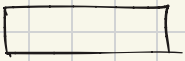
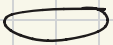
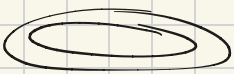
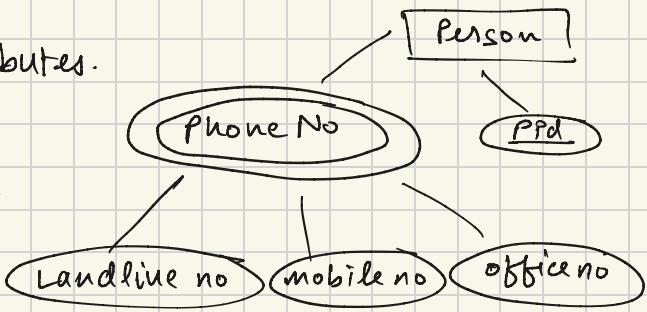
ER diagram.

 ← Entity set (eg. set of all persons)
↖ we make a table for persons

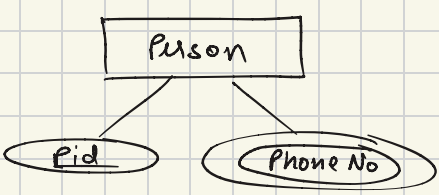
 ← attributes ⟹ columns in the table of the entity

Note: Multivalued attributes.

 eg ⟹



- Create new table with the name of phoneNo and has columns landline, mobile and office no.s and one primary key pid from the entity's corresponding table.




```
CREATE TABLE Person_Phone (
    Person_id INT,
    PhoneNo VARCHAR(15),
    CONSTRAINT comp-pk PRIMARY KEY (person_id, PhoneNo),
    CONSTRAINT fk_id FOREIGN KEY (Person_id)
    REFERENCES Person (Person_id) ON DELETE CASCADE);
```

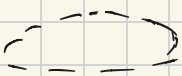
One way

Person_id	LandlineNo	MobileNo	officeNo

Table = PhoneNo

second way

Person_id	PhoneNo	Type
1	123456	Landline No
2	978356	Mobile No
3	- - - -	office No.

 ← Derived attribute

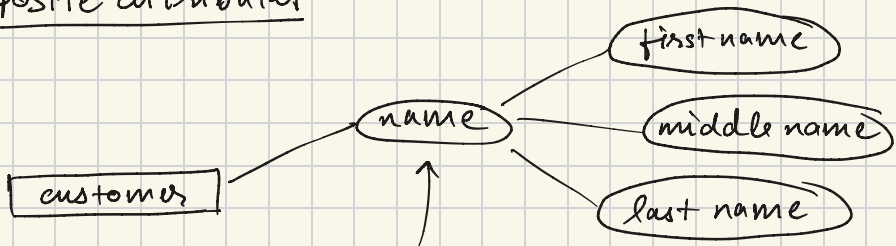
Something that can be calculated using the existing columns of the table.

eg:

- Age = today's date - date of birth
- Total marks = Compre marks + Midsem Marks + Quiz marks
- Salary = no. of working days * hourly wages.

↑ we dont make columns for derived attributes in the table., we can create view.

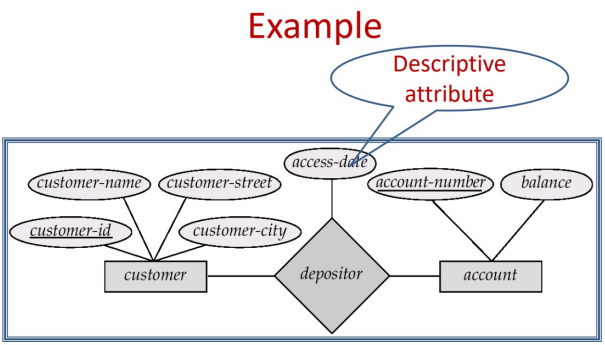
composite attributes



not a multivalued attribute

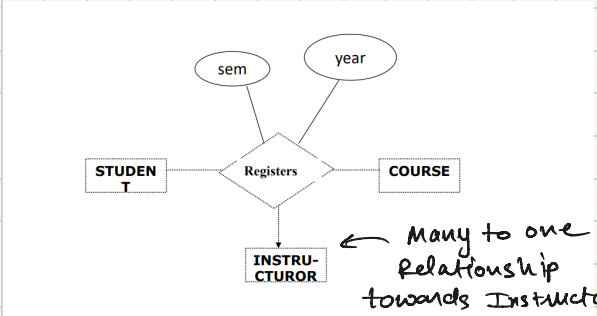
- instead of name make columns for first name, middle name, last name

Relationships



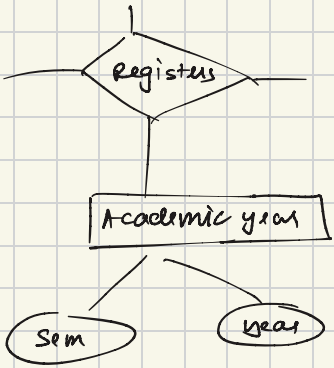

```sql
-- SQL code for ER diagram 1
CREATE TABLE customer (
    customer_ID INT,
    customer_name VARCHAR(10),
    customer_street VARCHAR(50),
    customer_city VARCHAR(50),
    CONSTRAINT pk_customer_id PRIMARY KEY(customer_ID)
);

CREATE TABLE account (
    account_number INT,
    balance INT,
    CONSTRAINT pk_acc_no PRIMARY KEY(account_number)
);

CREATE TABLE depositor (
    customer_id INT,
    account_number INT,
    access_date DATE,   -- the descriptive attribute
    -- composite primary key for this table
    PRIMARY KEY (customer_id, account_number),
    CONSTRAINT fk_acc_no FOREIGN KEY(account_number)
        REFERENCES account(account_number)
        ON DELETE CASCADE,
    CONSTRAINT fk_customer_id FOREIGN KEY(customer_id)
        REFERENCES customer(customer_ID)
        ON DELETE CASCADE
);
```

many to many relationship b/w customer and account.
so we create a new table depositor.



← Many to one Relationship towards Instructor

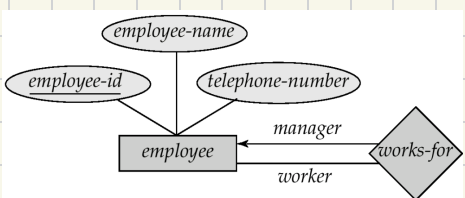
← in this we can make a new table




```sql
-- SQL code for Relationship
CREATE TABLE Registers (
    student_id INT,
    course_id INT,
    instructor_id INT,
    sem VARCHAR(10),
    year INT,
    -- primary key is student id, course_id, not instructor_id
    -- as we want students, course combination to have one instructor only
    PRIMARY KEY(student_id, course_id),
    -- now the 3 foreign keys
    CONSTRAINT fk_student_id FOREIGN KEY(student_id)
        REFERENCES students(student_id) ON DELETE CASCADE,
    CONSTRAINT fk_course_id FOREIGN KEY(course_id)
        REFERENCES courses(course_id) ON DELETE CASCADE,
    CONSTRAINT fk_instructor_id FOREIGN KEY(instructor_id)
        REFERENCES instructors(instructor_id) ON DELETE CASCADE
);
```

For total participation we add (not null).

example of Unary Relationship



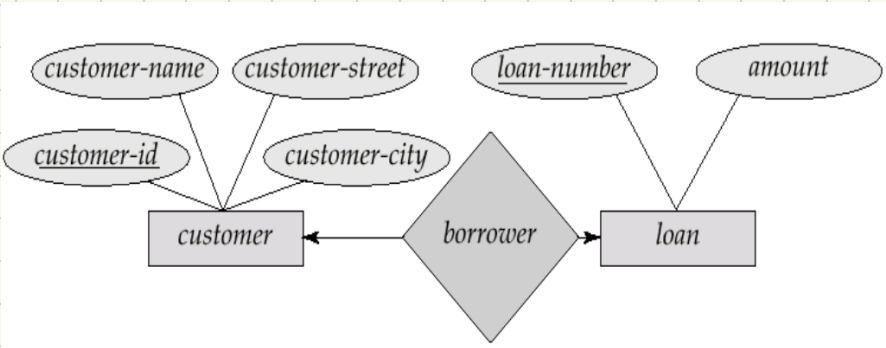

```sql
-- SQL code for unary relationship
CREATE TABLE employee (
    employee_id INT,
    employee_name VARCHAR(50),
    telephone_number VARCHAR(15),
    works_for INT
    PRIMARY KEY(employee_id),
    -- foreign key within the same table as we demonstrate many to one relationship
    CONSTRAINT fk_emp_id FOREIGN KEY(works_for)
        REFERENCES employee(employee_id)
        ON DELETE SET NULL
    -- If a manager leaves, the worker just has no manager.
);
```

Recurrence Relationship.

can also add NOT NULL constraint to emp_name.

- Many employees can have one Manager. ← so no new table is created
- For 1:1 relationship (one employee has only one manager) then we put "UNIQUE" constraint one work for column.

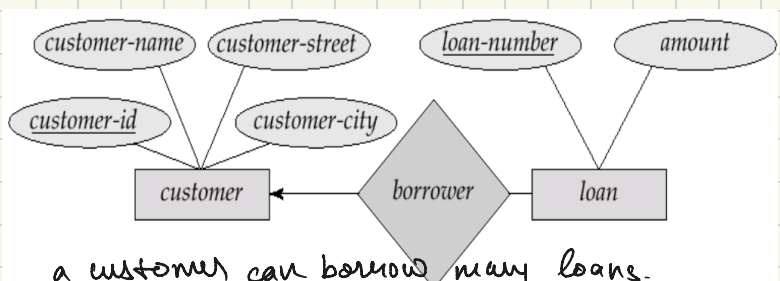
Note: 1: is part for employee side is taken care by primary key and to maintain 1:1 we put unique constraint.



customer is associated with at most 1 loan and vice versa

```sql
-- SQL code for one to one relationship
CREATE TABLE loan (
    loan_number INT PRIMARY KEY,
    amount DECIMAL(10, 2),
);

CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(50),
    customer_street VARCHAR(50),
    customer_city VARCHAR(10),
    loan_number INT UNIQUE -- unique constraint to enforce 1 to 1 relationship
    -- foreign key
    CONSTRAINT fk_loan_no FOREIGN KEY(loan_number)
        REFERENCES loan(loan_number)
);
```

Many to one Relationship



a customer can borrow many loans.

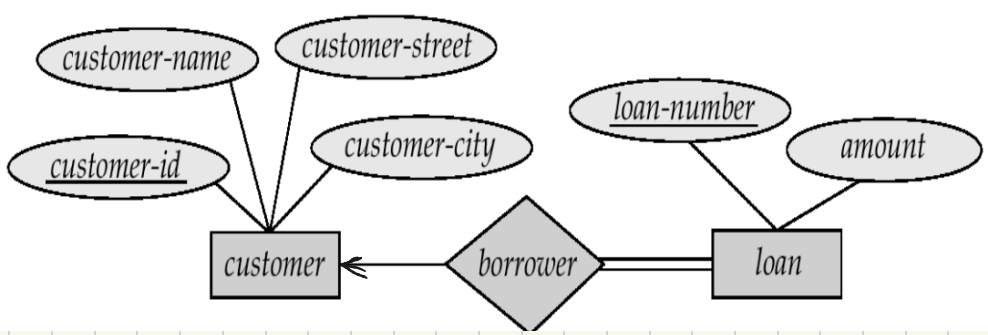
loans ⇒ M customer ⇒ 1

↳ foreign key is in Loans column

```sql
-- SQL code for Many to one relationship
CREATE TABLE customer (
    cutomer_id INT,
    customer_name VARCHAR(50),
    customer_street VARCHAR(50),
    customer_city VARCHAR(10),
    CONSTRAINT pk_customer PRIMARY KEY(customer_id)
);

CREATE TABLE loan (
    loan_number INT PRIMARY KEY,
    amount DECIMAL(5, 2),
    customer_id INT,
    -- foreign key is added in this table as A customer can borrow many loans
    CONSTRAINT fk_customer_id FOREIGN KEY(customer_id)
        REFERENCES customer(customer_id)
        ON DELETE CASCADE
    -- on deleting customer we should technically delete the loan
);
```

Very important

Total participation



loan has a total participation

Every loan must have a customer associated with it.

```sql
-- SQL code for total participation of loan
CREATE TABLE customer (
    cutomer_id INT,
    customer_name VARCHAR(50),
    customer_street VARCHAR(50),
    customer_city VARCHAR(10),
    CONSTRAINT pk_customer PRIMARY KEY(customer_id)
);

CREATE TABLE loan (
    loan_number INT PRIMARY KEY,
    amount DECIMAL(5, 2),
    customer_id INT NOT NULL,
    -- The not null constraint ensures that the each loan is associated to a customer
    CONSTRAINT fk_customer_id FOREIGN KEY(customer_id)
        REFERENCES customer(customer_id)
        ON DELETE CASCADE
);
```

↑ Here the not null indicates total participation.

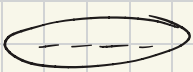
═══ denotes total participation

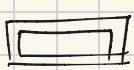
——— denotes partial participation

↑ Eg customer may or may not borrow a loan.

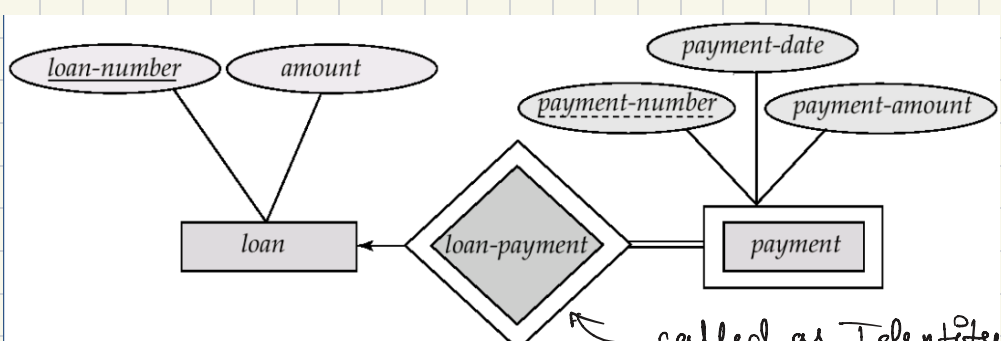
Weak Entity set

primary key of weak entity set is called discriminator.



Weak Entity set ⟹  ← cannot be standalone.

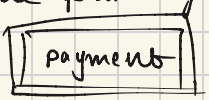
↳ has total participation always.



↑ called as Identifying Relationship

```sql
-- SQL Code for Weak Entity Sets
CREATE TABLE loan (
    loan_number INT PRIMARY KEY,
    amount DECIMAL(5, 2)
);

CREATE TABLE payment (
    payment_number INT,
    payment_date DATE,
    payment_amount DECIMAL(5, 2),
    loan_number INT NOT NULL,
    -- Not null for total partication i.e. every payment must be associated to a loan
    -- The primary key of a weak entity set is the discriminator and the primary key of foregin key table
    PRIMARY KEY (loan_number, payment_number),
    -- foreign key constraint
    CONSTRAINT fk_loan FOREIGN KEY(loan_number)
        REFERENCES loan(loan_number)
        ON DELETE CASCADE
);
-- It is a 1(Loan) to M(payment) relationship (One to Many)

```

Note: The primary key of the Weak Entity set is



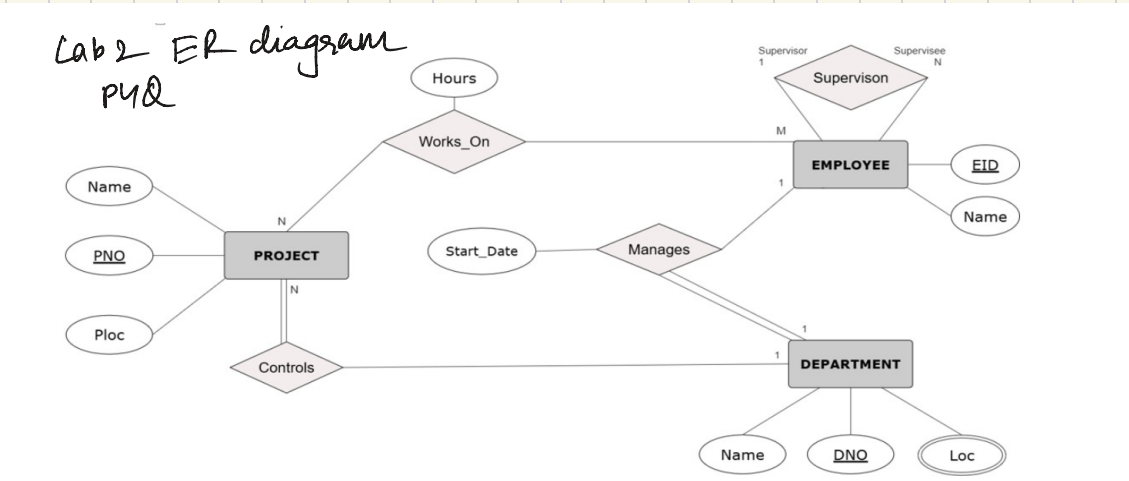
is a composite of loan number, payment number.

Composite PRIMARY KEY of parent table's primary key and discriminator.

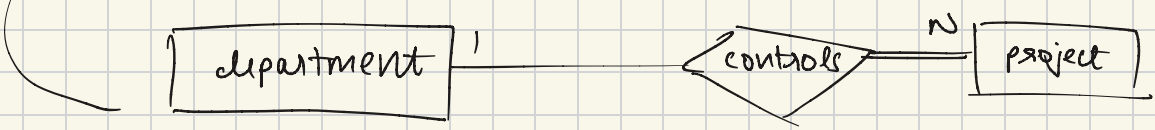
called as identifying entity set (loan in this case)

Final Question

Lab 2 ER diagram PYQ



1 department can have many N projects.
So, foreign key goes in projects table.



1 : Many

Conditions in Question

1. start_date ⟶ Date, not null
2. Employee cannot supervise himself
3. hours ⟶ decimal of precision of 5 and scale of 2 (Decimal(5,2)) not null, b/w 10 and 40 (inclusive).

SQL code for the PYQ Question.

```sql
-- PYQ Lab2 Question
CREATE TABLE EMPLOYEE (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(10),
    Supervision INT,
    CONSTRAINT chk1 CHECK Supervision != employee_id, -- Condition: Employee cannot be his own Supervisor
    CONSTRAINT fk_supervisor FOREIGN KEY(Supervision)
        REFERENCES EMPLOYEE(employee_id) ON DELETE SET NULL
        -- we will set null if the supervisor is deleted
);

CREATE TABLE DEPARTMENT (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(20),
    Manages INT NOT NULL UNIQUE,  -- 1:1 Total Participation so we add NOT NULL and UNIQUE
    Start_date DATE NOT NULL,  -- not null acc to the question
    CONSTRAINT fk_employee_id FOREIGN KEY(Manages)
        REFERENCES EMPLOYEE(employee_id) ON DELETE CASCADE
    -- on delete cascade makes sense beacuse of total participation of department with employee
);

-- create a new table for multivalued attribute of the DEPARTMENT table
CREATE TABLE Loc (
    department_id INT,
    Loc VARCHAR(20),
    -- we have the primary key as composite key of department_id and Loc itself as its members are unknown
    PRIMARY KEY(department_id, Loc),
    CONSTRAINT fk_department_id FOREIGN KEY(department_id) REFERENCES DEPARTMENT(department_id)
        ON DELETE CASCADE  -- in this cascade is needed as this is a part of the department table
);
```

```sql
CREATE TABLE PROJECT (
    project_id INT PRIMARY KEY,
    project_name VARCHAR(10),
    ploc VARCHAR(10),
    Controls INT NOT NULL,  -- not null due to the total participation of project
    CONSTRAINT fk_department_id FOREIGN KEY(Controls) REFERENCES DEPARTMENT(department_id)
        ON DELETE CASCADE
    -- cascade makes sense as project has total participation in the relationship with department
);

CREATE TABLE Works_On (
    employee_id INT,
    project_id INT,
    Hours DECIMAL(5, 2) NOT NULL, -- as we are applying a check on it (not null mentioned in Q)
    -- check if hours is between 10 and 40 inclusive
    CONSTRAINT chk2 CHECK (Hours >= 10 AND Hours <= 40),
    -- employee id and project id are the composite primary key as it is Many To Many Relationship
    PRIMARY KEY(employee_id, project_id),
    -- forign keys
    CONSTRAINT fk_emp_id FOREIGN KEY(employee_id)
        REFERENCES EMPLOYEE(employee_id) ON DELETE CASCADE,
    CONSTRAINT fk_proj_id FOREIGN KEY(project_id)
        REFERENCES PROJECT(project_id) ON DELETE CASCADE
    -- on delete cascade makes sense as the entry in this extra table should be removed
    -- if the employee or the project is removed
)
```