Chapter 6, The Relational Algebra and Relational Calculus

6.1 Unary Relational Operations: SELECT and PROJECT

6.1.1 The SELECT Operation

- SELECT a subset of tuples from R that satisfy a selection condition.
  - $\sigma_{<selection\ condition>}(R_1)$
  - $\sigma_{(DNO=4\ and\ SALARY>25000)\ or\ (DNO=5\ and\ SALARY>30000)}(EMPLOYEE)$
    See Figure 6.1(a) (Fig 7.8(a) on e3) for the result.
  - The resulting relation R_2 after applying selection on R_1, we have

$$degree(R_1) = degree(R_2)\ and$$

$$\mid R_2 \mid \leq \mid R_1 \mid\ \ for\ any\ selection\ condition$$

  - Commutative: $\sigma_{C_1}(\sigma_{C_2}(R)) = \sigma_{C_2}(\sigma_{C_1}(R))$
  - $\sigma_{C_1}(\sigma_{C_2}(\ldots(\sigma_{C_n}(R))\ldots)) = \sigma_{C_1\ and\ C_2\ and......and\ C_n}(R)$

6.1.2 The PROJECT Operation

- PROJECT some columns (attributes) from the table (relation) and discard the other columns.
  - $R_2 \leftarrow \pi_{<attribute\ list>}(R_1)$
  - $R_2 \leftarrow \pi_{LNAME,\ SALARY}(EMPLOYEE)$
  - R_2 has only the attributes in $<\ attribute\ list\ >$ with the same order as they appear in the list.
  - $degree(R_2) = |<\ attribute\ list\ >|$
  - PROJECT operation will remove any duplicate tuples (**duplication elimination**). This happens when attribute list contains only non-key attributes of R_1.

– $| R_2 | \leq | R_1 |$. If the $< attribute\ list >$ is a superkey of R_1, then $| R_2 | = | R_1 |$

– $\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$, where $< list2 >$ must be a subset of $< list1 >$.

6.1.3 Sequences of Operations and the RENAME Operation

- It may apply several relational algebra operations one after another to get the final result.
  - Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
  - we can apply one operation at a time and create intermediate result relations.
- For example, the algebra expression $\pi_{FNAME,\ LNAME,\ SALARY}(\sigma_{DNO=5}(EMPLOYEE))$ is equivalent to

$$\begin{cases} DEP5\_EMPS \longleftarrow \sigma_{DNO=5}\ (EMPLOYEE) \\ RESULT \longleftarrow \pi_{FNAME,\ LNAME,\ SALARY}(DEP5\_EMPS) \end{cases}$$

- We could rename the above intermediate (or final) relations by

$$\begin{cases} TEMP \longleftarrow \sigma_{DNO=5}(EMPLOYEE) \\ R(FN,\ LN,\ SALARY) \longleftarrow \pi_{FNAME,\ LNAME,\ SALARY}(TEMP) \end{cases}$$

  or we can define a RENAME operation and rewrite the query as follows.
  - $\rho_{S(B_1,B_2,\ldots,B_n)}(R)$ or
  - $\rho_S(R)$ or
  - $\rho_{(B_1,B_2,\ldots,B_n)}(R)$
  - Where S is the renamed relation name of R and B_i's are the renamed attribute names of R.
  - The query becomes

$$\begin{cases} \rho_{TEMP}(\sigma_{DNO=5}(EMPLOYEE)) \\ \rho_{R(FN,\ LN,\ SALARY)}(\pi_{FNAME,\ LNAME,\ SALARY}(TEMP)) \end{cases}$$

6.2 Relational Algebra Operations from Set Theory

6.2.1 The UNION, INTERSECTION, and MINUS Operations

- $R_1 \cup R_2$ (UNION), $R_1 \cap R_2$ (INTERSECTION), or $R_1 - R_2$ (SET DIFFERENCE) are valid operations iff R_1 and R_2 are **union compatible**.
- Two relations $R_1(A_1, A_2, \ldots, A_n)$ and $R_2(B_1, B_2, \ldots, B_n)$ are union compatible if $degree(R_1) = degree(R_2)$ and $dom(A_i) = dom(B_i)$ for $1 \le i \le n$.
- The resulting relation has the same attribute names as the first relation R_1
- Commutative: UNION, INTERSECTION
- Associative: UNION, INTERSECTION
- See Figure 6.4 (or Fig 7.11 on 3e)

6.2.2 The CARTESIAN PRODUCT (or CROSS PRODUCT) Operation

- $R_1 \times R_2$ (R_1 and R_2 do not need to be union compatible)
- $R_1(A_1, A_2, \ldots, A_n) \times R_2(B_1, B_2, \ldots, B_m) = Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$, where
  - $\mid Q \mid = \mid R_1 \mid \times \mid R_2 \mid$
  - For example,

B_1

A_1	A_2
a_{11}	a_{21}
a_{12}	a_{22}

$\times$ B_2

B_1
b_{11}
b_{12}
b_{13}

$=$ Q

A_1	A_2	B_1
a_{11}	a_{21}	b_{11}
a_{11}	a_{21}	b_{12}
a_{11}	a_{21}	b_{13}
a_{12}	a_{22}	b_{11}
a_{12}	a_{22}	b_{12}
a_{12}	a_{22}	b_{13}

  - See Figure 6.5 (Fig 7.12 on e3). This figures shows a possible sequence of steps to retrieve a list of names of each female employee's dependents.

6.3 Binary Relational Operations: JOIN and DIVISION

6.3.1 The JOIN Operation

- JOIN is used to combine related tuples from two relations into single tuples.
- $R_1(A_1, A_2, \ldots, A_n) \bowtie_{<join\ condition>} R_2(B_1, B_2, \ldots, B_m) = Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$; where each tuple in Q is the combination of tuples – one from R_1 and one from R_2 – whenever the combination satisfies the join condition.
- $R_1 \bowtie_{<condition>} R_2 \equiv \sigma_{<condition>}(R_1 \times R_2)$
  - For example, see Figure 6.6 (Figure 7.13 on e3),

$$\text{ACTUAL\_DEPENDENT} \longleftarrow \text{EMPLOYEE} \bowtie_{SSN=ESSN} \text{DEPENDENT}$$
$$\equiv \text{ACTUAL\_DEPENDENT} \longleftarrow \sigma_{SSN=ESSN}\ (\text{EMPLOYEE} \times \text{DEPENDENT})$$

- Usually the join condition is of the form:

$$< A_{i_1} \theta B_{j_1} > \ AND \ < A_{i_2} \theta B_{j_2} > \ AND \ldots AND \ < A_{i_p} \theta B_{j_p} >$$

  Each attribute pair in the condition must have the same domain; θ is one of the comparison operator.

6.3.2 The EQUIJOIN and NATURAL JOIN Variations

- All JOIN operations with only "=" operator used in the conditions are called EQUIJOIN.
- Each tuple in the resulting relation of an EQUIJOIN has the same values for each pair of attributes listed in the join condition.
- **NATUAL JOIN ($*$)** was created to get rid of the superfluous attributes in an EQUIJOIN.
  - NATUAL JOIN requires each pair of join attributes have the same name in both relations, otherwise, a renaming operation should be applied first.
  - For example, see 6.7 (Figure 7.14 on e3),

DEPT_LOCS $\longleftarrow$ DEPARTMENT $*$ DEPT_LOCATIONS

PROJ_DEPT $\longleftarrow$ $\rho_{(DNAME,\ DNUM,\ MGRSSN,\ MGRSTARTDATE)}$ (DEPARTMENT) $*$ PROJECT

- $0 \leq |\ R_1(A_1, A_2, \ldots, A_n)\ \bowtie_{<COND>}\ R_2(B_1, B_2, \ldots, B_m)\ | \leq n \times m$

6.3.3 A Complete Set of Relational Algebra Operations

- $\{\sigma, \pi, \cup, -, \times\}$ is a complete set; that is, any of the other relational algebra operations can be expressed as a sequence of operations from this set. For example,
  - $R \cap S \equiv (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{<COND>} S \equiv \sigma_{<COND>}(R \times S)$
  - A NATUAL JOIN can be specified as a CARTESIAN PRODUCT preceded by RENAME and followed by SELECT and PROJECT operations.

6.3.4 The DIVISION Operation

- $T(Y) \longleftarrow R(Z) \div S(X)$, where X, Y, Z are sets of attributes and $X \subseteq Z$ and $Y = Z - X$
- A tuple $t \in T$ if tuples $t_1 \in R$ with $t_1[Y] = t$ and with $t_1[X] = t_2$ for every tuple t_2 in S. See Figure 6.8(b) (Figure 7.15(b) on e3).
- Example for DIVISION operation: "Retrieve the names of employees who work on all the projects that 'John Smith' works on.

  JSMITH_SSN(ESSN) $\longleftarrow$ π_{SSN} ($\sigma_{FNAME='John'\ AND\ LNAME='Smith'}$ (EMPLOYEE))
  JSMITH_PROJ $\longleftarrow$ π_{PNO} (JSMITH_SSN $*$ WORKS_ON)
  WORKS_ON2 $\longleftarrow$ $\pi_{ESSN,PNO}$ (WORKS_ON)
  DIV_HERE(SSN) $\longleftarrow$ WORKS_ON2 $\div$ JSMITH_PROJ
  RESULT $\longleftarrow$ $\pi_{FNAME,\ LNAME}$ (EMPLOYEE $*$ DIV_HERE)
  See figure 6.8(a) (Figure7.15(a) on e3).

6.4 Additional Relational Operations

6.4.1 Aggregate Functions and Grouping

- Apply aggregate functions **SUM, AVERAGE, MAXIMUM, MINIMUM** and **COUNT** of an attribute to different groups of tuples.

- $R_2 \longleftarrow {}_{<grouping\ attribs>}\Im_{<<func\ attrib>,<func\ attrib>,......>}(R_1)$

  - The resulting relation R_2 has the **grouping attributes + one attribute for each element in the function list**.

  - Each group results in a tuple in R_2.

  - For example,

    * ${}_{DNO}\ \Im_{COUNT\ SSN,\ AVERAGE\ SALARY}(EMPLOYEE)$

DNO	COUNT_SSN	AVERAGE_SALARY
5	4	33250
4	3	31000
1	1	55000

    * $\Im_{COUNT\ SSN,\ AVERAGE\ SALARY}(EMPLOYEE)$

COUNT_SSN	AVERAGE_SALARY
8	35125

6.4.3 OUTER JOIN

- **LEFT OUTER JOIN:** $R_3(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m) \longleftarrow R_1(A_1, A_2, \ldots, A_n)$ $⟕_{<JOIN\ COND.>}$ $R_2(B_1, B_2, \ldots, B_m)$

  - This operation keeps every tuple t in left relation R_1 in R_3, and fills "NULL" for attributes $B_1, B_2, \ldots, B_m$ if the join condition is not satisfied for t.

  - For example,
    $TEMP \longleftarrow (EMPLOYEE ⟕_{SSN=MGRSSN} DEPARTMENT)$
    $RESULT \longleftarrow \pi_{FNAME,\ MINIT,\ DNAME}\ (TEMP)$
    The result is in Figure 6.12 (Figure 7.18 on e3)

- **RIGHT OUTER JOIN:** similar to LEFT OUTER JOIN, but keeps every tuple t in right relation R_2 in the resulting relation R_3.

  - Notation: ⟖

- **FULL OUTER JOIN:** $⟗$

6.4.4 The OUTER UNION Operation

- **OUTER UNION:** make union of two relations that are partially compatible.
  - $R_3(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m, C_1, C_2, \ldots, C_p) \longleftarrow R_1(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ OUTER UNION $R_2(A_1, A_2, \ldots, A_n, C_1, C_2, \ldots, C_p)$
  - The list of compatible attributes are represented only once in R_3.
  - Tuples from R_1 and R_2 with the same values on the set of compitable attributes are represented only once in R_3
  - In R_3, fill "NULL" if necessary
  - For example, STUDENT(NAME, SSN, DEPT, ADVISOR) and FACULTY(NAME, SSN, DEPT, RANK)

    The resulting relation schema after OUTER UNION will be R_3(NAME, SSN, DEPT, ADVISOR, RANK)

6.5 Examples of Queries in Relational Algebra

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

  $RESEARCH\_DEPT \longleftarrow \sigma_{DNAME='Research'}(DEPARTMENT)$

  $RESEARCH\_EMPS \longleftarrow (RESEARCH\_DEPT \bowtie_{DNUMBER=DNO} (EMPLOYEE))$

  $RESULT \longleftarrow \pi_{FNAME,LNAME,ADDRESS}(RESEARCH\_EMPS)$

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

  $STAFFORD\_PROJS \longleftarrow \sigma_{PLOCATION='Stafford'}(PROJECT)$

  $CONTR\_DEPT \longleftarrow (STAFFORD\_PROJS \bowtie_{DNUM=DNUMBER} (DEPARTMENT))$

  $PROJ\_DEPT\_MGR \longleftarrow (CONTR\_DEPT \bowtie_{MGRSSN=SSN} (EMPLOYEE))$

  $RESULT \longleftarrow \pi_{PNUMBET,DNUM,LNAME,ADDRESS,BDATE}(PROJ\_DEPT\_MGR)$

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

  $DEPT5\_PROJS(PNO) \longleftarrow \pi_{PNUMBER}(\sigma_{DNUM=5}(PROJECT))$

  $EMP\_PROJ(SSN, PNO) \longleftarrow \pi_{ESSN,PNO}(WORKS\_ON)$

  $RESULT\_EMP\_SSNS \longleftarrow EMP\_PROJ \div DEPT\_PROJS$

  $RESULT \longleftarrow \pi_{LNAME,FNAME}(RESULT\_EMP\_SSNS * EMPLOYEE)$

- 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.

  $SMITHS(ESSN) \longleftarrow \pi_{SSN}(\sigma_{LNAME='Smith'}(EMPLOYEE))$

  $SMITH\_WORKER\_PROJ \longleftarrow \pi_{PNO}(WORKS\_ON * SMITHS)$

  $MGRS \longleftarrow \pi_{LNAME,DNUMBER}(EMPLOYEE \bowtie_{SSN=MGRSSN} DEPARTMENT)$

  $SMITH\_MANAGED\_DEPTS(DNUM) \longleftarrow \pi_{DNUMBER}(\sigma_{LNAME='Smith'}(MGRS))$

  $SMITH\_MGR\_PROJS(PNO) \longleftarrow \pi_{PNUMBER}(SMITH\_MANAGED\_DEPTS * PROJECT)$

  $RESULT \longleftarrow (SMITH\_WORKER\_PROJS \cup SMITH\_MGR\_PROJS)$

- List the names of all employees who have two or more dependents.

  $T_1(SSN, NO\_OF\_DEPTS) \longleftarrow {}_{ESSN}\Im_{COUNT\ DEPENDENT\_NAME}(DEPENDENT)$

  $T_2 \longleftarrow \sigma_{NO\_OF\_DEPTS \geq 2}(T_1)$

  $RESULT \longleftarrow \pi_{LNAME,FNAME}(T_2 * EMPLOYEE)$

- Retrieve the names of employees who have no dependents.

  $ALL\_EMPS \longleftarrow \pi_{SSN}(EMPLOYEE)$

  $EMPS\_WITH\_DEPS(SSN) \longleftarrow \pi_{ESSN}(DEPENDENT)$

  $EMPS\_WITHOUT\_DEPS \longleftarrow (ALL\_EMPS - EMPS\_WITH\_DEPS)$

  $RESULT \longleftarrow \pi_{LNAME,FNAME}(EMPS\_WITHOUT\_DEPS * EMPLOYEE)$

- List the names of managers who have at least one dependent.

  $MGRS(SSN) \longleftarrow \pi_{MGRSSN}(DEPARTMENT)$

  $EMPS\_WITH\_DEPS(SSN) \longleftarrow \pi_{ESSN}(DEPENDENT)$

  $MGRS\_WITH\_DEPS \longleftarrow (MGRS \cap EMPS\_WITH\_DEPS)$

  $RESULT \longleftarrow \pi_{LNAME,FNAME}(MGRS\_WITH\_DEPS * EMPLOYEE)$

- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

  $EMPS\_DEPS \longleftarrow$

  $(EMPLOYEE \bowtie_{SSN=ESSN\ AND\ SEX=SEX\ AND\ FNAME=DEPENDENT\_NAME}\ DEPENDENT)$

  $RESULT \longleftarrow \pi_{LNAME,FNAME}(EMPS\_DEPS)$

- Retrieve the names of all employees who do not have supervisors.

  $RESULT \longleftarrow \pi_{LNAME,FNAME}(\sigma_{SUPERSSN=NULL}(EMPLOYEE))$

- Find the sum of salary of all employees, the maximum salary, the minimum salary, and the average salary for each department.

  $RESULT \longleftarrow$

  ${}_{DNO}\Im_{SUM\ SALARY,\ MAXIMUM\ SALARY,\ MINIMUM\ SALARY,\ AVERAGE\ SALARY}(EMPLOYEE)$

6.6 The Tuple Relational Calculus

- **Nonprocedural** Language: Specify what to do; Tuple (Relational) Calculus, Domain (Relational) Calculus.
- **Procedural** Language: Specify how to do; Relational Algebra.
- The expressive power of Relational Calculus and Relational Algebra is identical.
- A relational query language L is considered **relationally complete** if we can express in L any query that can be expressed in Relational Calculus.
- Most relational query language is relationally complete but have more expressive power than relational calculus (algebra) because of additional operations such as aggregate functions, grouping, and ordering.

6.6.1 Tuple Variables and Range Relations

- A **tuple variable** usually **range over** a particular database relation: the variable may take as its value any individual tuple from that relation.
- General Form: $\{t \mid COND(t)\}$
- Examples:
  - Find all employees whose salary is above $50,000.
    $\{t \mid EMPLOYEE(t) \textit{ and } t.SALARY > 50000\}$
  - Find the first and last names of all employees whose salary is above $50,000.
    $\{t.FNAME, t.LNAME \mid EMPLOYEE(t) \textit{ and } t.SALARY > 50000\}$
    Compare to:

```
SELECT T.FNAME, T.LNAME
FROM   EMPLOYEE AS T
WHERE  T.SALARY > 50000;
```

- Three information should be specified in a tuple calculus expression.

- For each tuple variable t, the **range relation** R of t is specified as R(t). (FROM clause in SQL)
- A condition to select particular combinations of tuples. (WHERE clause in SQL)
- A set of attributes to be retrieved, the **requested attributes**. (SELECT clause in SQL)

- Example: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.
$\{t.BDATE, t.ADDRESS \mid EMPLOYEE(t)\ and\ t.FNAME = 'John'\ and\ t.MINIT = 'B'\ and\ t.LNAME = 'Smith'\}$

6.6.2 Expressions and Formulas in Tuple Relation Calculus

- A general expression form:
$\{t_1.A_1, t_2.A_2, \ldots, t_n.A_n \mid COND(t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m})\}$
Where $t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m}$ are tuple variables, each A_i is an attribute of the relation on which t_i ranges, and COND is a **condition** or **formula**

- A **formula** is made up of one or more atoms, which can be one of the following.
  - An atom of the form $R(t_i)$ defines the range of the tuple variable t_i as the relation R.
    If the tuple variable t_i is assigned a tuple that is a member of R, then the atom is TRUE.
  - An atom of the form $t_i.A$ **op** $t_j.B$, where **op** is one of the comparison operators $\{=, >, \geq, <, \leq, \neq\}$.
    If the tuple variables t_i and t_j are assigned to tuples such that the values of the attributes $t_i.A$ and $t_j.B$ satisfy the condition, then the atom is TRUE.
  - An atom of the form $t_i.A$ **op** c or c **op** $t_j.B$.
    If the tuple variables t_i (or t_j) is assigned to a tuple such that the value of the attribute $t_i.A$ (or $t_j.B$) satisfies the condition, then the atom is TRUE.

- A **formula** is made up one or more atoms connected via the logical operators **and, or, not** and is defined recursively as follows.
  - Every atom is a formula.
  - If F_1 and F_2 are formulas, then so are $(F_1 \text{ and } F_2)$, $(F_1 \text{ or } F_2)$, $not(F_1)$, $not(F_2)$. And
    - $(F_1 \text{ and } F_2)$ is TRUE if both F_1 and F_2 are TRUE; otherwise, it is FALSE.
    - $(F_1 \text{ or } F_2)$ is FALSE if both F_1 and F_2 are FALSE; otherwise, it is TRUE.
    - $not(F_1)$ is TRUE if F_1 is FALSE; it is FALSE if F_1 is TRUE.
    - $not(F_2)$ is TRUE if F_2 is FALSE; it is FALSE if F_2 is TRUE.

6.6.3 The Existential and Universal Quantifiers

- There are two **quantifiers** can appear in formula, **universal quantifier** $\forall$ and **existential quantifier** $\exists$.
- **free** and **bound** for tuple variables in formula.
  - An occurrence of a tuple variable in a formula F that is an atom is free in F.
  - An occurrence of a tuple variable t is free or bound in $(F_1 \text{ and } F_2)$, $(F_1 \text{ or } F_2)$, $not(F_1)$, $not(F_2)$, depending on whether it is free or bound in F_1 or F_2. Notice that a tuple variable may be free in F_1 and bound in F_2.
  - All free occurrences of a tuple variable t in F are bound in formulas $F' = (\exists\, t)(F)$ or $F' = (\forall\, t)(F)$. For example:

    $F_1 : d.DNAME =' Research'$

    $F_2 : (\exists\, t)(d.DNUMBER = t.DNO)$

    $F_3 : (\forall\, d)(d.MGRSSN =' 333445555')$

    Where variable d is free in F_1 and F_2, but bound in F_3. Variable t is bound in F_2.
- A formula with **quantifiers** is defined as follows.
  - If F is a formula, then so is $(\exists\, t)(F)$, where t is a tuple variable. The formula $(\exists\, t)(F)$ is TRUE if the formula F evaluates to TRUE for some tuple assigned to free occurrences of t in F; otherwise $(\exists\, t)(F)$ is FALSE.

– If F is a formula, then so is $(\forall t)(F)$, where t is a tuple variable. The formula $(\forall t)(F)$ is TRUE if the formula F evaluates to TRUE for every tuple (in the universe) assigned to free occurrences of t in F; otherwise $(\forall t)(F)$ is FALSE.

6.6.4 Example Queries Using the Existential Quantifier

- Retrieve the name and address of all employees who work for the 'Research' department.
  $\{t.FNAME, t.LNAME, t.ADDRESS \mid EMPLOYEE(t)\ and\ (\exists\, d)$
  $(DEPARTMENT(d)\ and\ d.DNAME ='\ Research'\ and\ d.DNUMBER = t.DNO)\}$

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, birthdate, and address.
  $\{p.PNUMBER, p.DNUM, m.LNAME, m.BDATE, m.ADDRESS \mid PROJECT(p)$
  $and\ EMPLOYEE(m)\ and\ p.PLOCATION ='\ Stafford'\ and$
  $((\exists\, d)(DEPARTMENT(d)\ and\ p.DNUM = d.DNUMBER\ and\ d.MGRSSN =$
  $m.SSN))\}$

- For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.
  $\{e.FNAME, e.LNAME, s.FNAME, s.LNAME \mid EMPLOYEE(e)\ and$
  $EMPLOYEE(s)\ and\ e.SUPERSSN = s.SSN\}$

- Find the name of each employee who works on some project controlled by department number 5.
  $\{e.LNAME, e.FNAME \mid EMPLOYEE(e)\ and\ ((\exists\, x)(\exists\, w)$
  $(PROJECT(x)\ and\ WORKS\_ON(w)\ and\ x.DNUM = 5\ and\ w.ESSN = e.SSN\ and$
  $x.PNUMBER = w.PNO))\}$

- 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 controlling department for the project.
  $\{p.PNUMBER \mid PROJECT(p)\ and$

$(((\exists\, e)(\exists\, w)(EMPLOYEE(e)\ and\ WORKS\_ON(w)\ and$

$w.PNO = p.PNUMBER\ and\ e.LNAME =' Smith'\ and\ e.SSN = w.ESSN))$

or

$((\exists\, m)(\exists\, d)(EMPLOYEE(m)\ and\ DEPARTMENT(d)\ and$

$p.DNUM = d.DNUMBER\, and\, d.MGRSSN = m.SSN\, and\, m.LNAME =' Smith')))\}$

6.6.5 Transforming the Universal and Existential Quantifiers

- $(\forall\, x)(P(x)) \equiv not(\exists\, x)(not(P(x)))$
- $(\exists\, x)(P(x)) \equiv not(\forall\, x)(not(P(x)))$
- $(\forall\, x)(P(x)\ and\ Q(x)) \equiv not(\exists\, x)(not(P(x))\ or\ not(Q(x)))$
- $(\forall\, x)(P(x)\ or\ Q(x)) \equiv not(\exists\, x)(not(P(x))\ and\ not(Q(x)))$
- $(\exists\, x)(P(x)\ or\ Q(x)) \equiv not(\forall\, x)(not(P(x))\ and\ not(Q(x)))$
- $(\exists\, x)(P(x)\ and\ Q(x)) \equiv not(\forall\, x)(not(P(x))\ or\ not(Q(x)))$
- $(\forall\, x)(P(x)) \Rightarrow (\exists\, x)(P(x))$
- $not(\exists\, x)(P(x)) \Rightarrow not(\forall\, x)(P(x))$

6.6.6 Using the Universal Quantifier

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

  $\{e.LNAME, e.FNAME \mid EMPLOYEE(e)\ and\ ((\forall\, x)(not(PROJECT(x)\ or$

  $not(x.DNUM = 5)\ or\ ((\exists\, w)(WORKS\_ON(w)\ and\ w.ESSN = e.SSN\ and$

  $x.PNUMBER = w.PNO))))\}$

  BREAK INTO:

  $\{e..LNAME, e.FNAME \mid EMPLOYEE(e)\ and\ F'\}$

  $F' = ((\forall\, x)(not(PROJECT(x))\ or\ F_1))$

$F_1 = not(x.DNUM = 5)\ or\ F_2$

$F_2 = ((\exists\, w)(WORKS\_ON(w)\ and\ w.ESSN = e.SSN\ and\ x.PNUMBER = w.PNO))$

IS EQUIVALENT TO:

$\{e.LNAME, e.FNAME \mid EMPLOYEE(e)\ and\ (not(\exists\, x)(PROJECT(x)$
$and\ (x.DNUM = 5)\ and$
$(not(\exists\, w)(WORKS\_ON(w)\ and\ w.ESSN = e.SSN\ and$
$x.PNUMBER = w.PNO))))\}$

- Find the names of employees who have no dependents.

  $\{e.FNAME, e.LNAME \mid EMPLOYEE(e)\ and\ (not(\exists\, d)(DEPENDENT(d)$
  $and\ e.SSN = d.ESSN))$

  IS EQUIVALENT TO:

  $\{e.FNAME, e.LNAME \mid EMPLOYEE(e)\ and\ ((\forall\, d)$
  $(not(DEPENDENT(d))\ or\ not(e.SSN = d.ESSN)))\}$

- List the names of managers who have at least one dependent.

  $\{e.FNAME, e.LNAME \mid EMPLOYEE(e)\ and\ ((\exists\, d)(\exists\, p)$
  $(DEPARTMENT(d)\ and\ DEPENDENT(p)\ and\ e.SSN = d.MGRSSN\ and$
  $p.ESSN = e.SSN))\}$

6.6.7 Safe Expressions

- **Safe Expression:** The result is a finite number of tuples.

- For example, $\{t \mid not(EMPLOYEE(t))\}$ is unsafe.

- **Domain of a tuple relational calculus expression:** The set of all values that either appear as constant values in the expression or exist in any tuple of the relations referenced in the expression.

- An expression is **safe** if all values in its result are from the domain of the expression.

6.7 The Domain Relational Calculus

- Rather than having variables range over tuples in relations, the **domain variables** range over single values from domains of attributes,

- General form: $\{x_1, x_2, \ldots, x_n \mid COND(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m})\}$

  Domain Variables: $x_1, x_2, \ldots, x_n$ that range over the domains of attributes.

  Formula: $COND$ is the formula or condition of the domain relational calculus.

  A formula is made up of **atoms**.

  - An atom of the form $R(x_1, x_2, \ldots, x_j)$ (or simply $R(x_1 x_2 \ldots x_j)$), where R is the name of a relation of degree j and each x_i, $1 \leq i \leq j$, is a domain variable.

    This atom defines that $< x_1, x_2, \ldots, x_j >$ must be a tuple in R, where the value of x_i is the value of the i^{th} attribute of the tuple.

    If the domain variables $x_1, x_2, \ldots, x_j$ are assigned values corresponding to a tuple of R, then the atom is TRUE.

  - An atom of the form x_i **op** x_j, where **op** is one of the comparison operators $\{=, >, \leq, <, \geq, \neq\}$.

    If the domain variables x_i and x_j are assigned values that satisfy the condition, then the atom is TRUE.

  - An atom of the form x_i **op** c or c **op** x_j, where c is a constant value.

    If the domain variables x_i (or x_j) is assigned a value that satisfies the condition, then the atom is TRUE.

- Examples: we use lowercase letters $l, m, n, \ldots, x, y, z$ for domain variables.

  - Retrieve the birthdate and address of the employee whose name is 'John B Smith'.

    $\{uv \mid (\exists\, q)(\exists\, r)(\exists\, s)(\exists\, t)(\exists\, w)(\exists\, x)(\exists\, y)(\exists\, z)$

    $(EMPLOYEE(qrstuvwxyz)\ and\ q =' John'\ and\ r =' B'\ and\ s =' Smith')\}$

    An alternative notation for this query.

    $\{uv \mid EMPLOYEE('John', 'B', 'Smith', t, u, v, w, x, y, z)\}$

    For convenience, we quantify only those variables actually appearing

in a condition (these would be q, r and s in the above example) in the rest of examples

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

  $\{qsv \mid (\exists\, z)(\exists\, l)(\exists\, m)(EMPLOYEE(qrstuvwxyz)\ and$
  $DEPARTMENT(lmno)\ and\ l =' Research'\ and\ m = z)\}$

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

  $\{iksuv \mid (\exists\, j)(\exists\, m)(\exists\, n)(\exists\, t)(PROJECT(hijk)\ and\ EMPLOYEE(qrstuvwxyz)$
  $and\ DEPARTMENT(lmno)\ and\ k = m\ and\ n = t\ and\ j =' Stafford')\}$

- Find the names of employees who have no dependents.

  $\{qs \mid (\exists\, t)(EMPLOYEE(qrstuvwxyz)\ and\ (not(\exists\, l)(DEPENDENT(lmnop)$
  $and\ t = l)))\}$

  IS EQUIVALENT TO:

  $\{qs \mid (\exists\, t)(EMPLOYEE(qrstuvwxyz)\ and\ ((\forall\, l)(not(DEPENDENT(lmnop))$
  $or\ not(t = l))))\}$

- List the names of managers who have at least one dependent.

  $\{sq \mid (\exists\, t)(\exists\, j)(\exists\, l)(EMPLOYEE(qrstuvwxyz)\ and\ DEPARTMENT(hijk)$
  $and\ DEPENDENT(lmnop)\ and\ t = j\ and\ l = t)\}$