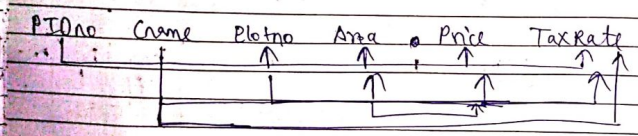
Date

SSN → {Ename, Address, Dnoo}
DNO → {Dname, DMGRSSN}

X = SSN Z = DNO.
Y = DMGRSSN ⇒ SSN ⇒ DMGRSSN
SSN → DNO.
DNO → DMGRSSN

Acc. to Codd's definition a relation is in 3rd Normal form if it satisfies 2nd Normal form and no non-prime attribute of R is transitively dependent on the primary key

A relation schema is in 3rd Normal form if whenever a non-trivial functional dependency X → A holds in R, either
(a) X is a superkey of R.
(b) A is a prime attribute of R.



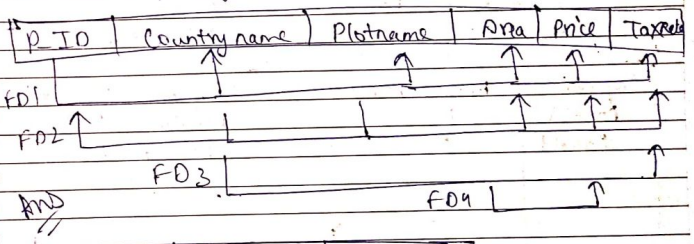
* Multivalued Dependency and 4th Normal form

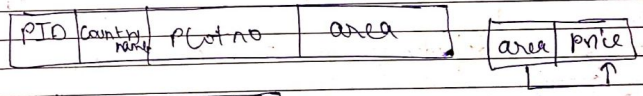
MVD is the FD where the dependency may

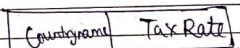
Spiral

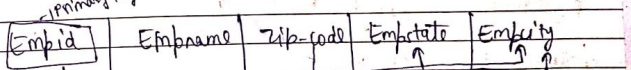
Date

be to a set and not just a single value is defined as
X →→ Y . In a relation R(X,Y,Z) if each X value is associated with a set of Y values in a way that does not depend on the Z values. X and Y both are subsets of R. A relation is said to be more NF if it is in BCNF and for every non-trivial MVD (X →→ Y) X is a superkey of R.









Find out candidate keys, non prime attributes

Candidate key - Empid
Non-prime attributes → empname, zip-code, empstate, empcity

Spiral

Empdate 2

Zipcode | Empstate | Empcity

Empdate 1

Empid | Empname | Zipcode

Q

Empname	Projectname	dependent name
Donald	P1	Mathew
Donald	P2	Rajesh
Donald	P1	Rajesh
Donald	P2	Mathew

Empname	Projectname
Donald	P1
Donald	P2

Empname	dependentname
Donald	Mathew
Donald	Rajesh

* Joint depedency in 5th Normal form

Joint depedency is said to exist if the joint of R_1 and R_2 over $C = R$ $R(A,B,C)$ and $R_1(A,B,C)$ and $R_2(C,D)$

Then R_1 and R_2 is loss less decomposition of R.

Spiral

Spiral