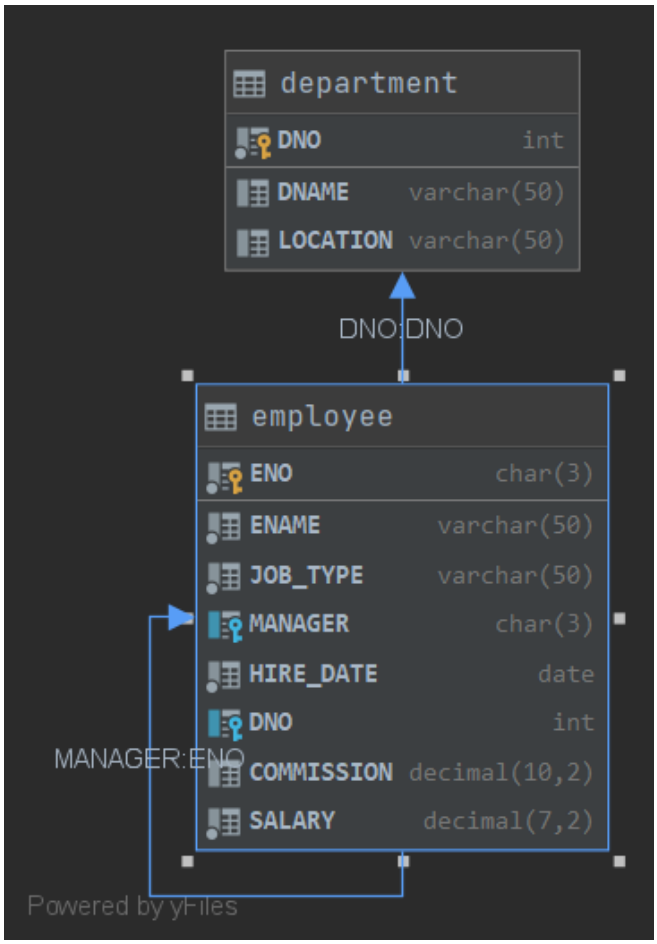
# Creating Schema – **COMPANYDB**




**creatingSchemas.sql**

```
CREATE SCHEMA COMPANYDB;
USE COMPANYDB;

-- EMPLOYEE TABLE
CREATE TABLE EMPLOYEE(
    ENO CHAR(3),
    ENAME VARCHAR(50) NOT NULL,
    JOB_TYPE VARCHAR(50) NOT NULL,
    MANAGER CHAR(3),
    HIRE_DATE DATE NOT NULL,
```

```
    DNO INT,
    COMMISSION DECIMAL(10,2),
    SALARY DECIMAL(7,2) NOT NULL,
    PRIMARY KEY(ENO)
);

-- DEPARTMENT TABLE
CREATE TABLE DEPARTMENT(
    DNO INT NOT NULL,
    DNAME VARCHAR(50),
    LOCATION VARCHAR(50) DEFAULT 'New Delhi',
    PRIMARY KEY(DNO)
);


-- POPULATING DEPARTMENT TABLE
INSERT INTO DEPARTMENT(DNO,DNAME,LOCATION) VALUES
    (1,'Research','Chennai'),
    (2,'Adminstration','New Delhi'),
    (7,'Headquaters','New Delhi');

-- POPULATING EMPLOYEE TABLE
INSERT INTO EMPLOYEE VALUES
    (001, 'Georgi Facello', 'Senior Engineer', NULL, '1986-06-
26', 1, 580, 60117),
    (911, 'Shay Casley', 'Senior Engineer', NULL, '1988-06-
25', 1, 555, 66074),
    (667, 'Makato Cusworth', 'Senior Staff', NULL, '1990-06-
25', 2, 736, 66961),
    (004, 'Chirstian Koblick', 'Engineer', NULL, '1986-12-
01', 1, NULL, 40054),
    (339, 'Tenius Liedekerke', 'Engineer', NULL, '1989-11-
30', 1, 200, 46065),
    (005, 'Kyoichi Maliniak', 'Staff', NULL, '1989-09-
12', 2, NULL, 58326),
    (882, 'Fun Heuser', 'CEO', NULL, '1994-11-29' ,7 ,1500, 78228);


-- ASSIGNING MANAGER
```

```
UPDATE EMPLOYEE SET MANAGER=667 WHERE JOB_TYPE = 'Staff';
UPDATE EMPLOYEE SET MANAGER=001 WHERE JOB_TYPE = 'Engineer';
UPDATE EMPLOYEE SET MANAGER=882 WHERE JOB_TYPE = 'Senior Engineer';
UPDATE EMPLOYEE SET MANAGER=882 WHERE JOB_TYPE = 'Senior Staff';

-- ALTERING THE CREATED TABLE TO ADD FOREIGN KEYS
ALTER TABLE EMPLOYEE ADD FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DNO);
ALTER TABLE EMPLOYEE ADD FOREIGN KEY(MANAGER) REFERENCES EMPLOYEE(ENO)
;
```

## Resulting Tables:

```
mysql> USE COMPANYDB;
Database changed
mysql> SHOW COLUMNS FROM EMPLOYEE;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| ENO        | char(3)       | NO   | PRI | NULL    |       |
| ENAME      | varchar(50)   | NO   |     | NULL    |       |
| JOB_TYPE   | varchar(50)   | NO   |     | NULL    |       |
| MANAGER    | char(3)       | YES  | MUL | NULL    |       |
| HIRE_DATE  | date          | NO   |     | NULL    |       |
| DNO        | int           | YES  | MUL | NULL    |       |
| COMMISSION | decimal(10,2) | YES  |     | NULL    |       |
| SALARY     | decimal(7,2)  | NO   |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
8 rows in set (0.12 sec)

mysql> SHOW COLUMNS FROM DEPARTMENT;
+----------+-------------+------+-----+-----------+-------+
| Field    | Type        | Null | Key | Default   | Extra |
+----------+-------------+------+-----+-----------+-------+
| DNO      | int         | NO   | PRI | NULL      |       |
| DNAME    | varchar(50) | YES  |     | NULL      |       |
| LOCATION | varchar(50) | YES  |     | New Delhi |       |
+----------+-------------+------+-----+-----------+-------+
3 rows in set (0.00 sec)

mysql>
```

# Queries:

31. Query to display the total number of supervisors without listing their names.

**Query**

```
SELECT COUNT(*) AS NO_OF_MANAGERS
FROM EMPLOYEE
WHERE ENO IN (SELECT DISTINCT MANAGER FROM EMPLOYEE);
```

**Output**

```
mysql> SELECT COUNT(*) AS NO_OF_MANAGER
    -> FROM EMPLOYEE
    -> WHERE ENO IN (SELECT DISTINCT MANAGER FROM EMPLOYEE);
+---------------+
| NO_OF_MANAGER |
+---------------+
|             3 |
+---------------+
1 row in set (0.00 sec)

mysql>
```

32. Query to display the Department Name, Location Name, No. of Employees and the average salary for all employees in that department.

**Query**

```
WITH SALARY_TABLE(DNO,AVERAGE_SALARY,NO_OF_EMPLOYEE) AS
    (
        SELECT DNO,AVG(SALARY),COUNT(*)
        FROM EMPLOYEE
        GROUP BY DNO
    )
SELECT D.*,S.NO_OF_EMPLOYEE,S.AVERAGE_SALARY
FROM DEPARTMENT AS D JOIN SALARY_TABLE AS S
ON D.DNO = S.DNO;
```

**Output**

```
mysql> WITH SALARY_TABLE(DNO,AVERAGE_SALARY,NO_OF_EMPLOYEE) AS
    ->     (
    ->         SELECT DNO,AVG(SALARY),COUNT(*)
    ->         FROM EMPLOYEE
    ->         GROUP BY DNO
    ->     )
    -> SELECT D.*,S.NO_OF_EMPLOYEE,S.AVERAGE_SALARY
    -> FROM DEPARTMENT AS D JOIN SALARY_TABLE AS S
    -> ON D.DNO = S.DNO;
+-----+--------------+----------+----------------+----------------+
| DNO | DNAME        | LOCATION | NO_OF_EMPLOYEE | AVERAGE_SALARY |
+-----+--------------+----------+----------------+----------------+
|   1 | Research     | Chennai  |              4 |   53077.500000 |
|   2 | Adminstration | New Delhi |              2 |   62643.500000 |
|   7 | Headquaters  | New Delhi |              1 |   78228.000000 |
+-----+--------------+----------+----------------+----------------+
3 rows in set (0.27 sec)

mysql>
```

33. Query to display Name and Hire Date for all employees in the same dept. as Blake.

**Query**

```
SELECT ENAME,HIRE_DATE
FROM EMPLOYEE
WHERE DNO IN (
    SELECT E.DNO
    FROM EMPLOYEE AS E
    WHERE E.ENAME LIKE '%Shay%'
);
```

**Output**

```
mysql> SELECT ENAME,HIRE_DATE
    -> FROM EMPLOYEE
    -> WHERE DNO IN (
    ->     SELECT E.DNO
    ->     FROM EMPLOYEE AS E
    ->     WHERE E.ENAME LIKE '%Shay%'
    -> );
+------------------+------------+
| ENAME            | HIRE_DATE  |
+------------------+------------+
| Georgi Facello   | 1986-06-26 |
| Tenius Liedekerke | 1989-11-30 |
| Chirstian Koblick | 1986-12-01 |
| Shay Casley      | 1988-06-25 |
+------------------+------------+
4 rows in set (0.00 sec)

mysql>
```

34. Query to display the Employee No. And Name for all employees who earn more than the average salary

**Query**

```
SELECT E.ENO,E.ENAME
FROM EMPLOYEE AS E
WHERE E.SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE);
```

**Output**

```
mysql> SELECT E.ENO,E.ENAME
    -> FROM EMPLOYEE AS E
    -> WHERE E.SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE);
+-----+-----------------+
| ENO | ENAME           |
+-----+-----------------+
| 1   | Georgi Facello  |
| 667 | Makato Cusworth |
| 882 | Fun Heuser      |
| 911 | Shay Casley     |
+-----+-----------------+
4 rows in set (0.02 sec)

mysql>
```

35. Query to display Employee Number and Name for all employees who work in a department with any employee whose name contains a ‘T’.

**Query**

```
SELECT E.ENO,E.ENAME
FROM EMPLOYEE AS E
WHERE E.DNO IN (
    SELECT DISTINCT DNO
    FROM EMPLOYEE
    WHERE ENAME LIKE '%T%'
);
```

**Output**

```
mysql> SELECT E.ENO,E.ENAME
    -> FROM EMPLOYEE AS E
    -> WHERE E.DNO IN (
    ->     SELECT DISTINCT DNO
    ->     FROM EMPLOYEE
    ->     WHERE ENAME LIKE '%T%'
    -> );
+-----+------------------+
| ENO | ENAME            |
+-----+------------------+
| 1   | Georgi Facello   |
| 339 | Tenius Liedekerke |
| 4   | Chirstian Koblick |
| 5   | Kyoichi Maliniak  |
| 667 | Makato Cusworth   |
| 911 | Shay Casley       |
+-----+------------------+
6 rows in set (0.00 sec)

mysql>
```

36. Query to display the names and salaries of all employees who report to supervisor named ‘Fun’

**Query**

```
SELECT E.ENAME,E.SALARY
FROM EMPLOYEE AS E
WHERE E.MANAGER IN
(SELECT ENO FROM EMPLOYEE WHERE ENAME LIKE '%Fun%');
```

**Output**

```
mysql> SELECT E.ENAME,E.SALARY
    -> FROM EMPLOYEE AS E
    -> WHERE E.MANAGER IN
    -> (SELECT ENO FROM EMPLOYEE WHERE ENAME LIKE '%Fun%');
+-----------------+----------+
| ENAME           | SALARY   |
+-----------------+----------+
| Georgi Facello  | 60117.00 |
| Makato Cusworth | 66961.00 |
| Shay Casley     | 66074.00 |
+-----------------+----------+
3 rows in set (0.00 sec)

mysql>
```

37. Query to display the department no, name and job for all employees in the Administration department.

**Query**

```
SELECT E.DNO,E.ENAME,E.JOB_TYPE
FROM EMPLOYEE AS E
WHERE E.DNO IN
(SELECT DNO FROM DEPARTMENT WHERE DNAME='Adminstration');
```

**Output**

```
mysql> SELECT E.DNO,E.ENAME,E.JOB_TYPE
    -> FROM EMPLOYEE AS E
    -> WHERE E.DNO IN
    -> (SELECT DNO FROM DEPARTMENT WHERE DNAME='Adminstration');
+------+------------------+--------------+
| DNO  | ENAME            | JOB_TYPE     |
+------+------------------+--------------+
|    2 | Kyoichi Maliniak | Staff        |
|    2 | Makato Cusworth  | Senior Staff |
+------+------------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

38. Display names of employees along with their department name who have more than 20 years' experience.

**Query**

```
SELECT E.ENAME,D.DNAME,E.HIRE_DATE,
       ROUND(DATEDIFF(NOW(),E.HIRE_DATE)/365) AS YEARS_EXP
FROM EMPLOYEE AS E JOIN DEPARTMENT AS D ON E.DNO = D.DNO
WHERE ROUND(DATEDIFF(NOW(),HIRE_DATE)/365) > 20;
```

**Output**

```
mysql> SELECT E.ENAME,D.DNAME,E.HIRE_DATE,
    ->         ROUND(DATEDIFF(NOW(),E.HIRE_DATE)/365) AS YEARS_EXP
    -> FROM EMPLOYEE AS E JOIN DEPARTMENT AS D ON E.DNO = D.DNO
    -> WHERE ROUND(DATEDIFF(NOW(),HIRE_DATE)/365) > 20;
+------------------+--------------+------------+-----------+
| ENAME            | DNAME        | HIRE_DATE  | YEARS_EXP |
+------------------+--------------+------------+-----------+
| Georgi Facello   | Research     | 1986-06-26 |        35 |
| Tenius Liedekerke| Research     | 1989-11-30 |        31 |
| Chirstian Koblick| Research     | 1986-12-01 |        34 |
| Shay Casley      | Research     | 1988-06-25 |        33 |
| Kyoichi Maliniak | Adminstration| 1989-09-12 |        31 |
| Makato Cusworth  | Adminstration| 1990-06-25 |        31 |
| Fun Heuser       | Headquaters  | 1994-11-29 |        26 |
+------------------+--------------+------------+-----------+
7 rows in set (0.00 sec)

mysql>
```

39. Display total number of departments at each location

**Query**

```
SELECT LOCATION,COUNT(*) AS NO_OF_DEPARTMENT
FROM DEPARTMENT
GROUP BY LOCATION;
```

**Output**

```
mysql> SELECT LOCATION,COUNT(*) AS NO_OF_DEPARTMENT
    -> FROM DEPARTMENT
    -> GROUP BY LOCATION;
+-----------+------------------+
| LOCATION  | NO_OF_DEPARTMENT |
+-----------+------------------+
| Chennai   |                1 |
| New Delhi |                2 |
+-----------+------------------+
2 rows in set (0.00 sec)

mysql>
```

40. Find the department name in which at least 2 employees work in.

**Query**

```
SELECT D.DNAME
FROM DEPARTMENT AS D
WHERE D.DNO IN (
    SELECT DNO
    FROM EMPLOYEE
    GROUP BY DNO
    HAVING COUNT(*) >= 2
);
```

**Output**

```
mysql> SELECT D.DNAME
    -> FROM DEPARTMENT AS D
    -> WHERE D.DNO IN (
    ->     SELECT DNO
    ->     FROM EMPLOYEE
    ->     GROUP BY DNO
    ->     HAVING COUNT(*) >= 2
    ->     );
+---------------+
| DNAME         |
+---------------+
| Research      |
| Adminstration |
+---------------+
2 rows in set (0.00 sec)

mysql>
```

41. Query to find the employee’ name who is not supervisor and name of supervisor supervising more than 5 employees.

**Query**

```
WITH SUPERVISOR(ENO) AS (
    SELECT ENO
    FROM EMPLOYEE
    WHERE ENO IN (SELECT E.MANAGER FROM EMPLOYEE AS E)
)
(
    SELECT E.ENO,E.ENAME
    FROM EMPLOYEE AS E
    WHERE E.ENO NOT IN (SELECT ENO FROM SUPERVISOR)
)
UNION
(
    SELECT E.ENO, E.ENAME
    FROM EMPLOYEE AS E,
         SUPERVISOR AS S
    WHERE E.MANAGER = S.ENO
    GROUP BY S.ENO
    HAVING COUNT(*) > 2
)
;
```

**Output**

```
mysql> WITH SUPERVISOR(ENO) AS (
    ->     SELECT ENO
    ->     FROM EMPLOYEE
    ->     WHERE ENO IN (SELECT E.MANAGER FROM EMPLOYEE AS E)
    -> )
    -> (
    ->     SELECT E.ENO,E.ENAME
    ->     FROM EMPLOYEE AS E
    ->     WHERE E.ENO NOT IN (SELECT ENO FROM SUPERVISOR)
    -> )
    -> UNION
    -> (
    ->     SELECT E.ENO, E.ENAME
    ->     FROM EMPLOYEE AS E,
    ->          SUPERVISOR AS S
    ->     WHERE E.MANAGER = S.ENO
    ->     GROUP BY S.ENO
    ->     HAVING COUNT(*) > 2
    -> );
+-----+-------------------+
| ENO | ENAME             |
+-----+-------------------+
| 339 | Tenius Liedekerke |
| 4   | Chirstian Koblick |
| 5   | Kyoichi Maliniak  |
| 911 | Shay Casley       |
| 1   | Georgi Facello    |
+-----+-------------------+
5 rows in set (0.00 sec)

mysql>
```

42. Query to display the job type with maximum and minimum employees.

**Query**

```
WITH JOB(JOB_TYPE,NO) AS
    (SELECT JOB_TYPE,COUNT(*)
    FROM EMPLOYEE
    GROUP BY JOB_TYPE)
SELECT JOB_TYPE,NO
FROM JOB
WHERE NO >= (SELECT MAX(NO) FROM JOB)
UNION
SELECT JOB_TYPE,NO
FROM JOB
WHERE NO >= (SELECT MIN(NO) FROM JOB);
```

**Output**

```
mysql> USE COMPANYDB;
Database changed
mysql> WITH JOB(JOB_TYPE,NO) AS
    ->      (SELECT JOB_TYPE,COUNT(*)
    ->      FROM EMPLOYEE
    ->      GROUP BY JOB_TYPE)
    -> SELECT JOB_TYPE,NO
    -> FROM JOB
    -> WHERE NO >= (SELECT MAX(NO) FROM JOB)
    -> UNION
    -> SELECT JOB_TYPE,NO
    -> FROM JOB
    -> WHERE NO >= (SELECT MIN(NO) FROM JOB);
+-----------------+----+
| JOB_TYPE        | NO |
+-----------------+----+
| Senior Engineer |  2 |
| Engineer        |  2 |
| Staff           |  1 |
| Senior Staff    |  1 |
| CEO             |  1 |
+-----------------+----+
5 rows in set (0.00 sec)

mysql>
```