DBMS – Arefin Sir

Relational algebra (Conceptual procedure of SQL)

* Take input from table and give output to table.

Selection (σ): row wise operation

$\sigma_p(r)$

σ → selection

p → logic formula

r → Relation (Table)

$\sigma_{age>18}(student)$ → Show those rows are have student age is greater than 18.

Projection (π): column wise operation

$\pi_{a_1, a_2 \ldots a_n}(r)$

π → projection

a_i → propositional logic

r → Relation (The table)

Q: Fetch ID, Name & Board data from student table for those students whose Board = "Dhaka"

$\pi_{ID, Name, Board}(\sigma_{Board = "Dhaka"}(Student))$ → At first select all the student have board Dhaka, the projection their ID, Name, Board.

$\sigma_{Board="Dhaka"}(\pi_{ID,Name,Board}(Student))$ → First projection ID, Name, Board of all student then select whoes board is Dhaka.

Rename (ρ): Rename the output table of relation

$\rho(R_1, R)$

R_1 → New name table
R → Old table name

Q: Query to fetch data corresponding Name, Age & Board and rename the relation as StdInfo.

$\rho(StdInfo, \pi_{Name,Age,Board}(Student))$ → Make new table with name StdInfo.

Q: Fetch Data for ID, Name from Student for those whose Board is Dhaka. Rename table as BoardInfo also rename ID = D.ID, Name = D.Name

$$\rho(BoardInfo(D.ID, D.Name), \pi_{ID,Name}(\sigma_{Board="Dhaka"}(Student)))$$

Union (Binary union between two table)

$R_1 \cup R_2$

→ R_1 and R_2 attribute is same.

→ Have same data type.

→ Duplicate tuple auto eliminate

R_1

ID	Name
1	2
1	3
3	4

R_2

EE	EA
2	3
2	7
3	8

$R_1 \cup R_2$

ID	Name
2	3
1	2
3	4
2	7
3	8

→ Keep attribute name as left table.

Q: Union with two attribute of R_1 with two attribute of R_2 but have 3-attribut of R_2

$R_1 \cup \pi_{ID, Name}(R_2)$ → First Take all info of R_2 using projection then make union.

Set Intersection (Return Common row)

$R_1 \cap R_2$

→ R_2 and R_2 same attribute.

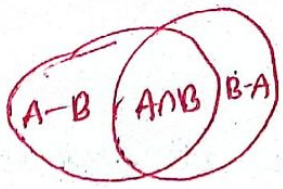
→ Same data type.

→ Return distinct attribute.

Set Difference (Those tuple in R_1 but not in R_2.

$R_1 - R_2$

→ Same as Intersection



π_{ID} (Tech Book) − π_{ID} (Non-tech-Book) → Show ID who have only takes tech book.

Cartesian Product (Merge all the columns from two relation)

$R_1 \times R_2$

* Totall Number of column: $R_1 + R_2$
* Totall no of record: $R_1 \times R_2$

Division Operator (Use when queries involve "all")

R_1 / R_2

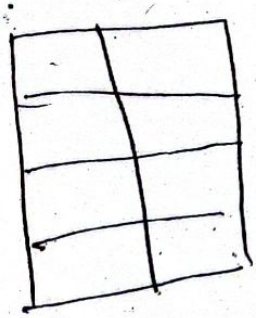
Q: Fetch those have taken all the course.

R_1

ID.	Course
1	C
2	Java
1	DBMS
3	C
1	Java
3	Java

R_2

Course
C
Java
DBMS



R_1/R_2

ID
2.

→ These attribute will remove who are matched.

Division operation using $(\pi, -, \times)$

~~$\pi_{ID}(R_1) - \pi_{ID}(R_2) \times R_2 - R_1$~~

$\pi_{ID}(R_1) - \pi_{ID}((\pi_{ID}(R_1) \times R_2) - R_1)$

~~$\pi_{ID}(R_2)$~~ → $(\pi_{ID, course}(R_2))$

Join (Return those combination of R_1, R_2 for a condition)

⇒ Inner join: only return the matching condition

i) Theta join → condition use any comparison operation.

ii) ~~Equal~~ Equi join → operation is "="

iii) Natural join → remove duplicate

⇒ Outer join: Both return matching combination and not matching condition.

i) Left outer join(⟕): Natural join + All in left.

ii) Right outer join(⟖): Natural join + All in right.

iii) Full outer join (⟗): All match record + all mismatch but empty cell is Null.

Natural Join(Those combination have common attribute)

$R_1 \bowtie R_2$

→ Only common record will come.

Functional Dependency:

A functional dependency $X \rightarrow Y$ holds, if for any two tuples t_1 and t_2 whenever-

if $t_1.x = t_2.x$

then must $t_1.y = t_2.y$

→ If two row have same value for X then they must have the same value for attribute Y.

Determinent attribute → X Determine Y ← Dependent Attribute

→ Here X, Y are one attribute or set of attribute.

→ If two X is same then corresponding y must be same.
If two X is not same no need to check for Y.

Types of F.D:

1. Trivial: Right side is subset of left side $\{A, B\} \rightarrow \{A\}$
$x \rightarrow y$ if $y \subseteq x$ (Always true) $\{S.ID, S.Roll\} \rightarrow S.Roll$
↳ subset

2. Non-trivial: Not subset $x \rightarrow y$ and $x \cap y = \emptyset$ $Y \notin X$
S.ID → S.Name

3. Multivalued: $x \twoheadrightarrow y$, for a single X there can be multiple value of Y. All value of y are independent. (Only 4NF)
$X \rightarrow \{Y, Z\}$ S.ID → S.Name, S.Roll

4. Transitive: $X \to Y$ then $Y \to Z$ so $X \to Z$ (3NF violation)

S.ID → S.Roll → S.Name

There occure composite $W,X \to Y$

(StudentID, CourseID) → Marks

Here marks depend on all parts of this composite key, so it is called full functional dependency.

There occure composite $X \to Y,Z$

EmployeID → (Name, Dept)

CarID → (Model, Company)

→ classification of different types of F.D help us to understood which normalization rules (2NF, 3NF, 4NF) we need to apply to reduce redandency and anomalies.

Armstrong's axioms: they are basic rules to split, convert find or derive required functional dependency in database. They also called অনুমান inference rules

Primary:

1. Reflexivity: If $Y \subseteq X$ then $X \to Y$ (Not must be proper subset) ↗ trivial
2. Transitivity: If $X \to Y$ and $Y \to Z$ then $X \to Z$

3. Augmentation: If $X \to Y$ then $X.Z \to Y.Z$

Secondary:

4. Union: If $X \to Y$ and $X \to Z$ then $X \to YZ$

5. Decomposition / splitting: If $X \to YZ$ the $X \to Y$ and $X \to Z$

6. Psudotransitivity: If $X \to Y$ and $YZ \to A$ then $XZ \to A$

7. Composition: If $X \to Y$ and $A \to B$ then $XA \to YB$

Attribute Closure: → Help to find all candidate key using f.d

Attribute closure for X is denoted as X^+ is mean the set of all attributes thoes are functionally determined by X using a given set of functional Dependency.

Superkey: set of attributes whoes closure contains all attributes of a given relation.

Candidat key: whoes proper subset is not a super key.

Super key is any set of attributes that uniquly identify value.
Candidate key is minimum attribute to uniquely identify value.
Primary key is one chosen candidate key used to identify values.

Find All Candidate Key:

$R=(A, B, C, D)$

$F.D = \{A \to B,\ B \to C,\ C \to A\}$

SK $ABCD^+ = \{A, B, C, D\}$ To find take all

SK $ACD^+ = \{A\ C\ D\ B\}$

SK $AD^+ = \{A\ D\ B\ C\}$

Minimize করার উদ্দেশ্য হল তাহলে কম subset এর CK বের করতে হবে।

Proper subset of AD^+ $A^+ = \{ABC\}$

$D^+ = \{D\}$

As, in proper subset closure there have not exist all attribute, so it is candidate key.

$C_K = AD$

So, prime attribute are $= \{A, D\}$

But in Prime attribute, they also remain at the right side of functional dependency. So, we need to check if more CK exist or not.

$C_K = AD$

SK $= CD$

Now, $C^+ = \{C, A, B\}$

$D^+ = \{D\}$

$\therefore C_K = CD$ $\therefore$ prime attribute $= \{A, D, C\}$

Again

$CK = CD$

$SK = BD$

Now, closure

$B^+ = \{B, C, A\}$

$D^+ = \{D\}$

$CK = BD$

Finally, prime attribute are $= \{A, B, C, B\}$

So, here candidate keys are $= \{AD, CD, BD\}$

→ যদি proper subset (closure) single attribute A, B, C, D এমন আসে তবে তাদের proper subset null হবে, তাই তারা তারা সবসময় candidate key. Like $A^+ = \{AS$ that is always CK.

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

$F.D = \{AB \to C,\ C \to DE,\ E \to F,\ D \to A,\ C \to B\}$

Soln:

$AB^+ = \{A\ B\ C\ D\ E\ F\}$

$ABD^+ = \{A\ B\ D\ E\ F\ C\}$

$AB^+ = \{A\ B\ F\ C\ D\ E\}$

$AB^+ = \{A, B, C, D, E, F\}$

$AB \to C \to E \to F$

Proper subset of AB^+ is

$A^+ = \{A\}$

$B^+ = \{B\}$

$CK = AB$

Prime attribute $= \{A, B\}$

Now $CK = AB$

CK = AB

SK = DB [D → A]

Closure of D+ = {D, A}

B+ = {B}

CK = DB

Prime attribute = {A, B, D}

CK = AB

SK = AC [C → B]

closure of C+ = {C D E F A B}

As C+ is super key, so AC+ must be a super key.

Now, check if A+, or C+ is candidate key or not, as after find a super key we need to check if there remain any nested candidate key. But if C+ will be a CK then AC will be a CK, we will avoid lower CK like A+ or C+.

super key হওয়ায় কোথায় check করতে হবে এটা CK কি না, A+ সুপার কি নয় তাই দেখাতে ছোট কোন লাগে নি।

so; new candidate key, CK = C

Prime attributes are = {A, B, D, C}

Again, CK = DB

SK = CB

closure of C+ is SK. so CB is also a SK.

Finally prime attribute are = {A, B, D, C}

All candidate keys are = {AB, DB, C}

Normalization: is a process in which a larger table is decomposed into several smaller table to remove or minimize the insertion, updat or deletion amongalies. Like-

Insertion → Add new course and teacher but no students.

Update → chang any Course or Sir need update everywhere.

Delet → For deleting student Dept. Course may delet.

First Normal Form:

1. The attributes must contain atomic values. (single values)
2. The values in a column must be in same domain.
3. No column name will be same. same type values in same column.
4. No records (row) will be duplicate.
5. No ordering in rows and columns.

→ For multivalues attribute make seperate table. Phone Number

→ For composite attribute make columns. Name, Nick Name

Second Normal Form:

1. If it is in 1NF (It must follow 1NF by default)
2. There is no partial dependency.

Partial Dependency: Proper subset of CK determine non-prime attribute (NPA).

→ The attributes which are part of CK or make CK are called prime attribute.

যদি কোনো একটি CK এর proper subset থেকে non-prime attribute এ functional dependency থাকে তবে সেটা partial dependency.

Ex: R(A, B, C, D, E, F)

F.D = {A→B, B→C, C→D, D→E}

$AB^+ = \{A, B, C, D, E, F\}$

$AF^+ = \{A, F, B, C, D, E\}$

Now, closures are, $A^+ = \{A, B, C, D, E\}$

$F = \{F\}$

CK = AF [A or F is not in any F.D right side]

Prime attribute = {A, F}

Non-prime attribute = {B, C, D, E}

Proper subsets are = {A, F}, have relation with non-prime attribute

(So, No 2NF)

Ex: R(A, B, C, D)

F.D = {AB → CD, C → A, D → B}

Now,

$AB^+ = \{A, B, C, D\}$

$AB^+ = \{A, B, C, D\}$

Attribute closure,

$A^+ = \{A\}$

$B^+ = \{B\}$

CK = AB

Prime attribute = {A, B}

Again, SK = CB

Attribute closure,

$C^+ = \{C, A\}$

$B^+ = \{B\}$

CK = CB

Prime attribute = {A, B, C}

Again, SK = AD

Attribute closure,

$A^+ = \{A\}$

$D^+ = \{D, B\}$

CK = {D B}

Prime attribute = {A, B, C, D}

Non-prime attribute = {∅}

Ex: R(A, B, C, D)

F.D = (A → B, B → C, C → D)

$ABCD^+ = \{ABCD\}$

$A^+ = \{A, B, C, D\}$

Though A^+ is SK but only A is CK because proper subset is ∅

So, prime attribute = {A}

A is not present in right side

Only CK is A and prime attribute is = {A} → Proper subset is null. ∴ SO 2NF

Ex: R(A, B, C, D)

F.D = {A → B, B → D)

$ABCD^+ = \{ABCD\}$

$AC^+ = \{A, C, B, D\}$

$A^+ = A, B, D$

$C^+ = \{C\}$

CK = AC

Prime attribute = {A, C}

Non-prime attribute = {B, D}

Proper subset = A, C

Here have A → B, so no 2NF

→ If a table have one primary key, it is in 2NF.

Third Normal Form:

SID S.Name DeptID Dept.Name Dept Building

3NF Solve

1. If it is in 2NF.
2. No transitive dependency in the relations for non-prime attribute. NPA → NPA

A table is in 3NF is and only if for each of its non-trivial functional dependency at least one of the following conditions holds-

i) L.H.S is super key.

ii) R.H.S is prime attribute.

Ex: R(A,B,C,D)

F.D = {A→B, B→C, C→D}

~~AB~~~~C~~~~D~~$^+$ = {ABCD}

A^+ = {ABCD}

$S_K = A^+$

$C_K = A$

Proper subset = ∅

P.A → {A}

NPA → {BCD}

Here have NPA→NPA

So, it is not 3NF

Ex: R(A,B,C,D,E,F)

F.D = {AB → CDEF, BD → F}

~~AB~~~~C~~~~D~~~~E~~~~F~~$^+$ = {ABCDEF}

AB^+ = {ABCDEF}

A^+ = {A}

B^+ = {B}

C_K = AB

P.A → AB

NPA → {CDEF}

AB → CDEF

PA → NPA

BD → F

NPA → NPA

if there exist any NPA letter then all is consider as NPA

Ex: R(A, B, C, D, E)

F.D = {A → B, B → C, C → D, D → A}

~~AB CD E~~ = ABCDE

AE^+ = {A, E, B, C, D}

A^+ = {A, B, C, D}

E^+ = {E}

CK = AE PA → {A, E}

For, SK = ~~RA → A~~ DE

~~DE+ = D.~~

D^+ = {D, A, B, C}

E^+ = {E}

CK = DE PA → {A, E, D}

SK = CE

C^+ = {C, D, A, B}

E^+ = {E}

CK = CE PA → {A, E, D, C}

SK = BE

B^+ = {B, C, D, A}

E^+ = {E}

CK = BE PA → {A, E, D, C, B}

Non-prime attribute is = ϕ

So, it is 3NF.

1NF → No repeating groups → Multiple values in any column.

2NF → No partial dependency → It occure when non-prime attribute depend only a part of composite key, not the whole.

3NF → No transitive dependency. (S.ID, CourseID) → S.Name, C

→ Proper subset of CK means a part of the CK that is not equall to the full candidate key.

BCNF → All determinents is super key.

BCNF (Boyce Codd Normal Form): Strong than 3NF

1. If it is in 3NF.
2. For each non-trivial functional dependency the left hand side must be super key.

Ex: R(A, B, C)

F.D ⇒ {A → B, B → C, C → A}

A^+ = {ABC}

A^+ = {A B C}

SK = A then the proper subset = ∅ Prime-attribute = {A}

so, and CK = A

Again CK = C as no proper subset. Prime-attribute = {A, C}

Again CK = B prime attribute = {A, C, B}

Candidate keys are = {A, B, C}

As left side have candidate key, they are also a super key, so BCNF is exist.

Ex: R(A,B,C,D,E)

F.D(A→BCDE, BC→ACE, D→E)

ABCDE+ = {ABCDE}

A+ = {ABCDE}

SK = A and CK = A prime-attribute = {A}

Again SK = BC

B+ = {B}

C+ = {C}

CK = BC prime-attribute = {A, B, C}

Again SK = BC

= AC

A+ = {A, B, C, D, E} → super-key

C+ = {C}

and SK = BC

= BA Right HS can take single letter of LHS

B+ = {B}

A+ = {A, B, C, D, E} → super-key

So, finally prime-attribute = {A, B, C}

All candidate-key = {A, BC}

→ Here, D→E, the D is not super key so, it is not BCNF.

→ Here, D→E, the L.H.S D is not super key or R.H.S E is not prime attribute, so it is not 3NF

Here, D→E, proper subset of CK not determine to nonprime attribute so it is 2NF

BCNF ✓ ✓ ✗

3NF ✓ ✓ ✗

2NF ✓ ✓ ✓

Ex: R(A, B, C, D, E)

F.D = { AB → CDE, D → A }

ABCDE+ = ABCDE

AB+ = {A, B, CDE}

A+ = {A}

B+ = {B}

So, CK = AB prime-attribute = {A, B}

Again, SK = AB
= DB

D+ = {D, A}

B+ = {B}

So, CK = DB prime-attribute = {A, B, D}

Again, SK = DB
= AB (But it is taken)

Final, candidate-key = {AB, DB}
prime-attribute = {A, B, D}

BCNF : ✓ ✗ ← D is not super key as it lies in candidate key DB

3NF : ✓ ✓ ← A is prime attribute

Fourth Normal Form:

1. If it is in BCNF
2. No multivalued dependency of attribute.

Multivalued Dependency (MVD):

If a relation R(X,Y,Z) exist the following dependency

$X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$ but there is no relation between Y and Z.

→ To be a multivalued dependency a relation must contain at least three columns

Course	Instructor	Supervi
C	Farhan Hossain	S1 S2
DBMS	Mawa Shoma	S3

→

Course	Instructor	Supervisor
C	Farhan	S1
C	Farhan	S2
C	Hossain	S1
C	Hossain	S2
DBMS	Mawa	S3
DBMS	Shoma	S3

PK

Course	Instructor
C	Farhan
C	Hossain
DBMS	Mawa
DBMS	Shoma

PK

Course	Supervisor
C	S1
C	S2
DBMS	S3

Identify Heighest Normal form:

Ex: R(ABCDEFGH)

F.D = {ABC → DE, E → GH, H → G, G → H, ABCD → EF}

$ABCDEFGH^+ = ABCDEFGH$

$ABC^+ = \{A, B, C, D, E, G, H, F\}$

$AB^+ = \{AB\}$

$BC^+ = \{BC\}$

$AC^+ = \{AC\}$

CK = ABC

Prime-attribute = {A, B, C}, Non-prime-attribute = {D, E, F, G, H}

BCNF:	✓	×	×	×	✓	→ SK + any key = SK
3NF:	✓	×	×	×	✓	LHS → SK or RHS → PA
2NF:	✓	✓	✓	✓	✓	proper subset determine NPA

Ex: R(ABCD)

F.D = {AB → CD, AC → BD, BC → D}

$AB(D)^+ = ABCD$

$AB^+ = ABCD$

$A^+ = \{A\}$

$B^+ = \{B\}$

CK = AB prime-attribute = {A, B}

Again CK = AB ← Have in RHS

SK = AC ← AC · A → AC

$A^+ = \{A\}$

$C^+ = \{C\}$

CK = {AC} Prime-attribute = {A, B, C}

Finall-prime-attribute = {A, B, C} C.K = {AB, AC}

BCNF : ✓ ✓ ✗

3NF : ✓ ✓ ✗

2NF : ✓ ✓ ✗

S.O, it is in 1NF

যাদের CK তারা সবাই SK কিন্তু নতুন কেউ SK কিনা তা বের করার জন্য অবশ্যই closure বের করবে।

$BC^+ = \{B, C, D\}$ Not CK and SK

→ Proper subset of one ck ∪ proper subset of another ck = Proper subset of ck

- If all CK are simple (single attribute) then it would be in 2NF
- If all attribute of a relation are prime attribute then it would be in 3NF
- If relation is in 3NF and all CK are simple then it is in BCNF.

Ex:

R(ABCDE)

AB → CDE

D → BE

CK = AB, AD

BCNF: ✓ ✗

3NF: ✓ ✗

2NF: ✓ ✗

Ex: R(ABCD)

AB → C

ABD → C

ABC → D

AC → D

CK = AB

BCNF: ✓ ✗ ✗ ✗

3NF: ✓ ✗ ✗ ✗

2NF: ✓ ✗ ✗ ✗

Ex: R(ABC)

A → B

B → AC

CK = A, B

BCNF: ✓ ✓

Ex: R(ABCDE)

AE → BC

AC → D

CD → BE

D → E

CK = AD, AC, AE

BCNF: ✓ ✓ ✗ ✗

3NF: ✓ ✓ ✗ ✓

2NF: ✓ ✓ ✗ ✓

Ex: R(ABCD)

A → BCD

BC → AD

D → B

CK = A, BC, CD

BCNF: ✓ ✓ ✗

3NF: ✓ ✓ ✓

Ex: R(ABCDE)

A → BC BC → AD D → E

CK: A, BC

BCNF: ✓ ✓ ✗

3NF: ✓ ✓ ✗

2NF: ✓ ✓ ✓

Ex: R(ABCDE)

A→B BC→E ED→A

CK = BCD, ACD, CDE

BCNF: ✗ ✗ ✗

3NF: ✓ ✓ ✓

Ex: R(ABCDE)

B→A, A→C, BC→D, AC→BE

CK = A; B

BCNF: ✓ ✓ ✓ ✓

Decomposition: means splitting a relation into two or more smaller relations to-

i) Eliminate redundancy.

ii) Avoid anomalies (insertion, update, deletion)

iii) Achieve higher normal form like 3NF or BCNF.

Two types of decomposition.

Dependency Preserving: All functional dependency of original relations are preserved in decomposed relations

The original relation is R and its functional dependency is F. Now it is decomposed into n new relation, then

$R_1 \quad R_2 \quad R_3 \quad \ldots\ldots \quad R_n$

$F_1 \cup F_2 \cup F_3 \quad \ldots\ldots \cup \ldots F_n \equiv F$

Ex: R(A, B, C, D, E)

F.D = { A → B, B → C, C → D, D → A }

New relations

R_1 (A, B, C)

R_2 (C, D, E)

$R_1(A, B, C)$

$A^+ = ABC$	$A \to BC$
$B^+ = BCA$	$B \to CA$
$C^+ = CAB$	$C \to AB$
$AB^+ = ABC$	$AB \to C$ Trivial
$BC^+ = BCA$	$BC \to A$ Trivial as A→ all, B→ all

$F_1 = A \to BC, B \to CA, C \to AB$

$R_2(C, D, E)$

$C^+ = CDAB$	$C \to D$
$D^+ = DABC$	$D \to C$
$E^+ = E$	
$CD^+ = CDAB$	Trivial
$DE^+ = DEABC$	$DE \to C$ Trivial
$CE^+ = CEDAB$	$CE \to D$ Trivial

$F_2 = C \to D, D \to C$

Now, $F_1 \cup F_2 = \{A \to BC, B \to CA, C \to AB, C \to D, D \to C\}$

For $D \to A$, show D^+ using this unioned new f.d

$D^+ = DCAB$

As D^+ determine A so all dependencys are preserved.

$F_1 \cup F_2 \equiv F$

If $F_1 \cup F_2 = G$ then G cover F and we need to check.

But if F cover G or not need not to check, because it is always true if G cover F is true.

→ Dependency preserving is only valid for 1NF, 2NF, 3NF.

Loss less Join Decomposition: we can construct the original relation by joining decomposed relations, no data is lossed or added.

1. the decomposition will be based on CK or SK of the relation R that is it will be common in both decomposed relations.
2. For relation R, if $R_1 \cup R_2 = R$
3. For relation R, if $R_1 \cap R_2 \neq \emptyset$

To check lossless decomposition-

1. Make cartesian product of relations (R_1, R_2)
2. Make natural join of them

R(ABC)

A	B	C
1	2	1
2	2	2
3	3	2

→

$R_1(A, B)$

A	B
1	2
2	2
3	3

$R_2(B, C)$

B	C
2	1
2	2
3	2

$R_1 \bowtie R_2$

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

→ Extra tuples

Spurious tuples ← fake or false

Rules for Decomposition

i) $R_1 \cup R_2 = R$

ii) $R_1 \cap R_2 \neq \{\phi\}$

iii) $R_1 \cap R_2 = R_1$ or $R_1 \cap R_2 = R_2$

→ Spurious tuples occure when the common attributes used in the join is not a candidate key.

~~Distinct value~~

→ so, to make a lossless join common attribute need to be candidate key or super key in at least one of the any two relations.

Ex:

$R_1(ABC)$ $R_2(D,E)$ → No common attribute

$D(ABC, CD)$ → Attribute E is lost here.

$R_1(ABC)$ $R_2(CDE)$ → In R_2 C is candidate key.

R(A	B	C	D	E)
1	1	2	1	3
2	2	2	1	3
3	1	6	3	6
4	2	8	5	7
5	3	4	5	7

$R_1(ABC)$. $R_2(ABDE)$ → AB is candidate key.

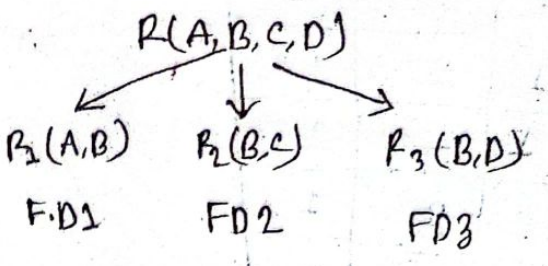
$R_1(A,B)$ $R_2(BCDE)$ → B is not a candidate key.

$R_1(AB)$ $R_2(CD)$ $R_3(DE)$ → For R_2R_3 here D in R_3 is CK. But later with R_1 and $R_{23}(CDE)$ have no common attribute.

Ex: Dependency preserving Decomposition.

$R(A, B, C, D)$

$F.D = \{A \to B,\ B \to C,\ C \to D,\ D \to B\}$



AB	BC	BD
$A \to B$	$B \to C$	$B \to D$
	$C \to B$	$D \to B$

→ Dependency should be non trivial. $R_1 \cap R_2 = \emptyset$

$FD_1 = A \to B$ $FD_2 = \{B \to C,\ C \to B\}$ $FD_3 = \{B \to D,\ D \to B\}$

$B^+ = BCD$

$C^+ = CDB$ ক্লোজারের ভেতরে থাকলেই ডেটা পাওয়া সম্ভব।

$FD_1 \cup FD_2 \cup FD_3 = \{A \to B,\ B \to C,\ C \to B,\ B \to D,\ D \to B\}$

$= FD$

Ex: lossless decomposition

$R(ABC)$

$F.D = \{A \to B,\ B \to C,\ C \to A\}$

$R_1(ABC)$ $R_2(CD)$

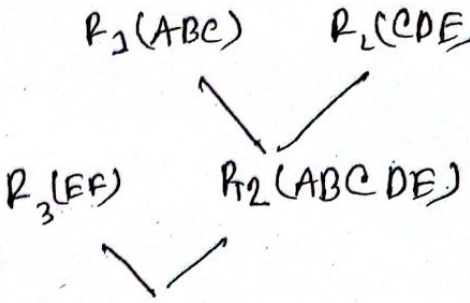
$C^+ = CAB$ As C can derive all attribute of R_1 so it is candidate key.

Ex: R(ABCDEF)

F.D={AB→C, C→D, D→EF, F→A, D→B}

D={ABC, CDE, EF}

$R_1(ABC)$ $R_2(CDE)$ $R_3(EF)$



C+= CDEFAB

E+= {E} It cannot derive all attribute(EF)

~~But E+ is not a candidate key so it is lossy decomposition.~~

~~Now~~

~~E+=~~

→ দুটি R এর মাঝে যেকোনো একটির Candidate Key পাওয়া গেলে সেটা lossless.

উপরের উদাহরণে $R_1(ABC)$ $R_2(CDE)$ এখানে C+ মূলে R_2 এর সবগুলো অথবা R_2 এর সবগুলোতে যেতে পারলেই সেটা CK. আর R(ABCDEF) পাওয়া লাগবে না।

→ Another way is (i) natural join (ii) cartesian product.

* No spurious attribute.

Note In Exam Preparation : 8.9.25

Data: Data is simply a value of real world things or object can be form in concrete or abstract.

Information: An information is the processed form of data which is meaningful and useful to the user.

Database: A database is a collection of interrelated data of a particular enterprise.

DBMS: A DBMS is one or more database and a set of programs to access data from those database.

Application of DBMS:

1. Education.
2. Bank and Insurance.
3. Healthcare.
4. E-commerce.
5. Social Media.
6. Space Exploration.
7. Government project.
8. Manufacturing company.

Drawbacks of conventional file processing system.

1. Data Redundancy and Inconsistency (Details)
2. Difficult in Accessing Data. (Details)
3. Integrity Problem. (Details)
4. Atomiticity Problem. (Details)
5. Concurrent Access Anomalies (Details)
6. Security Problem. (Details)

1. The same info maybe duplicated in several places. It lead to higher storage and access cost. It also may lead to data inconsistency. Various copy of same data may no longer agree.

2. Conventional file processing system do not allow to needed data to be retrived in convenient and effective manner. More and more responsive data retrival system need.

3. The data values must satisfy certain type of consistency constraints. It refer accuracy, consistency and trustworthiness.

4. It is a fundamental property of transection. It ensure that a transaction is treated as a single, indivisible unite of work. It occure in fail transaction.

Descriptive attribute: Give info about the relationship set. It is the own attribute of relationship.

Specialization: The process of dividing a single entity set into multiple, more specific entity sets based on some distinguishing characteristics

Generalization: The process of combining two or more entity sets into a single, generalized entity set.

Aggregation: It is an abstraction that allows modelling a relationship as an entity.

DBA is a person for managing, maintaining and controlling a DB system in Organization. They ensure availability, reliability, security and performence.

Responsibilities: ① Database design and implementation. ② Performence optimization. ③ security and access control. ④ Backup and recovery ⑤ user support and training ⑥ Stay Update.

Convert E-R Diagram to Schema:

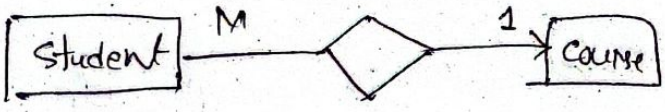
1. A strong entity set with only simple attributes will require only one table.
2. With composite attribute requir one table with all attribute.
3. With multi valued attribuet requir separate table with primary key.
4. Relational set will requir one table with primary key of participating set. Descriptive property will also include
5. Many to many will make seperate table for relationship.
   One to many marge relationship with many side entity se
   One to One may merge any side entity set.

* যে কোনো relationship যদি কোন relationship set এর সাথে যুক্ত entity র primary key নিয়ে seperate entity set এ key আকারে যায়।

→ total participation এগুলোর foreign key Not Null হয়। যে entity কোনো থাকতে পারে সেটার key foreign key. হলে Not Null হয়।

In weak entity set there must be totall participation. And always 1:n relationship from identifying entity set to weak entity set.

→ Partial (---) attribute is only in weak entity set.



→ Arrow means 1, Normal straight line mean many.

In Both side totall participation only a single table.

(Age) Derived attribute will not come into schema.

→ Totall participation is always many relation.

Divisional Operator: $R_1 \div R_2$ is possible $R_2 \subset R_1$ and $R_1 \neq R_2$

1. First make cartesian product all combination who take all things.
2. From this product subtract the given list who take some.
3. Then we will find whome who not take all.
4. Now substract them from whole list.

(a) Equi Join: Join table using an equality condition. It is standard and most common inner join.

(b) Theta Join: Inner join with a non-equality condition, uses other operator instead of "=".

File Storage / Disk Storage:

Redundant Arrays of Independent Disk.

Different Raid levels:

1. RAID 0 - Striping (Split Data into several disks)
2. RAID 1 - Mirroring (copy data in several disk)
3. RAID 2 - Bit level stripping with error correction codes. → Hamming code
4. RAID 3 - Byte level stripping with dedicated parity disk.
5. RAID 4 - Block level stripping with dedicated parity disk.
6. RAID 5 - Block level stripping with distributed parity disk.
7. RAID 6 - Block level stripping with double parity disk.

Transaction: is a set of operations used to perform a logical unite of work.

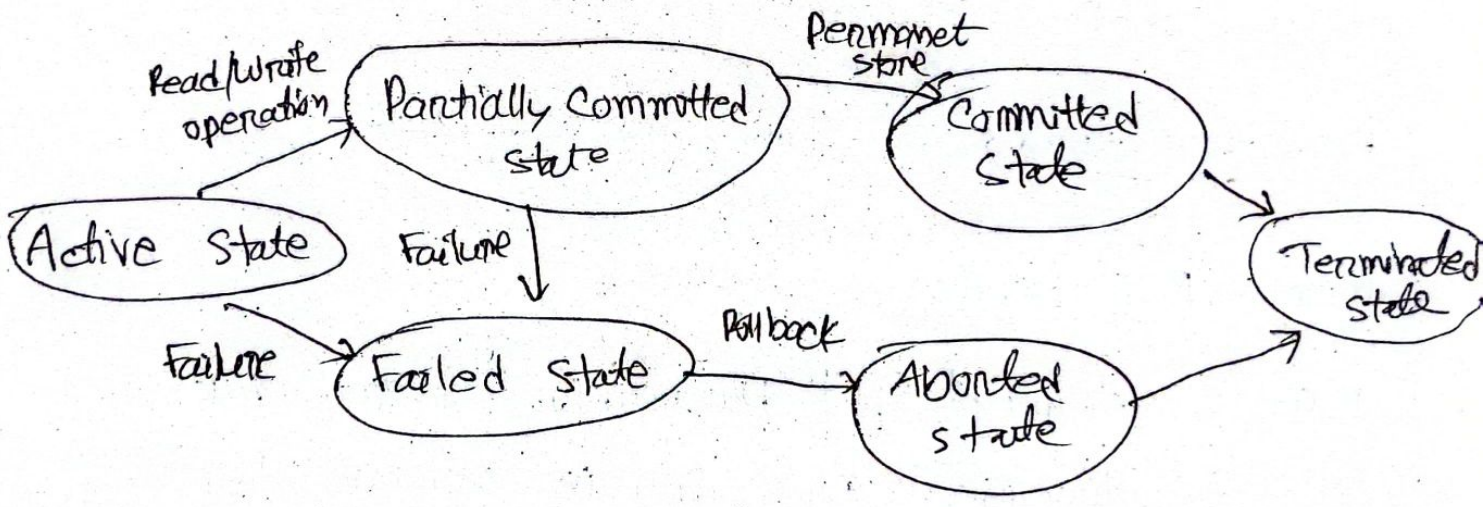
Acid Property:

Atomiticity: Either all or none.

consistency: The sum of money before and after will be same

Isolation: Conversion parallel transaction into serial.

Durability: Transaction data will save permently in storage.



Schedule: is a chronological sequence of execution of transa

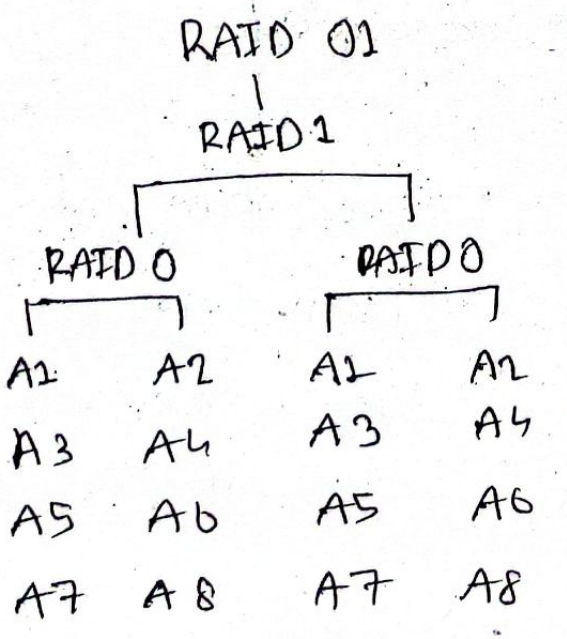
Transaction Scheduling:

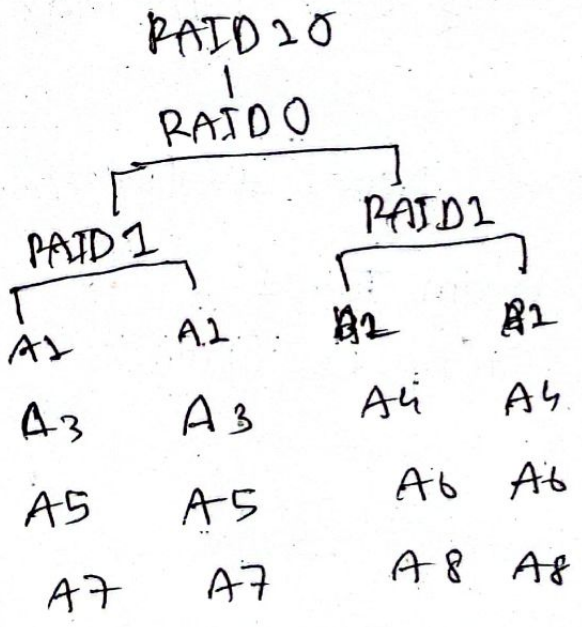
1. Serial Scheduling.
2. Parallel or concurrent scheduling.

Trigger: are procedures that are stored in DB and implicitly executed when the content of a table are changed due to insert, update, delet.

Row trigger: is fired each time the table is affected by the triggering statement. keep track of all affected record.

Statement trigger: executes once for the entire SQL statement, regardless of how many rows are affected or not affected.





Aggregate Function: take multiple row of data and return a single summarized value.

```
Select count(*) AS X
From table1
where name = "Nazim";
```

(*) → All
X → Variable
AS → alias

(Age)/(Distinct Age) → this will count and only the not null of this given column.

```
Select sum(salary) AS Y
From Table1;
```

→ sum not null value

```
Select AVG(salary) AS X
From Employee;
```

In this way MIN, MAX can be calculated.

```
Select Name, AVG(salary) As Z
From Table1
Group By (status);
```

// show name and avg
// From table1
// Make as much group by status, CSE EEE group.

"Group by" make group on the property are given and supply each group to the above function. Make group of same value for given column here column is "status".

WHERE filters rows before grouping but HAVING filter groups after grouping.

```
Select Name, AVG(salary)
    From Table1
    Group By (Status)
    Having AVG(salary) > 60000;
```

// Status অনুযায়ী group করে যে group এর avg sallary 60000 এর বেশি তাদের avg sallary আর নাম দেখাবে।

```
Select Name, Count(*) AS X
    From table1
    Group by (Status)
    Having COUNT(*)>1
```

// Status অনুযায়ী group করে যে group এর member 1 জনের চেয়ে বেশি তাদের Name আর কতজন এটা দেখাবে।

```
Select branch_name
    From Account
    Group By branch name
    Having AVG(balance) >= (
        Select AVG(balance)
        From Account
    );
```

// Having will work on each group made with branch_name

```
DELIMITER //
Create trigger account_before_insert
    before insert
    on Account for each row

begin
    Declair vuser varchar(30);
    select user() into vuser;
    Insert into students(name)
        values(vuser);
end //
DELIMETER;
```