Course Notes on Relational Algebra

Relational Algebra: Summary

- Operators
  - Selection
  - Projection
  - Union, Intersection, Difference
  - Cartesian Product
  - Join
  - Division
- Equivalences
- Outer Join, Outer Union
- Transitive Closure

What is the Relational Algebra?

- Relational algebra =
  - a collection of operations each acting on one or two relations and producing one relation as result, and
  - a language for combining those operations
- The algebra has played a central role in the relational model: algebraic operations characterize high-level set-at-a-time access
- The algebra in practice
  - it was never a real user language (calculus-based languages and SQL are simpler)
  - its semantics is clear and a de facto standard
  - a precise syntax for the algebra is more complicated than its semantics

Relational Schema for the Company Example

Employee

SSN	FName	LName	BDate	Address	Sex	Salary	SuperSSN	DNo

Department

DNumber	DName	DMgr	MgrStartDate

DeptLocations

DNumber	DLocation

Project

PNumber	PName	PLocation	DNumber

WorksOn

PNo	ESSN	Hours

Dependent

ESSN	DependentName	Sex	BDate	Relationship

Company Example: Population of the Database (1)

Employee

FName	MInit	LName	SSN	BDate	Address	Sex	Salary	SuperSSN	DNo
John	B	Smith	123456789	09-Jan-55	...	M	30000	333445555	5
Franklin	T	Wong	333445555	08-Dec-45	...	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	...	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-Jul-31	...	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	...	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	...	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	29-Mar-59	...	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	...	M	55000	null	1

Department

DNumber	DName	MgrSSN	MgrStartDate
5	Research	333445555	22-May-78
4	Administration	987654321	01-Jan-85
1	Headquarters	888665555	19-Jun-71

DeptLocations

DNumber	DLocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Project

PNumber	PName	PLocation	DNum
1	ProductX	Bellaire	5
2	ProductY	Sugarland	5
3	ProductZ	Houston	5
10	Computerization	Stafford	4
20	Reorganization	Houston	1
30	Newbenefits	Stafford	4

4

Company Example: Population of the Database (2)

WorksOn

ESSN	PNo	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40
453453453	1	20
453453453	2	20
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
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

Dependent

ESSN	DependentName	Sex	BDate	Relationship
333445555	Alice	F	05-Apr-76	Daughter
333445555	Theodore	M	25-Oct-73	Son
333445555	Joy	F	03-May-48	Spouse
987654321	Abner	M	29-Feb-32	Spouse
123456789	Michael	M	01-Jan-78	Son
123456789	Alice	M	31-Dec-78	Daughter
123456789	Elizabeth	F	05-May-57	Spouse

5

Selection (or Restriction)

- Select tuples satisfying a condition

$$\sigma_{\text{condition}}(\text{R}) = \{r \in R \mid \text{condition}(\text{r})\}$$

- Example: $\sigma_{\text{DNo}=5}(\text{Employee})$

SSN	FName	MInit	LName	BDate	Address	Sex	Salary	SuperSSN	DNo
123456789	John	B	Smith	...	...	M	30000	333445555	5
333445555	Franklin	T	Wong	...	...	M	40000	888665555	5
666884444	Ramesh	K	Narayan	...	...	M	38000	333445555	5
453453453	Joyce	A	English	...	...	F	25000	333445555	5

- Intuition: selection is a “horizontal” slice of relation Employee
- Operational meaning: the condition is applied to every tuple; if it is satisfied, the tuple is kept in the answer

6

Possible Forms of Conditions in Selection

- Simple Condition $= \left\{ \begin{array}{l} \langle\text{attribute}\rangle\ \langle\text{comparison}\rangle\ \langle\text{attribute}\rangle \\ \langle\text{attribute}\rangle\ \langle\text{comparison}\rangle\ \langle\text{constant}\rangle \end{array} \right.$
- Comparisons: $=, \neq, <, \leq, >, \geq$
- Condition: combination of simple conditions with AND, OR, NOT
- Simple conditions are the most frequent

7

Projection

- Retain a subset of the attributes (columns) of a relation

$$\boxed{\pi_{\text{attributes}}(\text{relation})}$$

- Example: $\pi_{\text{LName,FName,Salary}}(\text{Employee})$

LName	FName	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

- Intuition: projection is a "vertical" slice of relation Employee

8

"Duplicate Removal"

- The result of a projection is a relation ⇒ projection involves duplicate removal
- Example: $\pi_{\text{DNo}}(\text{Employee})$

DNo
5
4
1

DNo
5
5
4
4
5
5
4
1

9

- **What is wrong with duplicate tuples?**
  - ◇ "If something is true, then saying it twice does not make it more true" (Codd) Even if repetition is a proven pedagogical technique :-)
  - ◇ Identical things are indistinguishable, there is no need to represent them twice
  - ◇ "Objects in the real world have only one thing in common: they are all different" (anonymous, A. Taivalsaari, JOOP, Nov. 1997)
  - ◇ Distinct things should have some value to distinguish them
  - ◇ It is difficult to manipulate "duplicate tuples" unless for counting, averaging, etc
  - ◇ The obvious modeling of identical objects is as a common description plus the number of copies of the object
- **Duplicate removal** (from a multiset to a set) can be expensive; it can be done with
  - ◇ nested loops
  - ◇ sort/merge
  - ◇ hashing

A Precise Definition of the Relational Algebra

- Algebraic operations:
  - ◇ operate on one or two relations and produce a relation as result
  - ◇ the result relation has no name
  - ◇ rules are needed to specify the attribute names in the result of algebraic operations with two operands

10

- For a precise syntax and semantics of the algebra, see *A Precise Definition of Basic Relational Notions and the Relational Algebra*, A. Pirotte, ACM SIGMOD Record, 13-1, 1982, pp. 30-45.

Combining Algebraic Operations

List the name and salary of employees in department 5

(1) Nested form: $\pi_{\text{FName,LName,Salary}}(\boxed{\sigma_{\text{DNo}=5}(\text{Employee})})$

(2) Sequential form: $\text{Temp} \leftarrow \sigma_{\text{DNo}=5}(\text{Employee})$

Temp

SSN	FName	MInit	LName	BDate	Address	Sex	Salary	SuperSSN	DNo
123456789	John	B	Smith	09-Jan-55	...	M	30000	333445555	5
333445555	Franklin	T	Wong	08-Dec-45	...	M	40000	888665555	5
666884444	Ramesh	K	Narayan	15-Sep-52	...	M	38000	333445555	5
453453453	Joyce	A	English	31-Jul-62	...	F	25000	333445555	5

$$\text{R}(\text{FirstName}, \text{LastName}, \text{Salary}) \leftarrow \pi_{\text{FName,LName,Salary}}(\text{Temp})$$

R

FirstName	LastName	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

11

Nesting or Sequencing Operations

- Several relational algebra operations may be needed to express a given request:
  - ◇ by nesting several algebraic operations within a single **relational algebra expression**
  - ◇ by applying operations one at a time in a **sequence of steps** and creating named intermediate relations by assignment operations
  - ◇ the correspondence between nested form and sequential form is immediate
- **Nesting and closure**
  - ◇ the result of an algebraic operation is a relation
  - ◇ algebraic operations can be nested like functions
  - ◇ closure is essential for the full power of the algebra

12

- **Nesting**
  - ◇ nesting = classical functional composition:
    - ∗ $Y_3 \rightarrow f_3(f_1(X_1, X_2), X_4, f_2(X_3))$, is equivalent to the sequence
      - · $Y_1 \rightarrow f_1(X_1, X_2)$
      - · $Y_2 \rightarrow f_2(X_3)$
      - · $Y_3 \rightarrow f_3(Y_1, X_4, Y_2)$
- **Closure**
  - ◇ closure = make nesting freely usable for combining operations, i.e., the result of every algebraic operation is a relation and it can be used as operand of another algebraic operation
  - ◇ closure can be violated by
    - ∗ the definition of language structure (SQL does)
    - ∗ operations whose result is not a relation (e.g., "relations" with duplicate tuples)
  - ◇ What is really needed is **"compositionality"**
    - ∗ the result of a query can be used as argument for another query
    - ∗ this is a version of what is called **orthogonality** (i.e., the generality of combining pieces of the definition of a language)

Set-Theoretical Operations

- Standard operations on sets are automatically applicable to relations
- **Union compatibility**
  - ◇ relations have to be defined on the same domains
  - ◇ **type compatibility** would be more adequate
  - ◇ more precise definition: two relations R and S are union-compatible if there is a one-to-one correspondence between attributes of R and attributes of S such that corresponding attributes are associated with the same domain
- A mechanism for defining the attribute names of the result is needed

13

- A relation is a set of tuples
- **Union**, **intersection**, and **difference** on union-compatible relations R and S have their usual meaning:

◇ $R \cup S$ = all tuples (without duplication) in R, in S, or in both
◇ $R \cap S$ = all tuples in both R and S
◇ $R - S$ = all tuples in R but not in S

- Union compatibility is similar to type checking in programming languages

Union, Intersection, Difference

Student

FN	LN
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang

Prof

FName	LName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student ∪ Prof

FN	LN
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang
John	Smith
Ricardo	Brown
Francis	Johnson

Student ∩ Prof

FN	LN
Susan	Yao
Ramesh	Shah

Student - Prof

FN	LN
Barbara	Jones
Amy	Ford
Jimmy	Wang

14

- Rules must be stated to specify the attribute names of the result; for example
  ◇ the attributes of the first operand
  ◇ the attributes of the second operand
  ◇ explicitly-specified attributes, e.g., $R_1(FN, LN) \leftarrow \mathit{Student} \cup \mathit{Prof}$

Cartesian Product

Temp ← Department × DeptLocations

DNumber	DName	MgrSSN	MgrStartDate	DNo	DLocation
5	Research	333445555	22-May-78	1	Houston
4	Administration	987654321	01-Jan-85	1	Houston
1	Headquarters	888665555	19-Jun-71	1	Houston
5	Research	333445555	22-May-78	4	Stafford
4	Administration	987654321	01-Jan-85	4	Stafford
1	Headquarters	888665555	19-Jun-71	4	Stafford
5	Research	333445555	22-May-78	5	Bellaire
4	Administration	987654321	01-Jan-85	5	Bellaire
1	Headquarters	888665555	19-Jun-71	5	Bellaire
5	Research	333445555	22-May-78	5	Sugarland
4	Administration	987654321	01-Jan-85	5	Sugarland
1	Headquarters	888665555	19-Jun-71	5	Sugarland
5	Research	333445555	22-May-78	5	Houston
4	Administration	987654321	01-Jan-85	5	Houston
1	Headquarters	888665555	19-Jun-71	5	Houston

- 15 tuples in result, if 3 tuples in Department and 5 tuples in DeptLocations

15

- The Cartesian product associates every tuple of the first relation with every tuple of the second one

- The result relation has all the attributes of the operand relations: if tha attributes of the operand relations are not all distinct, some of them must be renamed

- The relational algebra thus needs a **renaming** operation, with as a possible syntax:
  ◇ rename (relation name, (oldname → newname, ...))
  ◇ in the example: rename(DeptLocations,(DNumber→DNo)

Semantics of Cartesian Product

- The Cartesian product associates every tuple of one relation with every tuple of the other (this is not a very useful operation)
- The following operation is more useful

$$\pi_{\text{DNumber,DName,MgrSSN,MgrStartDate,DLocation}}(\sigma_{\text{DNumber=DNo}}(\text{Temp}))$$

DNumber	DName	MgrSSN	MgrStartDate	DLocation
1	Headquarters	888665555	19-Jun-71	Houston
4	Administration	987654321	01-Jan-85	Stafford
5	Research	333445555	22-May-78	Bellaire
5	Research	333445555	22-May-78	Sugarland
5	Research	333445555	22-May-78	Houston

- The result is a **join** of relations Department and DeptLocations, that associates each department with its locations

Join

◇ **Join** (⋈) combines two relations into one on the basis of a condition

$$R \bowtie_{\text{condition}} S = \{\text{concat(r,s)} \mid r \in R \wedge s \in S \wedge \text{condition(r,s)}\}$$

◇ Add the location information to the information about departments

$$\text{DeptLocs} \leftarrow \text{Department} \bowtie_{\text{DNumber=DNo}} \text{DeptLocations}$$

DNumber	DName	MgrSSN	MgrStartDate	DNo	Location
5	Research	333445555	22-May-78	5	Bellaire
5	Research	333445555	22-May-78	5	Sugarland
5	Research	333445555	22-May-78	5	Houston
4	Administration	987654321	01-Jan-85	4	Stafford
1	Headquarters	888665555	19-Jun-71	1	Houston

- The restriction retains only the tuples of the Cartesian product where the values of DNumber and of DNo are equal
- The projection suppresses the DNo attribute
- All these operations (Cartesian product, selection, and projection) can can be expressed as a single algebra operation: the **join**
- The join is a fundamental operation for meaningfully creating bigger relations from smaller ones: but it is not always the inverse of projection (see later)

◇ Remember that, if R and S have an attribute in common, then some attributes of R and/or S have to be renamed

◇ The condition can be more general than a test for equality of 2 attribute values
  * simple condition = ⟨R's attribute⟩ ⟨comparison⟩ ⟨S's attribute⟩
  * comparison: $=, \neq, <, \leq, >, \geq$
  * condition: combination of simple conditions with AND

◇ **Two ways of defining the semantics of joins**
  * **declarative**: create a tuple in the result for each pair of tuples in the relation arguments that satisfy the condition
  * **operational** (evaluation strategy): the condition is applied to every tuple of the Cartesian product; if it is satisfied, the tuple is kept in the answer

◇ As seen above, the join can be evaluated as a combination of Cartesian product, selection, and projection but this is not an efficient evaluation strategy; there are various strategies for implementing joins (see later); a basic method to implement the operation definition above is with nested loops:

$R \bowtie_{\text{A=B}} S =$

```
for each r ∈ R
do
    for each s ∈ S
    do
      if r.A = s.B
      then concat(r,s) ⇒ result
      fi
    end
end
```

- Kinds of joins
  - **theta join**: it is the general join (when all the attributes of the operand relations appear in the result of the join and the join condition is not simply a test of equality of 2 attributes)
  - **equijoin**: when the join condition is a simple equality (e.g., A = B)
  - **natural Join** = equijoin + only one of the attributes tested for equality is included in the result
    - $R *_{A=B} S$: only one of A and B (say, A) is retained in the result
    - $R * S$: an equijoin is performed that tests equality of the attributes that have the same name in R and S, and only one of them is retained in the result (this version of join does not require the attributes of R and S to be all different)

Natural Join

- Add the location information to the information about departments (with the DNumber attribute of DeptLocs renamed as DNo)

$$\text{DeptLocs} \leftarrow \text{Department} *_{\text{DNumber}=\text{DNo}} \text{DeptLocations}$$

<u>DNumber</u>	DName	MgrSSN	MgrStartDate	<u>Location</u>
5	Research	333445555	22-May-78	Bellaire
5	Research	333445555	22-May-78	Sugarland
5	Research	333445555	22-May-78	Houston
4	Administration	987654321	01-Jan-85	Stafford
1	Headquarters	888665555	19-Jun-71	Houston

18

Other Example of Natural Join

- Create a relation associating project information with information about the department from which the projects depend

$$\text{ProjDept} \leftarrow \text{Project} * \text{Department}$$

<u>PNumber</u>	PName	PLocation	DNum	DName	MgrSSN	MgrStartDate
1	ProductX	Bellaire	5	Research	333445555	22-May-78
2	ProductY	Sugarland	5	Research	333445555	22-May-78
3	ProductZ	Houston	5	Research	333445555	22-May-78
10	Computerization	Stafford	4	Administration	987654321	01-Jan-85
20	Reorganization	Houston	1	Headquarters	888665555	19-Jun-71
30	NewBenefits	Stafford	4	Administration	987654321	01-Jan-85

- The join condition test equality of DNumber values in Project and Department
- It can also be written Project $*_{\text{DNumber}=\text{Dnumber}}$ Department

19

Relative Order of Selection and Join

Find name and address of employees who work for the Research department

- Selection on an operand of the join, or selection "before" or "inside" join

$$\begin{aligned}
&\text{ResDept} \leftarrow \sigma_{\text{DName}=\text{'Research'}}(\text{Department})\\
&\text{ResDeptEmps} \leftarrow \text{ResDept} \bowtie_{\text{DNumber}=\text{DNo}} \text{Employee}\\
&\text{Result} \leftarrow \pi_{\text{LName},\text{Address}}(\text{ResDeptEmps})
\end{aligned}$$

$$\pi_{\text{LName},\text{Address}}\left(\boxed{\boxed{\sigma_{\text{DName}=\text{'Research'}}(\text{Department})} \bowtie_{\text{DNumber}=\text{DNo}} \text{Employee}}\right)$$

- Selection on the result of the join, or selection "after" or "outside" join

$$\begin{aligned}
&\text{DeptsEmps} \leftarrow \text{Department} \bowtie_{\text{DNumber}=\text{DNo}} \text{Employee}))\\
&\text{ResDeptEmps} \leftarrow \sigma_{\text{DName}=\text{'Research'}}(\text{DeptsEmps})\\
&\text{Result} \leftarrow \pi_{\text{LName},\text{Address}}(\text{ResDeptEmps})
\end{aligned}$$

$$\pi_{\text{LName},\text{Address}}\left(\boxed{\sigma_{\text{DName}=\text{'Research'}}\left(\boxed{\text{Department} \bowtie_{\text{DNumber}=\text{DNo}} \text{Employee}}\right)}\right)$$

20

About query optimization

- The two formulations of the query above are equivalent (it should be clear that they produce the same result)
- The first one does the selection *before* the join (i.e., the result of the selection serves as operand of the join), while the second one evaluates the selection on the result of the join
- Such equivalences are frequent in the relational algebra
- If the algebraic formulation was taken as guidance for actual evaluation, then in general there would be differences in performance (in the example above, the first formulation would probably produce a more efficient execution, as the selection on DName in Department produces a small relation before the join is evaluated)
- Query optimizers of current relational technology are able to perform comparative evaluation of performance and select a good strategy
- Query optimizers for SQL will not beat the best Cobol programmer, but
  - ◇ good programmers are scarce and differences in individual productivity are enormous compared to other human activities
  - ◇ it is impossible, in practice and in theory, to similarly optimize the compilation of programs in imperative languages (e.g., Cobol or C)
  - ◇ the best strategy selected depends on the database populations; if populations change so much that the best evaluation strategy ceases to be the best
    - ∗ Cobol programs have to be rewritten to adjust to the new situation and remain optimum
    - ∗ SQL optimization is redone dynamically by the DBMS: they can be optimized without being rewritten
- Demonstrates the clear superiority of nonprocedural approaches over imperative ones: this was a key factor in establishing the relational model
- For actual evaluation, a useful heuristics is to perform selections before joins, because joins are expensive operations that should be evaluated on operands as small as possible
- The user has to choose one of the two algebraic formulations (but the query optimizer may decide to evaluate the query with the other one)

An Example with Two Joins

For every project located in 'Brussels', list the project number, the controling department number, and the department manager's last name, address, and birth date

- The result of joining Project and Department is joined with Employee

$$\begin{aligned}
&\text{BrusselsProjs} \leftarrow \sigma_{\text{PLocation}=\text{`Brussels'}}(\text{Project})\\
&\text{ProjDept} \leftarrow \text{BrusselsProjs} \bowtie_{\text{DNum}=\text{DNumber}} \text{Department}\\
&\text{ProjDeptMgr} \leftarrow \text{ProjDept} \bowtie_{\text{MgrSSN}=\text{SSN}} \text{Employee}\\
&\text{Result} \leftarrow \pi_{\text{PNumber,DNum,LName,Address,BDate}}(\text{ProjDeptMgr})
\end{aligned}$$

- The result of joining Department and Employee is joined with Project

$$\begin{aligned}
&\text{BrusselsProjs} \leftarrow \sigma_{\text{PLocation}=\text{`Brussels'}}(\text{Project})\\
&\text{DeptMgr} \leftarrow \text{Department} \bowtie_{\text{MgrSSN}=\text{SSN}} \text{Employee}\\
&\text{ProjDeptMgr} \leftarrow \text{BrusselsProjs} \bowtie_{\text{DNum}=\text{DNumber}} (\text{DeptMgr})\\
&\text{Result} \leftarrow \pi_{\text{PNumber,DNum,LName,Address,BDate}}(\text{ProjDeptMgr})
\end{aligned}$$

21

- The query requires two binary joins and formulations differ in the ordering of those joins
- For actual evaluation, the query optimizer will choose the most efficient order (in this case, most probably the second one).
- A third formulation would express a product of Employee and Project, and would join the result with Department

Join or Intersection

List the names of managers who have at least one dependent

$$\text{EmpDep} \leftarrow \text{Employee} *_{\text{SSN}=\text{ESSN}} \text{Dependent}$$
$$\text{MgrDep} \leftarrow \text{EmpDep} *_{\text{SSN}=\text{MgrSSN}} \text{Department}$$
$$\text{Result} \leftarrow \pi_{\text{LName}}(\text{MgrDep})$$

- Projections can be "moved down" to make join arguments smaller
- Joins of one-attribute relations are intersections
- For actual evaluation, the query optimizer chooses the location of projections

$$\text{Mgrs(SSN)} \leftarrow \pi_{\text{MgrSSN}}(\text{Department})$$
$$\text{EmpsWithDeps(SSN)} \leftarrow \pi_{\text{ESSN}}(\text{Dependent})$$
$$\text{MgrsWithDeps} \leftarrow \text{Mgrs} \cap \text{EmpsWithDeps}$$
$$\text{Result} \leftarrow \pi_{\text{LName}}(\text{MgrsWithDeps} * \text{Employee})$$

22

- The join Employee $*_{\text{SSN}=\text{ESSN}}$ Dependent illustrates the loss of information in a join (see later, outer join)
- The join MgrsWithDeps $*$ Employee is sometimes called a semi-join, it is similar to a selection: it selects the tuples of Employee whose SSN appears in the one-attribute relation MgrsWithDeps.

Difference

List the names of employees who have no dependent

(1) $\text{AllEmps} \leftarrow \pi_{\text{SSN}}(\text{Employee})$
$\text{EmpsWithDeps(SSN)} \leftarrow \pi_{\text{ESSN}}(\text{Dependent})$
$\text{EmpsWithoutDeps} \leftarrow \text{AllEmps} - \text{EmpsWithDeps}$
$\text{Result} \leftarrow \pi_{\text{LName}}(\text{EmpsWithoutDeps} * \text{Employee})$

(2) $\pi_{\text{LName}}\Big(\overbrace{(\underbrace{\pi_{\text{SSN}}(\text{Employee})}_{\text{AllEmps}} - \underbrace{\pi_{\text{ESSN}}(\text{Dependent})}_{\text{EmpsWithDeps}})}^{\text{EmpsWithoutDeps}} * \text{Employee}\Big)$

(3) $\pi_{\text{LName}}\Big(\text{Employee} - \overbrace{\text{Employee} *_{\text{SSN}=\text{ESSN}} \underbrace{(\pi_{\text{ESSN}}(\text{Dependent}))}_{\text{EmpsWithDeps}}}^{\text{EmpsWithDeps}'}\Big)$

23

Union

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

$$\text{Smiths(ESSN)} \leftarrow \pi_{\text{SSN}}(\sigma_{\text{LName}=\text{'Smith'}}(\text{Employee}))$$
$$\text{SmithWorkerProjs} \leftarrow \pi_{\text{PNo}}(\text{WorksOn} * \text{Smiths})$$
$$\text{Mgrs} \leftarrow \pi_{\text{LName,DNumber}}(\text{Employe} \bowtie_{\text{SSN}=\text{MgrSSN}} \text{Department})$$
$$\text{SmithMgrs} \leftarrow \sigma_{\text{LName}=\text{'Smith'}}(\text{Mgrs})$$
$$\text{SmithManagedDepts(DNum)} \leftarrow \pi_{\text{DNumber}}(\text{SmithMgrs})$$
$$\text{SmithMgrProjs(PNo)} \leftarrow \pi_{\text{PNumber}}(\text{SmithManagedDepts} * \text{Project})$$
$$\text{Result} \leftarrow \text{SmithWorkerProjs} \cup \text{SmithMgrProjs}$$

This query really is the union of two simpler queries

24

Division

- $R(A,B) \div S(B) =$
  - ◇ all tuples of $\pi_A(R)$ that appear in R with at least every element of S
  - ◇ $\{a \in \pi_A(R) \mid S \subseteq \pi_B(\sigma_{A=a}(R))\}$
  - ◇ $\{a \in \pi_A(R) \mid \forall b \in S \rightarrow (a,b) \in R\}$

25

An Example with Division

- Retrieve the name of employees who work on all the projects that Smith works on

$$\begin{aligned}
&\text{Smith} \leftarrow \sigma_{\text{LName}=\text{`Smith'}}(\text{Employee})\\
&\text{SmithPNos} \leftarrow \pi_{\text{PNo}}(\text{WorksOn} *_{\text{ESSN}=\text{SSN}} \text{Smith})\\
&\text{SSNPNos} \leftarrow \pi_{\text{PNo,ESSN}}(\text{WorksOn})\\
&\text{SSNS(SSN)} \leftarrow \text{SSNPNos} \div \text{SmithPNos}\\
&\text{Result} \leftarrow \pi_{\text{FName,LName}}(\text{SSNS} * \text{Employee})
\end{aligned}$$

26

Division: Operational Semantics

SSNPNos

ESSN	PNo
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SmithPNos

PNo
1
2

SSNS

SSN
123456789
453453453

Result

FName	LName
John	Smith
Joyce	English

27

- **Operational semantics** of division:
  - ◇ partition relation SSNPNos in classes of tuples with the same value of ESSN
  - ◇ relation SSNS is the result of the division SSNPNos ÷ SmithPNos, i.e., the set of ESSN values that occur in a class with at least the PNo values of SmithPNos
  - ◇ Relation Result is the result of the query

Semantics of Division

R

A	B
a1	b1
a1	b2
a1	b3
a1	b4
a2	b1
a2	b3
a3	b2
a3	b3
a3	b4
a4	b1
a4	b2
a4	b3

S

B
b1
b2
b3

T = R ÷ S

A
a1
a4

- Partition relation R according to the A values; to each value a_i is attached a set of B values associated with that a_i value in R
- Include in T each a_i such that the set of B values associated with a_i contains S

28

Another Example with Division

Find the names of employees who work on all the projects controled by department number 5

$$\mathrm{Dept5Projs(PNo)} \leftarrow \pi_{\mathrm{PNumber}}(\sigma_{\mathrm{DNum=5}}(\mathrm{Project}))$$
$$\mathrm{EmpProj(SSN, PNo)} \leftarrow \pi_{\mathrm{ESSN,PNo}}(\mathrm{WorksOn})$$
$$\mathrm{ResultEmpSSNs} \leftarrow \mathrm{EmpProj} \div \mathrm{Dept5Projs}$$
$$\mathrm{Result} \leftarrow \pi_{\mathrm{LName,FName}}(\mathrm{ResultEmpSSNs} * \mathrm{Employee})$$

29

Relational Completeness

- A language is "relational complete" if it has at least the power of the relational algebra
- Relational completeness is the only widely-accepted measure of power (besides computational completeness)
- Domain calculus and tuple calculus have the same power of expression as the relational algebra

30

Equivalences (Theory)

- Not all operations of the algebra are independent
- $\{\sigma, \pi, \cup, -, \times\}$ is a complete set, i.e., it has all the expression power of the algebra
- $\cap$, $\bowtie$, and $\div$ can be derived from them
- $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
- $R \bowtie_{\mathrm{condition}} S = \sigma_{\mathrm{condition}}(R \times S)$
- $R *_{\mathrm{condition}} S = \pi_{\mathrm{attr}}(\sigma_{\mathrm{condition}}(R \times S))$
- $R \div S = T$ can be reexpressed with difference and Cartesian product

31

Division Redefined

$R \div S = T$ is equivalent to

$T_1 \leftarrow \pi_A(R)$
$T_2 \leftarrow \pi_A((T_1 \times S) - R)$
$T \leftarrow T_1 - T_2$

R

A	B
a1	b1
a1	b2
a1	b3
a1	b4
a2	b1
a2	b3
a3	b2
a3	b3
a3	b4
a4	b1
a4	b2
a4	b3

S

B
b1
b2
b3

$R \div S$

A
a1
a4

T_1

A
a1
a2
a3
a4

T_2

A
a2
a3

$T_1 \times S$

A	B
a1	b1
a1	b2
a1	b3
a2	b1
a2	b2
a2	b3
a3	b1
a3	b2
a3	b3
a4	b1
a4	b2
a4	b3

32

- T_1 contains all the candidate a-values, the answer is a subset of T_1
- $T_1 \times S$ associates each a-value with all b-values of S
- if an a-value is in T_2, this means that a-value is associated in R with fewer b-values than the b-values with which it is associated in $T_1 \times S$: that a-value should not be part of the result

Equivalences (Practical)

(1) $\sigma_{c_1 \wedge c_2}(R) = \sigma_{c_1}(\sigma_{c_2}(R))$

(2) $\sigma_{c_1}(\sigma_{c_2}(R)) = \sigma_{c_2}(\sigma_{c_1}(R))$

(3) if $A \subseteq A_1$, then $\pi_A(R) = \pi_A(\pi_{A_1}(R))$

(4) $\pi_A(\sigma_c(R)) = \sigma_c(\pi_A(R))$ if attributes in $c \subseteq$ attributes in A

(5) $\sigma_c(R \bowtie S) = \sigma_c(R) \bowtie S$ if attributes in $c \subseteq$ attributes in R

(6) $\pi_{A,B}(R \bowtie_c S) = \pi_A(R) \bowtie_c \pi_B(S)$
if c involves only attributes in A of R and in B of S

(7) $\pi_{A,B}(R \bowtie_c S) = \pi_{A,B}(\pi_{A,A_1}(R) \bowtie_c \pi_{B,B_1}(S))$
if c involves attributes in A, A_1 of R and in B, B_1 of S

(8) $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

(9) $\sigma_c(R \cup S) = \sigma_c(R) \cup \sigma_c(S)$

(10) $\pi_A(R \cup S) = \pi_A(R) \cup \pi_A(S)$

33

(1) Break conjunctive selection $\sigma_{c_1 \wedge c_2}(R) = \sigma_{c_1}(\sigma_{c_2}(R))$

(2) Commute selections $\sigma_{c_1}(\sigma_{c_2}(R)) = \sigma_{c_2}(\sigma_{c_1}(R))$
Used for

- applying the most selective join first for efficiency
- commute a simple selection with another operation (join, projection)

(3) Sequence partial projections: if $A \subseteq A_1$, then $\pi_A(R) = \pi_A(\pi_{A_1}(R))$

(4) Commute selection and projection:

- $\pi_A(\sigma_c(R)) = \sigma_c(\pi_A(R))$, if attributes in $c \subseteq$ attributes in A
- $\sigma_c(\pi_A(R))$ can be evaluated as $\pi_A(\sigma_c(R))$ (but not the other way around)

(5) Enter selection into join: if attributes in $c \subseteq$ attributes in R, then $\sigma_c(R \bowtie S) = \sigma_c(R) \bowtie S$

(6) Enter projection into join: if c involves only attributes in A of R and in B of S, then $\pi_{A,B}(R \bowtie_c S) = \pi_A(R) \bowtie_c \pi_B(S)$

(7) Enter projection into join (general case):

- if c involves attributes in A, A_1 of R and in B, B_1 of S, then $\pi_{A,B}(R \bowtie_c S) = \pi_{A,B}(\pi_{A,A_1}(R) \bowtie_c \pi_{B,B_1}(S))$
- the attributes of R (idem for S) comprise A, A_1, and A_2; A is needed in the result, A_1 participates in the join but is not needed in the result; A_2 does not participate at all

(8) Associate, commute joins (also valid for set-theoretic operations)
$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
$R \bowtie S = S \bowtie R$

Used for choosing the order of joins for efficiency

(9) Enter selection into union (also intersection, difference)
$\sigma_c(R \cup S) = \sigma_c(R) \cup \sigma_c(S)$

(10) Enter projection into union (also intersection, difference)
$\pi_A(R \cup S) = \pi_A(R) \cup \pi_A(S)$

Motivation for Outer Joins: Ordinary Joins are often Lossy

- $\pi_{\text{FName},\ldots,\text{DNo}}(\text{Employee} \bowtie_{\text{SSN}=\text{MgrSSN}} \text{Department}) \subseteq \text{Employee}$
- $\pi_{\text{FName},\ldots,\text{DNo}}(\text{Employee} \sqsupset\!\!\bowtie_{\text{SSN}=\text{MgrSSN}} \text{Department}) = \text{Employee}$
- $\pi_{\text{FName},\ldots,\text{DNo}}(\sigma_{\text{DName}\neq\text{null}}(\text{Employee} \sqsupset\!\!\bowtie_{\text{SSN}=\text{MgrSSN}} \text{Department}) =$
  $\pi_{\text{FName},\ldots,\text{DNo}}(\text{Employee} \bowtie_{\text{SSN}=\text{MgrSSN}} \text{Department})$

34

- "Lossy" = information is lost in the result of the join (e.g. employees who are not department managers disappear in $\text{Employee} \bowtie_{\text{SSN}=\text{MgrSSN}} \text{Department}$)
- In $R \bowtie S$, only tuples satisfying the join condition contribute to the result and information may be lost (projections of $R \bowtie S$ on R or S may be smaller than R or S)

Outer Joins

- **Outer Joins** preserve information from the operands (outer joins are "lossless")
- **Left Outer Join** $R \sqsupset\!\!\bowtie S$: retains all tuples of the left operand relation R: if, for a tuple of R, no matching tuple is found in S, the attribute values corresponding to S in the result are set to null
- **Right Outer Join** S in $R \bowtie\!\!\sqsubset S$: retains all tuples of S
- **Full Outer Join** ($\sqsupset\!\!\bowtie\!\!\sqsubset$): retains all tuples in both relations

35

Example of Outer Join

- Retrieve the name of all employees, plus the name of the departments that they manage (if any)

$$\text{Temp} \leftarrow \text{Employee} \sqsupset\!\!\bowtie_{\text{SSN}=\text{MgrSSN}} (\text{Department})$$
$$\text{Result} \leftarrow \pi_{\text{FName},\text{MInit},\text{LName},\text{DName}}(\text{Temp})$$

Result

FName	MInit	LName	DName
John	B	Smith	null
Franklin	T	Wong	Research
Alicia	J	Zelaya	null
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	null
Joyce	A	English	null
Ahmad	V	Jabbar	null
James	E	Borg	Headquarters

36

Outer Union

- Union of tuples from two **partially compatible** relations (only some of their attributes are union compatible)
- Attributes that are not union compatible from either relation are kept in the result, and tuples with no values for these attributes are padded with null values
- Outer union of

  Student(Name,SSN,Department,Advisor)
  Professor(Name,SSN,Department,Rank)

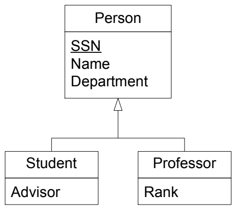
  is a relation R(Name,SSN,Department,Advisor,Rank) obtained from

  Student $*$ Professor $\cup$
  Professors that are not students (with null for Advisor) $\cup$
  Students that are not professors (with null for Rank)
- All tuples of the operand relations appear as a subtuple of the result

37

Outer Union as ER Generalization

- Outer union corresponds to generalization in the ER model



38

Transitive Closure

- Natural and frequent operation for exploring nested structures (e.g., part-subpart composition)
- Natural in algebra style but not available
- Applies to a **recursive relationship** between tuples of the same relation, e.g., between employee and supervisor in relation Employee

Example

- Retrieve all employees supervised by James Borg

  $=$ all employees directly supervised by James Borg
  $+$ all employees directly supervised by the previous ones
  $+$ ...
- Although it is possible to specify each level in relational algebra, the number of levels is not known since it depends on the extension

39

- Remember that the relational algebra (nor TRC, DRC, SQL - see later) is not computational complete (= does not have the expressive power of algorithmic languages)
- Computational completeness = Church-Turing thesis
  - thesis = any algorithm that you can think of can be formulated with any of the popular programming languages
  - revised with Gödel theorem and undecidable problems

Recursive Formulations of Transitive Closure

- Prolog-style

$$\begin{aligned} \text{Result}(s) &\leftarrow \text{BorgSSN}(s) \\ \text{Result}(s_1) &\leftarrow \text{Supervision}(s_1, s_2) \text{ and } \text{Result}(s_2) \end{aligned}$$

- Recursive equation

$$\begin{aligned} \text{Result}(\text{SSN} \leftarrow \text{SSN1}) \leftarrow\ & \text{BorgSSN}(\text{SSN1}) \cup \\ & \pi_{\text{SSN1}}(\text{Supervision} *_{\text{SSN2}=\text{SSN}} \text{Result}) \end{aligned}$$

- Alternative: usual algorithmic program

More about recursion ...

- A dictionary definition:
  - ◇ *recursive*: see "recursive"
- "Anything in computer science that is not recursive is no good" (Jim Gray?)
- Recursion = fundamental linguistic tool (like iteration) for expressing, in a well-defined finite way, patterns of action that repeat an unknown number of times (in natural language: "and so on", "etc.", "...")

Incomplete Algebra Solution for Transitive Closure

$$\begin{aligned} &\text{BorgSSN} \leftarrow \pi_{\text{SSN}}(\sigma_{\text{FName}=\text{'James'} \wedge \text{LName}=\text{'Borg'}}(\text{Employee})) \\ &\text{Supervision}(\text{SSN1}, \text{SSN2}) \leftarrow \pi_{\text{SSN},\text{SuperSSN}}(\text{Employee}) \\ &\text{Result1}(\text{SSN}) \leftarrow \pi_{\text{SSN1}}(\text{Supervision} \bowtie_{\text{SSN2}=\text{SSN}} \text{BorgSSN}) \\ &\text{Result2}(\text{SSN}) \leftarrow \pi_{\text{SSN1}}(\text{Supervision} \bowtie_{\text{SSN2}=\text{SSN}} \text{Result1}) \end{aligned}$$

BorgSSN

SSN
888665555

Supervision

SSN1	SSN2
123456789	333445555
333445555	888665555
999887777	987654321
987654321	888665555
666884444	333445555
453453453	333445555
987987987	987654321
888665555	null

Incomplete Algebra Solution for Transitive Closure

Result1

SSN
333445555
987654321

supervised by Borg

Result2

SSN
123456789
999887777
666884444
453453453
987987987

supervised by Borg's subordinates

Result

SSN
333445555
987654321
123456789
999887777
666884444
453453453
987987987