The Relational Algebra and Relational Calculus (cont.)

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

Examples of Queries in Relational Algebra

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

$$\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{Dname}=\text{'Research'}}(\text{DEPARTMENT})$$

$$\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{Dnumber}=\text{Dno}} \text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{RESEARCH\_EMPS})$$

Single in-line expression

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

Examples of Queries in Relational Algebra (cont.)

- **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.

$$\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{Plocation}=\text{‘Stafford’}}(\text{PROJECT})$$

$$\text{CONTR\_DEPTS} \leftarrow (\text{STAFFORD\_PROJS} \bowtie_{\text{Dnum}=\text{Dnumber}} \text{DEPARTMENT})$$

$$\text{PROJ\_DEPT\_MGRS} \leftarrow (\text{CONTR\_DEPTS} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ\_DEPT\_MGRS})$$

Examples of Queries in Relational Algebra (cont.)

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

$$DEPT5\_PROJS \leftarrow \rho_{(Pno)}(\pi_{Pnumber}(\sigma_{Dnum=5}(PROJECT)))$$

$$EMP\_PROJ \leftarrow \rho_{(Ssn,\ Pno)}(\pi_{Essn,\ Pno}(WORKS\_ON))$$

$$RESULT\_EMP\_SSNS \leftarrow EMP\_PROJ \div DEPT5\_PROJS$$

$$RESULT \leftarrow \pi_{Lname,\ Fname}(RESULT\_EMP\_SSNS * EMPLOYEE)$$

Examples of Queries in Relational Algebra (cont.)

- **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.

$$\text{SMITHS(Essn)} \leftarrow \pi_{\text{Ssn}} (\sigma_{\text{Lname='Smith'}}(\text{EMPLOYEE}))$$

$$\text{SMITH\_WORKER\_PROJS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} * \text{SMITHS})$$

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

$$\text{SMITH\_MANAGED\_DEPTS(Dnum)} \leftarrow \pi_{\text{Dnumber}} (\sigma_{\text{Lname='Smith'}}(\text{MGRS}))$$

$$\text{SMITH\_MGR\_PROJS(Pno)} \leftarrow \pi_{\text{Pnumber}}(\text{SMITH\_MANAGED\_DEPTS} * \text{PROJECT})$$

$$\text{RESULT} \leftarrow (\text{SMITH\_WORKER\_PROJS} \cup \text{SMITH\_MGR\_PROJS})$$

Examples of Queries in Relational Algebra (cont.)

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

$$T1(\text{Ssn, No\_of\_dependents}) \leftarrow {}_{\text{Essn}}\ \Im\ _{\text{COUNT Dependent\_name}}(\text{DEPENDENT})$$

$$T2 \leftarrow \sigma_{\text{No\_of\_dependents}>2}(T1)$$

$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(T2 * \text{EMPLOYEE})$$

Assume that dependents of the same employee have distinct Dependent_name values

Examples of Queries in Relational Algebra (cont.)

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

$$\text{ALL\_EMPS} \leftarrow \pi_{\text{Ssn}}(\text{EMPLOYEE})$$

$$\text{EMPS\_WITH\_DEPS(Ssn)} \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$$

$$\text{EMPS\_WITHOUT\_DEPS} \leftarrow (\text{ALL\_EMPS} - \text{EMPS\_WITH\_DEPS})$$

$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{EMPS\_WITHOUT\_DEPS} * \text{EMPLOYEE})$$

Examples of Queries in Relational Algebra (cont.)

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

$$\text{MGRS(Ssn)} \leftarrow \pi_{\text{Mgr\_ssn}}(\text{DEPARTMENT})$$

$$\text{EMPS\_WITH\_DEPS(Ssn)} \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$$

$$\text{MGRS\_WITH\_DEPS} \leftarrow (\text{MGRS} \cap \text{EMPS\_WITH\_DEPS})$$

$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{MGRS\_WITH\_DEPS} * \text{EMPLOYEE})$$

Today's Lecture

1. Relational Algebra
   - The basic set of operations for the formal relational model

2. Relational Calculus
   - Provides a high-level declarative language for specifying relational queries
   - Tuple relational calculus

Introduction to Relational Calculus

- **Relational calculus** is the *other* formal query language
- In relational calculus, a **query** (= selection of a set of tuples that satisfies **a certain condition**) is expressed using **logical conditions**

> ▪ **In contrast**, a query in **Relational Algebra** is expressed using ***operations*** **(e.g.,** `join`**,** `project`**, ...**

> ▪ **Relational Calculus** is a ***non-procedural*** **language**, **while**
>
> ▪ **Relational Algebra** (that we have learned in the previous chapter) is a ***procedural*** **language**

Tuple Variables and Range Relations (cont.)

- **Tuple relational calculus**
  - Based on specifying a number of **tuple variables**
  - Each tuple variable usually *ranges over* a particular database relation

- Range relation *R*(*t*) = the **relation** that is the **range** for a tuple variable *t*

  - ***R(t)* = true** if **tuple** t is a tuple from the **relation *R***
  - ***R(t)* = false** if **tuple** t is ***not*** a tuple from the **relation *R***

- Example:

```
Employee(t)
```

Tuple Variables and Range Relations (cont.)

- Example:
  - A simple tuple relational calculus query: **{*t* | COND(*t*)}**

All tuples *t* that evaluate COND(*t*) to TRUE

Tuple variable

Conditional (Boolean) expression

  - Find all employees whose salary is above $50,000,

{*t* | EMPLOYEE(*t*) **AND** *t*.Salary>50000}

Tuple Variables and Range Relations (cont.)

- Retrieve some of the attributes—say, the first and last names:
  - {*t*.Fname, *t*.Lname | EMPLOYEE(*t*) AND *t*.Salary>50000}

- Need to specify the following information in a tuple relational calculus expression:
  - For each tuple variable *t*, the **range relation** *R* of *t*.
  - A **condition** to select particular combinations of tuples.
  - A set of attributes to be retrieved, the **requested attributes**.

A Running Example

- **Query 0**:
  - Retrieve the birth date and address of the **employee** (or employees) whose name is John B. Smith.
  - **Q0**: {t.Bdate, t.Address | EMPLOYEE(*t*) AND *t*.Fname='John' AND *t*.Minit='B' AND *t*.Lname='Smith'}

Expressions and Formulas in Tuple Relational Calculus

- A **general expression** of the tuple relational calculus:
  - $\{t_1.A_j, t_2.A_k, \ldots, t_n.A_m \mid \mathrm{COND}(t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m})\}$

- A condition or formula can be one of the following:
  - $R(t_i)$: indicate the range of the variable t_i as the relation R
  - $t_i.A$ **op** $t_j.B$: where **op** is one of $\{=, <, \leq, >, \geq, \neq\}$
  - $t_i.A$ **p** c *or* c **op** $t_j.B$

- Be connected by **AND**, **OR**, and **NOT**

The Existential and Universal Quantifiers

- Quantifiers can appear in formulas
  - Universal quantifier (∀)
    - (∀*t*)(COND(*t*)) is TRUE if all (in the universe) tuples *t* satisfies the condition COND(*t*).
  - Existential quantifier (∃)
    - (∃*t*)(COND(*t*)) is TRUE if *there exists* some (at least one) tuple *t* satisfies the condition COND(*t*)

- Example:
  - (∃*t*) (Employee(*t*) and *t*.fname='John' and *t*.lname='Smith’)
  - (∃d) (Department(d) and d.dname='Research')
  - (∀*t*) (Employee(*t*) and *t*.salary > 40000)

A Running Example

- Example: Find the **department number** of the **'Research' department**
  - Solution 1: without using the existential quantifier
    - { *d*.dno | Department(*d*) and *d*.dname='Research' }
  - Solution 2: with the existential quantifier ∃

{ d.dno | Department(d)
AND (
(∃ t)
(Department(t)
AND t.dname = 'Research'
AND t.dno = d.dno
)
)
}

is **true** for the tuple *t* belonging to the Research department

is **true** only when: *t*.dno = *d*.dno and t is the tuple of the Research department

In other words: *d*.dno is the department number of the Research department

Sample Query 1

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

```
{ e.Fname, e.Lname, e.Address  |

          Employee(e)            // e is an employee
    AND
          (∃d)
          (
               Department(d)               // d is a department tuple
            AND  d.Dname = 'Research’      // d is the Research department
            AND  d.Dnumber = e.Dno               // e is an employee in R.dept
           )
}
```

Sample Query 1 (cont.)

- Execution

```
for ( e := every tuple in the database ) do
    for ( d := every tuple in the database ) do
    {
        if ( Employee(e) AND
                (∃d)
                (
                        Department(d)              // d is a department tuple
                    AND   d.dname = 'Research'     // d is the Research department
                    AND   d.dno = e.dno            // e is an employee in R. dept
                )
           )
        {
            output e.fname, e.lname;
        }
    }
```

Sample Query 2

- **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.

{ *p*.Pnumber, *p*.Dnum, *m*.Lname, *m*.Bdate, *m*.Address |

Project(*p*) // *p* is a project

AND

Employee (*e*) // *e* is an employee

AND

p.Plocation = ‘Stafford’

AND

(∃d)

(

Department(*d*) // *d* is a department tuple

AND *p*.Dnum = *d*.Dnumber // *p* is controlled by *d*

AND *d*.Mgr_ssn = *m*.Ssn // Join relation Department and Employee

)

}

Sample Query 3

- **Query 3**
  - For each employee, retrieve the **employee**'s first and last name and the first and last name of his or her immediate **supervisor**
  - **Q3**: {e.Fname, e.Lname, s.Fname, s.Lname | EMPLOYEE(e) **AND** EMPLOYEE(s) **AND** e.Super_ssn=s.Ssn}

tuple variables in a query can range over the same relation

Sample Query 4

- **Query 4**
  - List the name of **employees** who works on *some* project controlled by department number 5.

```
{ e.Fname, e.Lname    |

            Employee(e)                         // e is an employee
     AND
            (   (∃p) (∃w)
                (
                        Project(p)              // p is a Project tuple
                  AND   Works_on(w)             // w is a Works_on tuple

                  AND    p.dnum=5                       // p is controlled by dept 5
                  AND    p.pnumber=w.pnum               // join Project and Work_on
                  AND    w.essn=e.ssn                   // e works on the project
               )
            )
}
```

- **Query 5**
  - 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.

```
{ p.Pnumber   |
            Project(p)                  // p is a project
    AND
            (  (∃e) (∃w)
                (
                        Employee (e)                // e is an employee
                  AND   Works_on(w)                 // w is a Works_on tuple

                  AND    w.Pno=p.Pnumber            // join Project and Work_on
                  AND    e.Lname=‘Smith’            // an employee whose last name is ‘Smith’
                  AND    e.Ssn=w.Essn               // Join Employee and Wok_on
                 )
             )
    OR
            (  (∃m) (∃d)
                (
                              Employee (m)                  //m is an employee
                  AND   Department(d)                 // d is a department

                  AND    p.Dnum=d.Dnumber
                  AND    d.Mgr_ssn=m.Ssn
                  AND    m.Lname=‘Smith’
                )
            )
}
```

Expressive Power of Relational Algebra and Relational Calculus

- **Expressive power** of query language
  - The set of all queries that can be **written** using that **query language**.

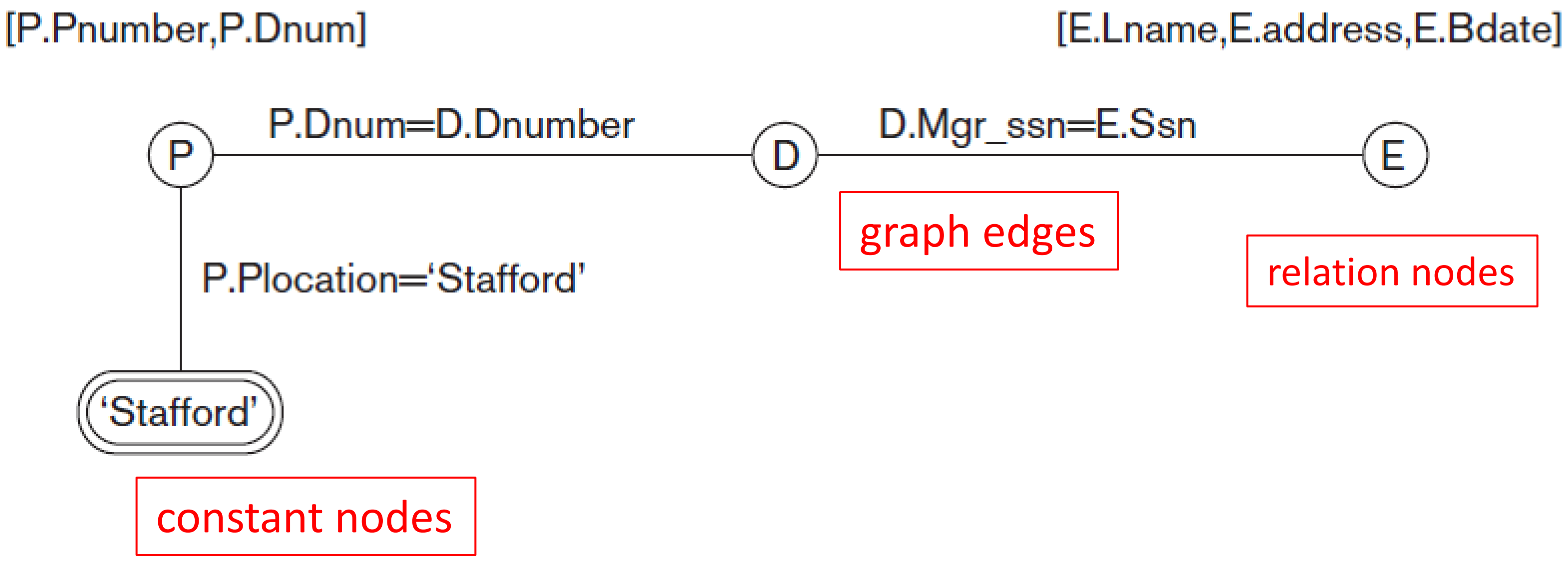
- Comparing different query language
  - A query language *A* is more expressive than a query language *B* if:
    - The **set of all queries** than can be written in A is a **superset** of the **set of all queries** than can be written in B

- Comparing Relational Algebra and Relational Calculus:
  - Relational Algebra and Relational Calculus are equally expressive

Notation for Query Graphs

- Graphical representation for **select-project-join** queries
- Query graph for Q2:

```
{ p.Pnumber, p.Dnum, m.Lname, m.Bdate, m.Address  |
          Project(p)          // p is a project
    AND
          Employee (e)        // e is an employee
    AND
          p.Plocation = 'Stafford'
    AND
          (∃d)
          (
                   Department(d)
               AND  p.Dnum = d.Dnumber
               AND  d.Mgr_ssn = m.Ssn
          )
 }
```



Transforming the Universal and Existential Quantifiers

- Transform a universal quantifier into an existential quantifier, and vice versa

$$(\forall x)\ (P(x)) \equiv \textbf{NOT}\ (\exists x)\ (\textbf{NOT}\ (P(x)))$$

$$(\exists x)\ (P(x)) \equiv \textbf{NOT}\ (\forall x)\ (\textbf{NOT}\ (P(x)))$$

$$(\forall x)\ (P(x)\ \textbf{AND}\ Q(x)) \equiv \textbf{NOT}\ (\exists x)\ (\textbf{NOT}\ (P(x))\ \textbf{OR NOT}\ (Q(x)))$$

$$(\forall x)\ (P(x)\ \textbf{OR}\ Q(x)) \equiv \textbf{NOT}\ (\exists x)\ (\textbf{NOT}\ (P(x))\ \textbf{AND NOT}\ (Q(x)))$$

$$(\exists x)\ (P(x))\ \textbf{OR}\ Q(x)) \equiv \textbf{NOT}\ (\forall x)\ (\textbf{NOT}\ (P(x))\ \textbf{AND NOT}\ (Q(x)))$$

$$(\exists x)\ (P(x)\ \textbf{AND}\ Q(x)) \equiv \textbf{NOT}\ (\forall x)\ (\textbf{NOT}\ (P(x))\ \textbf{OR NOT}\ (Q(x)))$$

≡ symbol stands for **equivalent to**

Transforming the Universal and Existential Quantifiers (cont.)

- Notice also that the following is **TRUE**, where the ⇒ symbol stands for implies:

$$(\forall x)(P(x)) \Rightarrow (\exists x)(P(x))$$

$$\textbf{NOT } (\exists x)(P(x)) \Rightarrow \textbf{NOT } (\forall x)(P(x))$$

Using the Universal Quantifier in Queries

- Need to follow a few rules to ensure that our expression makes sense
  - List the names of employees who work on all the projects controlled by department number 5.

$$
\begin{aligned}
\textbf{Q3: } & \{e.\text{Lname}, e.\text{Fname} \mid \text{EMPLOYEE}(e) \textbf{ AND } F'\} \\
& F' = ((\forall x)(\textbf{NOT}(\text{PROJECT}(x)) \textbf{ OR } F_1)) \\
& F_1 = \textbf{NOT}(x.\text{Dnum}=5) \textbf{ OR } F_2 \\
& F_2 = ((\exists w)(\text{WORKS\_ON}(w) \textbf{ AND } w.\text{Essn}=e.\text{Ssn} \\
& \textbf{AND } x.\text{Pnumber}=w.\text{Pno}))
\end{aligned}
$$

Using the Universal Quantifier in Queries

- Need to follow a few rules to ensure that our expression makes sense
  - List the names of employees who work on all the projects controlled by department number 5.

$$
\begin{aligned}
\textbf{Q3: } & \{e.\text{Lname}, e.\text{Fname} \mid \text{EMPLOYEE}(e) \textbf{ AND } F'\} \\
& F' = ((\forall x)(\textbf{NOT}(\text{PROJECT}(x)) \textbf{ OR } F_1)) \\
& F_1 = \textbf{NOT}(x.\text{Dnum}=5) \textbf{ OR } F_2 \\
& F_2 = ((\exists w)(\text{WORKS\_ON}(w) \textbf{ AND } w.\text{Essn}=e.\text{Ssn} \\
& \textbf{AND } x.\text{Pnumber}=w.\text{Pno}))
\end{aligned}
$$