5/Jan/17

CHAPTER - 1

Databases and Database Users.

* What is a database?

A database is a collection of related data. Data means known facts that can be recorded & that have implicit meaning.

A Database has following implicit properties-

- It represents some aspects of the real world.
- It is a logically coherent collection of data with some inherent meaning
- A database is designed, built & populated with data for a specific purpose. It has intended groups of users.

* What is DBMS ?

A DBMS is a collection of programs that enables users to create & maintain a database. It is a general purpose software system that facilitates the processes of defining, constructing, manipulating & sharing databases among various users & applications.

Defining - Specifying data types, structures & constraints Database descriptive / definition information is stored in database catalog called, metadata.

Constructing - Storing data on some storage medium is controlled by DBMS.

Manipulating - Updating, deleting or retrieving database

Sharing - Allows multiple users & programs to access the database simultaneously.

Characteristics of Database System

1. Self describing nature

A database system not only contains the database but also

contains the complete definition of the database structure and constraints. Meta-data describes the structure of primary database.

* In traditional file system, data definition is typically part of the application programs themselves. So the programs are constrained to work within only a specific database, whose structure is described in application program. But DBMS software can access diverse databases by extracting the database definitions from catalog.

2. Insulation b/w Programs & Data
In traditional file systems, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access the file. But in DBMS, program do not require such changes. DBMS provides data-abstraction through program-data independence and program-operation independence.

3. Support of Multiple views of Data
A DBMS, with multiple users, must provide facilities for defining multiple views.

4. Sharing of Data & Multiuser Transaction Processing
DBMS must allow multiple users to access the data at the same time. So DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner

Date:

Users of Database -

1. Administrator - Oversees & manages resources
2. Designer - Identifies data & appropriate structures.
3. End User - Access the database for quering, etc.
   - Casual Users (occassionally access database)
   - Native Users (perform canned transactions → standard types of transactions)
   - Sophisticated Users (engg., business analysts,
   - Stand alone Users (maintain personal databases by using ready made program pkgs)
4. System Analysts & Application programmers. (Software engg.) - Determines the requirements of end users, especially native & parametric end users

6/Jan/17

Advantages of Database Systems.

1. Controlling redundancy.
In traditional file system we suffer from- data inconsistency, duplication of efforts & more space is required. Database system takes care of these problems. and a same data is shared & hence controls & data redundancy.

2. Restricting unauthorised access.
To avoid accidental change of data.

3. Provides backup & recovery routines.

4. Various integrity constraints are there.

5. Provides multiple user interfaces.

6. Provides effiective storage structures.

7. Provides concurrency control.

When not to use DBMS ?

- When data is simple and do not need modification.
- Real time data is used.
- No multiple user access to data.

9/Jan/17

CH-2

DATABASE CONCEPTS & ARCHITECTURE.

Data Model - Collection of concept which is used to define structure of database.

Types of data model -

1. Higher level data model
   - Closer to real world.
   - Example - Entity relationship (ER) model
2. Intermediate or representational model
   -
   - Example - Relational Model
3. ~~Higher~~ lower level Model
   - Deals with storage & physical ~~detail~~ details

Schema - It is the description of the database.

STUDENT

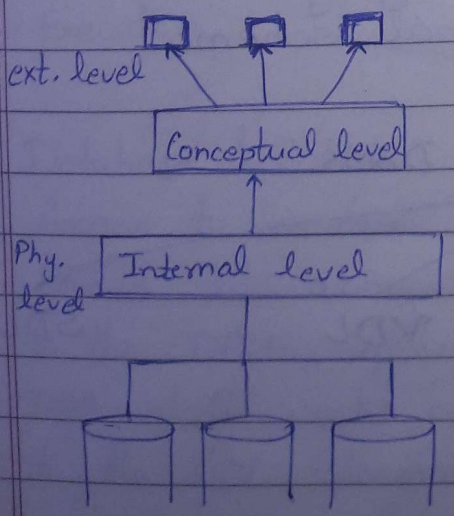
Rollno.	Name	age	course no.	- - -

COURSE

courseno.	coursename.	- - - -

Schema doesnot changes frequently whereas data changes.

Three level / Schema Architecture of Database.



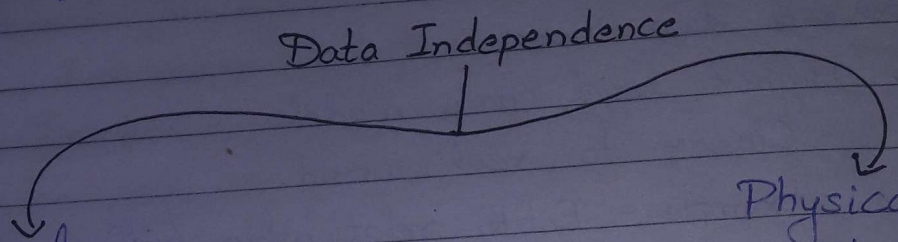
Provides interface to users.

contains conceptual detail.
data structure

How data is stored.

stored database

Data Independence - Capacity to change at one level without affecting next higher level. It is the property of database system.

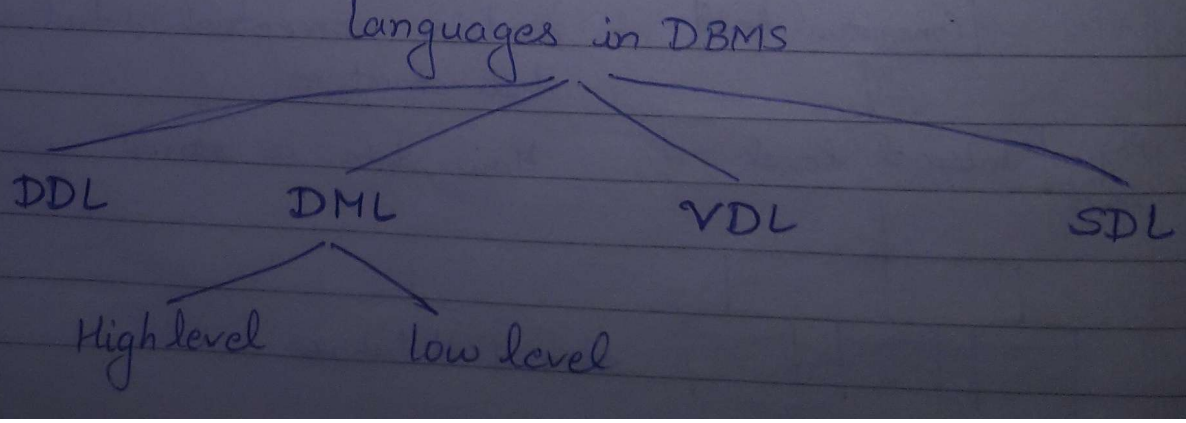


Logical
(Changes made at conceptual level should not affect the external level)

Physical.
(Changes made at phy. level should not affect ~~next~~ conceptual level)

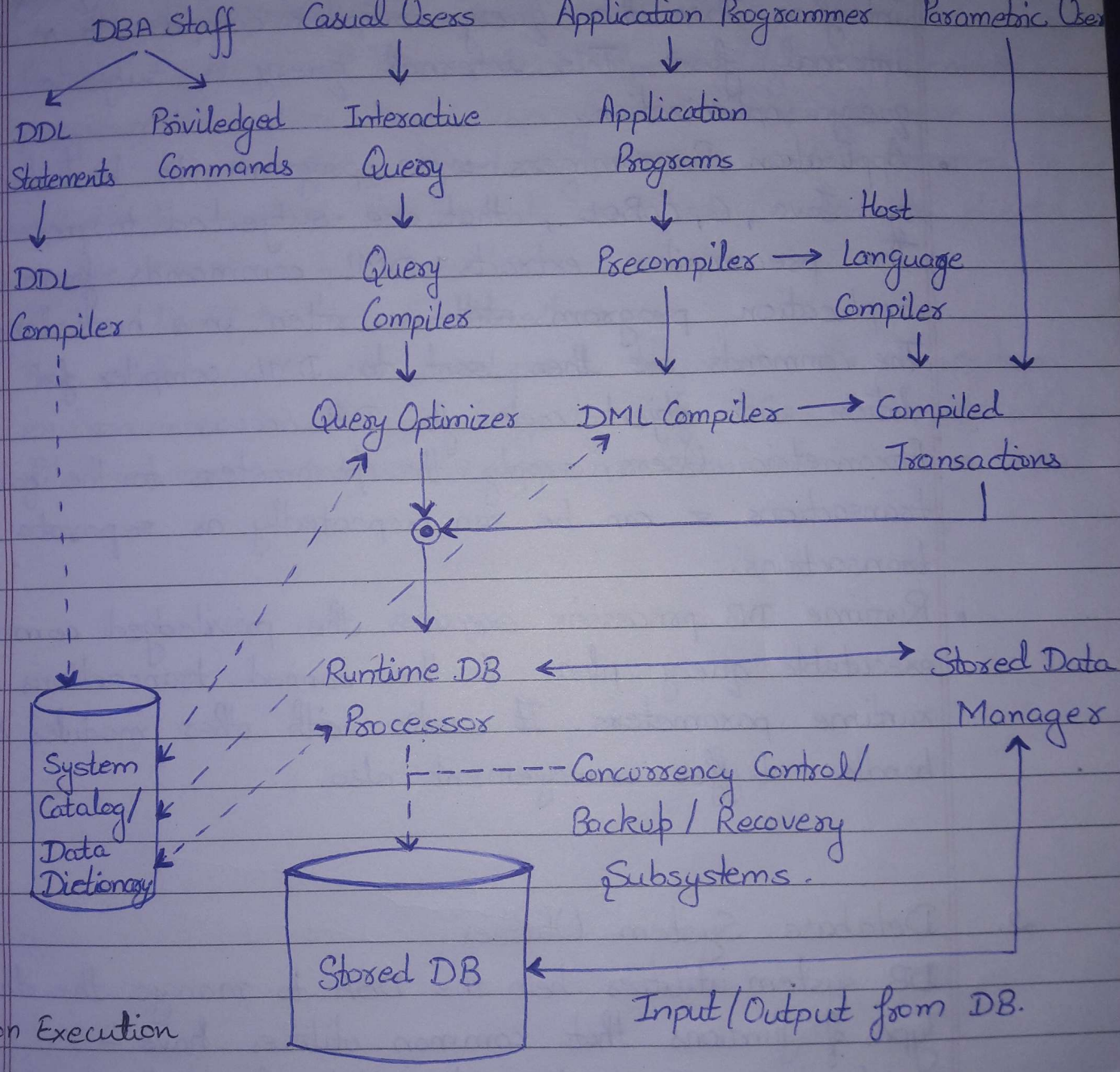
Database languages.

1. **DDL** - Data Definition language.
   Used by DBAs & by database designess to define conceptual & internal schemas.
2. **DML** - Data Manipulation Language
   It includes typical manipulations as - retrieval, insertion, deletion & modification of the data.
3. **VDL** - Veiw Definition Language.
   To specify uses ~~to~~ views & their mappings to conceptual schema.
4. **SDL** - Storage Definition language
   Used to specify the internal schema.



The Database System Environment

1. DBMS Component Modules



- Database & DBMS catalog are usually stored in the disk. Access to the disk is primarily controlled by the OS.
- Stored data manager module of the DBMS controls access to the DBMS information stored on disk.
- DBA staff works on defining the databases & making changes to the DDL commands & priviledged commands. Then the

DDL compiler processes DDL commands & stores description of schema in DBMS catalog.

- Casual Users interact with database using some form of interfaces, which we call as, interactive query interface. Then the query compiles them into an internal form. This internal query is subjected to query optimization.
- Application Programmers write program in host languages eg- Java, C, COBOL, that are subjected to precompiler The precompiler extracts DML commands from an application program ~~with~~ written in a host language The commands are then sent to DML compiler for compilation into object code for DB. access.
- Parametric Users supply the parameters to the canned transactions so can be run repeatedly as separate transactions.
- Runtime DB processor executes the priviledged commands, executable query plans & the canned transactions with runtime parameters. It works with other modules & handles buffer management also.

2. Database System Utilities

DB system utilities help the DBA to manage the db. system. Types of functions that common utilities have -

1. Loading - To load existing data files into the database
2. Backup - It creates a backup copy of the database, usually by dumping the ~~dos~~ database into tape.
3. Database storage reorganisation - Used to recognize a set of db files into a different file organisation to improve performance.
4. Performance monitoring - Provides monitoring of db

usuage & ~~stastisti~~ statistics to the DBA.

Client-Server Architectures for DBMS

1. Centralized DBMS Architecture

In this DBMSs, all the DBMS functionality, application program execution and user face processing were carried out on one machine. Gradually, DBMS system started to exploit the available power at user side & led to client/server DBMS architectures.

2. Basic Client/Server Architecture

This architecture was developed to deal with computing environments in which a large no. of PCs, workstations, file servers, database servers are connected via a n/w.

16/01/2017

Relational Data Model

- Collection of tables
- Tuples ⇒ rows & attributes ⇒ column
- Degree of Table = no. of attributes

Key - An attribute or set of attributes which can uniquely identify the record.

- Keys can be more than one, & ~~so~~ are known as candidate keys.
- The key which is selected to uniquely identify ~~the~~ the records is called primary key.

Domain - Range of valid values for a particular attribute.

Characteristics of Relational Model

1. Records can be entered in any order.
2. Values within a single record must be in order.
3. NULL values should be avoided, as it causes ~~am~~ more ambiguity.
4. Each value is atomic.

Constraints in Relational Data Model (Schema Based)

1. Domain constraint
2. Key constraint
3. Integrity constraint (cannot have null values for keys)
4. Refrential integrity constraint

18/1/17

Relational Model

Constraints

- Model Based (inherent in a data model)
  - Duplicacy of data records
  - Inconsistency
- Schema Based (related with the schema of the relation)
  - Domain Constraint
  - Key Constraint
  - Entity Integrity Constraint
  - Referential Integrity Constraint
- Application Based (Depends on user)

- Create Table in mysql

```
create table student (name varchar(30),
                      rollno int,
                      : );
```

- Inserting

```
insert into student values ("megha", 150418, ----
```

Super key ⇒ (Roll, name, course)

Primary key ⇒ Roll

↓

Minimal Super key.

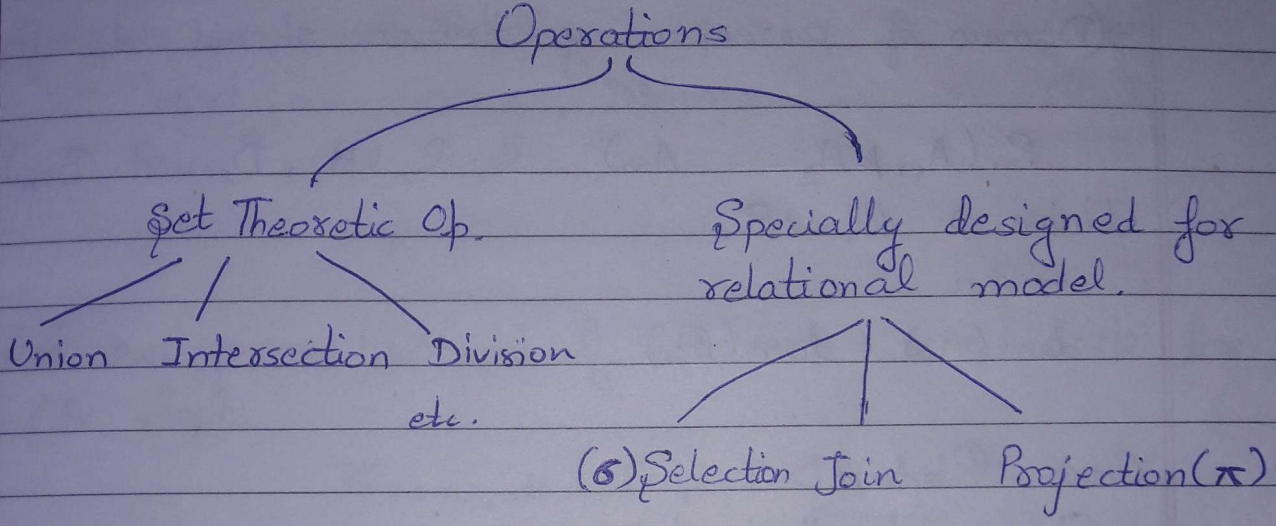
```
create table student(name varchar(30),
                    roll int AUTO_INCREMENT,
                    age int);
```

19/1/17

(CH.6)

Relational Algebra & Calculus

Basic set of op. which facilitates the user to access relational model.



Specially designed op. :

1. Selection $\sigma(R)$ (Selects a subset of R)

eg - $\sigma_{\text{course} = \text{"CS"} \text{ AND age} < 18}$ (Student)

2. Projection (displays selected attribute

eg - $\pi_{\text{name, roll}}$ (Student)

★ Combination of selection & projection

$$\pi\left(\sigma(R)\right)$$

eg - $\pi_{\text{name, age}}\left(\sigma_{\text{course} = \text{"CS"} \text{ AND age} < 18}(\text{Student})\right)$

⇒ select name, age
from student
where course = "CS" AND age < 18 ;

20/Jan/17

Set Operations

1. Union
- Check union compatibility ⇒(Degree of relations should be same)
⇒(Domain of corresponding attribute should be same)

$R_1(A_1, A_2 - - - A_m)$ & $R_2(B_1, B_2 - - - B_n)$

Union ⇒ $R_1 \cup R_2$

$m = n$

$dom(A_i) = dom(B_i)$ $\forall$ @ $1 <= i <= n$

2. Cross Product
- Gives redundant data

$R_1 \times R_2$

* Join ⇒ Subset of cross Product

$R_1 \bowtie_{cond^n} R_2$ $\quad$ $R_1 \bowtie_{A_i \theta B_i} R_2$ ← Theta join.

eg- Dept $\bowtie_{DNO = DNUM}$ EMP.

$R_1 \bowtie_{A_i \theta B_i} R_2$

Can be a single cond^n or an exp.

@ if θ = is "=" then join is called equijoin

Natural Join → When R_1 & R_2 are joined on the equality of 2 attributes with same name then the common attribute comes once & condition is not required.

$R_1 * R_2$

```
Select  Name , DName
From    emp , dept ;
where   Dno = Dnum ;
```

If Dno is common, then

```
Select  Name , DName
From    emp , dept ;
where   emp.Dno = emp.Dno ;
```

1. Retrieve name & add of all emp. who work for CS dept.

$$\Pi_{name, add} (Emp \bowtie_{Dname = "CS"} Dept)$$

2. Emp. who do not work for comp. sci. dept.

~~$\Pi_{name} (Emp \bowtie Dept)$~~

$$\Pi_{name} \left(\sigma_{Dname != "CS"} (Emp \bowtie Dept) \right)$$

23/1/2017

EMP

FNAME	LNAME	SSN	BDATE	SALARY	ADD	SEX	SUPERSSN	DNO

DEPT

~~NAME~~ DName	DNUM	MGRSSN	MGRSTARTDATE

DEPT_LOC

DNUMBER	DLOC

PROJECT

PNAME	PNUM	PLOC	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPEDENTNAME	SEX	BDATE	RELATIONSHIP

1. Retrieve name & add of all emp. who works for research dept.
2. For every project located in Bombay list project no, controlling dept no. & manager's name & add.
3. ~~Find name of emp.~~ Make a list of project no. for projects that involves "Smith" (lname) either as emp or manager for dept. that controls the ~~proj~~ dept.
4. Name of emp who have no dependents
5. Name of managers who have atleast one dependent

1. $\pi_{FNAME, ADD.}\left(EMP \bowtie_{DNO=DNUM}\left(\sigma_{DNAME="Research"}(DEPT)\right)\right)$

2. BOMBAY_PRO. $\leftarrow \left(\sigma_{PLOC="Bombay"}(PROJECT)\right)$

CONT_DEPT $\leftarrow$ (BOMBAY_PRO * DEPT)

MNGR $\leftarrow$ (CONT_DEPT $\bowtie_{MGRSSN=SSN}$ EMP)

RESULT $\leftarrow \pi_{PNUM, DNUM, FNAME, LNAME, ADD}$ (MNGR)

3. SMITH_EMP $\leftarrow \left(\sigma_{LNAME="Smith"}(EMP)\right)$

SMITH_SSN $\leftarrow \pi_{SSN}$ (SMITH_EMP)

SMITH_PNO $\leftarrow \pi_{Pno}$ (SMITH_SSN $\bowtie_{ESSN=ESSN}$ WORKS_ON)

4. EMP_DEP(ESSN, DEP) $\leftarrow$ ESSN $\mathcal{F}_{count(DEPENDENT\ NAME)}$ (DEPENDENT)

NO_DEP $\leftarrow \pi_{ESSN}\left(\sigma_{DEP=0}(EMP\_DEP)\right)$

RESULT $\leftarrow \pi_{NAME}$ (NO_DEP $\bowtie_{ESSN=SSN}$ EMP)

5. EMP_DEP (ESSN, DEP) $\leftarrow \left(ESSN\ \mathcal{F}_{count(DEPENDENT\ NAME)}(DEPENDENT)\right)$

EMP_1_DEP $\leftarrow \pi_{ESSN}\left(\sigma_{ESSN>0}(EMP\_DEP)\right)$

8

Page No.
Date:

$MGR\_SSN \leftarrow \pi_{MGRSSN}(DEPARTMENT)$

$MGR\_1\_DEP(SSN) \leftarrow (EMP\_1\_DEP \cap MGR\_SSN)$

$RESULT \leftarrow \pi_{NAME, SSN}(MGR\_1\_DEP * EMP)$

6.2 # Relational Algebra Operations from Set Theory

1. UNION

- Union is applied to the relations which must have same type of tuples. Two relations are said to be union compatible if they have the same degree n & if $dom(A_i) = dom(B_i)$ for $1 \leq i \leq n$.
- Denoted by R ∪ S, it includes all tuples that either in R or S or in both. Duplicate tuples are eliminated

example

To retrieve SSN of all emp. who either work for dept. 5 or directly supervise an emp. of dept 5.

$$DEPT5 \leftarrow \left(\sigma_{DNO=5}(EMP)\right)$$

$$SSN\_DEPT5 \leftarrow \left(\pi_{SSN}(DEPT5)\right)$$

$$SUPERSSN5(SSN) \leftarrow \left(\pi_{SUPERSSN}(DEPT5)\right)$$

(RENAME)

$$RESULT \leftarrow SSN\_DEPT5 \cup SUPERSSN5$$

2. INTERSECTION

- R ∩ S, includes all tuples that are in both R & S

  $R \cap S = S \cap R$

3. MINUS

- R − S, includes all tuples that are in R but not in S.

  $R - S \neq S - R$

$$\star \left(R \cap S = (R \cup S) - (R - S) - (S - R)\right)$$

4. CROSS PRODUCT

$R(A_1, A_2 \dots A_n) \times Q(B_1, B_2, \dots B_m) = S$

S should have $(m+n)$ attributes.

$|R| = n_R$

$|Q| = n_Q$

Then S have $n_R * n_Q$ tuples.

6.3 Binary Relational Operations

1. JOIN

- Join is a subset of Cartesian Product.
- $R \bowtie_{\text{Join condn}} S$

The result of join has $n+m$ attributes.

* In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the cartesian product all combinations of tuples are included in the result.

Join

- Equijoin (only comparison operator is used)
  eg- $R \bowtie_{A=B} S$
- Natural Join (2 attributes must have same name in the relations)
  eg- $R \bowtie_{A=A} S = R * S$

30/1/17

Relational Algebra

Aggregate functions & grouping

Applied on a group.

ex- Maximum salary from EMP Table & avg. Salary

$$\mathcal{J}_{\text{Max(Salary), Avg(Salary)}} (\text{EMP})$$

Symbol for aggregate function.

To count total no. of employees

~~count(SSN)~~

$$\mathcal{J}_{\text{count(SSN)}} (\text{EMP})$$

Some other aggregate f^n are sum(...), min(...), max(...) etc.

Grouping

$$\Rightarrow \quad {}_{\text{DNO}}\mathcal{J}_{\text{count(SSN)}} (\text{EMP})$$

Count no. of emp in each dept.

$$\boxed{\langle \text{Grouping Attribute} \rangle \mathcal{J}_{\langle \text{Aggregate f}^n \rangle} (\langle \text{Table} \rangle)}$$

⇒ Show dept name also for above query

$$\text{EMP\_DNO} \leftarrow \left({}_{\text{DNO}}\mathcal{J}_{\text{count(SSN)}} (\text{EMP}) \right)$$

$$\text{EMP\_DNAME} \leftarrow \left(\text{DEPT} \bowtie_{\text{DNUM=DNO}} \text{EMP\_DNO} \right)$$

$$\text{RESULT} \leftarrow \pi_{\text{DNAME, COUNT(SSN)}} (\text{EMP\_DNAM})$$

⇒ List the name of all emp. with ~~one~~ 2 or more dependents

$$NO\_OF\_D(ESSN, DEP) \leftarrow \left({}_{ESSN}\mathcal{J}_{COUNT(DEPENDENT\_NAME)}(DEPENDENT) \right)$$

$$EMP\_2\_D \leftarrow \left(\pi_{ESSN} \left(\sigma_{DEP \geq 2} (NO\_OF\_D) \right) \right)$$

$$RESULT \leftarrow \left(\pi_{FNAME} \left(EMP \bowtie_{SSN = ESSN} EMP\_2\_D \right) \right)$$

3/Feb/17

DIVISION OPERATOR

R

A	B
a_1	b_1
a_2	b_1
a_3	b_1
a_1	b_2
a_2	b_2
a_3	b_2
a_2	b_3
a_3	b_4

S

A
a_1
a_2
a_3

$T \leftarrow R \div S$.

(S should be a proper subset of R)

T

B
b_1
b_2

⇒ Retrieve name of all emp. who work on projects "John" works on.

$$JOHN\_EMP \leftarrow \pi_{SSN} \left(\sigma_{FNAME = "John"} (EMP) \right)$$

$$PNO\_JOHN \leftarrow \pi_{PNO} \left(WORKS\_ON \bowtie_{ESSN = SSN} JOHN\_EMP \right)$$

$$R_3 \leftarrow \pi_{ESSN, PNO} (WORKS\_ON)$$

$$R_4 \leftarrow R_3 \div R_2$$

Result ← π_{FName} (R_4 * Emp)

⇒ Display name of dept. & no. of emp.

EMP_NO ← ($_{DNO}\mathcal{J}_{count(SSN)}$ (EMP))

Result ← ($\pi_{DNAME, COUNT(SSN)}$ (DEPT $\bowtie_{DNUM = DNO}$ (EMP_NO)))

⇒ Find names of all emp who r directly supervised by "XYZ".

⇒ Retrieve a

SSN_SUP ← π_{SSN} ($\sigma_{FNAME = "XYZ"}$ (EMP))

SUPER ← $\pi_{SSN, SUPERSSN}$ (EMP)

RESULT(NAME) ← π_{FNAME} (SSN_SUP $\bowtie_{SSN = SUPERSSN}^{EMP}$)

⇒ & Name of emp who are not working on any project

WORK_EMP ← π_{SSN} (WORKS_ON)

ALL_EMP ← π_{SSN} (EMP)

NOT_WORKING ← ALL_EMP − WORK_EMP

RESULT ← π_{NAME} (NOT_WORKING * EMP)

⇒ Count & display no. of employees in each dept.

$_{Dno}\mathcal{J}_{count(SSN)}$ (EMP)

⇒ Retrieve name of emp. who do not work on any project

WORKING_EMP(SSN) ← π_{ESSN} (WORKS-ON)

ALL_EMP ← π_{SSN} (EMP)

NOTWORKING ← ALL_EMP − NOTWORKING

RESULT $\leftarrow \pi_{FNAME}$ (NOTWORKING * EMP)

⇒ Find average salary of all female employees.

$R_{(AVG\_SALARY)} \leftarrow \mathcal{J}_{avg(salary)} (\sigma_{sex = "F"} (EMP))$

⇒ Retrieve the names of all employees who work on every project.

R1 $\leftarrow \pi_{ESSN, PNO}$ (WORKS_ON)

R2 $\leftarrow \pi_{Pnumber}$ (PROJECT)

R3 (SSN) $\leftarrow$ R1 ÷ R2

RESULT $\leftarrow \pi_{FNAME}$ (R3 * EMP)

⇒ List the names of all department managers who have no dependents.

MGRS(SSN) $\leftarrow \pi_{MGRSSN}$ (DEPT)

MGRS_DEP $\leftarrow \pi_{SSN}$ (MGRS * DEPENDENT) (WORKS_ON)

MGR_NO_DEP $\leftarrow$ MGRS − MGRS_DEP

RESULT $\leftarrow \pi_{FNAME}$ (MGR_NO_DEP * EMP)

Exercises

Ques-21 Consider the given library database & write relational expressions for the queries :-

BOOK

Book_id	Title	Publisher_name

BOOK_AUTHORS

Book_id	Author_name

PUBLISHER

Name	Address	Phone

BOOK_COPIES

Book_id	Branch_id	No_of_copies

BOOK_LOANS

Book_id	Branch_id	Card_no	Date_out	Date_due

LIBRARY_BRANCH

Branch_id	Branch_name	Address

BORROWER

Card_no	Name	Address	Phone

a) How many copies of the book titled "The Lost Tribe" are owned by lib branch whose name is "Sharpstown"?

$$R1 \leftarrow \pi_{Book\_id}\left(\sigma_{Title = \text{"The Lost Tribe"}}(BOOK)\right)$$

$$R2 \leftarrow \pi_{Branch\_id}\left(\sigma_{Branchname = \text{"Sharpstown"}}(LIBRARY\_BRANCH)\right)$$

$$R3 \leftarrow (R1 * BOOK\_COPIES)$$

$$R4 \leftarrow (R2 * R3)$$

$$RESULT \leftarrow \pi_{No\_of\_copies}(R4)$$

b) How many copies of the book titled The Lost Tribe are

owned by each branch?

$$R1 \leftarrow \pi_{Book\_id}\left(\sigma_{Title = \text{"The lost Tribe"}}(BOOK)\right)$$

$$RESULT \leftarrow \pi_{Book\_id,\ Branchid,\ No\ of\ Copies}(R1 * BOOK\_COPIES)$$

c) Retrieve the names of all borrowers who do not have any books checked out.

$$ALL\_BORROWERS \leftarrow \pi_{card\_no}(BORROWER)$$

$$BORROWED \leftarrow \pi_{cardno}(BOOK\_LOANS)$$

$$NOT\_BORROWED \leftarrow ALL\_BORROWERS - BORROWED$$

$$RESULT \leftarrow \pi_{Name}(NOT\_BORROWED * BORROWERS)$$

d) For each book that is loaned out from the Sharpstown branch & whose Date_due is today, retrieve the book title, the borrower's name, & address of borrower.

$$R1 \leftarrow \pi_{Branchid}\left(\sigma_{Branchname = \text{"Sharpstown"}}(LIBRARY\_BRANCH)\right)$$

$$R2 \leftarrow \sigma_{Due\_date = today()}(R1 * BOOK\_LOANS)$$

$$R3 \leftarrow (R2 * BOOK)$$

$$RESULT \leftarrow \pi_{Title,\ Name,\ Address}(R3 * BORROWER)$$

e) For each library branch, retrieve the branch name & total number of books loaned out from that branch.

$$R1(Branchid,\ Books\ loaned) \leftarrow \left({}_{Branchid}\mathcal{J}_{count(Book\_id,\ card\_no)}(BOOK\_LOANS)\right)$$

$$RESULT \leftarrow \pi_{Branch\_name,\ Books\ loaned}(R1 * LIBRARY\_BRANCH)$$

f) Retrieve the names, addresses & number of books checked out for all borrowers who have more than five books checked out.

$R1(Cardno, BookIssued) \leftarrow {}_{Cardno}\mathcal{I}_{count(Book\text{-}id,\ Branchid)}(BOOK\_LOANS)$

$R2 \leftarrow \sigma_{BookIssued > 5}(R1)$

$RESULT \leftarrow \pi_{Name,\ Address,\ BookIssued.}(R2 * BORROWERS)$

g) For each book authored by Stephen King, retrieve the title & no. of copies owned by the library = "Central".

$KING\_BOOK \leftarrow \pi_{Book\text{-}id}(\sigma_{Authorname = "Stephen King"}(BOOKS))$

$CENTRALID \leftarrow \pi_{Branchid}(\sigma_{Branchname = "Central"}(LIBRARY\_BRANCH))$

$R1 \leftarrow (KING\_BOOK * BOOK\_COPIES)$

$R2 \leftarrow (R1 * CENTRAL\_ID)$

$RESULT \leftarrow \pi_{Title,\ No\_of\_Copies}(R2 * BOOK)$

Ques - 25 Consider T1 & T2. Show results of -

T1

P	Q	R
10	a	5
15	b	8
25	a	6

T2

A	B	C
10	b	6
25	c	3
10	b	5

a) $T1 \bowtie_{T1.P = T1.A} T2$

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3

b) $T1 \bowtie_{Q=B} T2$

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5

c) $T1 \bowtie_{P=A} T2$

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3
15	b	8	NULL	NULL	NULL

d) $T1 \bowtie_{Q=B} T2$

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5
NULL	NULL	NULL	25	c	3

e) $T1 \cup T2$

P	Q	R
10	a	5
15	b	8
25	a	6
10	b	6
25	c	3
10	b	5

f) $T1 \bowtie_{P=A \text{ AND } R=C} T2$

P	Q	R	A	B	C
10	a	5	10	a	5

Outer Join

Outer join is used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of JOIN.

- FULL OUTER JOIN - ⟗ Keeps all tuples in both the left & right relations when no matching tuples are found.
- LEFT OUTER JOIN - ⟕ Keeps every tuple of left relation R, in R ⟕ S; if no matching tuples are found in S.
- RIGHT OUTER JOIN - ⟖ Keeps a every tuple of right relation S, in R ⟖ S; if no matching tuples are found in R.

Outer Join Union

- Takes union of tuples of 2 relations if the relations are not union compatible, completely (i.e partially compatible).

STUDENT

Name	ID	Rank
XYZ	5	1
ABC	8	2
DEF	18	4

FACULTY

Name	ID	Salary
F	171	48,000
G	981	60,000
H	205	62,000

Rank & Salary are not compatible else all are.

STUDENT ∪ FACULTY ⇒

Name	ID	Rank	Salary
XYZ	5	1	~~48,000~~ NULL
ABC	8	2	~~60,000~~ NULL
DEF	18	4	~~62,000~~ NULL
F	171	NULL	48,000
G	981	NULL	60,000
H	205	NULL	62,000

(CH-3)

Relational Databases & ER Relational Mapping Model

Entity - Relationship Model (ER Model)

Entity → Object of concern which have either physical or conceptual existance
eg- Student, Dept.

Attribute - Property that describes an ~~enti~~ entity.
eg- Employee's name, age, sex may be attributes for an EMPLOYEE entity.
- An attribute can be simple (atomic) or composite.
- Direct or derived
- Single Valued or multivalued.

Entity Type - ~~Coll~~ Collection of entities that have the same attributes. eg- COMPANY & EMPLOYEE

Entities

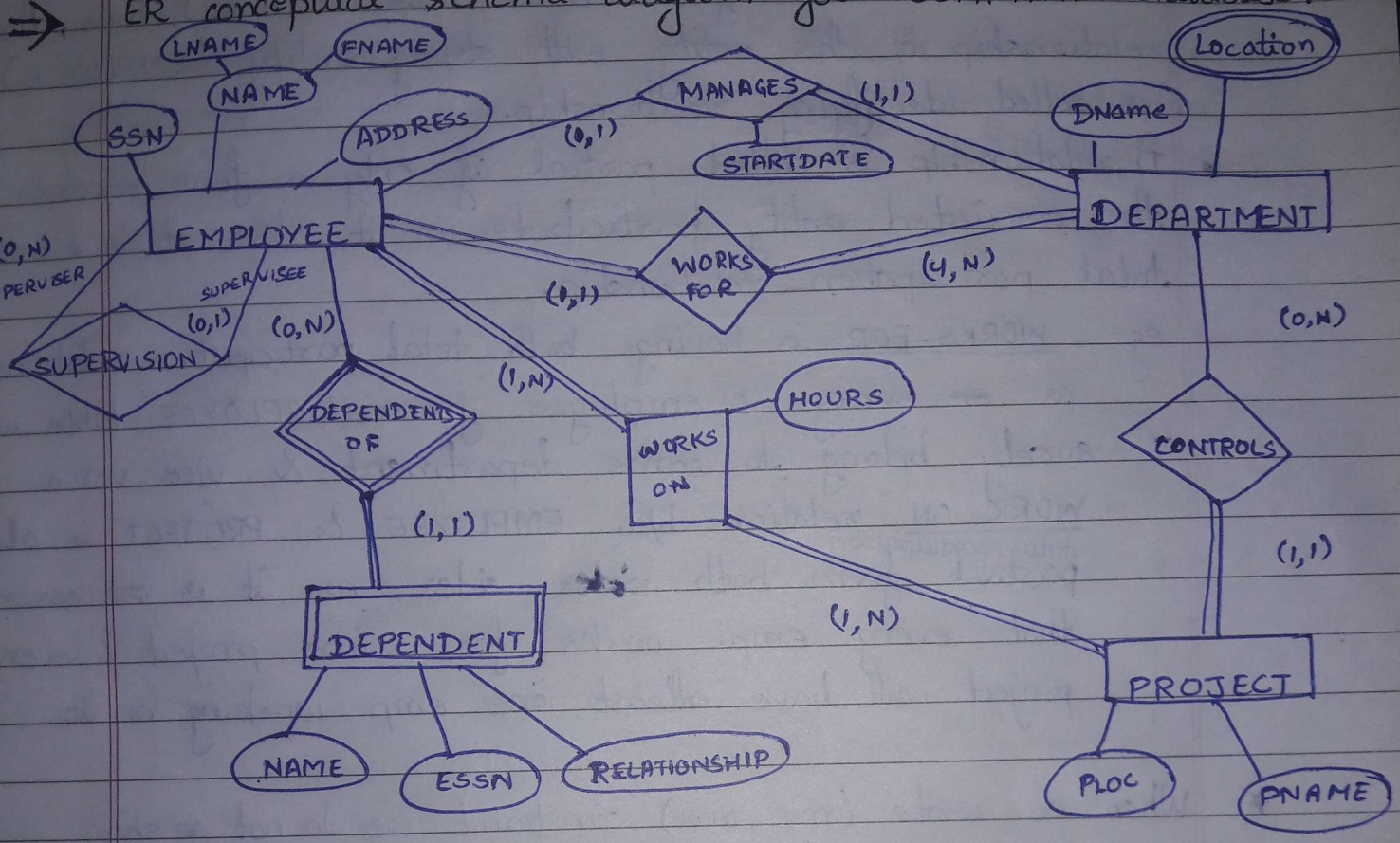
Weak	Strong
Have no recognisation as individual entity. eg- No. of Dependents entity	eg- Employee

Cardinality

Relationships - How entities are associated is defined by their relationship.
- Degree of a relationship type = no. of participating entity types
- Cardinality ration = Maximum no. of relationship instances that ~~any~~ an entity can participate in

The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1 & M:N.

⇒ ER conceptual schema diagram for COMPANY database.



	Entity	EMPLOYEE, PROJECT
	Weak entity	DEPENDENT
	Relationship	WORKS_FOR
	Identifying relationship (used for weak entities)	DEPENDENTS-OF
	Total participation of entities	~~CONTROLS~~ WORKS_FOR
	Partial participation	~~WORKS_ON~~ MANAGES
(a,b)	Multivalued attribute ~~Card~~ Structural ~~Cards~~ Constraint (min, max)	location

- SURERVISION is a recursive relationship, also called reflexive association.
- DEPENDENT is a weak entity as it cannot exist on its own (i.e. has no recognisation in this database). Thus, the relationship of this entity with strong entity (EMPLOYEE) is called identifying relationship.
- A relationship is called partial if only a few records of the associated entity participate in it else it is called total participation relationship.

  eg- WORKS-FOR is having both total participation relations as every employee from EMPLOYEE table will surely belong to some department & vice versa.

  WORK_ON relation b/w EMPLOYEE & PROJECT is also total participation from both sides, as it is essential that every emp. works for some project & every project will have atleast one emp working on it.

* When we write (min, max) constraint we do not show complete (total) or partial participation.

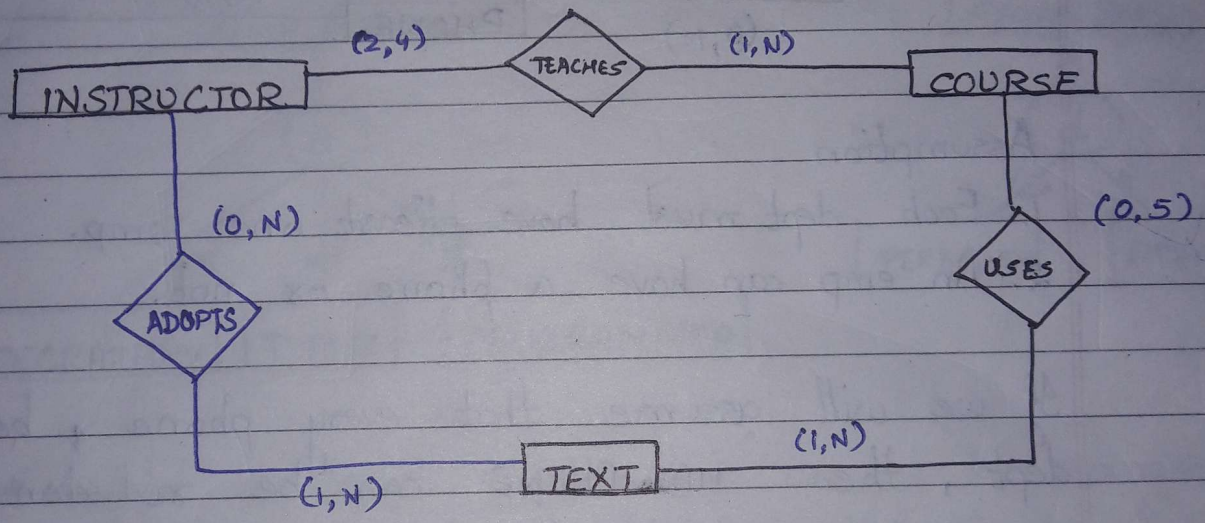
30/2/17

(CH-7)

Ques- 7.25

Practice Question

Assume that a course may or may not use a textbook, but that a text by definition is a book that is used in some course. A course may or may not use more than 5 books. Instructors teach from 2 to 4 courses. Supply (min, max) constraints. State clearly any additional assumption if you make. If we add the relationship **ADOPTS** b/w **INSTRUCTOR** and **TEXT** & add (min, max) constraint.



Assumptions -

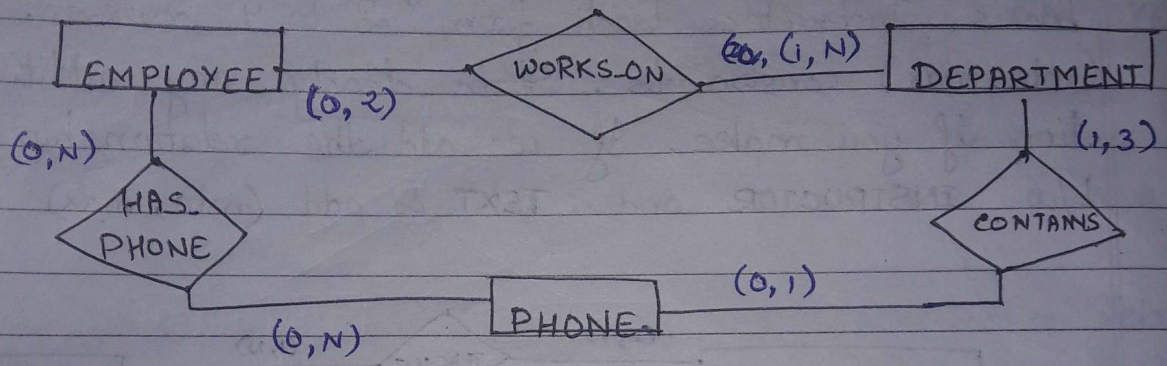
1. An instructor may use a text or not. And can use any no. of texts.
2. As given a text is used in any one of the course, this implies atleast one instructor is using the text & also any number of instructors can use it.

27/2/2017

Page No.
Date:

Ques - 7.24

Assume that an emp. may work in up to two departments or may not be assigned to any. Each dept. must have one or up to 3 phones. Supply (min, max) constraints. Under what cond" HAS-PHONE be redundant?



Assumption

1. Each dept must have atleast one emp.
2. An emp can have a phone or not.

If we will assume that every phone belongs to any dept, then HAS-PHONE can be ~~reduden~~ redundant.

Ques - 7.28

Refer - MOVIES DB (Pg - 240)

a) False

b) ~~True~~ May be

c) True

d)

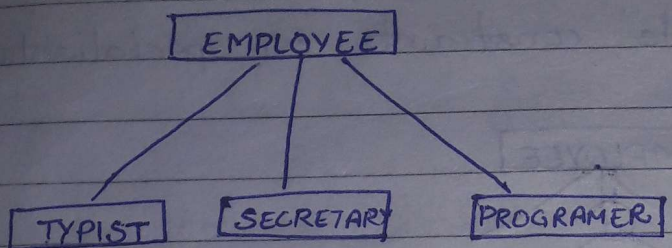
1/3/17

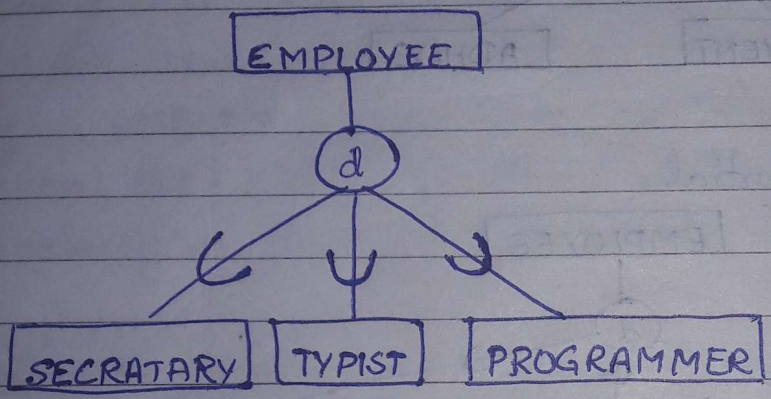
(CH-8)

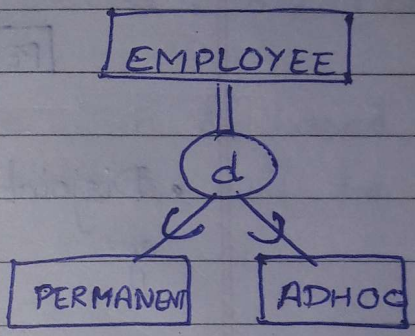
The Enhanced Entity - Relationship (EER) Model

Generalization / Specilization ≡ Subclass / Class concept.

eg-







d → disjoint constraint (an adhoc cannot be permanent)
= → Completeness constraint (total participation)

Classes Categories in specilisation-

- Disjoint or overlapping — Disjointness constraints
- Total or Partial — Completeness constraint

Specialisation - The process of defining a set of subclasses of an entity type.
eg- {SECRETARY, TYPIST, PROGRAMMER} is a specialisation of EMPLOYEE.
The attributes that apply only to entities of a particular subclass, are termed as specific attributes.

Generalisation - The process of defining a generalised type from the given entity type.

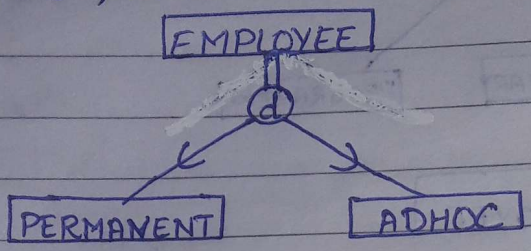
Page No.
Date:

- Constraints on specialisation & generalisation
  1. Disjointness constraints (disjoint, overlapping)
  2. Completeness constraints (total, partial)

Hence, 4 possible constraints on specialisation -

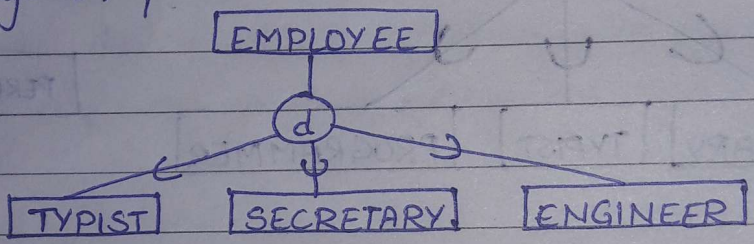
- Disjoint, total

eg-



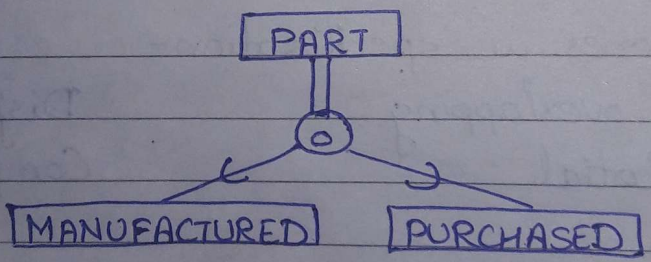
- Disjoint, partial

eg-



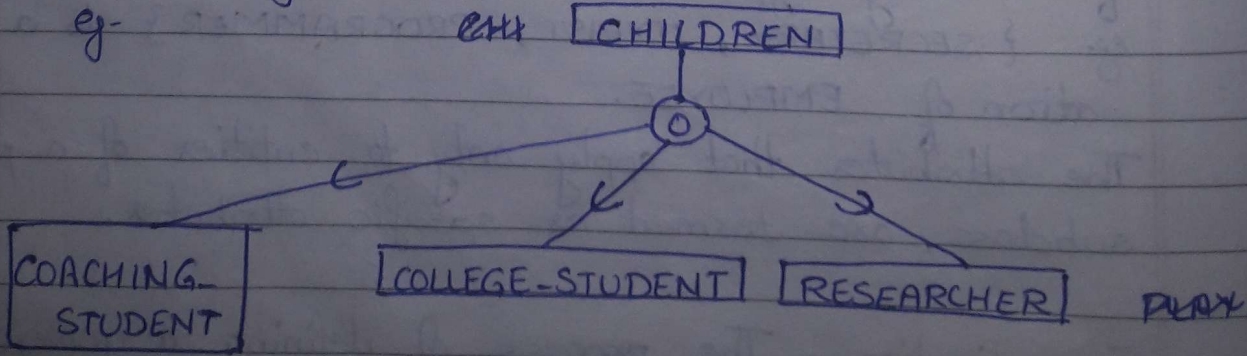
- Overlapping, total

eg-



- Overlapping, partial

eg-



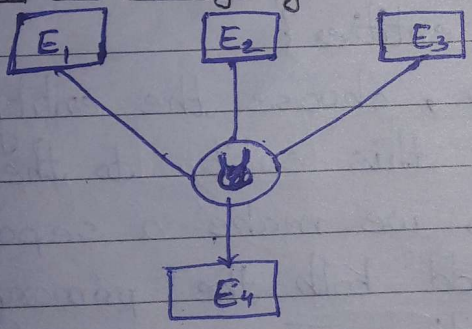
3/3/17

- Insertion
Inserting a new entity must be done in parent (class) entity first, then in subclass.

- Deletion
Entity must be deleted from subclass first

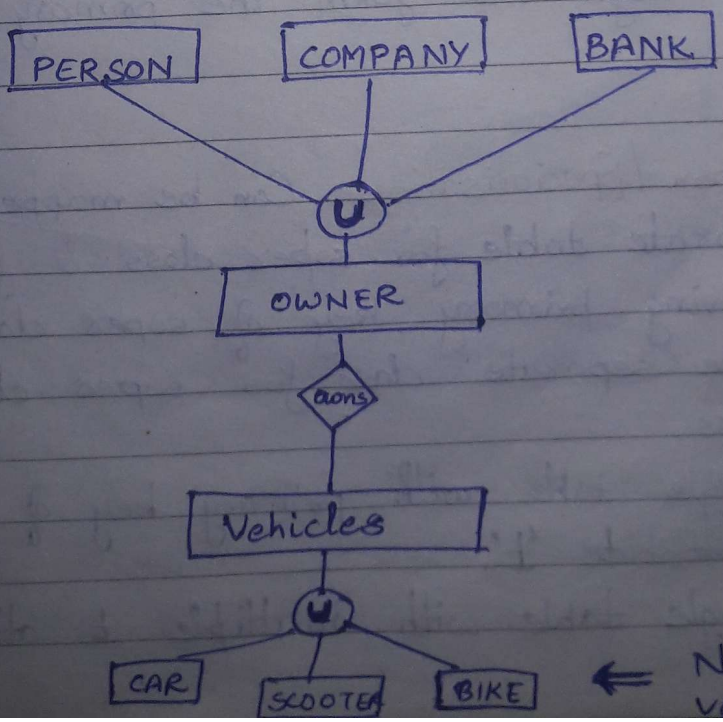
Predicate defined / attribute defined specialisation
- Entity is added to a particular subclass, based on certain cond^n.
- eg- EMPLOYEE can have an attribute JOB & based on the value of this attribute, entities will be classified in diff. subclasses.

Union or Category



$E_4 \subset (E_1 \cup E_2 \cup E_3)$

Example



⇐ Not inherited from vehicles.

9/3/2017

(CH-4)

ER- & EER-to-Relational Mapping

Considering EMPLOYEE Database. Map the database to relational model.

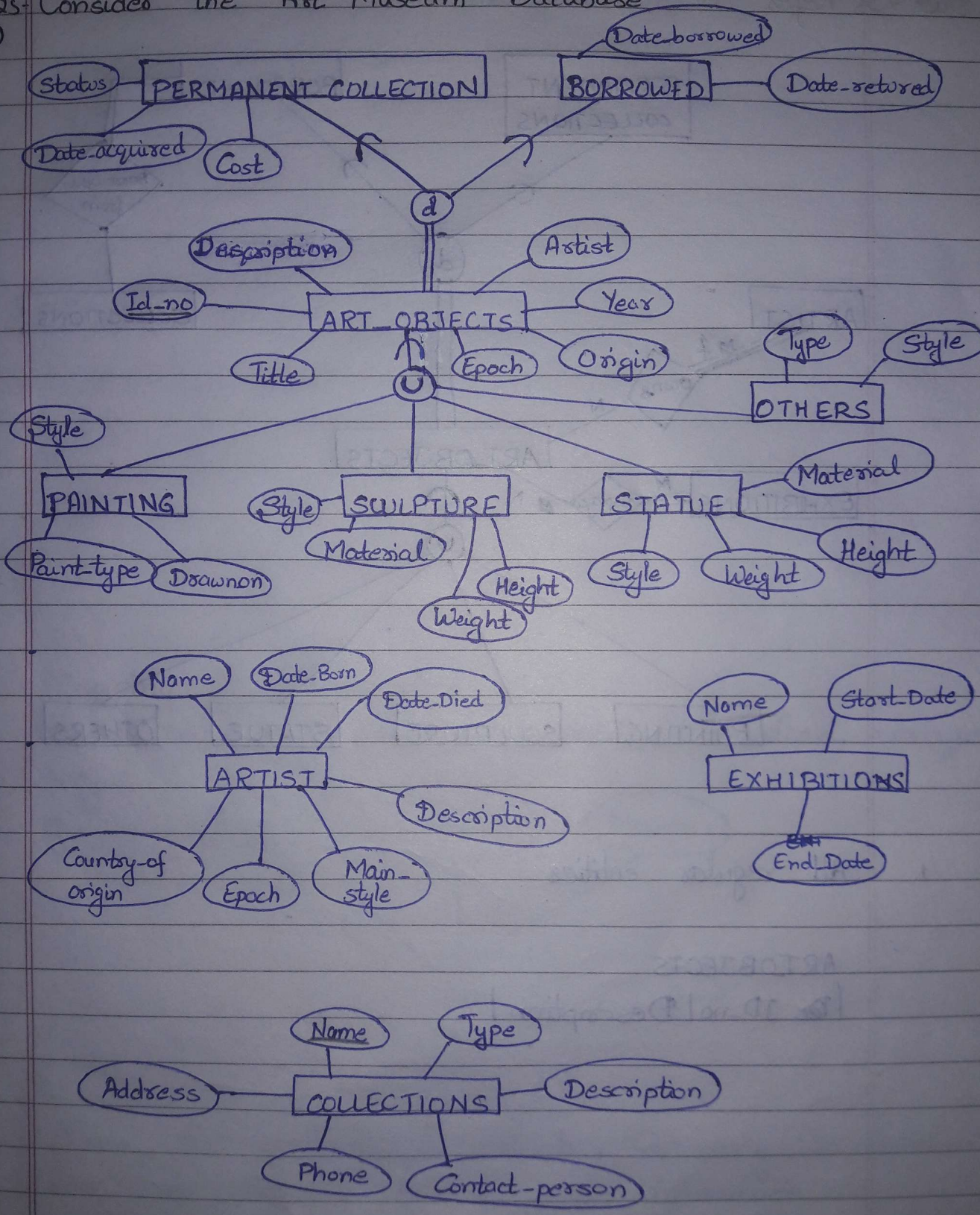
1. Write all the regular entities along with simple attributes of the entity.
2. Write weak entities & find the primary key & add the primary key of owner entity.
3. Make a relation for multivalued attributes.
4. Consider the relationships.
   - For 1:1 relationship, choose the entity with total participation & add the primary key of other entity to it & also add other attributes related to the relation (if any)
     {If both entities have total participation, then simply merge both the entities}
   - For 1:N relationship, choose the entity with N side Add primary key of this entity to the other entity
   - For M:N relationship, we make a separate table. And to this table add both the primary keys of both participating entities. The combination of these primary keys will form the primary key of this relation.

Generalisation / Specialisation. Can be mapped in any of 4 ways.

1. Make separate table for super class & for each of its subclass having primary key of super class in each.
2. Don't make separate class for super class merge it with subclass.
3. Make a single table with primary key of superclass & an additional attribute 't'.
4. Make a single table with multiple t attributes.

10/3/17

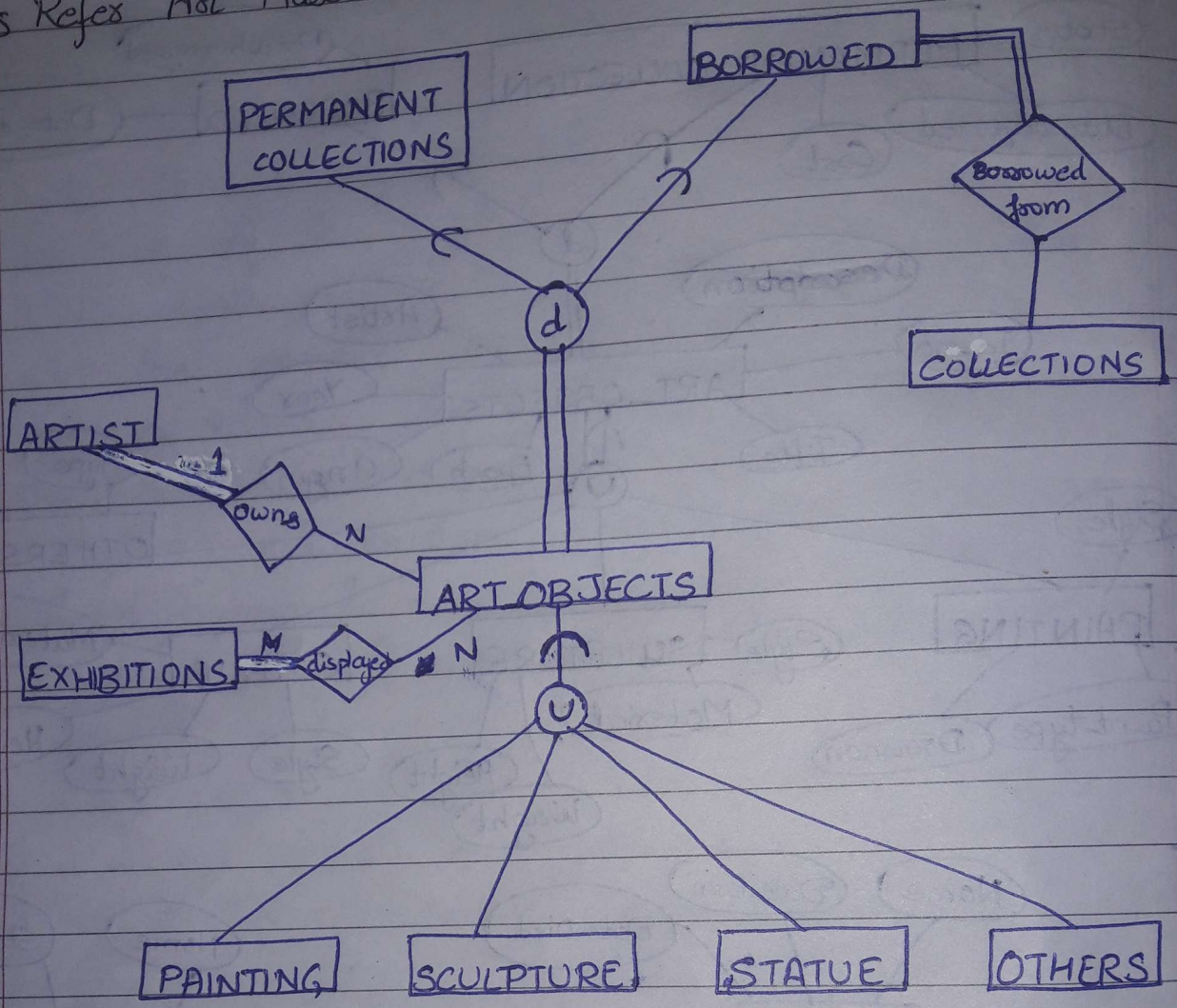
Ques 25- Consider the Art Museum Database
(4.25)



Entities in the database are - ART_OBJECTS, PAINTING, SCULPTURE, STATUE, OTHERS, PERMANENT_COLLECTIONS, BORROWED, COLLECTIONS, EXHIBITIONS, ARTIST

10/3/2017

Ques 4.25 Refer Art Museum Database



1. All regular entities

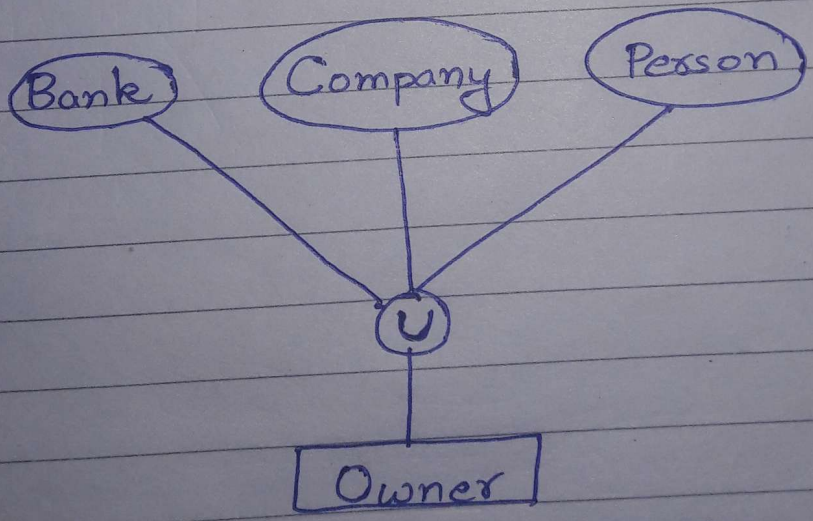
ART_OBJECTS

| ID_no | Description |

Mapping of Categorisation

Here, we use an additional attribute, called surrogate key.

eg-



Since, the ~~sub~~ classes Bank, Company & Person have no common attribute, so introduce an attribute (eg- Owner_Id to each class) and make a separate table with attributes Owner_Id & Type.