*Group By & Having Clauses;

- These clauses can be used where the Having query is executed on each group & if all elements satisfy, then that entire group is selected.

— * —

Note;

* DDL ⇒ Data Definition Language

* DML ⇒ Data Manipulation Language

* DCL ⇒ Data Control Language.

* (Grant or Revoke)

⇒ Rows also known as records or tuples

⇒ Columns also known as attributes or fields

- curs.fetchone()

  curs.dictfetchone() } Returns dictionary with values in column names.

— * —

ex; select avg(marks) from studies group by courseid

ex; select avg(marks) from studies group by courseid having count(rollno)>=

↑

Select average marks of course where more than 30 students are present

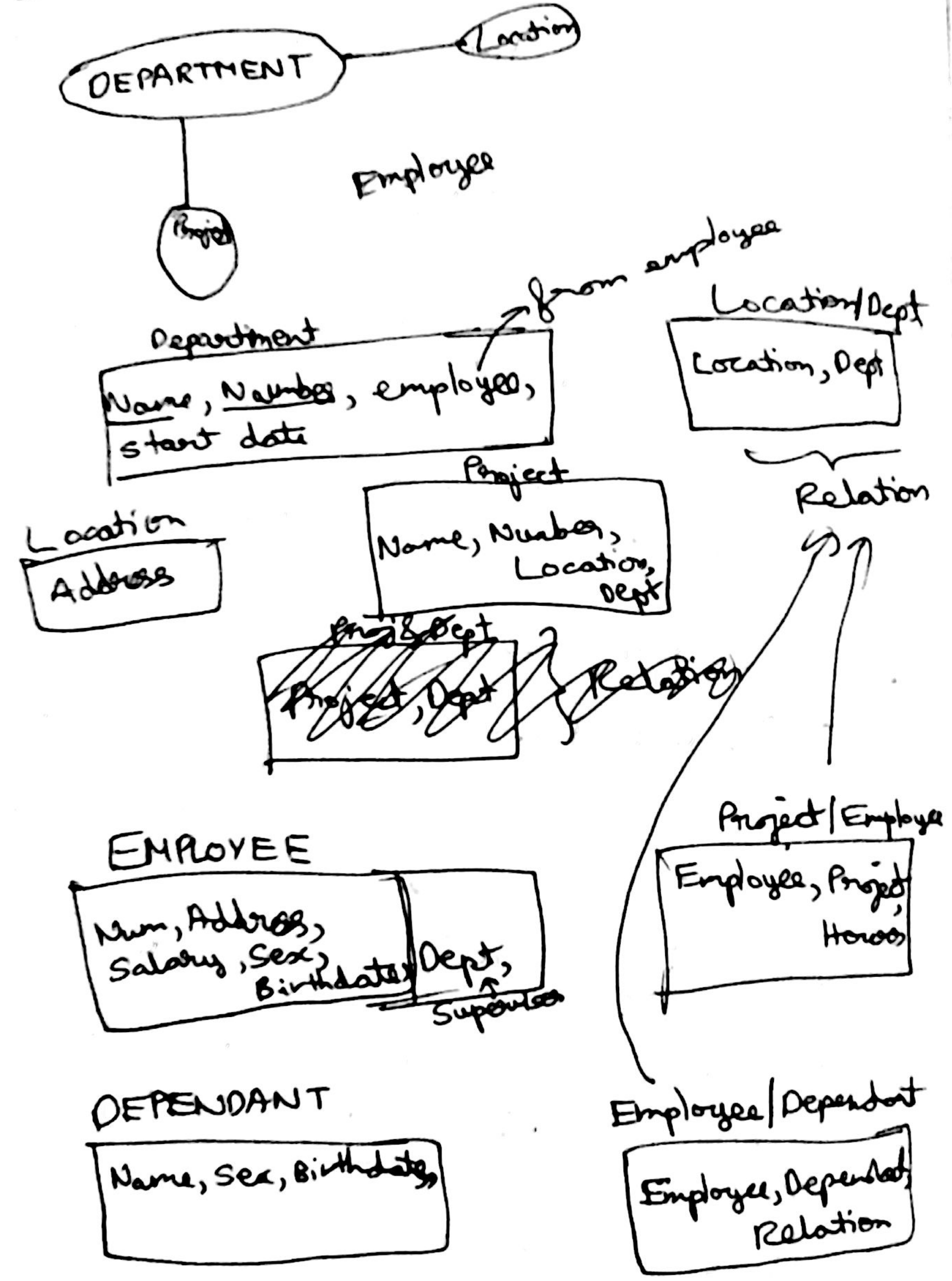
— * —

Scanned by CamScanner

Note; ER Diagrams } Entity-Relationship Diagram.

— x —

Company Database;



— x —

Types of Attributes;

1) Simple
2) Composite } List together describing one possibility
3) Multivalued } List of different possibilities,

——— × ———

Note; Joining 2 tables of same type can be done with aliases

⇒ ex; select e2.name from employee e1, ~~as~~ employee e2, supervision where e1.name ='Manoj' and supervision.supervisor=e1ssn and supervision.supervisee = e2.ssn

(Generally used for recursive relationships)
→ relationships b/w same type

Relationships;

- b/w 2 entities ⇒ Binary relationship
- b/w 3 entities ⇒ Ternery relationship

——— × ———

Note; Weak Entities;
- Entities with no primary key (They depend on the relationship as their identification)

——— × ———

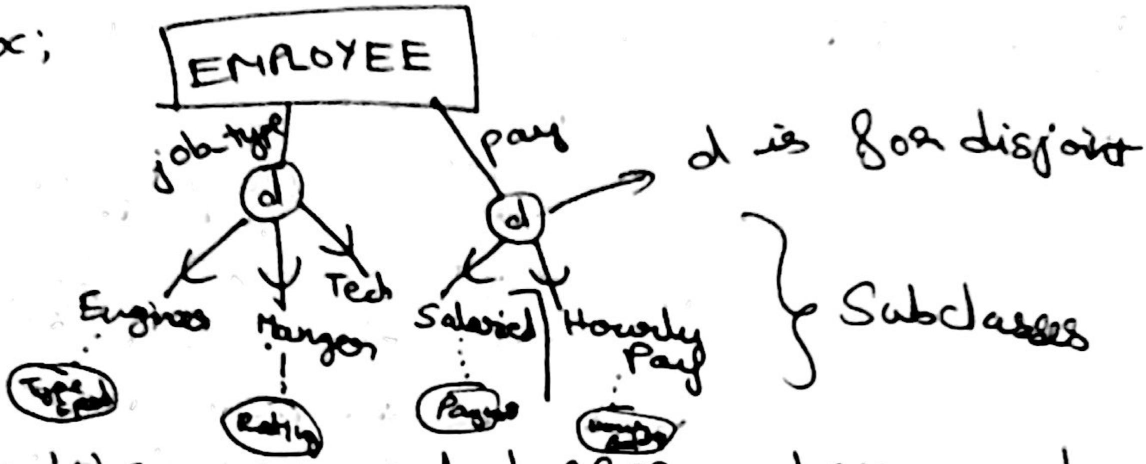
(Rajan Slides)

EER

(Enhanced Entity Relationship)
Modeling

⇒ Super Classes & Subclasses of entities can be done. (Specialization process)

ex;



d is for disjoint

Subclasses

- We use subclasses when each of the subclasses have attributes as well since we cannot just keep this subclass as an attribute to the super class.

⇒ In EER, all entities are called classes.

—— x ——

* d & o;

- disjoint constraint (d) is when the superc only belongs to one subclass.
- overlap constraint (o) is when its not disjoint.

—— x ——

Note; Single Inheritance ⇒ Tree structure
Multiple Inheritance ⇒ Lattice structure
(Exist in all superclasses)

⇒ ~~When~~

** Unions (of Superclasses)

- Its like multiple inheritance, but the subclass must belong to only one superclass.

——x——

Note; 1:1 relationships generally dont need relation tables, but n:m do need relations tables.

——x——

* Superkey; Superset of the key (superkey)
* Key; Minimal superkey

——x——

Note; Candidate Keys ⇒ User chooses one of them as primary key.

——x——

Types of Databases;

1) O.O databases (Object Oriented)
2) O.R databases (Object Relational) (EER Model)
(Objects stored in tables)

——x——

* Note; SQL etc as R-databases (Relational)

——x——

Note; In SQL databases (we try normalize as much as we can (reduce repetitions))
⇒ In NoSQL or BigData (we try not to normalize)

→ If we normalize, then we need to join to get queries. (These joins are expensive in BigData)

———×———

Normalized Forms;

1) ⇒ Get Functional Dependancies;

X → Y {X determines Y} Functional Dependancy

ex; RollNo → Name,

RollNo, CourseID → grade, attendance

- These Dependancies are reflexive & transitive.

- **Augmentation;**

If X → Y then X, Z → Y, Z

- Let f be the set of functional dependancies, then f^+ is the closure, when we apply the above 3 rules multiple times

⇒ f^+ ⇒ Closure of f

⇒ ~~Minimal set is f such that for all subsets of f^+, its minimal.~~

Minimal Set;

⇒ Minimal set is when every dependancy

in X has a single attribute on right hand side.

⇒ We cannot remove any dependancy & get an equivalent set.

→ We canot replace $Y \to A$ with $X \to A$ such that $Y \subset X$.

—x—

★★★★ 2) Create Normal form; (SLIDES!!!)

a) 1'FN; (1'st Normal Form)

- No composite attributes
- No multivalued attributes
- No nested relations

b) 2'nd NF;

- All attributes (non prime) should depend on entire (every candidate key) key.
- Prime attribute is a member of the primary key.

Transitively is fine as well.

- If $Y \to Z$, If we remove either A,B or C
  $\downarrow$
  $A,B,C \to Z$
  then the dependancy breaks, then this is a full functional dependacy. (ie. Only dependancy on key)

★ • Every non-prime attribute is fully dependant on primary key.

c) 3'rd NF;

- 2'nd NF and no non-prime & (Non-candidate key) attribute is transitively dependant on key- (ie. if in 2'nd NF we have $X \to Y$, $Y \to Z$ here we should have $X \to Y$, $X \to Z$)

—x—

d) BCNF : If X ⟶ A (Boyce Codd Normal Form)
then X is a Superkey in our relation

—— × ——

Note ; Relational Calculus etc , QBE etc } Altern math

—— × ——

* Note ; Relational Algebra ((Complete set of Operators)

—— × ——

* Note ; NOSQL (Big Data)

—— × ——