Database Management Systems

Database Management System is basically a collection of data related to an enterprise with some programs to manage or manipulate it.

Program | Data

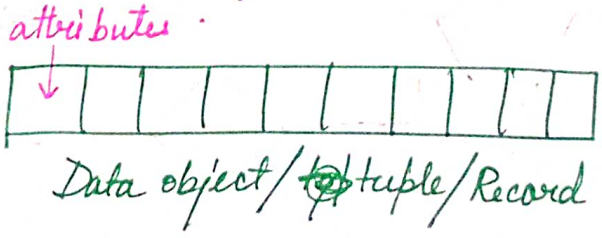
Isolation
Abstraction
(So that the low level details need not be known to the programmer)

— Convenience of Data
— Efficiency

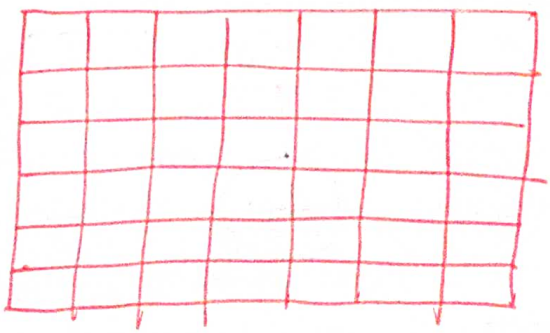
Database Model

• Relational Model

Every single unit of data is represented as a vector, where each component is called an attribute.



Multiple records gives rise to a "data table"



Eg.! Table / Students of DBMS

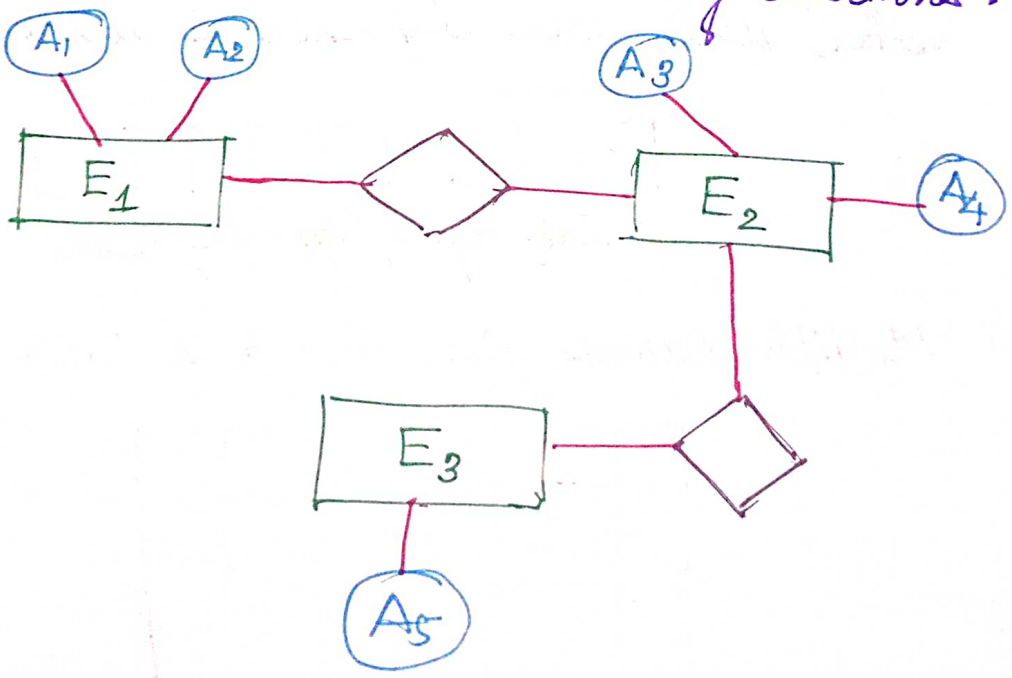
	RN	N	D	CG	SG	H	
S_1							
S_2							
S_3							
S_4							

We can name this relation "DBMS course students"

This is a subset of the cartesian product of everything.

- We will interchangibly use "Table" & "Relation"
- Structure of Record → Schema of data; All schema — Schema of DB
- Entity Relationship Model (ER-Model)

Data is represented in a diagramatic way. There are entities (described by a set of attributes) and these entities are related by relations.



Entity

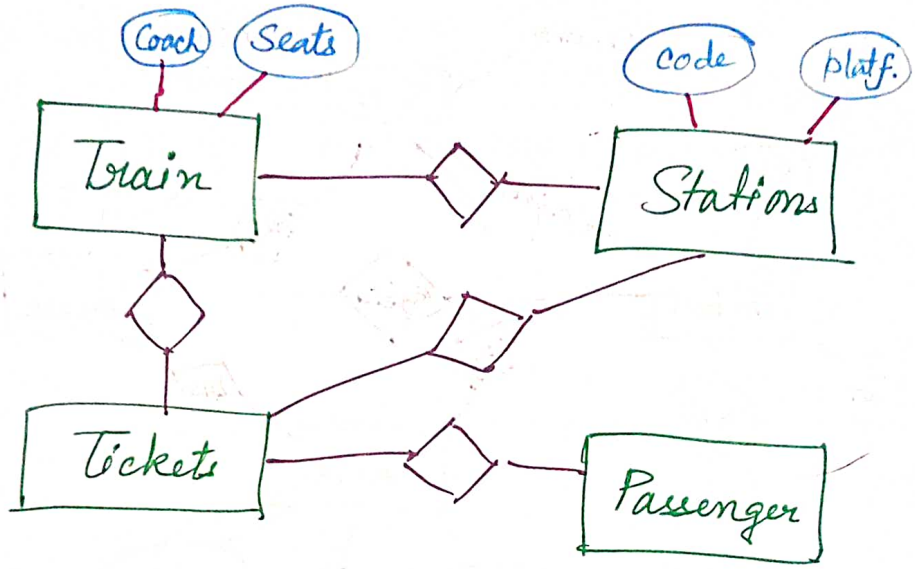
Attribute

Relation

Eg.: ER-Diagram of IRCTC Booking System

Entities

- Train
- Passengers
- Stops/Stations
- Ticket



We can convert an ER-Diagram to a Relational Schema & vice-versa.

(We can have entities as tables & or have a common table entry to identify relation).

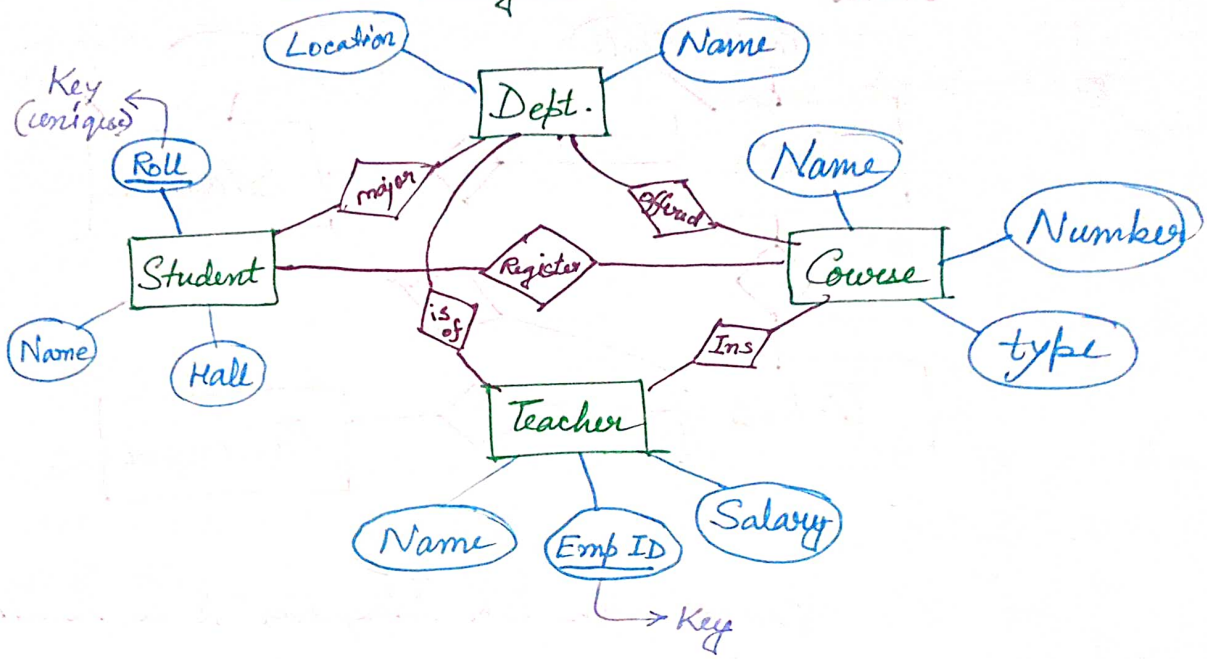
- E-R table & Relational schema is can be repeatation
- Relational schema cannot handle data without well defined set of attributes (unstructured data)
- Relational schema cannot handle data with varying number of attributes.

Semi Structure Data Model (XML)

Some markups or tags are used to describe data.

Used in E-commerce sites as Flipkart or Amazon

ER-Diagram



Step 1: Write down all entities and their attributes.
Step 2: Identify relations and incorporate them in the schema

Key: Unique attribute of an entity (no two instances of this entity has the same key)

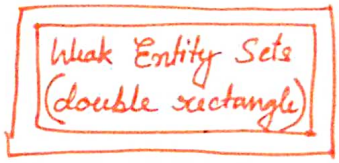
Superkey: Subset of attributes which has unique values for every instance of the entity

Minimal Superkey: Superkey whose no smaller subset as superkey (Candidate key)

- Superkey & Candidate Key are part of the schema
- There can be multiple Candidate Keys.
- Keys, when forced to be unique, ~~are called "Primary Key"~~ and is considered to be a part of the schema

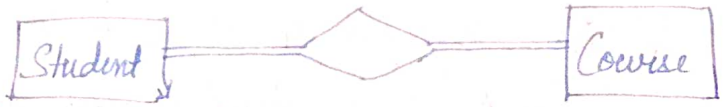
Primary Key: Candidate Key, which is forced to be unique.

Weak Entity Set: Entities which have no candidate key.



Participating Constraints:

(i) Total Participation: All ~~entity~~ instances of the entity has atleast one relation. (double line)



(ii) Cardinality

- Many to Many — —
- One to Many <——
- Many to One ——>
- One to One <——>

Converting ER-Diagram to a Table

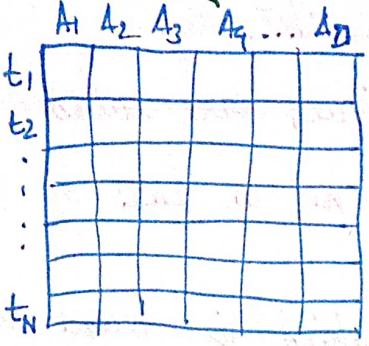


table r

The table/relation schema is specified by

$$r(A_1, A_2, \ldots, A_D)$$

A database schema may be specified by —

$$r_1(A_1, A_2, \ldots, A_{D_1})$$
$$r_2(A_1, A_2, \ldots, A_{D_2})$$
$$\vdots$$

Foreign Key:

r_1

A_2	A_5	A_7	A_8	A_9

r_2

A_1	A_2	A_3	...	A_D

If A_2 is a key for r_1 & r_2 is a weak entity set, then A_2 is a foreign key referencing r_2 from r_1

E.g.:

Courses

Cid	Cname

Student

Roll	Cid

Cid is a foreign key referencing Student table from Courses table

Conversion from ER-Diagram to Table

- Have a table for
  1. Each entity → { Normal : All attributes
     Weak : All attributes, plus attributes of all referencing relations entity (primary key) }
  2. Each relation. (key of both participating entities)

Example:

RELATIONAL ALGEBRA

"Queries" are fired on table (or a set of tables). The answers may also be represented by a table.

result-table = Query(table1, table2, ...);

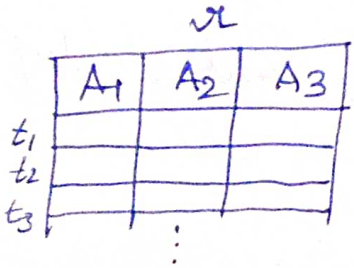
↑ can be specified by some Data Query Language.

Data Query Languages
- Procedural {Relational Algebra}
- Descriptive {Relational Calculus}

We start with Relational Algebra.

RELATIONAL ALGERA

Table ≡ Relation



, ~~r~~ r(A1, A2, A3); t1, t2, t3 ∈ r

This is a relation as, $r(A_1, A_2, A_3) \subseteq A_1 \times A_2 \times A_3$

Now, we consider a relational algebra expression.

$$r_4 = (r_1 \ op_1 \ r_2 \ op_2 \ r_3)$$

In relational algebra, there are six primitive operators.

UNITARY OPERATORS

• "SELECT" operator

$$result = \sigma_{\theta}(r)$$

("clause")

Where θ is a clause, like
- $(A_1 > 30) \ \& \ (A_2 == 0)$
- $(A_3 == 0)$

then, we have result as the set of tuples from r satisfying θ.

$$result = \{ t \in r \mid t \text{ satisfies } \theta \}$$

• "PROJECT" operator

$$result = \Pi_{\{A_{i_1}, \ldots, A_{i_k}\}}(r)$$

result will just be the table with only the columns of the attributes $A_{i_1}, \ldots, A_{i_k}$. Mathematically, removal of duplicates are necessary.

In SQL, implementation does lead to duplicates.

- "RENAME" operator

$\rho_{R_1}(R_2)$; renames R_2 as R_1

Example: student (Name, Roll, Hall, course).

Find the Name & Roll of students from RK, taking the DBMS course.

$$res = \Pi_{\{Name, Roll\}}\left(\sigma_{Hall == 'RK' \ \& \ course == 'DBMS'}(student)\right)$$

BINARY OPERATORS

- "SET UNION" operator

$R_3 = R_1 \cup R_2$ { The attributes in R_1 and R_2 must be the same }

Mathematically, R_3 is without duplicates.

- To save computational power, SQL implements union by allowing duplicates.

- "SET DIFFERENCE" operator

$R_3 = R_1 - R_2$ { Again, the attributes in R_1 & R_2 must be the same }

- "CARTESIAN PRODUCT" operator

$R_3 = R_1 \times R_2$

if $R_1(A_1, A_2, A_3)$ and $R_2(A_4, A_5)$; $R_3 = (A_1, A_2, A_3, A_4, A_5)$

if $R_1(A_1, A_2, A_3)$ and $R_2(A_4, A_5, A_3)$; $R_3 = (R_1.A_1, R_1.A_2, R_1.A_3, R_2.A_4, R_2.A_5, R_2.A_3)$

DERIVED OPERATORS

"NATURAL JOIN"

$$\mathcal{R}_1 \bowtie \mathcal{R}_2 = \Pi_{A_1 \cup A_2}\left(\sigma_{\mathcal{R}_1[A_1 \cap A_2] = \mathcal{R}_2[A_1 \cap A_2]}\left(\mathcal{R}_1 \times \mathcal{R}_2\right)\right)$$

$\mathcal{R}_1$

	A_1	A_2	A_3
t_{11}	a_{11}	a_{21}	a_{31}
t_{12}	a_{12}	a_{22}	a_{32}
t_{13}	a_{13}	a_{23}	a_{33}

$\mathcal{R}_2$

	A_2	A_3	B_1
t_{21}	a_{21}	a_{31}	b_{11}
$\bowtie t_{22}$	a_{22}	a_{32}	b_{12}
t_{23}	a_{21}	a_{31}	b_{13}

$\Downarrow$

$\mathcal{R}_1 \bowtie \mathcal{R}_2$

A_1	A_2	A_3	A_4	
a_{11}	a_{21}	a_{31}	b_{11}	$t_{11} \bowtie t_{21}$
a_{11}	a_{21}	a_{31}	b_{13}	$t_{11} \bowtie t_{23}$
a_{12}	a_{22}	a_{32}	b_{12}	$t_{12} \bowtie t_{22}$

Example:

"DIVISION"

$\mathcal{R}_1 \div \mathcal{R}_2$ ⟶ Take joinable tuples, but take only the attributes in $\mathcal{R}_1$ but not in $\mathcal{R}_2$.

$$\mathcal{R}_1 \div \mathcal{R}_2 = \Pi_{A_1 - A_2}\left(\sigma_{\mathcal{R}_1[A_1 \cap A_2] = \mathcal{R}_2[A_1 \cap A_2]}\left(\mathcal{R}_1 \times \mathcal{R}_2\right)\right)$$

"INTERSECTION" $\mathcal{R}_1 \cap \mathcal{R}_2$

EXTENDED RELATIONAL ALGEBRA

• "OUTER JOIN":

$r_1 \overline{\bowtie} r_2$ (left)

$r_1 \overline{\bowtie} r_2$ (right)

$r_1 \overline{\bowtie} r_2$ (full)

- For $r_1 \overline{\bowtie} r_2$, include $r_1 \bowtie r_2$ and include unjoined tuples in r_1 and append "Null" to attributes from r_2
- Similarly for $r_1 \overline{\bowtie} r_2$ and for $r_1 \overline{\bowtie} r_2$, we do both $r_1 \overline{\bowtie} r_2$ & $r_1 \overline{\bowtie} r_2$.

$$r_1 \overline{\bowtie} r_2 = (r_1 \overline{\bowtie} r_2) \cup (r_1 \overline{\bowtie} r_2).$$

• "AGGREGATION OPERATORS":

$${}_{A_2}\mathcal{G}_{r_1}(A_1)$$

$\mathcal{G}$ – count unique
– average
– max
– min
– sum.

gives the $\mathcal{G}$-aggregation of attribute A_1 grouped according to A_2.

DATABASE CONSISTENCY

Written down in terms of Functional Dependency.

Course_registration

Roll	Name	Cid	Cname

Things which are obvious:

→ If roll is same ⇒ name has to be same

→ If Cid is same ⇒ Cname has to be same

We write these as:

Roll → Name

Cid → Cname

Let $r(R)$ be a table where R is a set of attributes,

"$\alpha \rightarrow \beta$" , $\alpha, \beta \subseteq R$

means

$$[t_1[\alpha] = t_2[\alpha]] \Rightarrow [t_1[\beta] = t_2[\beta]]$$

F: { roll → name, cid → Cname }

F^+ { ... Set of implied dependency ... (e.g. roll, cid → Cname) }

ARMSTRONG'S AXIOMS:

1. Reflexivity:
   $\alpha \rightarrow \beta$ if $\beta \subseteq \alpha$
2. Augmentation:
   if $\alpha \rightarrow \beta$ then $\alpha\gamma \rightarrow \beta\gamma$
3. Transitivity:
   if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$

} All that is needed,

Let $r(R)$ be a table where R is a set of attributes,

"$\alpha \rightarrow \beta$" , $\alpha, \beta \subseteq R$

means

$$[t_1[\alpha] = t_2[\alpha]] \Rightarrow [t_1[\beta] = t_2[\beta]]$$

F: $\{$ roll → name, cid → Cname $\}$

F^+ $\{$ ... Set of implied dependency ... (e.g. roll, cid → Cname) $\}$

ARMSTRONG'S AXIOMS:

1. Reflexivity:
   $\alpha \rightarrow \beta$ if $\beta \subseteq \alpha$
2. Augmentation:
   if $\alpha \rightarrow \beta$ then $\alpha\gamma \rightarrow \beta\gamma$
3. Transitivity:
   if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$

} All that is needed,

* Union Rule:

if $\alpha \to \beta$ and $\alpha \to \gamma \Rightarrow \alpha \to \beta\gamma$

Proof:

$\alpha \to \beta$ and $\alpha \to \gamma$

$\Rightarrow \alpha, \alpha \to \beta.\alpha$ and $\beta, \alpha \to \beta, \gamma$

$\Rightarrow \alpha \to \alpha\beta$ and $\alpha\beta \to \beta\gamma$

$\Rightarrow \alpha \to \beta\gamma$.

* Pseudotransitivity:

if $\alpha \to \beta$ and $\beta\gamma \to \delta$ then $\alpha\gamma \to \delta$

Proof:

$\alpha \to \beta$ and $\beta\gamma \to \delta$

$\Rightarrow \alpha\gamma \to \beta\gamma$ and $\beta\gamma \to \delta$

$\Rightarrow \alpha\gamma \to \delta$

* Decomposition:

if $\alpha \to \beta\gamma$ then $\alpha \to \beta$ (and $\alpha \to \gamma$)

Proof:

$\alpha \to \beta\gamma$ and $\beta\gamma \to \beta$

$\Rightarrow \alpha \to \beta$.

- If $A \to R$ where R is the set of all attributes, then A is a candidate key.

CLOSURE OF AN ATTRIBUTE SET α

α^+ is the set of attributes in R such that $\alpha \rightarrow \alpha^+$, with respect to F^+, the closure of a functional definition, then α^+ is the closure of the attribute set α.

- If $\alpha^+ = R$, then α is a candidate key.

- If there exists F' (simpler than F) such that

$$F'^+ = F$$

then checking database consistency of F' suffices. This F' is called a 'canonical cover'.

F' is a canonical cover of F, if there does not exist F'' such that

$$F'' \subseteq F' \text{ and } F''^+ = F'^+$$

whereas $F'^+ = F^+$.

Example:

$$F = \left\{ \begin{array}{l} A \rightarrow B \\ B \rightarrow C \\ A \rightarrow BC \end{array} \right\} \Longrightarrow \text{then } F' = \left\{ \begin{array}{l} A \rightarrow B \\ B \rightarrow C \end{array} \right\}$$

Roll	Name	Cid	Cname
101	AB	CS1	PDS
101	AB	CS2	Algo
102	AG	CS1	PDS

Example : Roll → Name ∧ Cid → Cname

Extraneous Attribute :

Consider three functional dependencies :

FD1 : $\alpha \rightarrow \beta$
FD2 : $\beta \rightarrow \gamma$
FD3 : $\alpha \rightarrow \beta\gamma$

FD3 is implied from FD1 & FD2, we can drop γ from FD3 and it becomes FD1. Here γ is an extraneous attribute.

Obtaining the Canonical Cover F^c of F

~~1. Apply Union Rule (apparently to get closure)~~
~~2. Check for extraneous attributes on LHS & RHS remove all the~~

1. Apply Union Rule and add only the unions to F^c
2. Check for extraneous attributes on LHS & RHS, and remove this.

REDUNDANCIES IN TABLES

Faculty

	Name	Dept	Salary	Dept_addr	Ph_no
1	AB	CS	1000	Nalanda	123
2	AG	PHY	1000	Takshila	124
3	PM	PHY	1000	Takshila	125
4	NG	CS	1500	Nalanda	126
5	PM	CS	1000	Nalanda	125

Anomalies:

- **Update Anomaly:**
  - Update at one location may cause inconsistency.
- **Insertion Anomaly:**
  - Insertions causing inconsistency.
- **Deletion Anomaly:**
  - Deletion of one row may cause unnecessary loss in information.

* The following decomposition may be helpful.

Faculty

Name	Dept	Salary	Ph_no

Department

Dept	Addr

(Should be Lossless)

* "Lossy join" —— Cannot recover from the join
* "Lossless join" —— Possible to recover original table after naturally joining the tables.

Lossless Decomposition:

If R is split into R_1 & R_2, then it forms a lossless join, if —

- $R_1 \cap R_2 \neq \phi$
- Either $R_1 \cap R_2 \longrightarrow R_1$
  or $R_1 \cap R_2 \longrightarrow R_2$

[which is basically

$$\{R_1 \cap R_2 \to R_1 \,,\, R_1 \cap R_2 \to R_2\} \cap F^+ \neq \phi$$]

Lossless decomposition makes sense only under a set of functional ~~decomposition~~ dependencies.

Dependency Preservation

$R = R_1, R_2, \ldots, R_n$ be a split and F be the set of functional dependencies.

$$F_i := \{ [\alpha \to \beta] \in F \mid \alpha, \beta \subseteq R_i \}$$

This decomposition is Dependency Preserving if

$$(F_1 \cup F_2 \cup F_3 \ldots F_n)^+ = F^+$$

or $$(F_1 \cup F_2 \cup F_3 \ldots F_n)^c = F^c \quad \ldots\ldots (?)$$

A Decomposition must be

- Dependency Preserving
- Lossless join.

Algorithm to check if $(F_1 \cup F_2 \cdots F_n)^+ = F^+$:

for $\alpha \to \beta \in F - (F_1 \cup F_2 \cup \cdots \cup F_n)$

$\alpha_0 := \alpha$

for i in 1 to n

$\alpha_i :=$ closure of α_{i-1} in F_i

~~$\alpha^+ :=$ closure of α in F~~

if $\beta \notin \alpha_n$

return NOT_DP

return DP

SCHEMA REFINEMENT

We do a set of decomposition until we reach a "normal form"

Normal Forms:

"In a nutshell, these are non-redundant form"

• First Normal Format (1-NF):

A table is in 1-NF if it does not contain any multivalued or composite attribute.

[Just split up into another column if needed]

• Boyce Codd Normal Form (BCNF)

A relation $R(R)$ is in BCNF if for every FD $\alpha \to \beta$ in F^+ the following holds true.

$\alpha \to \beta$ is trivial or α is a superkey of R.

Example:

Faculty

Name	Dept	Salary	Ph

Dept

dept	address

~~Name → Dept~~

Name → Phone

Name → Salary

dept → address

NOT IN BCNF

Conversion to BCNF:

- Split R into R_1, R_2 for $\alpha \to \beta$ which are violating (non-trivial & α is not superkey) make $R_1(\alpha\beta)$ and $R_2(R-(\beta-\alpha))$

Further split R_1 and R_2 if needed for other violating dependencies.

* BCNF decomposition is always a Lossless join but may not be dependency preserving.

* BCNF can still have "transitive redundancy"

Transitive Redundancy:

In a set of FD F if

$\alpha \rightarrow \beta \in F^+$

$\alpha \rightarrow \gamma \in F^+$

$\beta \rightarrow \gamma \notin F^+$

then F consists transitive redundancy.

Third Normal Form (3NF):

A table $r(R)$ is in 3NF under a set of dependencies F, if $\forall \alpha \rightarrow \beta \in F^+$,

$\alpha \rightarrow \beta$ is trivial or α is a part of a candidate key.

Example:

Faculty (Name, dept, salary, address, ph) has candidate key (Name, dept), then under

$F = \{$

Name → ~~Address~~ Phone

Dept → ~~Address~~

$\}$

is not in BCNF but is in 3CNF.

3. CNF decomposition

Take union $\alpha \rightarrow \beta$ and make tables for each L.H.S.