OS - DBMS

ACID Properties

Atomicity :- all or none transaction

Consistency :- DB must be before & after the transaction

Isolation :- one trans. not impact on other.

Durability :- Recovery after system fail.

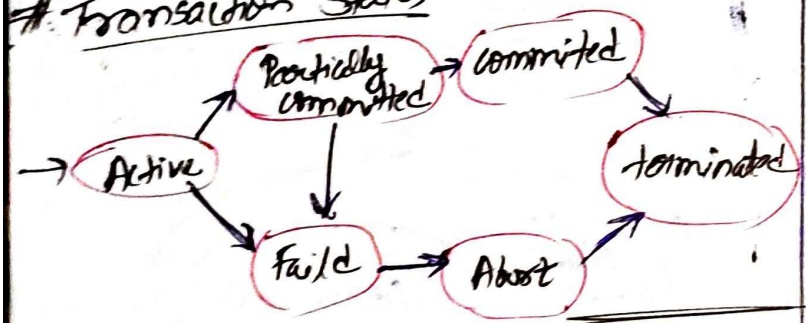
Mangers for ACID

A → Transaction control manager → Responsible for atomicity

C → DBMS and application Programs → Responsible for consistency

I → Concurrency control manager → Responsible for Isolation

Transaction States



Problems in concurrent execution

① Dirty Read :- Reading the data written by an uncommitted transaction. (write → Read conflict)

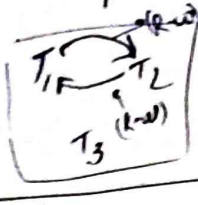
② Unrepeatable Read :- Read many times (Read → write conflict)

③ Overwriting Uncommitted data :- update performed by T_1, lost due to update made by T_2 (write → write conflict)

④ Phantom read :- If T_1 repeates the same query again, then T_1 will result in a row that not present previous

Note :- SELECT is executed twice, second time we get additional row.

$T_2 \to T_1$ [using] then can't be ① view serializable
$T_1 \to T_2$ [using]



NO view serializable

Note

- conflict serial → view serial

View serial — blind write → not conflict serial

Note if S1 and S2 have same precedence graph.

then S1 [conflict serial] → S2 [conflict serial]

(or) S2 is conflict equivalent to S1

Strict 2PL (S2PL)

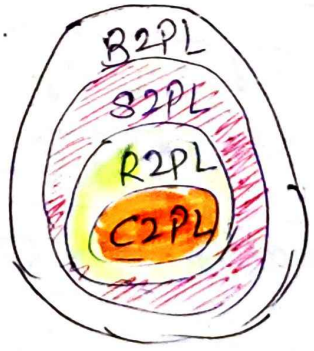
not release any exclusive lock until it commits or aborts

Rigorous 2PL (R2PL) (shared & Excl)

not release any lock until it commits or abort

Conservative 2PL (C2PL)

all the locks acquired will not be released until transaction commits.

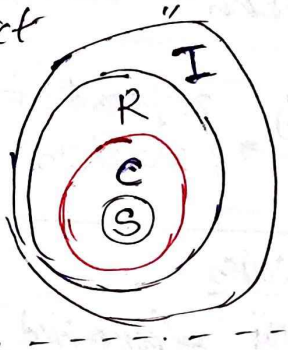


Basic 2PL (B2PL) :- lock acquire in growing phase. release lock in shrinking phase.

B2PL → conflict serializable

	Basic 2PL	Strict 2PL	Rigorous 2PL	Conservative 2PL	Basic TS	Thomas TS	Strict TS
Guaranteed Serializability	Yes	Yes	Yes	Yes	Yes (conflict)	Yes (view)	Yes (conflict)
Guaranteed Strict Recoverable	NO	Yes	Yes	NO	NO	NO	Yes
Free from deadlock	NO	NO	NO	Yes	Yes	Yes	Yes
Free from starvation	NO	NO	NO	NO	NO	NO	NO

I = Irrecoverable Schedule
R = Recoverable "
C = cascadeless "
S = strict "



Irrecoverable schedule:

T_1	T_2
R(A)	
W(A)	
⋮	
	R(A)
	commit
fail	

not possible to roll back to T_2.

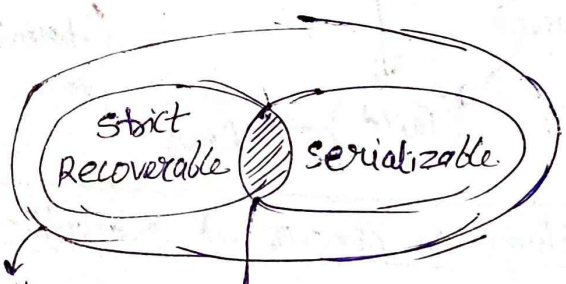
[$W_1(A) \rightarrow R_2(A)$ exist and, T_2 commit before then Irrecov.]

Recoverable schedule

T_i	T_j
R(A)	
W(A)	
⋮	
commit	R(A)
	W(A)
	commit

if T_i fail the T_j will be recoverable.

Recoverable
commit operation of T_i appear before commit of T_j for $W_i \rightarrow R_i$



neither Strict Recov. & Serializable

(i) S is strict recoverable
(ii) S is serializable

Cascadeless Recoverable schedule

T_i	T_j
R(A)	
W(A)	
⋮	
commit	
	R(A)
	commit

commit of T_i appear before the read of T_j

cascadeless → Recoverable

Strict Recoverable Schedule

T_1	T_j
W(A)	
commit or Rollback	
	R(A) (or) W(A)

if T_i updates A, T_j not allowed to R(A) or W(A) until commit or roll back of T_i.

Note
① if NO $W_i \rightarrow R_j$ then Recoverable
② if NO $W_i \rightarrow R_j$ then T_i & T_j Independent so, no cascading

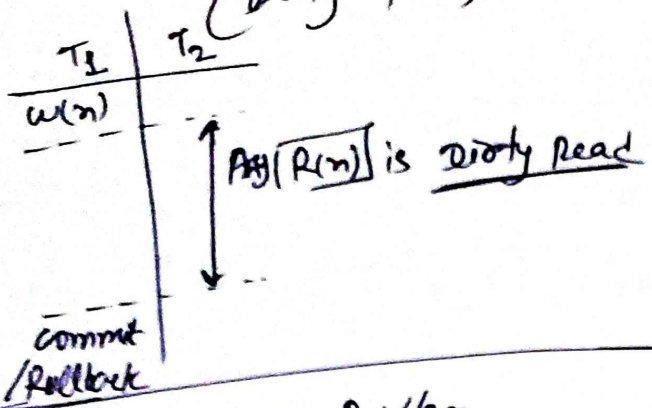
T_1	T_j
W(A)	
⋮	
	R(A)
commit	commit

not Recov.

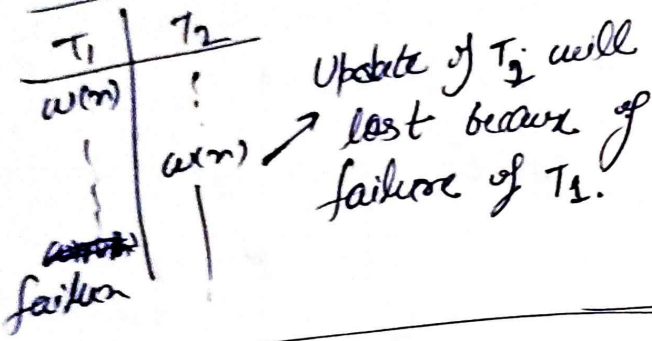
T_i	T_i
W(A)	
⋮	
Comm	R(A)
	comm

Recov.

Uncommitted Read (Dirty Read)



Lost update Problem



Update of T_2 will lost because of failure of T_1.

not conflict Serializable → Not 2PL

About 'Exist'.

Exist 'x' → TRUE if not Empty set
→ False if Empty set.

e.g. Exist 0, Exist 1, → true
because 0, 1 are not empty set.

Hashing

$h_1(n)$ → collision → $(h_1(n) + 1 \cdot h_2(n)) \bmod n$ → collision

(open addressing, single hash f^n)

$(h_1(n) + i\, h_2(n)) \bmod n$

Double hashing ↑ increase the value of i on each collision.

"i" is called probe.

Normalization

F = Function dependency set for R

X must be in all CK, $X \subseteq R$

if $[X]^+ = R$ then X is only CK

2NF

$\alpha \rightarrow \beta$

(or) (i) Non-Prime (ii) CK/SK

(i) Anything

Not 2NF (or)

(or) Prime → Prime
no prime → prime
no prime → no prime

$\alpha \rightarrow \beta$

(i) Prime (ii) Non-prime

---

if primekey == CK for R (imp)

then R should be 2NF

(imp) If R don't have non-prime

then R is in 2NF or 3NF

3NF

$\alpha \rightarrow \beta$

(i) Superkey (or) (ii) prime

Not 3NF

$\alpha \rightarrow \beta$

(i) Not prime (ii) Non prime

Extraneous attribute

$XY \rightarrow Z$, for $R(A, B .. X, Y, Z)$

if $[X]^+ = \{ \ldots Y, Z \ldots \}$ | Y or $Z \in [X]^+$

then $X \rightarrow Z$ and Y is extra

BCNF

$\alpha \rightarrow \beta$

(or) (i) Superkey (ii) CK

---

dependency preserve in decomposition (imp)

$\alpha \rightarrow \beta$

$R_1(A, B..)$
$R_2(..)$
$R_n(..)$

$Z = \alpha$

for ($i = 1$ to n)

$Z = Z \cup \{ R_i \cap [R_i \cap Z]^+ \}$

if $\beta \in Z$, then $\alpha \rightarrow \beta$ preserve

else not preserve

Note 3NF guarantee

(i) lossless
(ii) dependency preserving
(iii) All dependency must be in 3NF

Can't guarantee in BCNF

Note BCNF guarantee

(i) always lossless
(ii) may or may not Dep. Pre.

BCNF decomposition

(i) BCNF $R_1 R_2 .. R_n$
(ii) lossless
(iii) Dependency preservation may or may not

---

lossless decomposition

	A	B	C	D
R_1	x	x		
R_2		x		x
R_3	x	x		

→ find $B \rightarrow \beta$ and fill the cell and repeat till any of row become full.

if any row is full then lossless

else not

MVD

$\alpha \rightarrow\rightarrow \beta$

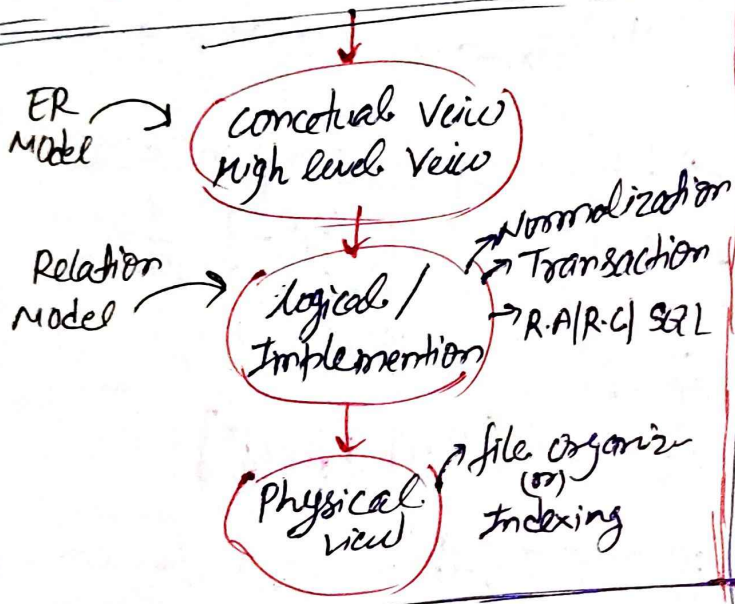
then $\alpha \rightarrow\rightarrow \beta \mid \alpha \rightarrow\rightarrow R-\beta$

find the $\beta \times R-\beta$ for all instance of $\alpha \in \{\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_n\}$

Note:
if $\alpha + \beta = R$ then that MVD always hold.

Primary key — any ck → can't NULL

Alternate key — All CK → can be NULL except PK



Degree — No. of entities Participate in a relationship.

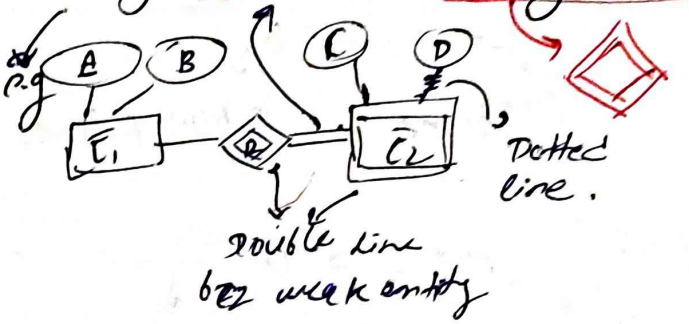
Cardinality Ratio — Maximum no. of times entity can participate in a Relation. { =1 → 1, >1 → M }

Participation :- Minimum no. of (or) Existance times entity can participate in a Relationship is known as existance.

Strong entity — Attribute determine by own PK.

Weak Entity — can't determine. don't have PK.

Note: Participation of weak entity always total with Identity Relationship

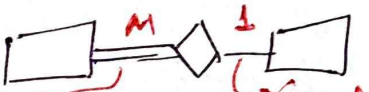


If T_1 and T_2 with 'n' and 'm' operation.

$$\text{No. of co-current schedule} = \frac{(n+m)!}{(n!)(m!)}$$

Probe sequence $= (h_1(n) + i\, h_2(n)) \% m$.

where 'i' is probe
'n' is key
$h_1(n)$ & $h_2(n)$ are hash f^n.



Total Participation

Partial participation

Min-Max Representation (min, max)

(1, 1) ↑ (cardinality)

(0, M) ↑ (participation)

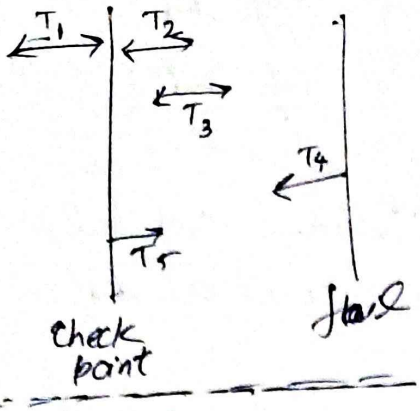
---

when participate is non zero then participate of an entity is total participation.

⇒ else Partial participate.

Cardinality & Participation → structure constraints.

Recovery System



REDO → T_2 T_3 T_5

UNDO - T_4

Note: the transaction b/w checkpoint and fail will be **REDO**.
and who touch of fail. will be **UNDO**.

B-Tree

Root node = 2 to n children

Internal Node = $\lceil n/2 \rceil$ to n children

Block Size = $P \times (BP) + (P-1)(key + RP)$

∴ P = order of node (no. of children)
BP = Size of Block pointer
key = Size of key field
RP = Size of Record pointer.

~~min. no. of nodes = $1 + \left(\frac{2\left[\lceil P/2 \rceil^{h-1} - 1\right]}{\lceil P/2 \rceil - 1}\right)$~~

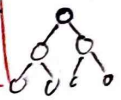
~~max no. of nodes = $\frac{p^{h+1} - 1}{p-1}$~~

height of root is 1.

of Node & record in B-TREE

m = max no. of children of node
c = ceil(m/2)
h = height of tree ; h of root is 1.
if h=0, then replace h → h+1

min record or keys = $2(c^{h-1} - 1) + 1$



min nodes = $\frac{2(c^{h-1} - 1)}{c-1} + 1$

(no. of min key in node)

max record or keys = $m^h - 1$

max nodes = $\frac{m^h - 1}{m-1}$

(no. of max key in node)

B+ Tree

Non-leaf node
$P \times BP + (P-1) \times key \leq$ Block size

Leaf node
$(P-1)(key + RP) + BP \leq$ Block size

B+ Tree formula

max record or key = $m^h - m^{h-1}$

min record or key = $2(c^{h-1} - c^{h-2})$

Short trick for $(a \times b) \bmod c$

Step ① convert a & b in c form

$$((c \pm x)(c \pm y)) \bmod c$$

Step ② find $z = (\pm x) \times (\pm y)$

$$d = z \bmod c$$

Step ③ if d is negative

than $c - d$

if d is positive

then $|d|$

File structure

Avg. no. of blocks accessed by

linear search = $\lceil \frac{\#\text{ of record blocks}}{2} \rceil$

binary search = $\lceil \log_2 \#\text{ of record blocks} \rceil$

Indexing Types

(single level Index)

(a) primary Index (sparse) : ordered with key field

(b) clustered Index (sparse) : ordered with non-key field.

(c) secondary Index (dense) : ordered with either key or non-key.

(multilevel Index)

(a) Indexed sequential access method

(b) B-tree Index

(c) B^+-Tree Index

ER Model ⇒ Relational Model

Strong Entity :- sep table

Weak Entity :- sep table

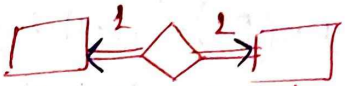
Attribute

- Multivalued :- sep table
- composite :- sep table

Relationship

- one-one :- No. sep table
- one-many :- No. sep table
- many-many :- sep table

case :



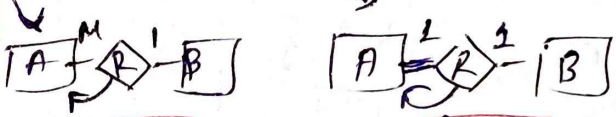
① one-one Relation (or) ~~many~~ ~~one~~

② total participate both side

1 table

Note: Relationship attribute will shift to (Many) Relation table

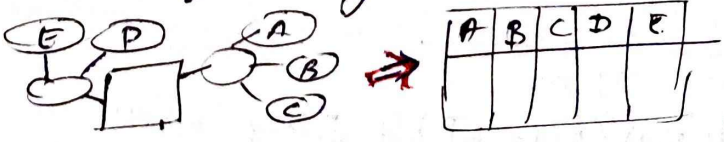
Note: Relationship attribute will ~~shift~~ to total participate.



must priority

less priority

Note: A strong Entity with any number of composite attributes will require only one table.



ER Diagram

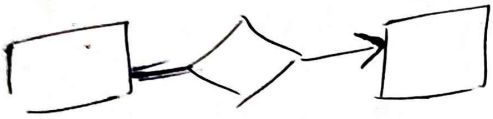
Multi-valued attribute

derived attribute

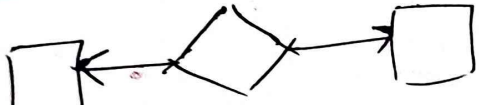
total participation

entity, weak, Relation

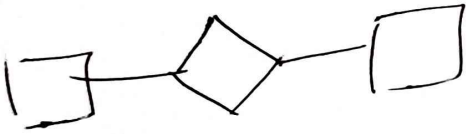
one-to-many



many-to-one



one-to-one



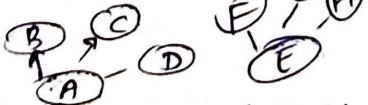
many-to-many

Conversion Rules of strong Entity set to Relational table

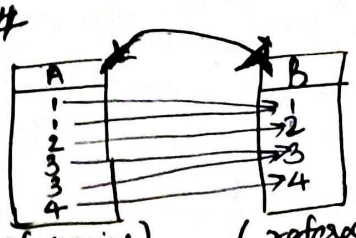
Rule 1: Strong Entity → separate table

Rule 2: simple attribute → add column in table.

Rule 3: Composite attribute → add B, C, D, E, F, G, H column, don't add A in table



#



A → B

(referencing) A : (refered) foreign key

B : Primary key.

#Note :-

⇒ A can be duplicate.

⇒ B can only primary

Operations

referencing (A)	refered (B)
Insert Inserting value must be in B. else violation.	just Insert
Delete just delete	(i) NO action (ii) cascading :- Delete all in A. (iii) NULL :- set Null in A.
Update Updating value must be in B else violation	(i) NO action (ii) cascade :- update all in A. (iii) Null :- set Null in A.

#Note :- A → B

A : violation I & U

B : cascading, Null D, U

#Rule 4

multivalued attribute → separate table (primary key)

eg (A) →

#Rule 5

derived attribute → don't add column

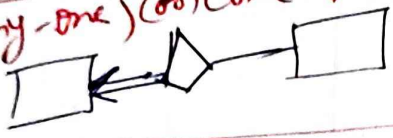
Total Participation

Partial Participation on both side

		merge
one → many	= 2 table	many
many → one	= 2 table	many
one → one	= 2 table	any one
many → many	= 3 table	sep table

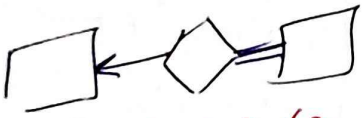
Full Participation on 'one' side

(many-one) (or) (one-many)



1 table

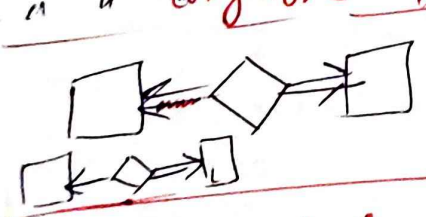
" " any "many".



2 table

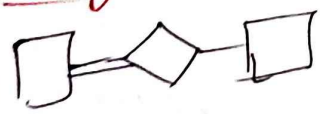
(many → one) (or) (one to many)

" " any "one" for one → one



1 table

" "any "many" for many → many.



2 table

Full Participation on both side

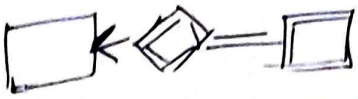
1:1
1:M
M:1
M:N

} 1 table

Note about weak entity set

mapping b/w identifying entity set & weak entity set must be 1:M

one → many.



weak Entity set and Multivalued attributes allowed in ER model but not in RDBMS table

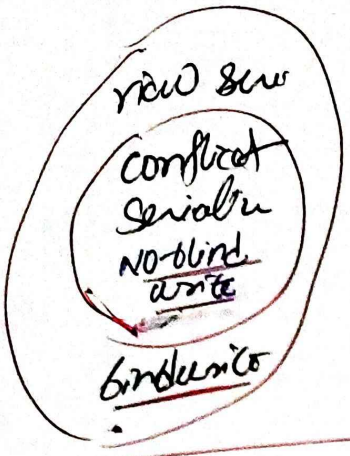
Note

one-one } → 1 Table
one-many }

many → one } → 2 table.
many → many }

any → any } → 1 table

#SQL



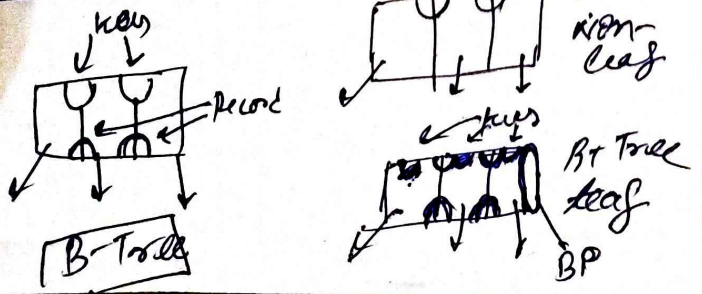
T	T
X(A)	X(A) or S(A)
: strict 2PL	: Rigorous 2PL
commit U(A)	commit U(A)

Guarantee Serializability ⇒ All

Guarantee Strict Recoverable ⇒ Strict 2PL, Rigorous 2PL

free from Deadlock ⇒ C2PL & All TS

free from Starvation ⇒ NO one



Procedural Qu Lang.
how to access
eg Relational Algebra

Non-Procedural Qu Lang
what to access
eg Relational Calculus

Relational Algebra
$\pi, \sigma, \times, \cup, -, \rho$

$\div$ (or) $/ = \pi_{AB}(E) / \pi_B(C)$

$\pi_{A(E)} - \pi_A[\pi_A(E) \times \pi_B(C) - \pi_{AB}(E)]$

Data Definition Lang (DLL)
① Create ② Drop ③ Alter
used to definite modify of table.

Data Manipulation Lan (DML)
used to modify or access record.
① Insert ② Delete ③ Update ④ Select

Data Control Lang. (DLL)
used for transaction control.
① lock ② unlock ③ commit
④ rollback

used for role access.
① Grant access ② Revoke access.

Execution flow

from → cross product
where → selection (or) condition
Group By
Having
Select
distinct
Order By

WITH clause

used to create sub-query result.

e.g WITH temp(a,b) as
(select E.a F.b from E,F...)

Note: ① Every Query of 'Having' can be re-written into 'WHERE'.

② Co-related Nested query inner Query allowed in 'where', 'Having' clause of outer Query.

---

func^{n} in Nested Query

① IN / NOT IN
② ANY
③ ALL
} Best for non-co-related Query

④ EXISTS / NOT EXISTS
} Best for co-related Query.

#Note: ① 'IN' ≡ '=ANY' [single attribute]

② 'IN' ≠ '=ANY' [more than one attribute]

③ 'NOT IN' ≡ '<> ALL'