Advanced Database Management Systems

Lecture 8 – Sections 8.4-8.5
SQL Queries

SQL

- ***SQL*** (Structured Query Language)
  is the standard language for commercial DBMSs
- ***SEQUEL*** (Structured English QUEry Language)
  was originally defined by IBM for SYSTEM R
  - mid 1970s
  - unofficial pronunciation (**see**-kw*uh*l) still sometimes used

- standardization of SQL began in the 80s
- current standard is SQL-99
  - subsequent revisions are not fully accepted by all vendors

- SQL is more than a query language:
  it includes a DDL, DML and admin commands

SQL commands

- Administration:
  - CREATE DATABASE
  - CREATE SCHEMA
  - SET ROLE
  - GRANT PRIVILEGES
- Data Definition:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
  - CREATE VIEW
- Data Modification:
  - INSERT
  - DELETE
  - UPDATE
- Queries:
  - SELECT

48 commands listed in *SQL in a Nutshell*

SQL Queries

- Queries in SQL are variations of the SELECT command

- Basic SQL queries correspond to the following relational algebra operations:
  - select σ
  - project π
  - cross product X
  - joins must be expressed as σ and X

Basic SELECT Command

project π

SELECT <attribute list>
FROM <table list>
WHERE <condition>;

cross product X

select σ

Single Table Queries (σ and π)

$$\pi_{Ssn}(\sigma_{Salary > 60000}(\text{EMPLOYEE}))$$

```
SELECT Ssn
FROM EMPLOYEE
WHERE Salary > 60000;
```

$$\pi_{City,State}(\sigma_{Airport\_code = \text{'SFO'}}(\text{AIRPORT}))$$

```
SELECT City, State
FROM AIRPORT
WHERE Airport_code = 'SFO';
```

Join as Select & Cross

- In the basic SELECT/FROM/WHERE form, joins must be expressed as using σ and x

$$\pi_{Lname,\ Dname}(\text{EMPLOYEE} \bowtie_{Ssn=Mgr\_ssn} \text{DEPARTMENT})$$

$$\pi_{Lname,\ Dname}(\sigma_{Ssn=Mgr\_ssn}(\text{EMPLOYEE X DEPARTMENT}))$$

```
SELECT Lname, Dname
FROM EMPLOYEE, DEPARTMENT
WHERE Ssn = Mgr_ssn;
```

Basic SQL Queries

- Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT    FNAME, LNAME, ADDRESS
FROM      EMPLOYEE, DEPARTMENT
WHERE     DNAME='Research' AND DNUMBER=DNO
```

selection condition

join condition

Basic SQL Queries

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
SELECT   PNUMBER, DNUM, LNAME, BDATE, ADDRESS

FROM     PROJECT, DEPARTMENT, EMPLOYEE

WHERE  DNUM=DNUMBER AND MGRSSN=SSN

       AND PLOCATION='Stafford'
```

join PROJECT and DEPARTMENT

select

join DEPARTMENT and EMPLOYEE

Tuple Variables (Aliases)

- We can give names to the tuples coming from each of the input relations

```
SELECT    E.Lname, D.Dname
FROM      EMPLOYEE E, DEPARTMENT D
WHERE     E.Ssn = D.Mgr_ssn;
```

- This can disambiguate common attribute names and improve readability

Renaming Attributes

- Attributes can also be renamed in the FROM clause
  - similar to alternate rename syntax in the algebra

```
SELECT    Fn, Ln
FROM      EMPLOYEE E(Fn, Mi, Ln, Bd, Ad, Sx, Sl, Sssn, Dn)
WHERE     Dn = 4;
```

Self Join

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     EMPLOYEE E, EMPLOYEE S
WHERE    E.SUPERSSN=S.SSN
```

- Aliases are necessary for this query
- Think of E and S as two different copies of EMPLOYEE
  - E represents employees in role of *supervisees* and
    S represents employees in role of *supervisors*

Aliases: alternate syntax

- Can also use the AS keyword to specify aliases

```
SELECT    E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM      EMPLOYEE AS E, EMPLOYEE AS S
WHERE     E.SUPERSSN=S.SSN
```

- Can also simply use the relation names (when non-ambiguous)

```
SELECT    EMPLOYEE.Lname, DEPARTMENT.Dname
FROM      EMPLOYEE, DEPARTMENT
WHERE     EMPLOYEE.Ssn = DEPARTMENT.Mgr_ssn;
```

No σ → No WHERE

- If there are no selection (or join) conditions, the WHERE clause can be omitted

SELECT Ssn
FROM EMPLOYEE

π_{Ssn} EMPLOYEE

- Two or more relations in FROM clause with no join is a *CROSS PRODUCT*

SELECT Lname, Dname
FROM EMPLOYEE, DEPARTMENT

$\pi_{Lname,\ Dname}$ (EMPLOYEE X DEPARTMENT)

No π → *

- To retrieve all the attribute values of the selected tuples, use *, which stands for *all the attributes*

```
SELECT      *
FROM        EMPLOYEE
WHERE       DNO=5

SELECT      *
FROM        EMPLOYEE, DEPARTMENT
WHERE       DNAME='Research' AND
                  DNO=DNUMBER
```

Tables as Sets → DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used

```
SELECT    SALARY
FROM      EMPLOYEE
```

may contain duplicates

```
SELECT    DISTINCT SALARY
FROM      EMPLOYEE
```

duplicates eliminated

Set Operations

- union operation (UNION)
  intersection (INTERSECT)
  set difference (MINUS, sometimes called EXCEPT)
  - some implementations of SQL do not support all set operations

- Set operation results ***are sets*** of tuples
  duplicate tuples are eliminated from the result

- The set operations apply only to *union compatible relations*:
  the two relations must have the same attributes and
  the attributes must appear in the same order

Set Operations: Example

- List project numbers for all projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

```
(SELECT     PNAME
FROM        PROJECT, DEPARTMENT, EMPLOYEE
WHERE       DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')

    UNION

(SELECT     PNAME
FROM        PROJECT, WORKS_ON, EMPLOYEE
WHERE       PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')
```

Multiset Operations

- UNION ALL, INTERSECT ALL, EXCEPT ALL
- Multiset operation results are multisets of tuples

  duplicate tuples are not eliminated

```
(SELECT     PNAME
FROM        PROJECT, DEPARTMENT, EMPLOYEE
WHERE       DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')
      UNION ALL
(SELECT     PNAME
FROM        PROJECT, WORKS_ON, EMPLOYEE
WHERE       PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')
```

WHERE Clause

- WHERE clause is a general boolean expression

- Boolean operators:
  AND, OR, NOT
- Comparison operators:
  =, <, <=, >, >=, <>
- String comparison operators:
  LIKE

- Parentheses can be used to set precedence
- String literals can be enclosed in "…" or '…'

String Comparison

- The **LIKE** comparison operator is used to compare partial strings

- Two wildcard characters are used:

  '%' replaces an arbitrary number of characters

  '_' replaces a single arbitrary character

String Comparison Example

- Retrieve all employees whose address is in Houston, Texas.

- The value of the ADDRESS attribute must contain the substring "Houston, TX".

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     ADDRESS LIKE '%Houston, TX%'
```

zero or more characters, before and after substring

String Comparison Example

- Retrieve all employees who were born during the 1960s.
  - '6' must be the 3rd character of the 10 character date string

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     BDATE LIKE   '_ _ 6 _ _ _ _ _ _ _'
```

  - Following would also work:

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     BDATE LIKE   '_ _ 6 %'
```

assumes date format is YYYY-MM-DD

Arithmetic Operation

- The standard arithmetic operators **'+', '-'. '*', and '/'** can be applied to numeric values in an SQL query result

- Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
SELECT    FNAME, LNAME, 1.1*SALARY
FROM      EMPLOYEE, WORKS_ON, PROJECT
WHERE     SSN=ESSN AND PNO=PNUMBER
          AND PNAME='ProductX'
```

Aggregate Functions

- Aggregate functions are applied to result attributes
  COUNT, SUM, MAX, MIN, and AVG

- Find the maximum salary, the minimum salary, and the average salary among all employees.

```
SELECT    MAX(Salary), MIN(Salary), AVG(Salary)
FROM      EMPLOYEE
```

- Find the total salary paid to employees who work for the 'Research' department.

```
SELECT    SUM(Salary)
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dno=Dnumber AND Dname='Research'
```

Aggregate Functions

- Retrieve the total number of employees in the company and the number of employees in the Research' department.

```
SELECT    COUNT (*)
FROM      EMPLOYEE

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

Join as X and σ

```
mysql> SELECT * FROM r;
+---+------+
| x | y    |
+---+------+
| 3 |    4 |
| 5 |    6 |
| 7 |    8 |
| 9 |    6 |
+---+------+
```

```
mysql> SELECT * FROM s;
+---+------+
| a | b    |
+---+------+
| 2 |    3 |
| 4 |    7 |
+---+------+
```

```
mysql> SELECT * FROM r,s WHERE y=a;
+---+------+---+------+
| x | y    | a | b    |
+---+------+---+------+
| 3 |    4 | 4 |    7 |
+---+------+---+------+
```

$$R \bowtie_{y=a} S$$

$$\sigma_{y=a}(R \times S)$$

Explicit Join

- Joins can be explicitly stated in the FROM clause.

```
SELECT * FROM (r JOIN s ON y=a);
+---+------+---+------+
| x | y    | a | b    |
+---+------+---+------+
| 3 |    4 | 4 |    7 |
+---+------+---+------+
```

$R \bowtie_{y=a} S$

Left/Right Outer Join

```
SELECT * FROM (r LEFT JOIN s ON y=a);
+---+------+------+------+
| x | y    | a    | b    |
+---+------+------+------+
| 3 |    4 |    4 |    7 |
| 5 |    6 | NULL | NULL |
| 7 |    8 | NULL | NULL |
| 9 |    6 | NULL | NULL |
+---+------+------+------+
```

```
SELECT * FROM (r RIGHT JOIN s ON y=a);
+------+------+---+------+
| x    | y    | a | b    |
+------+------+---+------+
| NULL | NULL | 2 |    3 |
|    3 |    4 | 4 |    7 |
+------+------+---+------+
```

Full Outer Join

```
SELECT * FROM r FULL OUTER JOIN s ON y=a;
```

mysql doesn't support full outer join,
so we'll substitute an equivalent query:

```
(SELECT * FROM r LEFT JOIN s ON y=a) UNION
(SELECT * FROM r RIGHT JOIN s ON y=a);
```

x	y	a	b
3	4	4	7
5	6	NULL	NULL
7	8	NULL	NULL
9	6	NULL	NULL
NULL	NULL	2	3

Ordering Results

- An ORDER BY clause can be added to order the result tuples

```
SELECT * FROM t;
+----+--------+
| i  | j      |
+----+--------+
| 10 | ten    |
| 11 | eleven |
| 20 | twenty |
|  4 | four   |
+----+--------+
```

```
SELECT * FROM t
ORDER BY j;
+----+--------+
| i  | j      |
+----+--------+
| 11 | eleven |
|  4 | four   |
| 10 | ten    |
| 20 | twenty |
+----+--------+
```

```
SELECT * FROM t
ORDER BY i;
+----+--------+
| i  | j      |
+----+--------+
|  4 | four   |
| 10 | ten    |
| 11 | eleven |
| 20 | twenty |
+----+--------+
```

ORDER BY Examples

- order by Lname first,
  then by Fname if Lname is the same:

```
SELECT Lname, Fname
FROM Employee
WHERE salary > 60000
ORDER BY Lname, Fname
```

- order by Lname in ascending order,
  then by salary in descending order

```
SELECT Lname, salary
FROM Employee
WHERE salary > 60000
ORDER BY Lname ASC, salary DESC
```

Grouping

- Forms groups (subsets) of result tuples before applying aggregate functions
- Example: count the number of employees in each department

(group employees by DNO, then count tuples in each group)

```
SELECT Dno, COUNT(*)
FROM Employee
GROUP BY Dno
```

$_{DNO}\mathcal{F}_{COUNT\ *}(EMPLOYEE)$

Dno	COUNT
8	120
22	238
7	82
20	169

GROUP BY Example

- For each project, get the project name, project number and the number of employees working on that project

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

Attributes in SELECT clause must be aggregates or must appear in the GROUP BY clause

Filtering Groups: HAVING

- We can throw away some groups
  by adding a condition in a HAVING clause

- example:
  for each project *that has more than two employees*,
  get the project name, project number and
  the number of employees working on that project

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

GROUP BY Examples

```
SELECT * FROM e;
+-----+--------+-------+
| eid | salary | dept  |
+-----+--------+-------+
| E01 |  65000 | ADMIN |
| E12 |  58400 | ENGR  |
| E08 |  76900 | ENGR  |
| E23 |  63800 | ADMIN |
| E07 |  56900 | ADMIN |
| E27 |  76400 | ENGR  |
| E14 |  48000 | TEST  |
+-----+--------+-------+
```

```
SELECT COUNT(*) FROM e
GROUP BY dept;
+----------+
| count(*) |
+----------+
|        3 |
|        3 |
|        1 |
+----------+
```

```
SELECT dept, COUNT(*)
FROM e GROUP BY dept;
+-------+----------+
| dept  | count(*) |
+-------+----------+
| ADMIN |        3 |
| ENGR  |        3 |
| TEST  |        1 |
+-------+----------+
```

GROUP BY Examples

```
SELECT dept, COUNT(*)
FROM e
GROUP BY dept
HAVING COUNT(*) > 1;
+-------+----------+
| dept  | count(*) |
+-------+----------+
| ADMIN |        3 |
| ENGR  |        3 |
+-------+----------+
```

```
SELECT dept, AVG(salary)
FROM e
GROUP BY dept
HAVING COUNT(*) > 1;
+-------+-------------+
| dept  | AVG(salary) |
+-------+-------------+
| ADMIN |       61900 |
| ENGR  | 70566.66667 |
+-------+-------------+
```

Nested Queries

- Nested queries can be used as set values in the WHERE clause

- Set comparison operators
  - IN – set membership (“is in”, ∈)
  - EXISTS – set not empty (∃)
  - ALL – applies to all set members (∀)
  - ANY – applies to any set member
  - CONTAINS – proper superset

Nested Queries

- find all employees who work on a project with John Smith

```
SELECT Lname, Fname
FROM EMPLOYEE E1, WORKS_ON W1
WHERE E1.SSN = W1.ESSN
  AND W1.Pno IN (SELECT Pno
                 FROM EMPLOYEE E2, WORKS_ON W2
                 WHERE E2.SSN = W2.ESSN
                   AND E2.Fname = "John"
                   AND E2.Lname = "Smith")
```

Nested Queries

- find the highest paid employee in department 5

```
SELECT Lname, Fname
FROM EMPLOYEE E1
WHERE E1.Dno=5
  AND E1.Salary > ALL (SELECT E2.Salary
                       FROM EMPLOYEE E2
                       WHERE E2.Dno=5)
```

Nested Queries

- List names of managers who have dependents

```
SELECT Lname, Fname
FROM EMPLOYEE E1
WHERE EXISTS (SELECT *
              FROM DEPENDENT D1
              WHERE E1.Ssn = D1.Essn)
      AND
      EXISTS (SELECT *
              FROM DEPARTMENT D2
              WHERE E1.Ssn = D2.Mgr_ssn)
```

This is an example of a *correlated nested query*,
since the nested queries refer to the relations in the outer query.

Nested Queries

- List names of employees who work on all projects controlled by department 5

```
SELECT Lname, Fname
FROM EMPLOYEE E
WHERE (SELECT W.Pno
               FROM WORKS_ON W
               WHERE E.Ssn = W.Essn)
      CONTAINS
      (SELECT P.Pnumber
               FROM PROJECT P
               WHERE P.Dnum=5)
```

Nested Queries

- List names of all projects controlled by department 5 or department 7

```
SELECT P.Pname
FROM PROJECT P
WHERE P.Dnum IN (5,7)
```

explicit set of values

SELECT: Syntax Summary

SELECT <attribute and function list>
FROM <table list>
WHERE <condition>
GROUP BY <grouping attributes>
HAVING <group condition>
ORDER BY <attribute list>

required

optional

SELECT: conceptual execution

1. FROM: cross product of tables
2. WHERE: select tuples
3. GROUP BY: group tuples
4. HAVING: filter groups
5. SELECT: project attributes and apply aggregates
6. ORDER BY: sort the tuples

This is not an efficient way to execute the query,
simply a way to define the meaning of the query conceptually.

EXERCISE 1: Queries

1. First and last name of employees who have no supervisor.
2. First and last name of employees supervised by Franklin Wong.
3. Last name of employees who have dependents.
4. Last name of employees who have daughters.
5. Last name of employees in department 5 who work more than 10 hours/week on ProductX.
6. Last name of supervisors of employees in department 5 who work more than 10 hours/week on ProductX.
7. First and last names of all department managers.
8. Salaries of all employees who have worked on the Reorganization project.
9. SSN of all employees who have worked on a project that is controlled by a department different than the department that they are assigned to.
10. Last name of all employees who are not married.

EXERCISE 2: Queries

1. List all airplane types that can land at any airport in San Francisco.
2. List the ids and number of seats for all airplanes that can land at any airport in Chicago.
3. List the name and phone number of all customers with a seat reserved on a flight that leaves Chicago O'Hara airport (ORD) on October 31, 2008.
4. List all airlines that have seats available for flights leaving Los Angeles (LAX) on September 25, 2008.
5. List all airlines that operate at San Jose International Airport (SJC).

EXERCISE 3: Queries

1. Count the number of overdue books.

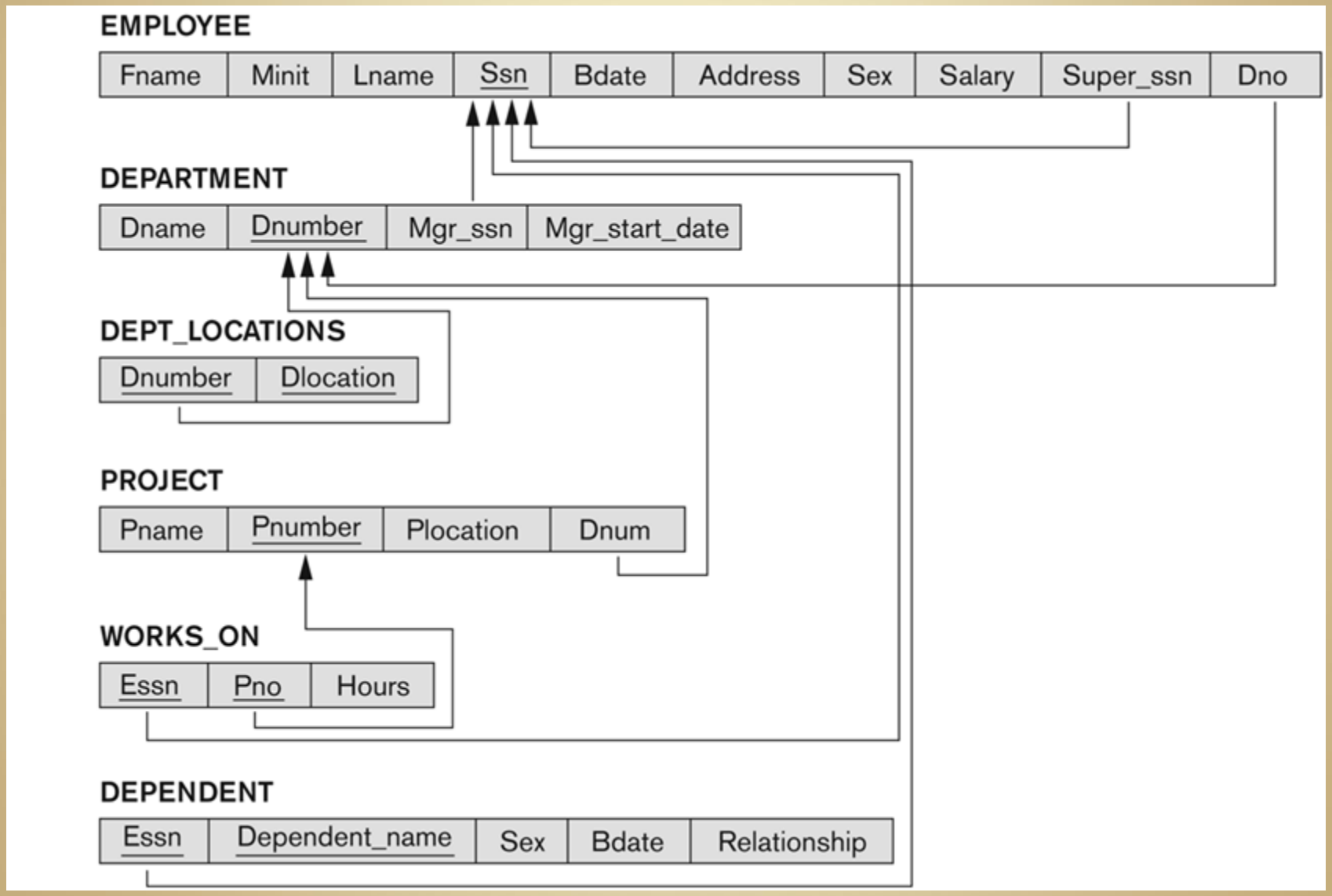
2. How many books by author Harry Crews are in the database?

3. Determine the number of library cards assigned to each borrower phone number.

4. Find names of all borrowers who do not have any book loans.

5. Do any library branches have every book?

EXERCISE 1: Schema



EXERCISE 1: Instance

EMPLOYEE

Fname	Minit	Lname	Ssn	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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	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	Pnumber	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

Essn	Dependent_name	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

Exercise 2: Schema

AIRPORT

Airport_code	Name	City	State

FLIGHT

Flight_number	Airline	Weekdays

FLIGHT_LEG

Flight_number	Leg_number	Departure_airport_code	Scheduled_departure_time	Arrival_airport_code	Scheduled_arrival_time

LEG_INSTANCE

Flight_number	Leg_number	Date	Number_of_available_seats	Airplane_id	Departure_airport_code	Departure_time	Arrival_airport_code	Arrival_time

FARE

Flight_number	Fare_code	Amount	Restrictions

AIRPLANE_TYPE

Airplane_type_name	Max_seats	Company

CAN_LAND

Airplane_type_name	Airport_code

AIRPLANE

Airplane_id	Total_number_of_seats	Airplane_type

SEAT_RESERVATION

Flight_number	Leg_number	Date	Seat_number	Customer_name	Customer_phone

EXERCISE 3: Schema