

**Figure 3.6**

One possible database state for the COMPANY relational database schema.

**EMPLOYEE**

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

**DEPARTMENT**

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

**DEPT\_LOCATIONS**

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

**WORKS\_ON**

<u>Essn</u>	<u>Pno</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

**PROJECT**

Pname	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

**DEPENDENT**

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

# DBMS QUERIES

## CREATE TABLE EMPLOYEE

```
( Fname VARCHAR(15) NOT NULL,
Minit CHAR,
Lname VARCHAR(15) NOT NULL,
Ssn CHAR(9) NOT NULL,
Bdate DATE,
Address VARCHAR(30),
Sex CHAR,
Salary DECIMAL(10,2),
Super_ssn CHAR(9),
Dno INT NOT NULL,
PRIMARY KEY (Ssn),+
FOREIGN KEY (Super_ssn) REFERENCES
EMPLOYEE(Ssn),
FOREIGN KEY (Dno) REFERENCES
DEPARTMENT(Dnumber) );
```

## CREATE TABLE DEPARTMENT

```
( Dname VARCHAR(15) NOT NULL,
Dnumber INT NOT NULL,
Mgr_ssn CHAR(9) NOT NULL,
Mgr_start_date DATE,
PRIMARY KEY (Dnumber),
UNIQUE (Dname),
FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
);
```

## CREATE TABLE DEPT\_LOCATIONS

```
( Dnumber INT NOT NULL,
Dlocation VARCHAR(15) NOT NULL,
PRIMARY KEY (Dnumber, Dlocation),
FOREIGN KEY (Dnumber) REFERENCES
DEPARTMENT(Dnumber) );
```

## CREATE TABLE PROJECT

```
( Pname VARCHAR(15) NOT NULL,
Pnumber INT NOT NULL,
Plocation VARCHAR(15),
Dnum INT NOT NULL,
PRIMARY KEY (Pnumber),
UNIQUE (Pname),
FOREIGN KEY (Dnum) REFERENCES
DEPARTMENT(Dnumber) );
```

## CREATE TABLE WORKS\_ON

```
( Essn CHAR(9) NOT NULL,
Pno INT NOT NULL,
Hours DECIMAL(3,1) NOT NULL,
PRIMARY KEY (Essn, Pno),
FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber)
);
```

## CREATE TABLE DEPENDENT

```
( Essn CHAR(9) NOT NULL,
Dependent_name VARCHAR(15) NOT NULL,
Sex CHAR,
Bdate DATE,
Relationship VARCHAR(8),
PRIMARY KEY (Essn, Dependent_name),
FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

**Query 1. Retrieve the name and address of all employees who work for the 'Research' department.**

Q1:

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
```

WHERE Dname='Research' AND Dnumber=Dno;

**Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.**

Q2:

```
SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn AND
Plocation='Stafford';
```

**Query 8. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.**

Q8:

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn=S.Ssn;
```

**Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of**

**EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.**

Q9:

```
SELECT Ssn
FROM EMPLOYEE;
```

Q10:

```
SELECT Ssn, Dname
FROM EMPLOYEE, DEPARTMENT;
```

**Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).**

```
Q11: SELECT ALL Salary
FROM EMPLOYEE;
```

```
Q11A: SELECT DISTINCT Salary
FROM EMPLOYEE;
```

**Query 4. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.**

Q4A:

```
(SELECT DISTINCT Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
AND Lname='Smith' )
UNION
( SELECT DISTINCT Pnumber
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Essn=Ssn
AND Lname='Smith' );
```

**Query 12. Retrieve all employees whose address is in Houston, Texas.**

Q12:

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston,TX%';
```

**Query 12A. Find all employees who were born during the 1950s.**

Q12:

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Bdate LIKE '__5 _____';
```

**Query 13. Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.**

Q13:

```
SELECT E.Fname, E.Lname, 1.1 * E.Salary AS
Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT
AS P
WHERE E.Ssn=W.Essn AND W.Pno=P.Pnumber AND
P.Pname='ProductX';
```

**Query 14. Retrieve all employees in department 5 whose salary is between \$30,000 and \$40,000.**

Q14:

SELECT \*

FROM EMPLOYEE

WHERE (Salary BETWEEN 30000 AND 40000) AND  
Dno = 5;

The condition (Salary BETWEEN 30000 AND 40000) in  
Q14 is equivalent to the condition ((Salary >= 30000)  
AND (Salary <= 40000)).

**Query 15. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.**

Q15:

SELECT D.Dname, E.Lname, E.Fname, P.Pname

FROM DEPARTMENT D, EMPLOYEE E, WORKS\_ON W,  
PROJECT P

WHERE D.Dnumber= E.Dno AND E.Ssn= W.Essn AND

W.Pno= P.Pnumber

ORDER BY D.Dname, E.Lname, E.Fname;

**SELECT <attribute list>**

**FROM <table list>**

**[ WHERE <condition> ]**

**[ ORDER BY <attribute list> ]**

**INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)**

**VALUES ('Richard', 'Marini', 4, '653298653');**

INSERT INTO WORKS\_ON\_INFO ( Emp\_name,  
Proj\_name,

Hours\_per\_week )

SELECT E.Lname, P.Pname, W.Hours

FROM PROJECT P, WORKS\_ON W, EMPLOYEE E

WHERE P.Pnumber=W.Pno AND W.Essn=E.Ssn;

**DELETE FROM EMPLOYEE**

**WHERE Lname='Brown';**

DELETE FROM EMPLOYEE

WHERE Ssn='123456789';

U4C: DELETE FROM EMPLOYEE

WHERE Dno=5;

U4D: DELETE FROM EMPLOYEE;

**UPDATE PROJECT**

**SET Plocation = 'Bellaire', Dnum = 5**

**WHERE Pnumber=10;**

UPDATE EMPLOYEE

SET Salary = Salary \* 1.1

WHERE Dno = 5;

**Query 18. Retrieve the names of all employees who do not have supervisors.**

Q18:

SELECT Fname, Lname

FROM EMPLOYEE

WHERE Super\_ssn IS NULL;

**SELECT DISTINCT Pnumber**

**FROM PROJECT**

**WHERE Pnumber IN**

**( SELECT Pnumber**

**FROM PROJECT, DEPARTMENT, EMPLOYEE**

**WHERE Dnum=Dnumber AND**

**Mgr\_ssn=Ssn AND Lname='Smith' )**

**OR**

**Pnumber IN**

**( SELECT Pno**

**FROM WORKS\_ON, EMPLOYEE**

**WHERE Essn=Ssn AND Lname='Smith' );**

SELECT DISTINCT Essn

FROM WORKS\_ON

```
WHERE (Pno, Hours) IN ( SELECT Pno, Hours
FROM WORKS_ON
WHERE Essn='123456789' );
```

```
SELECT Lname, Fname
FROM EMPLOYEE
WHERE Salary > ALL ( SELECT Salary
FROM EMPLOYEE
WHERE Dno=5 );
```

**Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.**

Q16:

```
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E
WHERE E.Ssn IN ( SELECT Essn
FROM DEPENDENT AS D
WHERE E.Fname=D.Dependent_name
AND E.Sex=D.Sex );
```

**Q16A: SELECT E.Fname, E.Lname  
FROM EMPLOYEE AS E, DEPENDENT AS D  
WHERE E.Ssn=D.Essn AND E.Sex=D.Sex  
AND E.Fname=D.Dependent\_name;**

```
Q16B: SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E
WHERE EXISTS ( SELECT *
FROM DEPENDENT AS D
WHERE E.Ssn=D.Essn AND E.Sex=D.Sex
AND E.Fname=D.Dependent_name);
```

**Query 6. Retrieve the names of employees who have no dependents.**

Q6:

```
SELECT Fname, Lname
```

```
FROM EMPLOYEE
WHERE NOT EXISTS ( SELECT *
FROM DEPENDENT
WHERE Ssn=Essn );
```

**Query 7. List the names of managers who have at least one dependent.**

Q7:

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE EXISTS ( SELECT *
FROM DEPENDENT
WHERE Ssn=Essn )
AND
EXISTS ( SELECT *
FROM DEPARTMENT
WHERE Ssn=Mgr_ssn );
```

**The query Q3: Retrieve the name of each employee who works on all the projects controlled by department number 5 can be written using EXISTS and NOT EXISTS in SQL**

Q3A:

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE NOT EXISTS ( ( SELECT Pnumber
FROM PROJECT
WHERE Dnum=5)
EXCEPT ( SELECT Pno
FROM WORKS_ON
WHERE Ssn=Essn) );
```

Q3B:

```
SELECT Lname, Fname
FROM EMPLOYEE
WHERE NOT EXISTS ( SELECT *
FROM WORKS_ON B
```

```

WHERE ( B.Pno IN ( SELECT Pnumber
FROM PROJECT
WHERE Dnum=5 )
AND
NOT EXISTS ( SELECT *
FROM WORKS_ON C
WHERE C.Essn=Ssn
AND C.Pno=B.Pno ));

```

**Query 17. Retrieve the Social Security numbers of all employees who work on project numbers 1, 2, or 3.**

```

Q17:

SELECT DISTINCT Essn
FROM WORKS_ON
WHERE Pno IN (1, 2, 3);

```

**Query 19. Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.**

```

Q19:

SELECT SUM (Salary), MAX (Salary), MIN (Salary),
AVG (Salary)
FROM EMPLOYEE;

```

**Query 20. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.**

```

Q20:

SELECT SUM (Salary), MAX (Salary), MIN (Salary),
AVG (Salary)
FROM (EMPLOYEE JOIN DEPARTMENT ON
Dno=Dnumber)
WHERE Dname='Research';

```

**Queries 21 and 22. Retrieve the total number of employees in the company (Q21) and the number of employees in the 'Research' department (Q22).**

```

Q21:

SELECT COUNT (*)
FROM EMPLOYEE;

```

Q22:

```

SELECT COUNT (*)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research';

```

Here the asterisk (\*) refers to the rows (tuples), so COUNT (\*) returns the number of

rows in the result of the query. We may also use the COUNT function to count values

in a column rather than tuples, as in the next example.

**Query 23. Count the number of distinct salary values in the database.**

```

Q23:

SELECT COUNT (DISTINCT Salary)

```

```

FROM EMPLOYEE;

```

**Query 24. For each department, retrieve the department number, the number of employees in the department, and their average salary.**

```

Q24:

SELECT Dno, COUNT (*), AVG (Salary)
FROM EMPLOYEE
GROUP BY Dno;

```

**Query 25. For each project, retrieve the project number, the project name, and the number of employees who work on that project.**

```

Q25:

SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE Pnumber=Pno

```

```

GROUP BY Pnumber, Pname;

```

**Query 26. For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.**

```

Q26:

SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE Pnumber=Pno
GROUP BY Pnumber, Pname
HAVING COUNT (*) > 2;

```

**Query 27. For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.**

Q27:

```
SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Ssn=Essn AND Dno=5
GROUP BY Pnumber, Pname;
```

**Query 28. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than \$40,000.**

Q28:

```
SELECT Dnumber, COUNT (*)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno AND Salary>40000 AND
( SELECT Dno
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT (*) > 5);
```

**SELECT <attribute and function list>**

**FROM <table list>**

**[ WHERE <condition> ]**

**[ GROUP BY <grouping attribute(s)> ]**

**[ HAVING <group condition> ]**

**[ ORDER BY <attribute list> ];**

## Relational algebra

**Query 1. Retrieve the name and address of all employees who work for the 'Research' department**

```
RESEARCH_DEPT ←
σDname='Research'(DEPARTMENT)

RESEARCH_EMPS ← (RESEARCH_DEPT
Dnumber=DnoEMPLOYEE)

RESULT ← πFname, Lname,
Address(RESEARCH_EMPS)
```

As a single in-line expression, this query becomes:

```
πFname, Lname, Address
(σDname='Research'(DEPARTMENT
Dnumber=Dno(EMPLOYEE)))
```

## Tuple Relational algebra

**Query 1. List the name and address of all employees who work for the 'Research' department.**

```
Q1: {t.Fname, t.Lname, t.Address | EMPLOYEE(t)
AND (∃d)(DEPARTMENT(d)
AND d.Dname='Research' AND d.Dnumber=t.Dno)}
```

**Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, birth date, and address.**

Q2:

```
{p.Pnumber, p.Dnum, m.Lname, m.Bdate, m.Address
| PROJECT(p) AND
EMPLOYEE(m) AND p.Plocation='Stafford' AND
((∃d)(DEPARTMENT(d)
AND p.Dnum=d.Dnumber AND d.Mgr_ssn=m.Ssn))}
```

**Query 3 . List the name of each employee who works on some project controlled by department number 5. This is a variation of Q3 in which all changed to some. In this case we need two join conditions and two existential quantifiers.**

Q0 :

```
{e.Lname, e.Fname | EMPLOYEE(e) AND
((∃x)(∃w)(PROJECT(x) AND
```

```
WORKS_ON(w) AND x.Dnum=5 AND w.Essn=e.Ssn
AND x.Pnumber=w.Pno))}
```

**Query 4. Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as manager of the controlling department for the project.**

Q4:

```
{ p.Pnumber | PROJECT(p) AND
(((∃e)(∃w)(EMPLOYEE(e)
AND WORKS_ON(w) AND w.Pno=p.Pnumber
AND e.Lname='Smith' AND e.Ssn=w.Essn) )
```

OR

```
((∃m)(∃d)(EMPLOYEE(m) AND DEPARTMENT(d)
```

AND p.Dnum=d.Dnumber AND d.Mgr\_ssn=m.Ssn

AND m.Lname='Smith'))))}

**Query 3. List the names of employees who work on all the projects controlled by department number 5. One way to specify this query is to use the universal quantifier as shown:**

Q3:

{e.Lname, e.Fname | EMPLOYEE(e) AND  
(( $\forall x$ )(NOT(PROJECT(x)) OR NOT

(x.Dnum=5) OR (( $\exists w$ )(WORKS\_ON(w) AND  
w.Essn=e.Ssn AND

x.Pnumber=w.Pno))))}

Q3A:

{e.Lname, e.Fname | EMPLOYEE(e) AND (NOT ( $\exists x$ )  
(PROJECT(x) AND

(x.Dnum=5) AND (NOT ( $\exists w$ )(WORKS\_ON(w) AND  
w.Essn=e.Ssn

AND x.Pnumber=w.Pno))))}

We now give some additional examples of queries that use quantifiers.

**Query 6. List the names of employees who have no dependents.**

Q6:

{e.Fname, e.Lname | EMPLOYEE(e) AND (NOT  
( $\exists d$ )(DEPENDENT(d)

AND e.Ssn=d.Essn))}

Using the general transformation rule, we can rephrase Q6 as follows:

Q6A: {e.Fname, e.Lname | EMPLOYEE(e) AND  
(( $\forall d$ )(NOT(DEPENDENT(d))

OR NOT(e.Ssn=d.Essn))})

**Query 7. List the names of managers who have at least one dependent.**

Q7: {e.Fname, e.Lname | EMPLOYEE(e) AND  
(( $\exists d$ )( $\exists p$ )(DEPARTMENT(d)

AND DEPENDENT(p) AND e.Ssn=d.Mgr\_ssn AND  
p.Essn=e.Ssn))}

## Domain Relational algebra

**Query 0. List the birth date and address of the employee whose name is 'John B. Smith'.**

Q0:

{u, v | ( $\exists q$ ) ( $\exists r$ ) ( $\exists s$ ) ( $\exists t$ ) ( $\exists w$ ) ( $\exists x$ ) ( $\exists y$ ) ( $\exists z$ )  
(EMPLOYEE(qrstuvwxyz) AND q='John' AND r='B'  
AND s='Smith'))}

**Query 1. Retrieve the name and address of all employees who work for the 'Research' department.**

Q1:

{q, s, v | ( $\exists z$ ) ( $\exists l$ ) ( $\exists m$ ) (EMPLOYEE(qrstuvwxyz) AND  
DEPARTMENT(lmno) AND l='Research' AND m=z)}

**Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, birth date, and address.**

Q2:

{i, k, s, u, v | ( $\exists j$ )( $\exists m$ )( $\exists n$ )( $\exists t$ )(PROJECT(hijk) AND  
EMPLOYEE(qrstuvwxyz) AND DEPARTMENT(lmno)  
AND k=m AND

n=t AND j='Stafford'))}

**Query 6. List the names of employees who have no dependents.**

Q6:

{q, s | ( $\exists t$ )(EMPLOYEE(qrstuvwxyz) AND

(NOT( $\exists l$ )(DEPENDENT(lmnop) AND t=l))})

Q6 can be restated using universal quantifiers instead of the existential quantifiers,

as shown in Q6A:

Q6A:

{q, s | ( $\exists t$ )(EMPLOYEE(qrstuvwxyz) AND

(( $\forall l$ )(NOT(DEPENDENT(lmnop)) OR NOT(t=l))})}

**Query 7. List the names of managers who have at least one dependent.**

Q7:

{s, q | ( $\exists t$ )( $\exists j$ )( $\exists l$ )(EMPLOYEE(qrstuvwxyz) AND  
DEPARTMENT(hijk)

AND DEPENDENT(lmnop) AND t=j AND l=t)}



```

RESEARCH_DEPT  $\leftarrow \sigma_{\text{Dname}='Research'}(\text{DEPARTMENT})$ 
RESEARCH_EMPS  $\leftarrow (\text{RESEARCH_DEPT} \bowtie_{\text{Dnumber}=\text{Dno}} \text{EMPLOYEE})$ 
RESULT  $\leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Address}}(\text{RESEARCH_EMPS})$ 

```

As a single in-line expression, this query becomes:

```

 $\pi_{\text{Fname}, \text{Lname}, \text{Address}}(\sigma_{\text{Dname}='Research'}(\text{DEPARTMENT} \bowtie_{\text{Dnumber}=\text{Dno}} \text{EMPLOYEE}))$ 

```

This query could be specified in other ways; for example, the order of the JOIN and SELECT operations could be reversed, or the JOIN could be replaced by a NATURAL JOIN after renaming one of the join attributes to match the other join attribute name.

**Query 2.** For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

```

STAFFORD_PROJS  $\leftarrow \sigma_{\text{Plocation}='Stafford'}(\text{PROJECT})$ 
CONTR_DEPTS  $\leftarrow (\text{STAFFORD_PROJS} \bowtie_{\text{Dnum}=\text{Dnumber}} \text{DEPARTMENT})$ 
PROJ_DEPT_MGRS  $\leftarrow (\text{CONTR_DEPTS} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE})$ 
RESULT  $\leftarrow \pi_{\text{Pnumber}, \text{Dnum}, \text{Lname}, \text{Address}, \text{Bdate}}(\text{PROJ_DEPT_MGRS})$ 

```

In this example, we first select the projects located in Stafford, then join them with their controlling departments, and then join the result with the department managers. Finally, we apply a project operation on the desired attributes.

**Query 3.** Find the names of employees who work on *all* the projects controlled by department number 5.

```

DEPT5_PROJS  $\leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum}=5}(\text{PROJECT})))$ 
EMP_PROJ  $\leftarrow \rho_{(\text{Ssn}, \text{Pno})}(\pi_{\text{Essn}, \text{Pno}}(\text{WORKS\_ON}))$ 
RESULT_EMP_SSNS  $\leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJS}$ 
RESULT  $\leftarrow \pi_{\text{Lname}, \text{Fname}}(\text{RESULT\_EMP\_SSNS} * \text{EMPLOYEE})$ 

```

In this query, we first create a table DEPT5\_PROJS that contains the project numbers of all projects controlled by department 5. Then we create a table EMP\_PROJ that holds (Ssn, Pno) tuples, and apply the division operation. Notice that we renamed the attributes so that they will be correctly used in the division operation. Finally, we join the result of the division, which holds only Ssn values, with the EMPLOYEE table to retrieve the desired attributes from EMPLOYEE.

**Query 4.** Make a list of project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.

```

SMITHS(Essn)  $\leftarrow \pi_{\text{Ssn}}(\sigma_{\text{Lname}='Smith'}(\text{EMPLOYEE}))$ 
SMITH_WORKER_PROJS  $\leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} * \text{SMITHS})$ 
MGRS  $\leftarrow \pi_{\text{Lname}, \text{Dnumber}}(\text{EMPLOYEE} \bowtie_{\text{Ssn}=\text{Mgr\_ssn}} \text{DEPARTMENT})$ 
SMITH_MANAGED_DEPTS(Dnum)  $\leftarrow \pi_{\text{Dnumber}}(\sigma_{\text{Lname}='Smith'}(\text{MGRS}))$ 
SMITH_MGR_PROJS(Pno)  $\leftarrow \pi_{\text{Pnumber}}(\text{SMITH\_MANAGED\_DEPTS} * \text{PROJECT})$ 
RESULT  $\leftarrow (\text{SMITH\_WORKER\_PROJS} \cup \text{SMITH\_MGR\_PROJS})$ 

```

In this query, we retrieved the project numbers for projects that involve an employee named Smith as a worker in SMITH\_WORKER\_PROJS. Then we retrieved the project numbers for projects that involve an employee named Smith as manager of the department that controls the project in SMITH\_MGR\_PROJS. Finally, we applied the **UNION** operation on SMITH\_WORKER\_PROJS and SMITH\_MGR\_PROJS. As a single in-line expression, this query becomes:

$$\pi_{Pno} (WORKS\_ON \bowtie_{Essn=Ssn} (\pi_{Ssn} (\sigma_{Lname='Smith'}(EMPLOYEE)))) \cup \pi_{Pno} ((\pi_{Dnumber} (\sigma_{Lname='Smith'}(\pi_{Lname, Dnumber}(EMPLOYEE)))) \bowtie_{Ssn=Mgr\_ssn} DEPARTMENT)) \bowtie_{Dnumber=Dnum} PROJECT)$$

**Query 5.** List the names of all employees with two or more dependents.

Strictly speaking, this query cannot be done in the *basic (original) relational algebra*. We have to use the AGGREGATE FUNCTION operation with the COUNT aggregate function. We assume that dependents of the *same* employee have *distinct* Dependent\_name values.

$$T1(Ssn, No\_of\_dependents) \leftarrow \pi_{Ssn} \mathcal{A}_{COUNT\ Dependent\_name} (DEPENDENT) \\ T2 \leftarrow \sigma_{No\_of\_dependents > 2} (T1) \\ RESULT \leftarrow \pi_{Lname, Fname} (T2 * EMPLOYEE)$$

**Query 6.** Retrieve the names of employees who have no dependents.

This is an example of the type of query that uses the MINUS (SET DIFFERENCE) operation.

$$ALL\_EMPS \leftarrow \pi_{Ssn} (EMPLOYEE) \\ EMPS\_WITH\_DEPS(Ssn) \leftarrow \pi_{Essn} (DEPENDENT) \\ EMPS\_WITHOUT\_DEPS \leftarrow (ALL\_EMPS - EMPS\_WITH\_DEPS) \\ RESULT \leftarrow \pi_{Lname, Fname} (EMPS\_WITHOUT\_DEPS * EMPLOYEE)$$

We first retrieve a relation with all employee Ssns in ALL\_EMPS. Then we create a table with the Ssns of employees who have at least one dependent in EMPS\_WITH\_DEPS. Then we apply the SET DIFFERENCE operation to retrieve employees Ssns with no dependents in EMPS\_WITHOUT\_DEPS, and finally join this with EMPLOYEE to retrieve the desired attributes. As a single in-line expression, this query becomes:

$$\pi_{Lname, Fname} ((\pi_{Ssn} (EMPLOYEE) - \rho_{Ssn} (\pi_{Essn} (DEPENDENT))) * EMPLOYEE)$$

**Query 7.** List the names of managers who have at least one dependent.

$$MGRS(Ssn) \leftarrow \pi_{Mgr\_ssn} (DEPARTMENT) \\ EMPS\_WITH\_DEPS(Ssn) \leftarrow \pi_{Essn} (DEPENDENT) \\ MGRS\_WITH\_DEPS \leftarrow (MGRS \cap EMPS\_WITH\_DEPS) \\ RESULT \leftarrow \pi_{Lname, Fname} (MGRS\_WITH\_DEPS * EMPLOYEE)$$

In this query, we retrieve the Ssns of managers in MGRS, and the Ssns of employees with at least one dependent in EMPS\_WITH\_DEPS, then we apply the SET INTERSECTION operation to get the Ssns of managers who have at least one dependent.

Bapatla Engineering College :: Bapatla  
Department of Information Technology  
DBMS LAB

**WORK SHEET-1( USING EMP TABLE)**

1. List empno, empname and salary.
2. List the names of all MANAGERS.
3. list all clerks in deptno. 30.
4. List the employees to who manager is 7698.
5. List jobs in dept 20.
6. List employee names whose salary is between 2000 and 3000.
7. List employees in the departments 10, 20.
8. List employee names which begin with S.
9. List employee names having 'A' in their names.
10. List employees who have joined in JAN.
11. List employees who have joined in the year 81.
12. List all distinct jobs.
13. List employee names in alphabetical order.
14. List employee names alphabetically department wise.
15. List employee names alphabetically job wise.
16. List employee numbers, name sal, DA(15% OF SAL) and PF (10% of sal).
17. List employee names having an experience more than 15 years.
18. List employee names whose commission is NULL.
19. list employees who do not report to anybody.
20. List maximum sal, minimum sal, average sal.
21. List the numbers of jobs.
22. List the numbers of people and average salary in deptno 30.
23. List maximum sal and minimum sal in the designations SALESMAN and CLERK.
24. List the numbers of people and average salary of employees joined in 81, 82 and 83.
25. List jobs that are unique to deptno 20 set operations (Add more problems).
26. Display today's date and present time.
27. List employee names and their joining date in the following formats
  - A. SMITH 17<sup>th</sup> DEC NINETEEN EIGHTY
  - B. SMITH SEVENTEENTH DEC NINETEEN EIGHTY
  - C. SMITH Week day of joining
  - D. SMITH 17/12/80
28. List employee names and their experience in years
29. List employee names who joined in DEC and on Monday or Friday.
30. Display a given date as a string in different formats.

Bapatla Engineering College :: Bapatla  
Department of Information Technology  
DBMS LAB

**WORKSHEET-II (USING EMP TABLE)**

- 1) List employee names and their hire dates sorted in the order of their experience.
- 2) List managers names and their joining dates completely spelled in alphabetical order of names.
- 3) List employee names and their experience in years with names arranged in descending order.
- 4) List employee names having a minimum of 2 years experience sorted on experience.
- 5) List employee names with all capital letters, with all small letters and with first letter only as capital.
- 6) List employee names with length of the name sorted on length.
- 7) List employee names appending Sri to the beginning and Garu to the end.
- 8) List employee names and month names of joining.
- 9) List employee names and year of joining in words.
- 10) List employees names, job and salary with 5 hyphens in between.
- 11) List employee names and position of first occurrence of I in their name.
- 12) List employee names and the string without first character and last character in their name.
- 13) List employees who joined between Apr 81 and Apr 82.
- 14) List max sal, min sal and average sal of depts. 10, 30.
- 15) List the designation in dept 30 but not in 20.
- 16) List the number of employees in each department along with dept numbers.
- 17) List number of employees joined year wise.
- 18) List number of employees job wise.
- 19) List max sal, min sal, average salary dept wise.
- 20) List max sal, min sal, average salary job wise.
- 21) List max sal, min sal for the jobs MANAGER and CLERK.
- 22) List max sal, min sal AND average salary of the depts. Having a minimum 3 employees.
- 23) List the number of employees in each job in each department.
- 24) List MGR and the number of employees report to them in the sorted order.
- 25) List emp numbers of employees to whom a minimum of 3 people report.
- 26) List dept numbers having a minimum of 3 persons.
- 27) List names of jobs having a minimum of 3 persons in that job.
- 28) List names of months in which a minimum of 3 persons joined.
- 29) List hiredates of employees having 2 or more employees having the same hiredate.
- 30) List departments having minimum of 3 people having a minimum of 17 years of experience.

Bapatla Engineering College :: Bapatla  
Department of Information Technology  
DBMS LAB

**WORKSHEET-III (USING EMP TABLE)**

1. List employee names and dept names with which they are associated.
2. List employee names, salary and their grade.
3. List employee name, dept name along with grade.
4. List employee names and their manager names.
5. List dept name and Manager name.
6. List managers of various depts.. Along with grade sorted on grade.
7. List employees having commission along with grade.
8. List employees names with job manager along their manager names to whom they have to report.
9. List names of employees who are working in the same dept of their manager.
10. List names of employees who are not working in the same dept of their manager.
11. List names of employees having first character in their name first character in their dept name same.
12. List employees who joined in the present month in any year and having grade and last digit in the year are same.
13. List names of employees whose empno, mgr and grade given the same remainder when divided by 2.
14. List the names of employees having grade and tens position in the deptno same.
15. List the names of employees having grade and tens position in the deptno different.
16. List employee name,deptname and dept location of those employees having any of these three same length
17. List names of employees having month number of hiredate and grade same
18. List names of clerks who are reporting to analyst.
19. List emp names and thrie manager names having same grade.
20. List emp names of employees who joined before their manager's joining date.

Bapatla Engineering College :: Bapatla  
Department of Information Technology  
DBMS LAB

**TABLES CREATION FOR LABSHEET-1**

Create the following tables

**1. EMPLOYEE( FNAME,MINIT,LNAME,SSN,SEX,SALARY,SUPERSSN,DNO)**

CONSTRAINTS:

FNAME,LNAME,SSN,DNO                NOT NULL  
PRIMARY KEY(SSN)  
FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)  
FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DNUMBER)

**2.DEPARTMENT(DNAME,DNUMBER,MGRSSN) CONSTRAINTS:**

DNAME,DNUMBER,MGRSSN    NOTNULL  
PRIMARY KEY (DNUMBER)  
UNIQUE (DNAME),  
FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN)

**3. DEPT\_LOCATIONS(DNUMBER,DLOCATION)**

CONSTRAINTS:

DNUMBER.DLOCATION                NOTNULL  
PRIMARY KEY(DNUMBER,DLOCATION)  
FOREIGN KEY(DNUMBER) REFERENCES DEPARTMENT(DNUMBER)

**4. PROJECT(PNAME,PNUMBER,PLOCATIOIM,DNUM)**

CONSTRAINTS:

PNAME.PNUMBER.DNUM    NOTNULL  
PRIMARY KEY(PNUMBER)  
UNIQUE(PNAME)  
FOREIGN KEY(DNUM) REFERENCES DEPARTMENT(DNUMBER)

**5. WORKS\_ON(ESSM,PNO,HOURS)**

CONSTRAINTS:

ESSN,PNO                                NOTNULL  
PRIMARY KEY(ESSN,PNO)  
FOREIGN KEY(ESSN) REFERENCES EMPLOYEE(SSN)  
FOREIGN KEY(PNO) REFERENCES PROJECT(PNUMBER)

**6. DEPENDENT(ESSN,D\_NAME,SEX,RELATIONSHIP)**

CONSTRAINTS':

ESSN,D\_NAME                                NOTNULL  
PRIMARY KEY(ESSN,D\_NAME)  
FOREIGN KEY(ESSN) REFERENCES EMPLOYEE(SSN)

Bapatla Engineering College :: Bapatla  
Department of Information Technology  
DBMS LAB

**EMP DATABASE**

**EMPLOYEE**

ENAME	MINIT	LNAME	SSN	SEX	SALARY	SUPERSSN	DNO
JOHN	B	SMITH	2345	M	30000	3344	5
FRANKLIN	T	WONG	3344	M	40000	8866	5
ALICIA	J	ZELAYA	9988	F	25000	8765	4
JENNIFER	S	WALLACE	8765	F	43000	8866	4
RAMESH	K	NARAYANA	6688	M	38000	3344	5
JOYCE	A	ENGLISH	5345	F	25000	3344	5
AHMAD	V	JABBER	8798	M	25000	8765	4
JAMES	E	BORG	8866	M	55000	NULL	1

**DEPARTMENT**

DNAMNE	DNUMBER	MGRSSN
RESEARCH	5	3344
ADMINISTRATION	4	8765
HEADQUATERS	1	8866

**DEPT\_LOCATION**

DNUMBER	DLOCATION
1	HOUSTON
4	STAFFORD
5	BELLARIE
5	SUGARLAND
5	HOUSTON

**WORKS\_ON**

ESSN	PNO	HOURS
2345	1	32.5
2345	2	7.5
6688	3	40
5345	1	20
5345	2	20
3344	2	10
3344	3	10
3344	10	10
3344	20	10
9988	30	30
9988	10	10
8798	10	35
8798	20	5
8765	20	20
8765	30	15
8866	30	NULL
8866	1	NULL

**DEPENDENT**

ESSN	D_NAME	SEX	RELATIONSHIP
3344	ALICE	F	DAUGHTER
3344	THEODORE	M	SON
3344	JOY	F	SPOUSE
8765	ABNER	M	SPOUSE
2345	MICHAEL	M	SON
2345	ALICE	F	DAUGHTER
2345	ELIZABETH	F	SPOUSE

**PROJECT**

PNAME	PNUMBER	PLOCATION	DNUM
PRODUCT_X	1	BELLARIE	5
PRODUCT_Y	2	SUGARLAND	5
PRODUCT_Z	3	HOUSTON	5
COMPUTERIZATION	10	STAFFORD	4
REORGANIZATION	20	HOUSTON	1
NEWBENEFITS	30	STAFFORD	4

Bapatla Engineering College :: Bapatla  
Department of Information Technology  
DBMS LAB

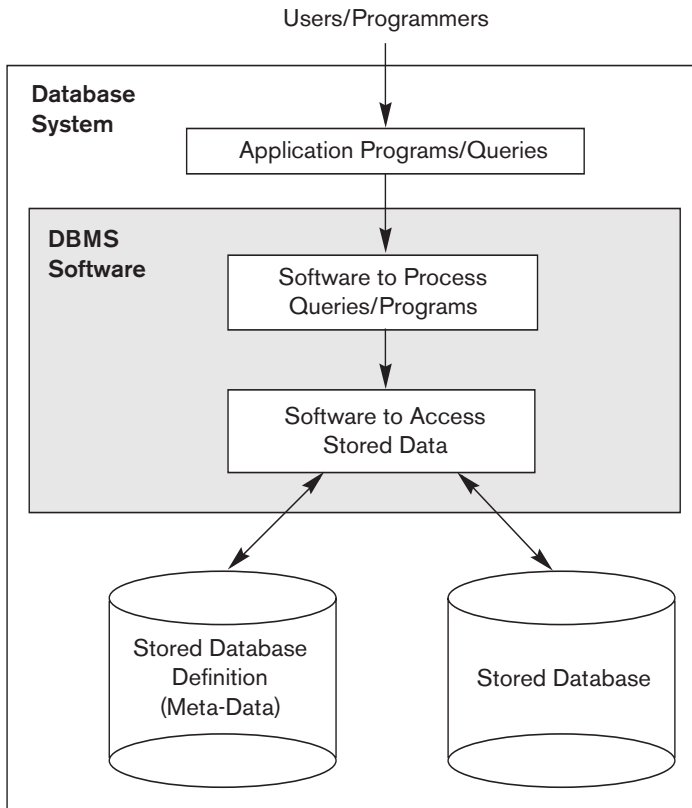
**Labsheet –1**

1. Retrieve names of Employees who work for the Research Department.
2. For each project located in Stafford, list the project number, controlling department number and the department manager's last name.
3. For each Employee retrieve the employee's first and last name of his/her supervisor.
4. List all project number for projects that involve an Employee whose last name is SMITH, either as a worker or as a MANAGER of the Department that controls the project.
5. Retrieve the list of Employees and the projects they are working on order by department and with in each department ordered Alphabetically by last name and first name.
6. Retrieve the name of each employee who has the dependent with the same first name and same sex as the employee.
7. Retrieve the name of each employee who works on all the projects controlled by department no. 5.
8. Retrieve the names of employees who have no dependents.
9. List the names of Managers who have at least one dependent.
10. For each project on which more than two employees work, retrieve the project number, the project name, and the no. of employees who work on the project.
11. For each project, retrieve the project number, the project name and the no. of employees from department 5 who work on the project.
12. For each department that has more than 5 employees, retrieve the department number and the no. of its employees who are making more than 40,000.

**PL/SQL Programs**

1. Write a PL/SQL program to print employee number of an employee as well as the corresponding MGR NO.
2. Write a PL/SQL program using FOR/WHILE LOOPS to list out month names and month numbers.
3. Write a PL/SQL program to update commission of an employee (employee number as input) As per the following norms.
  - i) If commission is NULL, make it as 10% of salary
  - ii) If comm. < 200 make comm. = 200
  - iii) If comm. < 300 make comm. = 300
4. Write a PL/SQL program to list out .  
DEPT NO, DNAME, NO OF EMPLOYEES, MAX(SAL), MIN(SAL),  
AVG(SAL) In each dept. If a dept has no employees then display  
"employees are not there in this dept".
5. write a PL/SQL program to get no. of employees whose salary is in between given range.



**Figure 1.1**

A simplified database system environment.

Major (such as mathematics or 'MATH' and computer science or 'CS'); each COURSE record includes data to represent the Course\_name, Course\_number, Credit\_hours, and Department (the department that offers the course); and so on. We must also specify a **data type** for each data element within a record. For example, we can specify that Name of STUDENT is a string of alphabetic characters, Student\_number of STUDENT is an integer, and Grade of GRADE\_REPORT is a single character from the set {'A', 'B', 'C', 'D', 'F', 'T'}. We may also use a coding scheme to represent the values of a data item. For example, in Figure 1.2 we represent the Class of a STUDENT as 1 for freshman, 2 for sophomore, 3 for junior, 4 for senior, and 5 for graduate student.

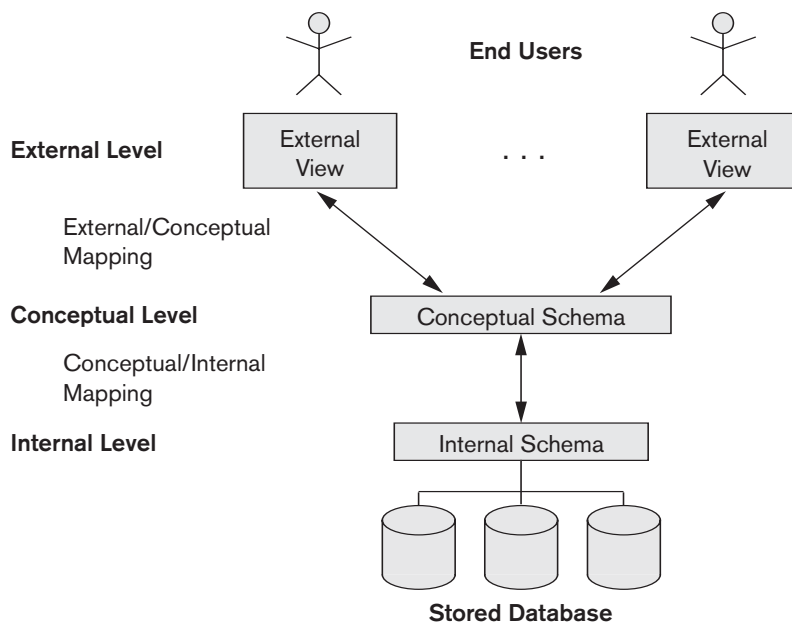
To *construct* the UNIVERSITY database, we store data to represent each student, course, section, grade report, and prerequisite as a record in the appropriate file. Notice that records in the various files may be related. For example, the record for Smith in the STUDENT file is related to two records in the GRADE\_REPORT file that specify Smith's grades in two sections. Similarly, each record in the PREREQUISITE file relates two course records: one representing the course and the other representing the prerequisite. Most medium-size and large databases include many types of records and have *many relationships* among the records.

### 2.2.1 The Three-Schema Architecture

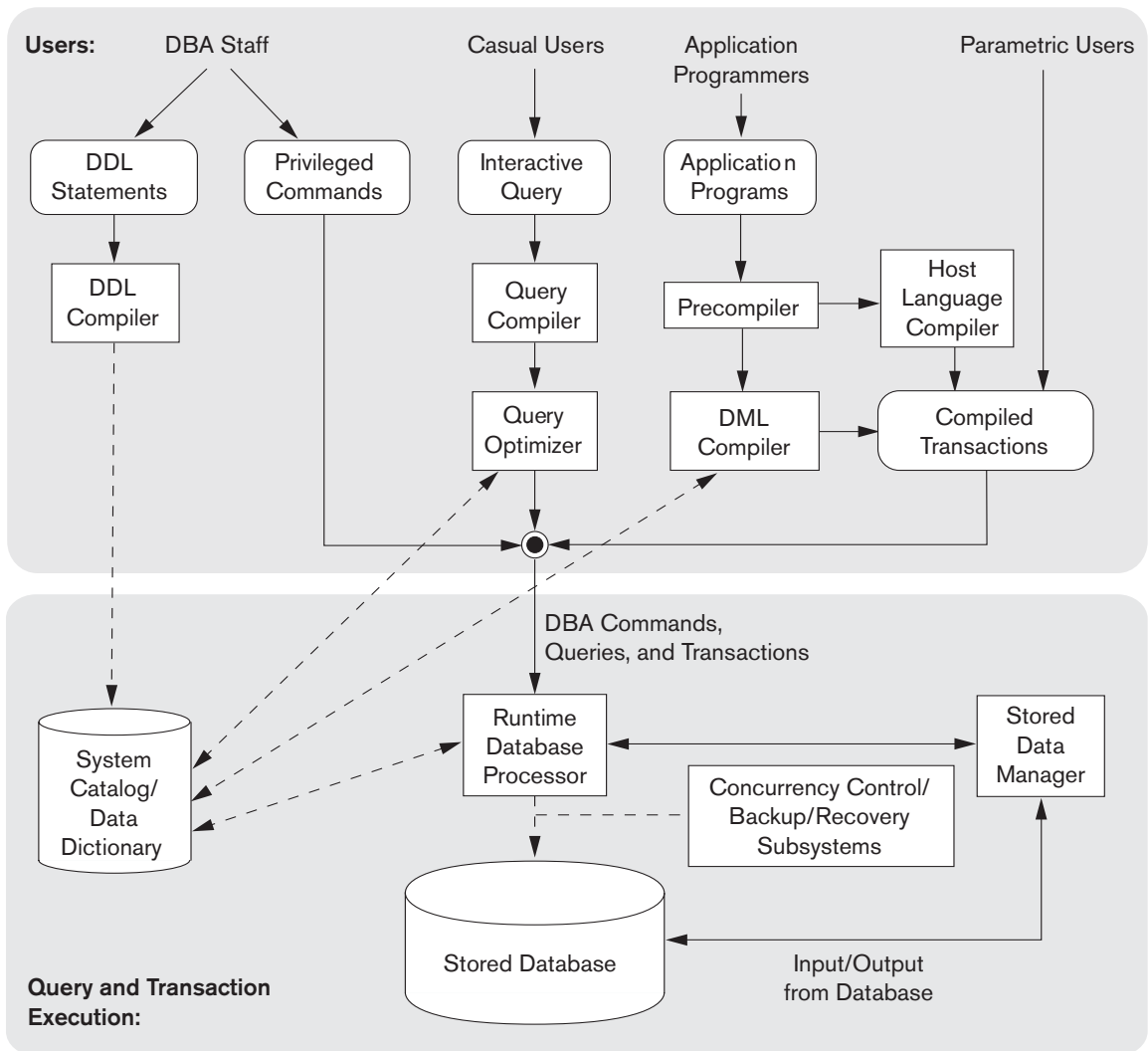
The goal of the three-schema architecture, illustrated in Figure 2.2, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The **internal level** has an **internal schema**, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This *implementation conceptual schema* is often based on a *conceptual schema design* in a high-level data model.
3. The **external or view level** includes a number of **external schemas** or **user views**. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

**Figure 2.2**  
The three-schema architecture.

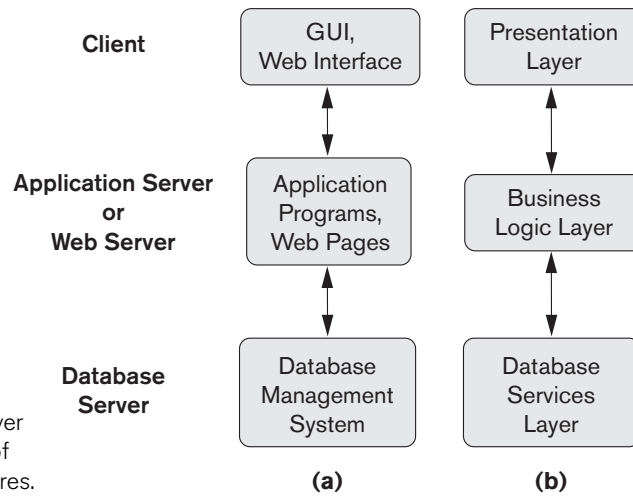


SRIHARI

**Figure 2.3**

Component modules of a DBMS and their interactions.

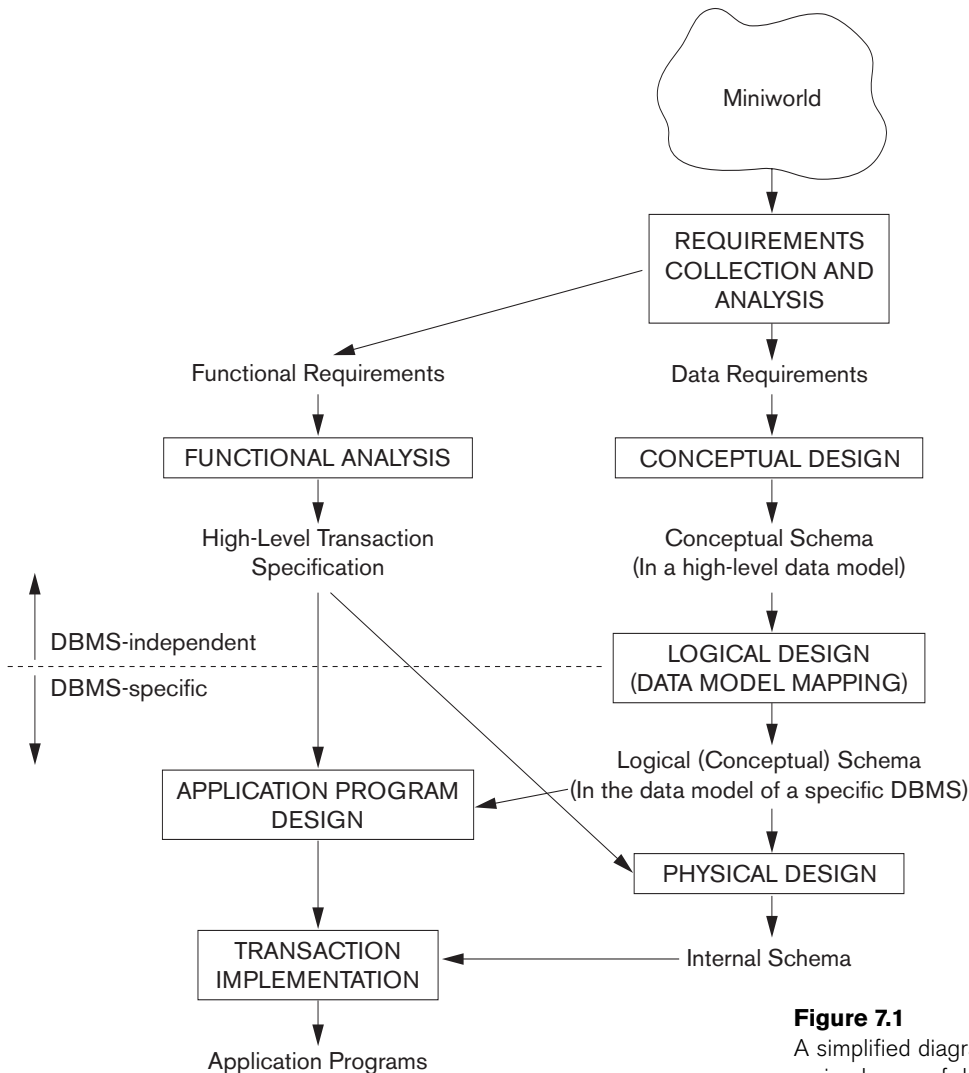
data elements, and so on by a **query compiler** that compiles them into an internal form. This internal query is subjected to query optimization (discussed in Chapters 19 and 20). Among other things, the **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution. It consults the system catalog for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor.



**Figure 2.7**  
Logical three-tier client/server  
architecture, with a couple of  
commonly used nomenclatures.

This intermediate layer or **middle tier** is called the **application server** or the **Web server**, depending on the application. This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client's credentials before forwarding a request to the database server. Clients contain GUI interfaces and some additional application-specific business rules. The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database server to the clients, where it may be processed further and filtered to be presented to users in GUI format. Thus, the *user interface*, *application rules*, and *data access* act as the three tiers. Figure 2.7(b) shows another architecture used by database and other application package vendors. The presentation layer displays information to the user and allows data entry. The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS. The bottom layer includes all data management services. The middle layer can also act as a Web server, which retrieves query results from the database server and formats them into dynamic Web pages that are viewed by the Web browser at the client side.

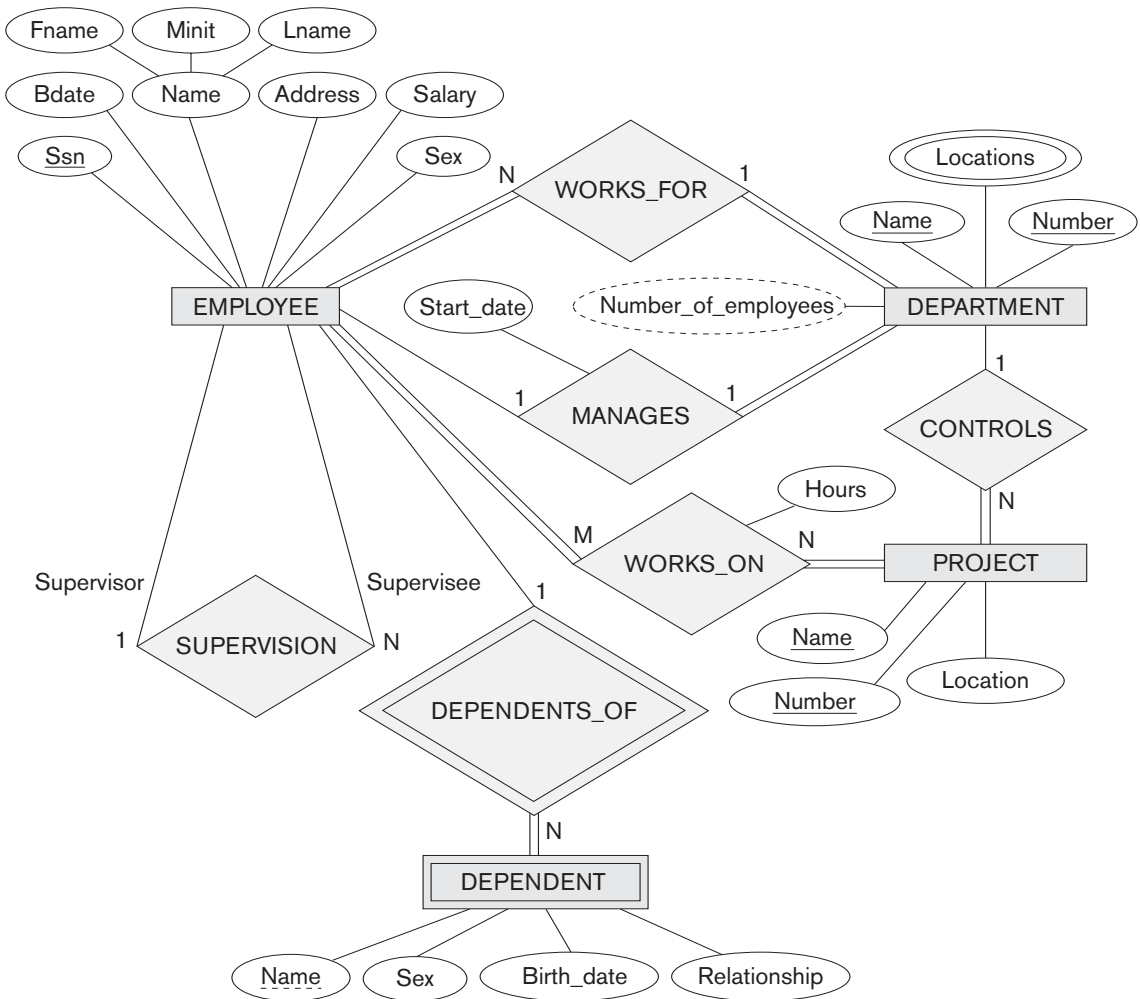
Other architectures have also been proposed. It is possible to divide the layers between the user and the stored data further into finer components, thereby giving rise to  $n$ -tier architectures, where  $n$  may be four or five tiers. Typically, the business logic layer is divided into multiple layers. Besides distributing programming and data throughout a network,  $n$ -tier applications afford the advantage that any one tier can run on an appropriate processor or operating system platform and can be handled independently. Vendors of ERP (enterprise resource planning) and CRM (customer relationship management) packages often use a *middleware layer*, which accounts for the front-end modules (clients) communicating with a number of back-end databases (servers).

**Figure 7.1**

A simplified diagram to illustrate the main phases of database design.

the known **functional requirements** of the application. These consist of the user-defined **operations** (or **transactions**) that will be applied to the database, including both retrievals and updates. In software design, it is common to use *data flow diagrams*, *sequence diagrams*, *scenarios*, and other techniques to specify functional requirements. We will not discuss any of these techniques here; they are usually described in detail in software engineering texts. We give an overview of some of these techniques in Chapter 10.

Once the requirements have been collected and analyzed, the next step is to create a **conceptual schema** for the database, using a high-level conceptual data model. This step is called **conceptual design**. The conceptual schema is a concise description of

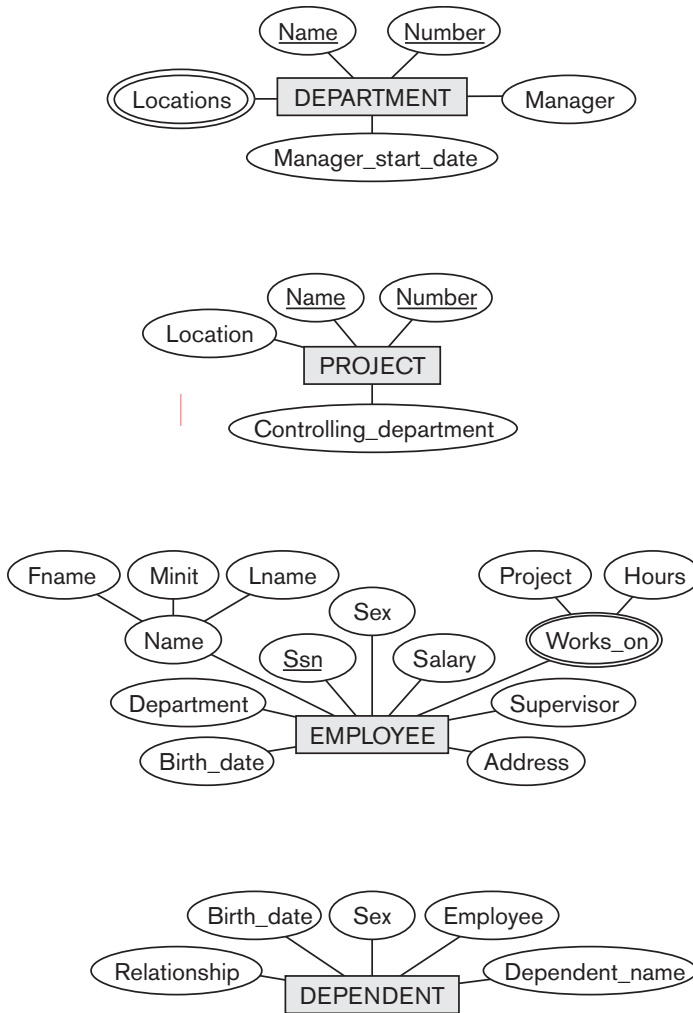
**Figure 7.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.


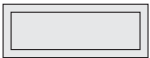

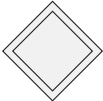


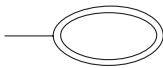
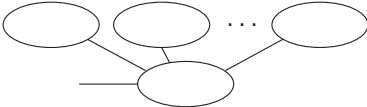

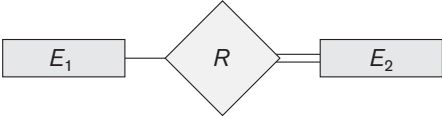
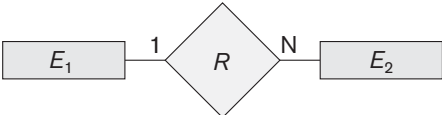
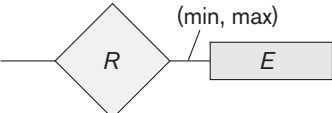
Figure 7.3 shows two entities and the values of their attributes. The EMPLOYEE entity  $e_1$  has four attributes: Name, Address, Age, and Home\_phone; their values are 'John Smith,' '2311 Kirby, Houston, Texas 77001,' '55', and '713-749-2630', respectively. The COMPANY entity  $c_1$  has three attributes: Name, Headquarters, and President; their values are 'Sunco Oil,' 'Houston', and 'John Smith', respectively.

Several types of attributes occur in the ER model: *simple* versus *composite*, *single-valued* versus *multivalued*, and *stored* versus *derived*. First we define these attribute

**Figure 7.8**

Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

So far, we have not represented the fact that an employee can work on several projects, nor have we represented the number of hours per week an employee works on each project. This characteristic is listed as part of the third requirement in Section 7.2, and it can be represented by a multivalued composite attribute of **EMPLOYEE** called Works\_on with the simple components (Project, Hours). Alternatively, it can be represented as a multivalued composite attribute of **PROJECT** called Workers with the simple components (Employee, Hours). We choose the first alternative in Figure 7.8, which shows each of the entity types just described. The Name attribute of **EMPLOYEE** is shown as a composite attribute, presumably after consultation with the users.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1 : N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

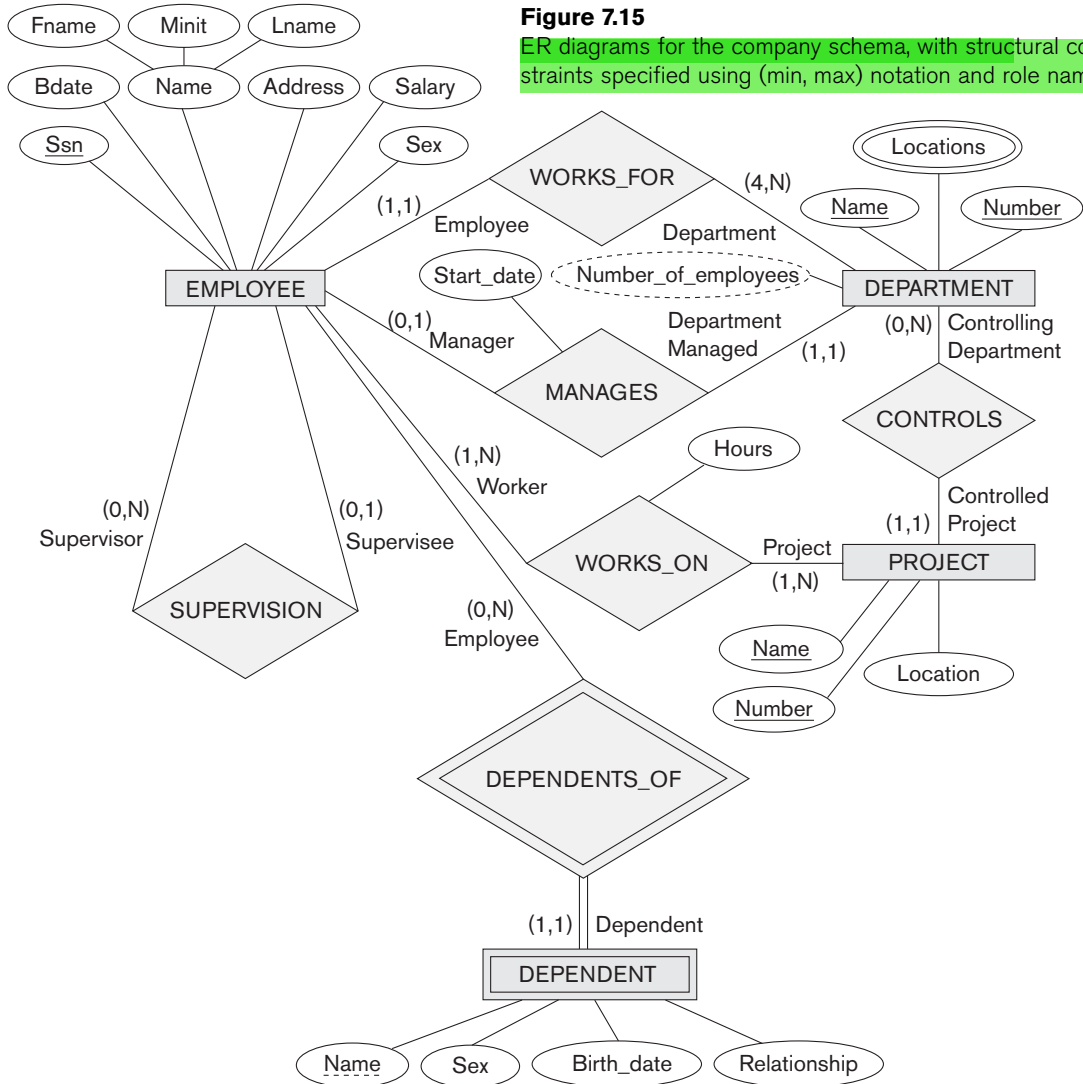
**Figure 7.14**

Summary of the notation for ER diagrams.



max relationship instances in  $R$  at any point in time. In this method,  $\min = 0$  implies partial participation, whereas  $\min > 0$  implies total participation.

Figure 7.15 displays the COMPANY database schema using the (min, max) notation.<sup>14</sup> Usually, one uses either the cardinality ratio/single-line/double-line notation or the (min, max) notation. The (min, max)



<sup>14</sup>In some notations, particularly those used in object modeling methodologies such as UML, the (min, max) is placed on the *opposite sides* to the ones we have shown. For example, for the WORKS\_FOR relationship in Figure 7.15, the (1,1) would be on the DEPARTMENT side, and the (4,N) would be on the EMPLOYEE side. Here we used the original notation from Abrial (1974).

notation is more precise, and we can use it to specify some structural constraints for relationship types of *higher degree*. However, it is not sufficient for specifying some key constraints on higher-degree relationships, as discussed in Section 7.9.

Figure 7.15 also displays all the role names for the COMPANY database schema.

## 7.8 Example of Other Notation: UML Class Diagrams

The UML methodology is being used extensively in software design and has many types of diagrams for various software design purposes. We only briefly present the basics of **UML class diagrams** here, and compare them with ER diagrams. In some ways, class diagrams can be considered as an alternative notation to ER diagrams. Additional UML notation and concepts are presented in Section 8.6, and in Chapter 10. Figure 7.16 shows how the COMPANY ER database schema in Figure 7.15 can be displayed using UML class diagram notation. The *entity types* in Figure 7.15 are modeled as *classes* in Figure 7.16. An *entity* in ER corresponds to an *object* in UML.

**Figure 7.16**  
The COMPANY conceptual schema  
in UML class diagram notation.

