Unit 1

Relational Algebra

- It is a procedural query language.
- It consist of set of operations that take one or two relations as input and produce a new relation as their result.
- The fundamental operations are SELECT, PROJECT, UNION, SET DIFFERENCE, CARTESIAN PRODUCT and RENAME.
- In addition other operations include SET Intersection, natural join, division and Assignment.

SELECT operation:

- Represented by greek letter (σ)
- The predicate appears as a subscript to sigma.
- The argument relation is in parenthesis after the sigma.

- It allows comparisons using =, !=, >, <, >=, <= in the relation predicate.
- We can combine several predicates into large predicates using connectives $\wedge$ (and), $\vee$ (or), $\neg$ (not)

Eg:

table name (relation)

$\sigma_{bank\_Branch = 'ABC'}(Loan)$

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

$\sigma_{branch\_name = 'ABC' \wedge amount > 1200}(Loan)$

$\sigma_{cust\_bank = banker\_name}(Loan\_officer)$

Explain

→ validate → read → compute → write	read → compute → validate → write
→ Help in protecting the system from concurrency conflict.	→ Allows conflict to happen.
→ Suitable for small dB.	→ large dB.
-	Also known as <u>validation based protocol</u>.

21/04/23

Implimentation of Isolation:

Isolation levels are defined by following phenominon:

1. Dirty read
2. Non-repeatable read
3. Phantom read

Based on these phenominon 4 isolation lvls are:

1. Read uncommited
2. Read committed
3. Repeatable read
4. Serializable read

level

Implimentation of isolation levels is achieved using:

1. Locking
2. Timestamping.

① for each data item Q, if transaction T_i reads the initial values of Q in schedule S, then same should occur on S' also.

② for each data item Q, if transaction T_i executes read(Q) in S and if that value was produced by write(Q) operation in T_j, then same should occur in S'

③ for each data item Q the final write(Q) operation in S, then the same should occur in S' also.

* Testing of Serializability

→ Done by using directed graph called precedence graph.

→ Conditions include:

① T_i executes write(Q) before T_j executes read(Q)

② T_i executes read(Q) before T_j executes write(Q)

③ T_i executes write(Q) before T_j executes write(Q)

Test for conflict serializability

→ If the precedence graph contains no cycle, then schedule is conflict serializable.

→ Topological sorting is used → if acyclic.

Test for view serializability:

→ labled precedence graph is used.

* Recoverability

Types of schedules include:

① Recoverable schedules - for each pair of transaction T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appear before the commit oper. of T_j.

Division operation:

→ The division operator is used for queries which involve 'ALL'

→ $R_1 /: R_2$ = Tuples of R_1 associated with all tuples of R_2

Cartesian Product:

Cartesian product on two relations that is, on two set of tuples, it will take every tuple one by one from left relation and will pair it up with all the tuples in the right relation.

It is denoted by $R_1 X R_2$

Eg: ...

Tuple Relational Calculus (TRC):

It is a non procedural query language used in RDBMS to retrieve data from tables.
TRC is based on the concept of tuple which are ordered set of attribute values that represent a single row or record in a dB table.
The basic syntax of TRC is:

TRC — $\{t \mid P(t)\}$

where 't' is a tuple variable and 'P(t)' is a logical formula that describes the condition that the tuple in the result must satisfy.

Employee Table/tuple
(EmpID, name, Salary, Dept)

Q Retrive the name of all employees who earn more than 50000 per year.

Sol: ~~TRC — {Name / Salary}~~

$\{t \mid Employe(t) \wedge t.salary > 50000\}$

Name

* **RENAME op^n:**

- The RENAME op^n is used to rename the output of a relation.
  It is represented by a greek letter rho (ρ)

It is represented by $\rho_x(E)$ where the symbol ρ is used to denote RENAME operator & E is the result of the sequence of operations or expression which is saved with the name x.

Eg: query to rename relation Student as maleStudent and the attributes of student roll no., studentname as SNo., SN is represented as:

$$\rho_{MaleStudent(SNo., SN)} \pi_{Rollno, SName} (\sigma_{condition} (Student))$$

Q Query to rename the attributes Name, age of table department to A, B.

Ans ~~$\rho_{Department}(A, B)$~~ $\rho_{(A,B)}(Department)$

Q Query to rename table name project to pro, and its attributes to P, Q, R.

Ans ~~$\rho_{Pro}(P, Q, R)(Project)$~~

$\rho_{Pro(P,Q,R)}(Project)$

Q Query to rename the first attribute of table student with attributes A, B, C to P

Ans ~~$\rho\ (P, B, C)\ (Student)$~~

$\rho_{(P,B,C)}(Student)$

② Second Normal Form (2NF) :

→ There must not be any partial dependency of any column on primary key.

→ It means for a table that has concatenated primary key, each column in the table that is not the part of primary key must depend upon the entire concatenated key for its existence.

Steps are :

① Write each key component on a seperate line

② Assign corresponding dependent attributes.

→ Therefore, 2NF has following characteristics:

- It is in 1NF.
- No partial dependencies

(1,2,3) (1,4)
R_1 & R_2

Eg :

Roll no	Name	Subject	Age

Eg STUDENT

Subject	Lecturer	Semester
Computer	Anshika	Sem 1
Computer	John	Sem 1
Math	John	Sem 1
Math	Akash	Sem 2
Chemistry	Praveen	Sem 1

suppose a new semester is added and we don't know the subject and lecturers ⟶ NULL.

Here all 3 columns together act as primary key so we cannot leave other two columns blank.

P_1

Semester	Subject
Sem 1	Computer
Sem 1	Math
Sem 1	Chemistry
Sem 2	Praveen

P_2

Subject	Lecturer
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P_3

Semester	Lecturer
Sem 1	Anshika
Sem 1	John
Sem 1	John
Sem 2	Akash
Sem 1	Praveen

Unit 4 end

Closure set of Attributes and Irreducible set of FD:

It is a linear algorithm.
It is also known as complete set of func. dependency.

$[\alpha \longrightarrow \beta]$ where α is set of attributes which are super key and we need to find the set of attributes which is functionally determined by α.

Q. $R(A, B, C)$

$A \rightarrow B$
$B \rightarrow C$

find closure for all?
$A^+ = ABC$
$B^+ = BC$
$C^+ = C$

In an irreducible set of FD, we reduce all the transaction to lessen the waste of set of attributes.

⇒ Irreducible set of FD is also known as Canonanical cover / canonanical set / form.

Steps are:

① Decompose all possible right side attribute only.

② Find closure of all transaction after decomposition of attribute including and excluding the same transaction.

③ If any changes are done in closure set then we cannot ignore the transaction, otherwise we ignore the transaction if closure is same in both the cases.

④ Follow this process in all decomposed transactions and then check for all closure, If their closure is different then transac. is in reducible form otherwise follow the steps again.

Q. $R(W, X, Y, Z)$ Find the irreducible set of FD.

$X \longrightarrow W$
$WZ \longrightarrow XY$
$Y \longrightarrow WXZ$

Sol: Step 1:
$X \longrightarrow W$
$WZ \rightarrow X$
$WZ \rightarrow Y$
$Y \rightarrow W$
$Y \rightarrow X$
$Y \rightarrow Z$

Natural join

- If we join R_1 & R_2 on equal condition it is called natural join or equijoin.

Eg

$R_1 . Regno = R_2 . Regno$.

Outer join :

It is an extension of natural join to deal with missing values of relation.

It is of 3 types.

① **Left outer join:** All the tuples of the left relation appears in output and the mismatching values of R_2 are filled with NULL.

Left outer join is equal to natural join + $\frac{\text{mismatch}}{\text{extra tuple of } R_1}$

② **Right outer join:**

Right outer join = natural join + $\frac{\text{mismatch}}{\text{extra tuple of } R_2}$

③ **Full outer join:**

Full outer join = left outer join ∪ Right outer join

Difference

Calculus	Relational Algebra
① Relational calculus is a declarative language	① It is a procedural language.
② Rel. calculus means "what" result we must obtain.	② Rel. algebra means "how" to obtain the result.
③ In Rel. calculus the order is not specified	③ In Rel. Algebra the order is specified in which operations have to be performed.
④ Rel. calculus can be domain dependent	④ Rel. algebra is independent on domain
⑤ Rel. calculus is not nearer to programming languages	⑤ It is nearer to prog. lang.
⑥ It is denoted by "$\{(t) \mid P(t)\}$".	⑥ Basic operations are SELECT(σ), π, $\cup$, SET DIFFRENCE, CARTESIAN PRODUCT, RENAME(ρ), etc.

6 marks/10

Pitfalls in Relational Database Design

① Repetition of Information

② Inability to represent certain information

③ Design goal for rel. dB to be achieved.

④ Avoid redundant data.

⑤ Ensure that relationship among attributes are represented

⑥ Facilitate checking of updates for violation of integrity constraints

⑦ Poor Design / Planning

⑧ Ignoring normalization

⑨ Poor naming standards.

⑩ Lack of Documentation

⑪ One table to hold all domain values.

⑫ Ignoring frequency or purpose of data

⑬ Insufficient Indexing.

③ Third Normal Form (3NF):

→ Every non prime attribute of table must be dependent on primary key.

→ Transitive FD must be removed.

Steps for conversion to 3NF are:

① Identify the dependent attributes.

② Remove the dependent attributes from transitive dependencies.

→ 3NF satisfies following characteristics:

- It is in 2NF.
- It contains no transitive dependencies

Eg:

Stud-id	Stud-Name	DOB	Street	City	State	Zipcode

In this table "stud-id" is p.k but street, City & state depend upon zipcode.

R1

Stud-id	Name	DOB	Zipcode

Zipcode	Street	City	State

Step 2:

$x^+ = xw$

Normalization :

* Problems caused by Redundancy

1. Redundant storage
2. Update anomalies
3. Insertion anomalies
4. Deletion anomalies

* What is Normalization :

→ Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like insertion, update, and deletion anomalies.

→ It is a multistep process that puts data into tabular form by removing the duplicated data from relation tables.

→ Normalization is mainly used for two purpose: → eliminating useless data
→ Ensuring data dependencies

② Cascadeless Schedules: For each pair of transac. T_i and T_j, such that T_j reads the data item previously written by T_i, the commit operation of T_i appears before read operation of T_j

→ Transactions that are dependent on other transactions should be rolled back.

→ This phenominon in which a single transaction failure leads to series of transaction roll back is called Cascading roll back.

*● Concurrency Control:

Measures are:

① Lock based protocol:
  → Shared lock, exclusive lock

② Simplilistic lock protokol:

③ Preclaiming lock protocol

④ Basic 2PL (phase lock)
  → growing phase
  → Shrinking phase

⑤ Conservative 2PL

⑥ Strict 2PL

⑦ Graph based protocol / Tree based Protocol.

⑧ Timestamp ordering protocol.
  → Basic timestamp ordering
  → strict timestamp ordering.

⑨ Deadlock - Prevention mechanism includes 2 schemes.
  (a) Wait die scheme
  (b) Wound weight scheme

→ Deadlock detection is performed with the help of wait for graph.

→ Algorithms for detecting deadlocks are:
  (a) Wait for graph
  (b) Banker's algorithm
  (c) Resource allocation graph
  (d) Detection by system modeling
  (e) Time stamping

Storage Structure

Storage types are:

1. Volatile storage
2. Non volatile "
3. Stable storage - RAID

Stable storage implimentation is done

- successful completion
  Partial failure
  Total failure

Data Access:

Recovery & Atomicity:

1. Log based recovery
2. Recovery with concurrent transac^n
3. Checkpoint
4. Differed dB modification
5. Immediate dB modification

* Boyce & Codd Normal form (BCNF)

→ It is higher version of 3NF

→ Deals with certain type of anomaly that is not handled by 3NF.

→ A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

→ For a table to be in BCNF following conditions must be satisfied

① 'R' must be in 3NF.

② for each FD $(X \rightarrow Y)$, X should be super key

Consider a relation R(A,B,C,D)
$A \rightarrow BCD$, $BC \rightarrow AD$, $D \rightarrow B$.

Above relationship is in 3NF and keys are: A and BC.

Functional dependency $A \rightarrow BCD$, A is super key.

$BC \rightarrow AD$, BC is also a key but in $D \rightarrow B$, D is not a key.

we can break the relationship 'R' into R_1 & R_2
R_1(A, D, C)
R_2(D, B)

Eg:

* Projection Operation:

→ The projection operation allows to produce or list the relation. This is represented by a greek letter (π) pie.

Eg: $\pi_{loan\text{-}no,\, amount}(Loan)$

→ list 2 columns.

* Set operations

The SQL operations UNION, INTERSECT and EXCEPT on relations correspond to the relational algebra operations $\cup$, $\cap$ and $-$.

* UNION operation:

→ It automatically eliminates duplicate. To retain all duplicates UNION ALL is used

Eg:
```
SELECT CN from depositor
UNION
SELECT C-N from borrower
```

Eg:
```
SELECT CN from depositor
UNION ALL
SELECT C-N from borrower
```

* INTERSECT operations

The INTERSECT operation automatically eliminate duplicates. To retain all duplicates INTERSECT ALL is used.

```
SELECT CN from borrower
INTERSECT
SELECT C-N from depositor
```

```
SELECT CN from borrower
INTERSECT ALL
SELECT C-N from depositor.
```

* EXCEPT operation:

```
SELECT CN from depositor
EXCEPT
SELECT C-N from borrower
```

```
SELECT CN from depositor
EXCEPT ALL
SELECT C-N from borrower.
```

→ Without normalization handling and updating the database without facing data loss is difficult.

→ Insertion, updation & deletion anomalies are very frequent if data is not normalized.

→ Normalization rule can be divided into following normal form:

① 1st Normal form (1NF):

→ No two rows of data must contain repeating group of info. ie each set of column must have unique value.

→ Each table should be organized into rows and should have a primary key.

Steps for conversion to 1NF are:

① Eliminate repeating groups

② Identify the primary key.

③ Identify all dependencies.

Therefore,

→ 1NF describes the tabular format in which:

- all key attributes are defined
- No repeating groups
- All attributes are dependent on primary key

Eg:

P.K

Roll no	Name	Subject
1	Rajat	Bio, Maths
2	Antas	DBMS, CD
3	Himanshu	DAA, DSA
4	Harsh	English, Maths

$\xrightarrow{1NF}$

Roll no.	Name	Subject
1	Rajat	Bio
1	Rajat	Maths
2	Antas	DBMS
2	Antas	CD
3	Himanshu	DAA
3	Himanshu	DSA
4	Harsh	English
4	Harsh	Maths.

* Join :

Join operation combines a relation R_1 & R_2 wrt a condition. It is denoted by $\bowtie$.

The different type of join operation are:

1. Theta Join
2. Natural Join
3. Outer Join – It is further classified into following types:
   1. Left outer join
   2. Right outer join
   3. Full outer join

* Theta Join

If we join R_1 & R_2 other than the equal to condition, it is called **theta Join** or **non-equijoin**.

R_1

RegNo	Branch	Section
1	CSE	A
2	ECE	B
3	Civil	A
4	IT	B
5	IT	A

R_2

Name	RegNo
Bhanu	2
Priya	4

Q. Perform Theta join with condition $R_1.RegNo > R_2.RegNo$.

Sol:

Cartesian Product

Reg (R_1)	B	S	N	R (R_2)	
1	CSE	A	Bhu	2	×
1	CSE	A	Priya	4	×
2	ECE	B	Bhanu	2	×
2	ECE	B	Priya	4	×
3	Civil	A	Bhanu	2	✓
3	Civil	A	Priya	4	×
4	IT	B	Bhanu	2	✓
4	IT	B	Priya	4	×
5	IT	A	Bhanu	2	✓
5	IT	A	Priya	4	✓

Reg	Branch	Sec.	Name
3	Civil	A	Bhanu
4	IT	B	Bhanu
5	IT	A	Bhanu
5	IT	A	Priya

→ Advantages of Deadlock detection algo are :

(i) Improved system stability
(ii) Better resource utilization
(iii) Easy implimentation

→ <u>Disadv</u> :

(i) Performance overhead
(ii) Complexity
(iii) False positives & negetives
(iv) Tradeoffs b/w performance, complexity, accuracy and risk.

(10) Multiple granuality protocol

(11) Multiversion protocol
(12) Intention mode lock → intention shared
→ " exclusive
→ shared & intention exclusive

Pessimistic approach v/s Optimistic approach

Pessimistic approach	Optimistic approach
→ It locks record so that selected record for update will not be changed meantime by another user	→ It doesnot log the record as it ensures record was not changed in time b/w select and submit operation
→ Conflict b/w transac. are large	→ Less.
→ synchronization of transactions is conducted in start phase of lifecycle of execution of transac^n.	→ in later phase.
→ Simple in designing & programming	→ Complex
→ Higher storage cost	→ Low as compared
→ Lower degree of concurrency.	→ Higher degree.
→ Approach is useful where more transac^n conflict	→ for fewer transac^n conflict.
→ The flow of transac^n phases are:	→ The flow of transac^n phases are:

Database Recovery Management

① Failure Classification

(i) Transac^n failure

(a) Logical error
(b) System error

(ii) System crash

(iii) Disk failure

Log recovery

DB recovery Techniques:

① Rollback/Undo Recovery Technique

② Commit Redo recovery Technique

→ There are two major techniques from recovery from non-catastrophic transac^n.

① Differed update/No undo/redo algo.

② Immediate update

③ Cashing/Buffing

④ Shadow paging

⑤ Backward recovery

⑥ Forward recovery

Some of the backup techniques are:

① Full dB backup
② Differential backup
③ Transaction log backup

Log based recovery:

→ Undo ~~Redo~~
→ Redo
→ Differed modification tech
→ Immediate " "
→ Use of checkpoints

Concurrency problem in DBMS transac^n

① Temporary update prob^m
② Incorrect summary probl^m
③ Lost update probl^m
④ Unrepeatable read probl^m
⑤ Phantom read probl^m
⑥ Dirty read problem

Multivalued Dependency:

- It occurs whenever two seperate attributes in a given table happens to be independent of each other.

Eg: Car model → Manufacturing Month

Car mode → colour

Colour & Manu. Month are dependent on car model but they are independent of each other.
Therefore we can call both of these columns as multivalued.

MD means that for some single value of attribute 'x' multiple values of attribute 'y' can exist.

Shown as: $X \to\to Y$

attr. name

Eg:

		a	b	c
		Name	Project	Hobby
Tuples	P_1	Rita	MS	Reading
	P_2	Rita	Oracle	
	P_3	Rita	MS	
	P_4	Rita	Oracle	

Here project and hobby are multivalued attributes bcz they contain different values for the same name ie Rita.

Conditions for MVD according to above example are:

An attribute 'x' can define another attribute 'y' if a legal relation 'Q', for all the pair of tuples P_1 and P_2 include such that
1. $P_1[a] = P_2[a]$

Then there exist P_3 and P_4 such that
2. $P_1[a] = P_2[a] = P_3[a] = P_4[a]$
3. $P_1[b] = P_3[b]$, $P_2[b] = P_4[b]$
4. $P_1 = P_4$, $P_2 = P_3$

Thus MVD exist in this case.

Fourth Normal form: (4NF)

→ A relation will be in 4NF, if it is in BCNF and has no multivalued depend[ency]

For a dependency $A \rightarrow B$, if for a sin[gle] value of A, multiple values of B exist then the relation will be a multivalued dependency.

Eg STUDENT

Student_ID	Course	Hobby

The given student table is in 3NF but Course and hobby are two independ[ent] attributes, in the relation a student can have more than one course and hobby. So ~~entity~~ MVD exist on student_ID which leads to repetition of data

Stud-Course

Stud_ID	Course

Student Hobby

Stud_ID	Hobby

Join Dependency

It is generalization of MVD.

If join of R_1 & R_2 over C is equal to relation R, then JD exist.

→ R_1 and R_2 are lossless decomposition of R

→ A JD is said to hold over a relation R and its attribute if $R_1, R_2, \ldots R_n$ is a lossless join decomposition.

Fifth Normal form (5NF):

→ A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless.

→ 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid repetition of data.

→ 5NF is also known as Project join Normal form (PJ/NF)

Domain Relational Calculus (DRC):

It is a non-procedural query language equivalent to TRC.

DRC provides only the description of the query but does not provide the method to solve it.

It is expressed as

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

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

The predicate calculus represents:

1) set of all comparison operator
2) set of connectives live $\wedge, \vee, \neg$
3) set of Quantifiers ($\exists$, $\forall$, $\exists$)

Q Find the loanNo., Branch, Amount of loan (L, B, A) of >= 100 amount that belong to relation Loan.

Ans:

$$\{ \langle L, B, A \rangle \mid \langle L, B, A \rangle \in Loan \wedge (A >= 100) \}$$

Q Find the loan no. for each loan of an amt. >= 150 that belong to relation Loan.

$$\{ \langle l \rangle \mid \exists b, a (\langle l, b, a \rangle \in Loan \wedge a >= 150) \}$$

20/4/23

Unit 5

Transaction Management & Concurrency control.

Concepts

① Transaction concepts & properties.
→ ACID (prop)

② Transaction states
→ Active
→ Partially commited
→ Failed
→ Aborted
→ commited

③ Concurrent execution
→ improved throughput & resource utilization
→ reduced waiting time

Serializability - need of serializability is:
① Lost update
② Dirty read
③ Unrepeatable read

Conflict & view Serializability

Conflict

I_i = read (Q), I_j = read (Q)
→ order doesnot matter

I_i = read (Q), I_j = write(Q)
→ order matters

I_i = write(Q), I_j = read (Q)
→ order matters

I_i = write (Q), I_j = write (Q)
→ order doesnot matter.

View

Consider 2 schedules, S and S' where the same set of transactions participate in both schedules.

The schedules S & S' are said to be view equivalent if 3 conditions are met:

* Types of FDs are:

① **Trivial Func. Dependency**: If $X \to Y$ and Y is subset of X.

② **Non-trivial FD**: If $X \to Y$ and Y is not subset of X.

③ **Multivalued FD**: If $A \to (B,C)$ and there exist no functional dependency b/w B and C.

④ **Transitive FD**: If $A \to B$, $B \to C$ then $A \to C$.

* **Attribute Closure**: of an attribute set can be defined as set of attributes which can be functionally determined from it.

Steps are:

① Add elements of attribute set to the result set

② Recursively add elements to the result set which can be functionally determined from the elements of the result set.

③ If attribute closure of an attr. set contains all attribute of relation, the attribute set will be Superkey of relation.

④ If no subset of this attribute set can functionally determine all attributes of relation, the set will be candidate key.

Eg: $R(A, B, C, D, E, F)$

FD:
$E \to A$
$E \to D$
$A \to C$
$A \to D$
$AE \to F$
$AG \to K$

Find E^+ or E.

$E^+ = E$
$= EA$ { for $E \to A$ add A }
$= EAD$ { for $E \to D$ add D }
$= EADC$ { for $A \to C$ add C }
$= EADC$ { for $A \to D$, D is already added }
$= EADCF$ { for $AE \to F$ add F }
$= EADCF$ { for $AG \to K$, don't add K as $AG \not\subset D^+$ }

(14) Lack of Testing.

Relational Decomposition: / Decomp. bad Scheme

- When a relation in the relational model is not in appropriate normal form then the decomposition of relation is required.
- It may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like inconsistency, anomalies and redundancy.

* Types of Decomposition:

① Lossless Decomposition: If natural join of all the decomposition gives the original relation.

② Dependency preserving: If relation R is decomposed into relation R_1 and R_2, then dependencies of R must either be a part of R_1 or R_2 or must be derivable from functional dependencies of R_1 or R_2.

* Types of Functional dependency in DBMS

A functional dependency is a constraint that specifies the relationship b/w two set of attributes where one set can accurately determine the value of other set, is denoted as $X \rightarrow Y$

, where X is called Determinant and Y is is dependent.

* Armstrong properties of FD:

① Reflexivity: If Y is a subset of X then X and Y holds.

② Argumentation: If $X \rightarrow Y$ is a valid dependency, then $XZ \rightarrow YZ$ is also valid.

③ Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.