DATABASE ASSIGNMENT 5

Vinayak Raghupathy

Netid-vr840

8.15 Show the result of each of the sample queries in Section 8.5 as it would apply

to the database state in Figure 5.6.

Query 1: Find the name and address of all employees who work for the 'Research‘ department.

|  |  |  |
| --- | --- | --- |
| FNAME | LNAME | ADDRESS |
| John | Smith | 731 Fondren,Houston,TX |
| Franklin | Wong | 638 Voss, Houston, TX |
| Joyce | English | 5631 Rice, Houston, TX |
| Ramesh | Narayan | 975 Fire Oak, Humble, TX |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PNUMBER | DNUM | LNAME | ADDRESS | BDATE |
| 10 | 4 | Wallace | 291Berry, Bellaire, TX | 20-JUN-31 |
| 30 | 4 | Wallace | 291Berry, Bellaire, TX | 20-JUN-31 |

Query 3: Find the names of all employees who work on all the projects controlled by dpt no 5

No Results. Dept 5 controls projects 1 , 2 ,3 and there are no employees who control all 3 projects

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

|  |
| --- |
| PNO |
| 1 |
| 2 |

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

|  |  |
| --- | --- |
| FNAME | LNAME |
| John | Smith |
| Franklin | Wong |

QUERY 6: List the names of employees who have no dependents.

|  |  |
| --- | --- |
| FNAME | LNAME |
| Alicia | Zeleiya |
| Ramesh | Narayan |
| Joyce | English |
| James | Borg |
| Ahmad | Jabbar |

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

|  |  |
| --- | --- |
| FNAME | LNAME |
| Jennifer | Wallace |
| Franklin | Wong |

8.18)

1. How many copies of the book titled *The Lost Tribe* are owned by the library branch whose name is ‘Sharpstown’?

Ans)

|  |  |  |
| --- | --- | --- |
| Sharps\_id | ← | *π*Branch\_id(*σ*Branch name=’Sharpstown’(Library Branch)) |
| Tribe\_id | ← | *π*Book\_id(*σ*Title=’TheLostTribe’(Book)) |
| Answer | ← | *π*No\_of\_copies(Book Copies ∗ Sharps id ∗ Tribe id) |

The Above can be written in a single line as below:

*π*No of copies(*σ*Branch name=’Sharpstown’∧Title=’TheLostTribe’(Book Copies ∗ Library Branch ∗ Book))

1. How many copies of the book titled *The Lost Tribe* are owned by each library branch?

*π*Name, No\_of\_copies(*σ*Branch name=’TheLostTribe’(Book Copies ∗ Library Branch ∗ Book))

1. Retrieve the names of all borrowers who do not have any books checked out.

None\_id ← *π*Card no(Borrower) −*π*Card no(Book\_Loans)   
Answer ← *π*Name(Borrower ∗ None\_id)

1. For each book that is loaned out from the Sharpstown branch and whose Due\_date is today, retrieve the book title, the borrower’s name, and the borrower’s address.

(temp1,Branch\_id(Branch\_name=’Sharpstown’(LIBRARY\_BRANCH)))

(temp2, Due\_date=TODAY(BOOK\_LOANS)  temp1)

(temp3, temp2  BOOK  BORROWER)

 Title, Name, Address (temp3)

1. For each library branch, retrieve the branch name and the total number of books loaned out from that branch.

(temp1, LIBRARY\_BRANCHBOOK\_LOANS)

 Branch\_name, count(γ Branch\_id, count()(temp1))

1. Retrieve the names, addresses, and number of books checked out for all borrowers who have more than five books checked out.

|  |  |  |
| --- | --- | --- |
| Loan\_counts | ← | *ρ*(Card\_no*,* Book\_count)(Card no=Count(Book id)(Book Loans)) |
| Big\_borrowers | ← | *σ*Book\_count*>*5(Loan counts) |
| Answer | ← | *π*Name*,* Address*,* Book\_count(Big borrowers ∗ Borrower) |
|  |  |  |

1. For each book authored (or coauthored) by Stephen King, retrieve the title and the number of copies owned by the library branch whose name is Central.

(temp1, Branch\_id(Branch\_name=’Central’LIBRARY\_BRANCH) )

(temp2, temp1BOOK\_COPIES)

(temp3, Author\_name =’Stephen King’ BOOK\_AUTHORS))

Title, No\_of\_copies(temp2temp3BOOK)

**8.22)**

* + 1. T1  T1.P=T2.A T2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | R | A | B | C |
| 10 | a | 5 | 10 | b | 6 |
| 10 | a | 5 | 10 | b | 5 |
| 25 | a | 6 | 25 | c | 3 |

* + 1. T1  T1.q=T2.b T2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | R | A | B | C |
| 15 | b | 8 | 10 | b | 6 |
| 15 | b | 8 | 10 | b | 5 |

* + 1. T1 T1.p=T2.a T2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | R | A | B | C |
| 10 | a | 5 | 10 | b | 6 |
| 10 | a | 5 | 10 | b | 5 |
| 25 | a | 6 | 25 | c | 3 |
| 15 | b | 8 | NULL | NULL | NULL |

* + 1. T1 T1.q= T2.b T2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | R | A | B | C |
| 15 | b | 8 | 10 | b | 6 |
| 15 | b | 8 | 10 | b | 5 |
| NULL | NULL | NULL | 25 | c | 3 |

* + 1. T1 U T2

|  |  |  |
| --- | --- | --- |
| P | Q | R |
| 10 | a | 5 |
| 15 | b | 8 |
| 25 | a | 6 |
| 10 | b | 6 |
| 25 | c | 3 |
| 10 | b | 5 |

* + 1. T1  (T1.p=T2.a AND T1.R=T2.C) T2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | R | A | B | C |
| 10 | a | 5 | 10 | b | 6 |

8.30)

1. Tuple relational Calculus: { t | R(t) AND t.A=c}

Domain relational Calculus: { xyz | R(xyz) AND x=c }

1. Tuple relational Calculus: { t.A, t.B | R(t) }

Domain relational Calculus: { xy | R(xyz) }

1. Tuple relational Calculus {t.A, t.B, t.C, q.D, q.E | R(t) AND S(q) AND t.C=q.C }

Domain relational Calculus: { xyzvw | R(xyz) AND (EXISTS u) ( S(uvw) AND z=u ) }

1. Tuple relational Calculus: { t | R(t) OR S(t) }

Domain relational Calculus: { xyz | R(xyz) OR S(xyz) }

1. Tuple relational Calculus: { t | R(t) AND S(t) }

Domain relational Calculus: { xyz | R(xyz) AND S(xyz) }

1. Tuple relational Calculus: { t | R(t) AND NOT(S(t)) }

Domain relational Calculus: { xyz | R(xyz) AND NOT(S(xyz)) }

1. Tuple relational Calculus: { t.A, t.B, t.C, q.D, q.E, q.F | R(t) AND S(q) }

Domain relational Calculus: ( xyzuvw | R(xyz) AND S(uvw) }

1. Tuple relational Calculus:

{ t.B | R(t) AND (FORALL s) ( NOT(S(s)) OR (EXISTS q) ( R(q) AND s.A=q.A AND q.B=t.B ) ) }

Domain relational Calculus:

{ y | R(xy) AND (FORALL z) ( NOT(S(z)) OR (EXISTS u) ( R(uy) AND z=u ) }

6.8) The following referential integrity constraints should hold (we use the notation:

R.(A1, ..., An) --> S.(B1, ..., Bn) to represent a foreign key from the attributes A1, ..., An of R (the referencing relation) to S (the referenced relation)):

PREREQUISITE.(CourseNumber) --> COURSE.(CourseNumber)

PREREQUISITE.(PrerequisiteNumber) --> COURSE.(CourseNumber)

SECTION.(CourseNumber) --> COURSE.(CourseNumber)

GRADE\_REPORT.(StudentNumber) --> STUDENT.(StudentNumber)

GRADE\_REPORT.(SectionIdentifier) --> SECTION.(SectionIdentifier)

One possible set of CREATE TABLE statements to define the database is given below:

CREATE TABLE STUDENT (Name VARCHAR(30) NOT NULL,

StudentNumber INTEGER NOT NULL,

Class CHAR NOT NULL,

Major CHAR(4), PRIMARY KEY (StudentNumber) );

CREATE TABLE COURSE ( CourseName VARCHAR(30) NOT NULL,

CourseNumber CHAR(8) NOT NULL,

CreditHours INTEGER,

Department CHAR(4),

PRIMARY KEY (CourseNumber),

UNIQUE (CourseName));

CREATE TABLE PREREQUISITE ( CourseNumber CHAR(8) NOT NULL,

PrerequisiteNumber CHAR(8) NOT NULL,

PRIMARY KEY (CourseNumber, PrerequisiteNumber),

FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber),

FOREIGN KEY (PrerequisiteNumber) REFERENCES COURSE (CourseNumber));

CREATE TABLE SECTION (SectionIdentifier INTEGER NOT NULL,

CourseNumber CHAR(8) NOT NULL,

Semester VARCHAR(6) NOT NULL,

Year CHAR(4) NOT NULL,

Instructor VARCHAR(15),

PRIMARY KEY (SectionIdentifier),

FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber));

CREATE TABLE GRADE\_REPORT ( StudentNumber INTEGER NOT NULL,

SectionIdentifier INTEGER NOT NULL,

Grade CHAR, PRIMARY KEY (StudentNumber, SectionIdentifier),

FOREIGN KEY (StudentNumber) REFERENCES STUDENT (StudentNumber),

FOREIGN KEY (SectionIdentifier) REFERENCES SECTION (SectionIdentifier));

6.12)

1. Retrieve the names of all senior students majoring in ‘cs’ (computer science).

SELECT Name

FROM student

WHERE Major = "CS"

1. Retrieve the names of all courses taught by Professor King in 2007 and 2008.

SELECT c.Course\_name

FROM section s JOIN course c ON s.Course\_number = c.Course\_number

WHERE s.Instructor = "King" AND (s.Year = "08" OR s.Year = "07");

1. For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.

SELECT s.Course\_number, s.Semester, s.Year, COUNT(g.Student\_number) AS Num\_Students

FROM section s JOIN grade\_report g ON s.Section\_identifier = g.Section\_identifier

WHERE s.Instructor = "King"

1. Retrieve the name and transcript of each senior student (Class = 4) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

SELECT st.name, c.Course\_name, c.Course\_number, c.Credit\_hours, s.Semester, s.Year, g.Grade

FROM student st, course c, grade\_report g, section s

WHERE st.Student\_number = g.Student\_number

AND g.Section\_identifier = s.Section\_identifier

AND s.Course\_number = c.Course\_number

AND st.Class = "2";

SELECT st.name, c.Course\_name, c.Course\_number, c.Credit\_hours, s.Semester, s.Year, g.Grade

FROM course c

INNER JOIN section s ON c.Course\_number = s.Course\_number

INNER JOIN grade\_report g ON s.Section\_identifier = g.Section\_identifier

INNER JOIN student st ON g.Student\_number = st.Student\_number

WHERE st.Class = "2";