Chapter 6 Outline

- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from Set Theory
- Binary Relational Operations: JOIN and DIVISION
- Additional Relational Operations
Chapter 6 Outline (cont’d.)

- Examples of Queries in Relational Algebra
- The Tuple Relational Calculus
- The Domain Relational Calculus
The Relational Algebra and Relational Calculus

- **Relational algebra**
  - Basic set of operations for the relational model

- **Relational algebra expression**
  - Sequence of relational algebra operations

- **Relational calculus**
  - Higher-level declarative language for specifying relational queries
Unary Relational Operations: SELECT and PROJECT

- The SELECT Operation
  - Subset of the tuples from a relation that satisfies a selection condition:
    \[ \sigma_{<\text{selection condition}>}(R) \]
  - Boolean expression contains clauses of the form
    <attribute name> <comparison op> <constant value>
  - or
  - <attribute name> <comparison op> <attribute name>
Unary Relational Operations: SELECT and PROJECT (cont’d.)

- Example:

\[ \sigma_{(Dno=4 \ \text{AND} \ Salary>25000) \ \text{OR} \ (Dno=5 \ \text{AND} \ Salary>30000)}(\text{EMPLOYEE}) \]

- <selection condition> applied independently to each individual tuple \( t \) in \( R \)
  - If condition evaluates to TRUE, tuple selected
- Boolean conditions \textbf{AND}, \textbf{OR}, and \textbf{NOT}
- Unary
  - Applied to a single relation
Unary Relational Operations: SELECT and PROJECT (cont’d.)

- **Selectivity**
  - Fraction of tuples selected by a selection condition

- **SELECT operation commutative**

- **Cascade** SELECT operations into a single operation with **AND** condition
The PROJECT Operation

- Selects columns from table and discards the other columns:
  \[ \pi_{<\text{attribute list}>}(R) \]

- Degree
  - Number of attributes in <attribute list>

- Duplicate elimination
  - Result of PROJECT operation is a set of distinct tuples
Sequences of Operations and the RENAME Operation

- **In-line expression:**
  \[\pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))\]

- **Sequence of operations:**
  \[\text{DEP5_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})\]
  \[\text{RESULT} \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\text{DEP5_EMPS})\]

- **Rename attributes in intermediate results**
  - RENAME operation
    \[\rho_{S(B_1, B_2, ..., B_n)}(R) \text{ or } \rho_{S}(R) \text{ or } \rho_{(B_1, B_2, ..., B_n)}(R)\]
Relational Algebra Operations from Set Theory

- **UNION, INTERSECTION, and MINUS**
  - Merge the elements of two sets in various ways
  - Binary operations
  - Relations must have the same type of tuples

- **UNION**
  - $R \cup S$
  - Includes all tuples that are either in $R$ or in $S$ or in both $R$ and $S$
  - Duplicate tuples eliminated
Relational Algebra Operations from Set Theory (cont’d.)

- **INTERSECTION**
  - $R \cap S$
  - Includes all tuples that are in both $R$ and $S$

- **SET DIFFERENCE (or MINUS)**
  - $R - S$
  - Includes all tuples that are in $R$ but not in $S$
The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- **CARTESIAN PRODUCT**
  - CROSS PRODUCT or CROSS JOIN
  - Denoted by $\times$
  - Binary set operation
  - Relations do not have to be union compatible
  - Useful when followed by a selection that matches values of attributes
Binary Relational Operations: JOIN and DIVISION

- **The JOIN Operation**
  - Denoted by $\Join$
  - Combine related tuples from two relations into single “longer” tuples
  - General join condition of the form $<\text{condition}> \text{ AND } <\text{condition}> \text{ AND} \ldots \text{ AND} <\text{condition}>$
  - Example:

    \[
    \text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \Join_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE} \\
    \text{RESULT} \leftarrow \pi_{\text{Dname, Lname, Fname}}(\text{DEPT\_MGR})
    \]
Binary Relational Operations: JOIN and DIVISION (cont’d.)

- **THETA JOIN**
  - Each <condition> of the form $A_i \theta B_j$
  - $A_i$ is an attribute of $R$
  - $B_j$ is an attribute of $S$
  - $A_i$ and $B_j$ have the same domain
  - $\theta$ (theta) is one of the comparison operators:
    - $\{=, <, \leq, >, \geq, \neq\}$
Variations of JOIN: The EQUIJOIN and NATURAL JOIN

- **EQUIJOIN**
  - Only = comparison operator used
  - Always have one or more pairs of attributes that have identical values in every tuple

- **NATURAL JOIN**
  - Denoted by *
  - Removes second (superfluous) attribute in an EQUIJOIN condition
Variations of JOIN: The EQUIJOIN and NATURAL JOIN (cont’d.)

- **Join selectivity**
  - Expected size of join result divided by the maximum size $n_R \times n_S$

- **Inner joins**
  - Type of match and combine operation
  - Defined formally as a combination of CARTESIAN PRODUCT and SELECTION
A Complete Set of Relational Algebra Operations

- Set of relational algebra operations \{\sigma, \pi, U, \rho, -, \times\} is a **complete set**
  - Any relational algebra operation can be expressed as a sequence of operations from this set
The DIVISION Operation

- Denoted by ÷
- Example: retrieve the names of employees who work on all the projects that ‘John Smith’ works on
- Apply to relations $R(Z) \div S(X)$
  - Attributes of $R$ are a subset of the attributes of $S$
# Operations of Relational Algebra

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>PURPOSE</th>
<th>NOTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>Selects all tuples that satisfy the selection condition from a relation $R$.</td>
<td>$\sigma_{\text{selection condition}}(R)$</td>
</tr>
<tr>
<td>PROJECT</td>
<td>Produces a new relation with only some of the attributes of $R$, and removes duplicate tuples.</td>
<td>$\pi_{\text{attribute list}}(R)$</td>
</tr>
<tr>
<td>THETA JOIN</td>
<td>Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.</td>
<td>$R_1 \bowtie_{\text{join condition}} R_2$</td>
</tr>
<tr>
<td>EQUIJOIN</td>
<td>Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.</td>
<td>$R_1 \bowtie_{\text{join condition}} R_2$, OR $R_1 \bowtie_{(&lt;\text{join attributes 1}&gt;), (\text{&lt;join attributes 2&gt;})} R_2$, OR $R_1 \bowtie_{(&lt;\text{join attributes 1}&gt;), (\text{&lt;join attributes 2&gt;})} R_2$</td>
</tr>
<tr>
<td>NATURAL JOIN</td>
<td>Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.</td>
<td>$R_1 \bowtie_{\text{join condition}} R_2$, OR $R_1 \bowtie_{(&lt;\text{join attributes 1}&gt;), (\text{&lt;join attributes 2&gt;})} R_2$, OR $R_1 \bowtie_{(&lt;\text{join attributes 1}&gt;), (\text{&lt;join attributes 2&gt;})} R_2$</td>
</tr>
</tbody>
</table>
# Operations of Relational Algebra (cont’d.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNION</td>
<td>Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$; $R_1$ and $R_2$ must be union compatible.</td>
<td>$R_1 \cup R_2$</td>
</tr>
<tr>
<td>INTERSECTION</td>
<td>Produces a relation that includes all the tuples in both $R_1$ and $R_2$; $R_1$ and $R_2$ must be union compatible.</td>
<td>$R_1 \cap R_2$</td>
</tr>
<tr>
<td>DIFFERENCE</td>
<td>Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$; $R_1$ and $R_2$ must be union compatible.</td>
<td>$R_1 - R_2$</td>
</tr>
<tr>
<td>CARTESIAN PRODUCT</td>
<td>Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$.</td>
<td>$R_1 \times R_2$</td>
</tr>
<tr>
<td>DIVISION</td>
<td>Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.</td>
<td>$R_1(Z) \div R_2(Y)$</td>
</tr>
</tbody>
</table>
Notation for Query Trees

- **Query tree**
  - Represents the input relations of query as leaf nodes of the tree
  - Represents the relational algebra operations as internal nodes
Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.
Additional Relational Operations

- **Generalized projection**
  - Allows functions of attributes to be included in the projection list
  \[ \pi_{F_1, F_2, ..., F_n}(R) \]

- **Aggregate functions and grouping**
  - Common functions applied to collections of numeric values
  - Include SUM, AVERAGE, MAXIMUM, and MINIMUM
Additional Relational Operations (cont’d.)

- Group tuples by the value of some of their attributes
  - Apply aggregate function independently to each group

\[ \langle \text{grouping attributes} \rangle \bigcap \langle \text{function list} \rangle (R) \]
Figure 6.10
The aggregate function operation.

a. \( \rho_{R(Dno, No\_of\_employees, Average\_sal)}(Dno \bowtie COUNT Ssn, AVERAGE Salary(EMPLOYEE)). \)

b. \( Dno \bowtie COUNT Ssn, AVERAGE Salary(EMPLOYEE). \)

c. \( \bowtie COUNT Ssn, AVERAGE Salary(EMPLOYEE). \)

| R |  
|---|---|---|
| Dno | No\_of\_employees | Average\_sal |
| 5 | 4 | 33250 |
| 4 | 3 | 31000 |
| 1 | 1 | 55000 |

| (b) |  
|---|---|---|
| Dno | Count\_ssn | Average\_salary |
| 5 | 4 | 33250 |
| 4 | 3 | 31000 |
| 1 | 1 | 55000 |

| (c) |  
|---|---|---|
| Count\_ssn | Average\_salary |
| 8 | 35125 |

\(^8^\)Note that this is an arbitrary notation we are suggesting. There is no standard notation.
Recursive Closure Operations

- **Operation applied to a recursive relationship** between tuples of same type

\[
\begin{align*}
\text{BORG}_{\text{SSN}} & \leftarrow \pi_{\text{SSn}}\left(\sigma_{\text{Fname}=\text{James}} \quad \text{AND} \quad \text{Lname}=\text{Borg}\left(\text{EMPLOYEE}\right)\right) \\
\text{SUPERVISION}(\text{Ssn}1, \text{Ssn}2) & \leftarrow \pi_{\text{Ssn}.\text{Super_ssn}}\left(\text{EMPLOYEE}\right) \\
\text{RESULT1}(\text{Ssn}) & \leftarrow \pi_{\text{Ssn}1}\left(\text{SUPERVISION} \times_{\text{Ssn}2=\text{Ssn}} \text{BORG}_{\text{SSN}}\right)
\end{align*}
\]
OUTER JOIN Operations

- **Outer joins**
  - Keep all tuples in $R$, or all those in $S$, or all those in both relations regardless of whether or not they have matching tuples in the other relation

- **Types**
  - LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN

- **Example:**
  
  $$\text{TEMP} \leftarrow \text{(EMPLOYEE ⋈}_{\text{Ssn}=\text{Mgr_ssn}} \text{DEPARTMENT)}$$

  $$\text{RESULT} \leftarrow \pi_{\text{Fname, Minit, Lname, Dname}}(\text{TEMP})$$
The OUTER UNION Operation

- Take union of tuples from two relations that have some common attributes
  - Not union (type) compatible
- Partially compatible
  - All tuples from both relations included in the result
  - Tuples with the same value combination will appear only once
Examples of Queries in Relational Algebra

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

```
RESEARCH_DEPT ← σ_{Dname=’Research’}(DEPARTMENT)
RESEARCH_EMPS ← (RESEARCH_DEPT △ Dnumber=Dno EMPLOYEE)
RESULT ← π_{Fname, Lname, Address}(RESEARCH_EMPS)
```

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

```
π_{Fname, Lname, Address}(σ_{Dname=’Research’}(DEPARTMENT △ Dnumber=Dno(EMPLOYEE))
```
Examples of Queries in Relational Algebra (cont’d.)

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.

\[
\begin{align*}
\text{STAFFORD_PROJS} & \leftarrow \sigma_{\text{Location} = 'Stafford'}(\text{PROJECT}) \\
\text{CONTR_DEPTS} & \leftarrow (\text{STAFFORD_PROJS} \bowtie_{\text{Dnum} = \text{Dnumber}} \text{DEPARTMENT}) \\
\text{PROJ_DEPT_MGRS} & \leftarrow (\text{CONTR_DEPTS} \bowtie_{\text{Mgr_ssn} = \text{Ssn}} \text{EMPLOYEE}) \\
\text{RESULT} & \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ_DEPT_MGRS})
\end{align*}
\]

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

\[
\begin{align*}
\text{DEPT5_PROJS} & \leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum} = 5}(\text{PROJECT}))) \\
\text{EMP_PROJ} & \leftarrow \rho_{(\text{Ssn}, \text{Pno})}(\pi_{\text{Essn}, \text{Pno}}(\text{WORKS_ON})) \\
\text{RESULT_EMP_SSNS} & \leftarrow \text{EMP_PROJ} \div \text{DEPT5_PROJS} \\
\text{RESULT} & \leftarrow \pi_{\text{Lname, Fname}}(\text{RESULT_EMP_SSNS} \times \text{EMPLOYEE})
\end{align*}
\]
Examples of Queries in Relational Algebra (cont’d.)

**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.

\[
\begin{align*}
\text{ALL\_EMPS} & \leftarrow \pi_{\text{Ssn}}(\text{EMPLOYEE}) \\
\text{EMPS\_WITH\_DEPS}(\text{Ssn}) & \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT}) \\
\text{EMPS\_WITHOUT\_DEPS} & \leftarrow (\text{ALL\_EMPS} \setminus \text{EMPS\_WITH\_DEPS}) \\
\text{RESULT} & \leftarrow \pi_{\text{Lname, Fname}}(\text{EMPS\_WITHOUT\_DEPS} \times \text{EMPLOYEE})
\end{align*}
\]

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

\[
\begin{align*}
\text{MGRS}(\text{Ssn}) & \leftarrow \pi_{\text{Mgr\_ssn}}(\text{DEPARTMENT}) \\
\text{EMPS\_WITH\_DEPS}(\text{Ssn}) & \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT}) \\
\text{MGRS\_WITH\_DEPS} & \leftarrow (\text{MGRS} \cap \text{EMPS\_WITH\_DEPS}) \\
\text{RESULT} & \leftarrow \pi_{\text{Lname, Fname}}(\text{MGRS\_WITH\_DEPS} \times \text{EMPLOYEE})
\end{align*}
\]
The Tuple Relational Calculus

- Declarative expression
  - Specify a retrieval request nonprocedural language
- Any retrieval that can be specified in basic relational algebra
  - Can also be specified in relational calculus
Tuple Variables and Range Relations

- **Tuple variables**
  - Ranges over a particular database relation

- **Satisfy** $\text{COND}(t): \{ t \mid \text{COND}(t) \}$

- **Specify**:
  - **Range relation** $R$ of $t$
  - Select particular combinations of tuples
  - Set of attributes to be retrieved (**requested attributes**)
Expressions and Formulas in Tuple Relational Calculus

- General expression of tuple relational calculus is of the form:

\[
\{t_1.A_j, t_2.A_k, \ldots, t_n.A_m \mid \text{COND}(t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m})\}
\]

- **Truth value** of an atom
  - Evaluates to either TRUE or FALSE for a specific combination of tuples

- **Formula** (Boolean condition)
  - Made up of one or more atoms connected via logical operators **AND, OR,** and **NOT**
Existential and Universal Quantifiers

- Universal quantifier ($\forall$)
- Existential quantifier ($\exists$)
- Define a tuple variable in a formula as free or bound
Sample Queries in Tuple Relational Calculus

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

\[ Q1: \{ t.\text{Fname}, t.\text{Lname}, t.\text{Address} \mid \text{EMPLOYEE}(t) \land (\exists d)(\text{DEPARTMENT}(d) \land d.\text{Dname}=\text{Research} \land d.\text{Dnumber}=t.\text{Dno}) \} \]

**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.

\[ Q4: \{ p.\text{Pnumber} \mid \text{PROJECT}(p) \land ((\exists e)(\exists w)(\text{EMPLOYEE}(e) \land \text{WORKS_ON}(w) \land w.\text{Pno}=p.\text{Pnumber} \land e.\text{Lname}=\text{Smith} \land e.\text{Ssn}=w.\text{Essn}) \lor ((\exists m)(\exists d)(\text{EMPLOYEE}(m) \land \text{DEPARTMENT}(d) \land p.\text{Dnum}=d.\text{Dnumber} \land d.\text{Mgr_ssn}=m.\text{Ssn} \land m.\text{Lname}=\text{Smith}))) \} \]
Q4A in SQL with Existential Quantifier

\{ p.PNUMBER | PROJECT(P) AND (Ee)(Ew) (EMPLOYEE(e) AND WORKS_ON(w) AND
w.Pno = p.Pnumber AND e.Lname = 'Smith' AND e.SSN = w.ESSN)
OR (Em)(Ed) (EMPLOYEE(m) AND DEPARTMENT(d) AND
p.Dnum = d.Dnumber AND m.Lname = 'Smith' AND d.Mgr_SSN = m.ESSE ) \}

SELECT P.Pnumber
FROM PROJECT P
WHERE EXISTS ( SELECT P.Pnumber
FROM PROJECT P,
DEPARTMENT D, EMPLOYEE E
WHERE D.Mgr_SSN = E.SSN AND
P.Dnum = D.Dnumber AND
E.Lname = 'Smith'
)

OR EXISTS ( SELECT W.Pno
FROM WORKS_ON W,
EMPLOYEE E
WHERE W.ESSE = E.SSN AND
E.Lname = 'Smith'
);
Q4B Same Query with UNION:

(SELECT Distinct Pnumber
 FROM PROJECT P,
 DEPARTMENT D, EMPLOYEE E
 WHERE P.Dnum = D.Dnumber AND
   D.Mgr_SSN = E.SSN AND
   E.Lname = 'Smith')

UNION

(SELECT Distinct Pnumber
 FROM WORKS_ON W,
 PROJECT P, EMPLOYEE E
 WHERE P.Pnumber = W.Pno AND
   W.Essn = E.Ssn AND
   E.Lname = 'Smith');
Expressing Query with Existantial Quantifier

Q:
Retrieve the name of employees who have no dependents.

Q in Relational calculus:
\{e.FNAME, e.LNAME \mid \text{EMPLOYEE (e) AND (Not(Ed)} (\text{DEPENDENT (d) AND e. SSN=d.ESSN}) ) \}

Q in SQL:
SELECT e.FNAME, e.LNAME
FROM \text{EMPLOYEE e}
WHERE \text{NOT EXISTS (SELECT *) FROM \text{EMPLOYEE d WHERE e. SSN=d.ESSN});}
Notation for Query Graphs

Figure 6.13
Query graph for Q2.

[P.Pnumber,P.Dnum]

P.Dnum=D.Dnumber

D.Mgr_ssn=E.Ssn

P.Plocation=‘Stafford’

‘Stafford’

[E.Lname,E.address,E.Bdate]
Transforming the Universal and Existential Quantifiers

- Transform one type of quantifier into other with negation (preceded by **NOT**)
  - **AND** and **OR** replace one another
  - Negated formula becomes unnegated
  - Unnegated formula becomes negated
Using the Universal Quantifier in Queries

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 \mid \text{EMPLOYEE}(e) \land ((\forall x)(\text{NOT}(\text{PROJECT}(x))) \lor \text{NOT}(x.Dnum=5) \lor ((\exists w)(\text{WORKS_ON}(w) \land w.Essn=e.Ssn \land x.Pnumber=w.Pno)))\}\n
Q3A: \{e.Lname, e.Fname \mid \text{EMPLOYEE}(e) \land (\text{NOT}(\exists x)(\text{PROJECT}(x)) \land (x.Dnum=5) \land (\text{NOT}(\exists w)(\text{WORKS_ON}(w) \land w.Essn=e.Ssn \land x.Pnumber=w.Pno)))\}\
Q3A: Expressing Universal Quantifier For All P(x) with NOT EXISTS (~P(x)) in SQL

```
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.ESN=SSN AND C.PNO = B.PNO));
```
Expressing Universal Quantifier For All $P(x)$ with NOT EXISTS ($\neg P(x)$)

{e.LNAME, e.FNAME | EMPLOYEE (e) AND
  ((\forall x) (\neg (\text{PROJECT}(x)) \text{ OR } \neg (x. \text{DNUM}=5))
  \text{ OR } ((\exists w) (\text{WORKS_ON (w) AND w.ESSN}=e.\text{SSN AND x.PNUMBER}=w.PNO))))

SELECT e.fname, e.lname
FROM employee e
WHERE NOT EXISTS ( (SELECT pnumber
                   FROM project p
                   WHERE p.dnum = 5)
                   \text{MINUS}
                   (SELECT w. pno
                    FROM works_on w
                    WHERE e.ssn = w.essn));
Safe Expressions

- Guaranteed to yield a finite number of tuples as its result
  - Otherwise expression is called **unsafe**
- Expression is **safe**
  - If all values in its result are from the domain of the expression
The Domain Relational Calculus

- Differs from tuple calculus in type of variables used in formulas
  - Variables range over single values from domains of attributes
- Formula is made up of **atoms**
  - Evaluate to either TRUE or FALSE for a specific set of values
    - Called the **truth values** of the atoms
The Domain Relational Calculus (cont’d.)

- QBE language
  - Based on domain relational calculus

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

\[
q_1: \{q, s, v \mid (\exists z) (\exists l) (\exists m) (\text{EMPLOYEE}(qrstuvwxyz) \land \text{DEPARTMENT}(lmno) \land l=’Research’ \land m=z)\}
\]

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.

\[
q_2: \{i, k, s, u, v \mid (\exists j)(\exists m)(\exists n)(\exists t) (\text{PROJECT}(hijk) \land \text{EMPLOYEE}(qrstuvwxyz) \land \text{DEPARTMENT}(lmno) \land k=m \land n=t \land j=’Stafford’)\}
\]
Summary

- Formal languages for relational model of data:
  - Relational algebra: operations, unary and binary operators
  - Some queries cannot be stated with basic relational algebra operations
    - But are important for practical use

- Relational calculus
  - Based predicate calculus