Motivation

- We know how to store data …
- How can we retrieve (interesting) data?
- We need a query language
  - *declarative* (to allow for abstraction)
  - *optimisable* (➔ less expressive than a programming language, not Turing-complete)
  - relations as input and output

E.F. Codd (1970): Relational Algebra

Characteristics of an Algebra

- Expressions
  - Are constructed with operators from atomic operands (constants, variables, ….)
  - can be evaluated
- Expressions can be equivalent
  - …if they return the same result for all values of the variables
  - Equivalence gives rise to identities between (schemas of) expressions
- The value of an expression is independent of its context
  - e.g., 5 + 3 has the same value, no matter whether it occurs as

    $10 - (5 + 3)$ or $4 \cdot (5 + 3)$

  - Consequence: subexpressions can be replaced by equivalent expressions without changing the meaning of the entire expression

Example: Algebra of Arithmetic

- Atomic expressions:

  numbers and variables

- Operators: +, -, ·, :

- Identitities:

$$x + y = y + x$$
$$x \cdot (y + z) = x \cdot y + x \cdot z$$
$$\ldots$$

Relational Algebra: Principles

Atoms are relations

Operators are defined for arbitrary instances of a relation

Two results have to be defined for each operator

1. result schema
   (depending on the schemas of the argument relations)
2. result instance
   (depending on the instances of the arguments)

- *"Equivalent" to SQL query language*
- *Relational Algebra concepts reappear in SQL*
- *Used inside a DBMS, to express query plans*

Classification of Relational Algebra Operators

- Set theoretic operators

  union "$\cup$", intersection "$\cap$", difference "\"
- Renaming operator ρ
- Removal operators

  projection π, selection σ
- Combination operators

  Cartesian product "$\times$", joins "$\bowtie$"
- Extended operators

  duplicate elimination, grouping, aggregation, sorting, outer joins, etc.

Set Theoretic Operators

Observations:

- Instances of relations are sets
  ➔ we can form unions, intersections, and differences
- Results of algebra operations must be relations, i.e., results must have a schema

Would it make sense to apply these operators to bags (= multisets)?

Hence:

- Set theoretic algebra operators can only be applied to relations with identical attributes, i.e.,
  - same *number* of attributes
  - same *names*
  - same *types*

Union

CS-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4
s4	Maurer	2

Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4

CS-Student $\cup$ Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4
s4	Maurer	2

Note: relations are sets
$\rightarrow$ duplicates are eliminated

Intersection

CS-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4
s4	Maurer	2

Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4

CS-Student ∩ Master-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4

Difference

CS-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4
s4	Maurer	2

Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4

CS-Student \ Master-Student

Studno	Name	Year
s4	Maurer	2

Not Every Union That Makes Sense is Possible

Father-Child

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

Mother-Child

Mother	Child
Eve	Abel
Eve	Seth
Sara	Isaac

Father-Child $\cup$ Mother-Child

??

Renaming

- The renaming operator ρ changes the name of one or more attributes
- It changes the schema, but not the instance of a relation

Father-Child

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

$\rho_{\text{Parent} \leftarrow \text{Father}}$ (Father-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

Father-Child

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

$\rho_{\text{Parent} \leftarrow \text{Father}}$ (Father-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

Mother-Child

Mother	Child
Eve	Abel
Eve	Seth
Sara	Isaac

$\rho_{\text{Parent} \leftarrow \text{Mother}}$ (Mother-Child)

Parent	Child
Eve	Abel
Eve	Seth
Sara	Isaac

$\rho_{\text{Parent} \leftarrow \text{Father}}$ (Father-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

$\rho_{\text{Parent} \leftarrow \text{Mother}}$ (Mother-Child)

Parent	Child
Eve	Abel
Eve	Seth
Sara	Isaac

$\rho_{\text{Parent} \leftarrow \text{Father}}$ (Father-Child)

$\cup$

$\rho_{\text{Parent} \leftarrow \text{Mother}}$ (Mother-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac
Eve	Abel
Eve	Seth
Sara	Isaac

Projection and Selection

Two "orthogonal" operators

- Selection:
  - horizontal decomposition
- Projection:
  - vertical decomposition

Projection

General form:

$$\pi_{A1,\ldots,Ak}(R)$$

where R is a relation and A1,…,Ak are attributes of R.

Result:

- Schema: (A1,…,Ak)
- Instance: the *set* of all subtuples t[A1,…,Ak] where $t \in R$

Intuition: Deletes all attributes that are not in projection list

In general, needs to *eliminate duplicates*

Real systems do projection without this!

… but not if A1,…,Ak comprises a key (why?)

Projection: Example

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$\pi_{tutor}(STUDENT)$ =

tutor
bush
kahn
goble
zobel

Note: result relations don't have a name

Selection

General form:

$$\sigma_C(R)$$

with a relation R and a condition C on the attributes of R.

Result:

- Schema: the schema of R
- Instance: the *set* of all $t \in R$ that satisfy C

Intuition: Filters out all tuples that do not satisfy C

No need to eliminate duplicates (Why?)

Selection: Example

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$\sigma_{name='bloggs'}(STUDENT)$ =

studno	name	hons	tutor	year
s4	bloggs	ca	goble	1

Selection Conditions

Elementary conditions:

<attr> op <val> or <attr> op <attr> or <expr> op <expr>

where op is "=", "<", "≤", *(on numbers and strings)*
"LIKE" *(for string comparisons),…*

Example: age ≤ 24, phone LIKE '0039%',
salary + commission ≥ 24 000

No specific set of elementary conditions is built into rel alg …

Combined conditions (using Boolean connectives):

C1 and C2 or C1 or C2 or not C

Operators Can Be Nested

Who is the tutor of the student named “Bloggs”?

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

π_{tutor} ($\sigma_{name='bloggs'}$ (STUDENT))

=

tutor
goble

Operator Applicability

Not every operator can be applied to every relation:

- Projection: $\pi_{A1,\ldots,Ak}$ is applicable to R if

  R has attributes with the names A1,…, Ak

- Selection: σ_C is applicable to R if

  all attributes mentioned in C
  appear as attributes of R
  and have the right types

Identities for Selection and Projection

For all conditions C1, C2 and relations R we have:

- $\sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C2}(\sigma_{C1}(R)) = \sigma_{C1 \text{ and } C2}(R))$

What about

- $\pi_{A1,\ldots,Am}(\pi_{B1,\ldots,Bn}(R)) = \pi_{B1,\ldots,Bn}(\pi_{A1,\ldots,Am}(R))$?

And what about

- $\pi_{A1,\ldots,Am}(\sigma_{C}(R)) = \sigma_{C}(\pi_{A1,\ldots,Am}(R))$?

Any ideas for more identities?

Cartesian Product

General form: $R \times S$

where R and S are arbitrary relations

Result:

- Schema: (A1,…,Am,B1,…,Bn), if (A1,…,Am) is the schema of R and (B1,…,Bn) is the schema of S.

  (If A is an attribute of both, R and S, then $R \times S$ contains the *disambiguated attributes* R.A and S.A.)

- Instance: the set of all *concatenated tuples*

  (t,s)

  where $t \in R$ and $s \in S$

Cartesian Product: Student × Course

STUDENT

studno	name
s1	jones
s2	brown
s6	peters

COURSE

courseno	subject	equip
cs250	prog	apple
cs150	prog	apple
cs390	specs	apple

STUDENT × COURSE

studno	name	courseno	subject	equip
s1	jones	cs250	prog	apple
s1	jones	cs150	prog	apple
s1	jones	cs390	specs	apple
s2	brown	cs250	prog	apple
s2	brown	cs150	prog	apple
s2	brown	cs390	specs	apple
s6	peters	cs250	prog	apple
s6	peters	cs150	prog	apple
s6	peters	cs390	specs	apple

Cartesian Product: Student × Staff

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

STAFF

lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

What's the point of this?

studno	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	kahn	IT206
s1	jones	ca	bush	2	bush	2.26
s1	jones	ca	bush	2	goble	2.82
s1	jones	ca	bush	2	zobel	2.34
s1	jones	ca	bush	2	watson	IT212
s1	jones	ca	bush	2	woods	IT204
s1	jones	ca	bush	2	capon	A14
s1	jones	ca	bush	2	lindsey	2.10
s1	jones	ca	bush	2	barringer	2.125
s2	brown	cis	kahn	2	kahn	IT206
s2	brown	cis	kahn	2	bush	2.26
s2	brown	cis	kahn	2	goble	2.82
s2	brown	cis	kahn	2	zobel	2.34
s2	brown	cis	kahn	2	watson	IT212
s2	brown	cis	kahn	2	woods	IT204
s2	brown	cis	kahn	2	capon	A14
s2	brown	cis	kahn	2	lindsey	2.10
s2	brown	cis	kahn	2	barringer	2.125
s3	smith	cs	goble	2	kahn	IT206
s3	smith	cs	goble	2	bush	2.26
s3	smith	cs	goble	2	goble	2.82
s3	smith	cs	goble	2	zobel	2.34
s3	smith	cs	goble	2	watson	IT212
s3	smith	cs	goble	2	woods	IT204
s3	smith	cs	goble	2	capon	A14
s3	smith	cs	goble	2	lindsey	2.10
s3	smith	cs	goble	2	barringer	2.125
s4	bloggs	ca	goble	1	kahn	IT206

• • •

“Where is the Tutor of Bloggs?”

To answer the query

“For each student, identified by name and student number, return the name of the tutor and their office number”

we have to

- combine tuples from Student and Staff
- that satisfy “Student.tutor=Staff.lecturer”
- and keep the attributes studno, name, tutor and lecturer.

In relational algebra:

$$\pi_{studno,name,lecturer,roomno}(\sigma_{tutor=lecturer}(Student \times Staff))$$

The part $\sigma_{tutor=lecturer}(Student \times Staff)$ is a “join”.

Example: Student Marks in Courses

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

"For each student, show the courses in which they are enrolled and their marks!"

ENROL

stud no	course no	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

First, do

$$R \leftarrow \sigma_{\text{Student.studno= Enrol.studno}} (\text{Student} \times \text{Enrol}),$$

then

$$\text{Result} \leftarrow \pi_{\text{studno,name, …,exam\_mark}}(R)$$

Natural Join

Suppose: R, S are two relations

with attributes A1,…,Am and B1,…,Bn, resp.
and with common attributes D1,…,Dk

The natural join of R and S is a relation that has as

Schema: all attributes occurring in R or S,
where common attributes occur only once,
i.e., the set of attributes **Attr** = {A1,…,Am,B1,…,Bn}

Instance: all tuples t over the attributes **Att** such that
t[A1,…,Am] $\in$ R and t[B1,…,Bn] $\in$ S

Notation: R $\bowtie$ S

Natural Join is a Derived Operation

The natural join of R and S can be written using

- Cartesian Product
- Selection
- Projection

$$R \bowtie S = \pi_{\mathbf{Attr}}(\sigma_{R.D1=S.D1 \text{ and } \ldots \text{ and } R.Dk=S.Dk}(R \times S))$$

θ-Joins (read, Theta-Joins), Equi-Joins

Most general form of join

- First, form Cartesian product of R and S
- Then, filter $R \times S$ with operators (abstractly written "θ") relating attributes of R and S

Notation:

$$R \bowtie_C S = \sigma_C (R \times S)$$

Special case: If C is a conjunction of equalities, i.e.,

C = R.A1=S.B1 and … and R.Al=S.Bl

then the θ-Join with condition C is called an equi-join.

Example:

$\sigma_{\text{tutor=lecturer}}(\text{Student} \times \text{Staff}) = \text{Student} \bowtie_{\text{tutor=lecturer}} \text{Staff}$

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

STAFF

lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

$$\text{Student} \bowtie_{\text{tutor=lecturer}} \text{Staff} \quad (= \sigma_{\text{tutor=lecturer}}(\text{Student} \times \text{Staff}))$$

stud no	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	bush	2.26
s2	brown	cis	kahn	2	kahn	IT206
s3	smith	cs	goble	2	goble	2.82
s4	bloggs	ca	goble	1	goble	2.82
s5	jones	cs	zobel	1	zobel	2.34
s6	peters	ca	kahn	3	kahn	IT206

Self Joins

For some queries we have to combine data coming from a single relation.

> *"Give all pairings of lecturers and appraisers, including their room numbers!"*

We need two identical versions of the STAFF relation.

Question: How can we distinguish between the versions and their attributes?

Idea:

- Introduce temporary relations with new names
- Disambiguate attributes by prefixing them with the relation names

LEC ← STAFF, APP ← STAFF

Self Joins (cntd.)

"Give all pairings of lecturers and appraisers, including their room numbers!"

$$R \leftarrow \pi_{\text{LEC.lecturer,LEC.roomno, LEC.appraiser,APP.roomno}} (\text{LEC} \bowtie_{\text{LEC.appraiser=APP.lecturer}} \text{APP})$$

$$\text{Result} \leftarrow \rho_{\text{lecturer} \leftarrow \text{LEC.lecturer,roomno} \leftarrow \text{LEC.roomno, appraiser} \leftarrow \text{LEC.appraiser,approom} \leftarrow \text{APP.roomno}} (R)$$

STAFF

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

lecturer	roomno	appraiser	approom
kahn	IT206	watson	IT212
bush	2.26	capon	A14
goble	2.82	capon	A14
zobel	2.34	watson	IT212
watson	IT212	barringer	2.125
woods	IT204	barringer	2.125
capon	A14	watson	IT212
lindsey	2.10	woods	IT204

Duplicate Elimination

Real DBMSs implement a version of relational algebra that operates on multisets ("bags") instead of sets.

(Which of these operators may return bags, even if the input consists of sets?)

For the bag version of relational algebra, there exists a duplicate elimination operator δ.

If R =

A	B
1	2
3	4
3	4
1	2

, then $\delta(R)$ =

A	B
1	2
3	4

Aggregation

- Often, we want to retrieve aggregate values, like the “sum of salaries” of employees, or the “average age” of students.
- This is achieved using aggregation functions, such as SUM, AVG, MIN, MAX, or COUNT.
- Such functions are applied by the grouping and aggregation operator γ.

If R =

A	B
1	2
3	4
3	5
1	1

, then $\gamma_{SUM(A)}(R)$ =

SUM(A)
8

and $\gamma_{AVG(B)}(R)$ =

AVG(B)
3

Grouping and Aggregation

- More often, we want to retrieve aggregate values for groups, like the "sum of employee salaries" per department, or the "average student age" per faculty.
- As additional parameters, we give γ attributes that specify the criteria according to which the tuples of the argument are grouped.
- E.g., the operator $\gamma_{A,SUM(B)}(R)$
  - partitions the tuples of R in groups that agree on A,
  - returns the sum of all B values for each group.

If R =

A	B
1	2
3	4
3	5
1	3

, then $\gamma_{A,SUM(B)}(R)$ =

A	SUM(B)
1	5
3	9

Summary

- Relational algebra is a query language for the relational data model
- Expressions are built up from relations and unary and binary operators
- Operators can be divided into set theoretic, renaming, removal and combination operators (plus extended operators)
- Relational algebra is the target language into which user queries are translated by the DBMS
- Identities allow one to rewrite expressions into equivalent ones, which may be more efficiently executable ($\rightarrow$ query optimization)

References

In preparing these slides I have used several sources.
The main ones are the following:

Books:

- A First Course in Database Systems, by J. Ullman and J. Widom
- Fundamentals of Database Systems, by R. Elmasri and S. Navathe

Slides from Database courses held by the following people:

- Enrico Franconi (Free University of Bozen-Bolzano)
- Carol Goble and Ian Horrocks (University of Manchester)
- Diego Calvanese (Free University of Bozen-Bolzano) and Maurizio Lenzerini (University of Rome, “La Sapienza”)

In particular, a number of figures are taken from their slides.