6 The Relational Data Model: Algebraic operations on tabular data

Kemper / Eickler: 3.4, Elmasri /Navathe: chap. 74-7.6,
Garcia-Molina, Ullman, Widom: chap. 5

Context

Requirements analysis

Conceptual Design

Schema design
- logical ("create tables")

Schema design
- physical
("create access path")

Loading, administration, tuning, maintenance, reorganization

Part 1: Designing and using database

Database Design
- developing a relational database schema
Design
- formal theory

Data handling in a relational database
- Algebra,-calculus -SQL

Extensions

Using Databases from application progs

Special topics,

Physical Schema,

Part 2:
Implementation of DBS
Transaction,
Concurrency control
Recovery

6.1 Basic idea of relational languages

Important concepts

- Data Model:

  Language for definition and handling (manipulation) of data

  – Languages for handling data:
    - Relational Algebra (RA) as a semantically well defined applicative language
    - Relational tuple calculus (domain calculus): predicate logic interpretation of data and queries
    - SQL / DML (or simply SQL)

  – Kernel of SQL built upon RA as well as calculus, extended by operations like arithmetic expressions not available in RA or calculus

Relational Languages

Goal of language design

Given a relational database like the Video shop DB

Design a language, which allows to express queries like:

- Customers who rented videos for more than 100 $ last month
- List of all movies no copy of which have been on loan since 2 month
- List the total sales volume of each movie within the last year
- Is there anybody whose rented movies all have category "horror"?

  ....

Privacy?

Language should be declarative ("descriptive")

Historically: "Make query formulation 'as easy as in natural language' "

Relational Algebra

- Idea of Relational Algebra:
  – Given relations
    R(a1,...,an), S(b1,...,bm)
  – Define operators which transform one or more tables into a result table.

Example

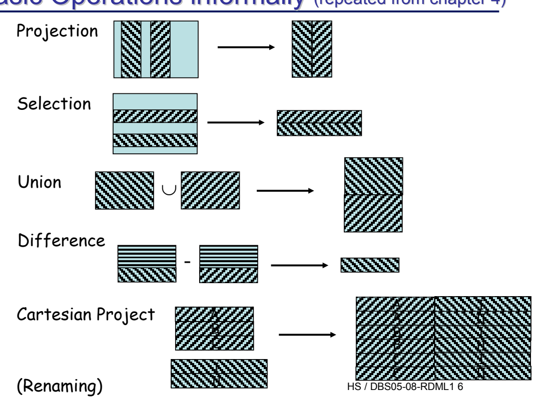
```
Tape ( id   movieId )
       1    'B'
       5    'A'
       6    'B'
```

Join tables!

```
Movie ( m_Id   title )
         'A'    "Frenzy"
         'B'    "Matrix"
```

How could we find the number of tapes of "Matrix?"

Basic Operations informally (repeated from chapter 4)



Relational Algebra Basics

- Why "algebra"?
  - Mathematically, algebraic structures basically defined by a base set S of values and operations which map one or more elements of S to S and obey certain laws (e.g. groups, lattices, …)
  - The base set of Relational Algebra is the set of all relations (tables) with attributes from a given set A of attributes.
  - Operations on tables projection, cartesian product, join, ... as introduced intuitively above
  - Note: Result of an operation is time dependent

6.2 Relational Algebra operations

Terminological update

Let A be a set universal of attributes

- A Relation Schema is a named n-tuple of attributes RS = R (a_1,...,a_n), {a1,…an} $\subseteq$ A
- R_A is the set {a_1,...,a_n} of attributes (columns) of RS called the type signature of R
- The operation Σ applied to a relation R results in the type signature of R: Σ (R) = R_A
- A Relational Database Schema is a set of relation schemas
- A Database Relation R (conforming to Relation Schema RS) is a subset of D(a_1) X ...X D(a_n), the cross product of the domains of the attributes of R

Set operations

Tape (id	movieId)
1	'B'
5	'A'
6	'B'

$\cup$

Movie (movieId	title)
'A'	"Frenzy"
'B'	"Matrix"

?

R, S relations,

R and S are called union-compatible if the domains of Σ (R) and Σ (S) are pair wise the same

or: two tables are union-compatible if they have the same number of columns and have the same domains in corresponding colums

R and S union-compatible, then set union and set difference R $\cup$ S and R \ S are defined as usual on mathematical sets

Other set operations may be easily defined using $\cup$ and \

Relational Algebra Basic Operations

- Cartesian (Cross) Product
  - Cross product of two sets R and S: a set of pairs with type signature Σ(R) $\subseteq$ A and Σ(S) $\subseteq$ A
  - Result relation T should be a relation over Σ(R) $\cup$ Σ(S) = A' $\subseteq$ A (assumed Σ(R) $\cap$ Σ(S) = $\varnothing$)

R (a1	a2)
1	'A'
5	'Z'

X

S (b1	a2)
3	'A'
1	'B'

=

T ((a1 , a2) (b1 | a2))
(1 'A') (3 'A')
(5 'Z') (3 'A')
(1 'A') (1 'B')
(5 'Z') (1 'B')

Not a relation schema in 1NF. Therefore NOT an operation of Relational Algebra

Relational Algebra Basic Operations

Extended cross product X

Let R and S be relations, Σ(R) ={a_1,...,a_n} $\subseteq$ A , Σ(S) = {b_1,...,b_m} $\subseteq$ A, Σ(R) $\cap$ Σ(S) = $\varnothing$

then

- Schema Σ(R X S): {R.a_1,...R.a_n, S.b_1, …,S.b_m} = {R.a | a $\in$ A} $\cup$ {S.b | b $\in$ A}

  Omit relation qualifiers "R." and "S." - no naming conflict.

- Extended cross product R X S : R X S = {(a_1,...a_n,b_1,...,b_m) | (a_1,...,a_n) $\in$ R, (b_1,...,b_m) $\in$ S}

Renaming, if Σ(R) $\cap$ Σ(S) != $\varnothing$:

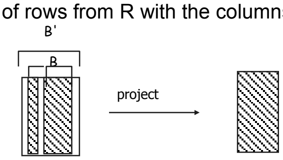
ρ <attrname> $\leftarrow$ <newAttrname> (<relname>)

Projection π

Let Σ(R) = B', B $\subseteq$ B'

Projection π_B (R) of R on B:

Set of rows from R with the columns not in B eliminated



- No duplicates in π_B (R) (in theory!)

π_B (R) = {r restricted to B | r $\in$ R}
= {r' | there is a tuple r $\in$ R such that r' is the restriction of r to the attributes in B}

Relational Algebra Basic Operations

```
Parent (id, mother, father)
```

25	Mary, Paul
47	Mary, John
55	Mary, Paul

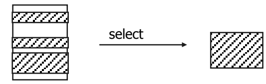
π_{mother} (Parent) = Mary

A relation with only one column and one row (no duplicates)

- Property of projection:
  B contains a key of R $\Rightarrow \pi_B$ (R) contains as many tuples as R: $|\pi_B (R)| = |R|$
- Useful for estimating the size of query results
- Important for optimization

Selection σ

"Find movies directed by Billy Wilder made 1960 or later"



Selection of tuples from a table R according to a predicate defined on R

Predicate P :: R -> {TRUE, FLASE}
defined on tuples of R.
For each r ∈ R : P(r) = TRUE | FALSE

Primitive predicates

Primitive (simple) predicates:

Let a, b be attributes, w value from dom (a)
a θ b and a θ w are primitive predicates
where θ ∈ { =, !=, <, <=, >, >=}

Primitive predicates compare either an attribute and a value or two attributes

e.g. `Movie.director = 'Wilder, Billy'`

or `Rental.until > Rental.from + 1 month`

Row predicates

Boolean row predicates

Row predicates combine primitive (simple) predicates by **and**, **or**, **not** and parenthesis '(', ')'

Inductive definition of syntax for (row) predicates ....as usual:

- Primitive predicates are predicates
- If Q, Q' are predicates, then Q ∧ Q' , Q ∨ Q' and ¬Q are predicates
- Operator preference and brackets as usual
- There are no other predicates

Movies directed by Spielberg before 1999 or an entertainment movie :

`movie.director='Spielberg'` ∧ `(year <= TO_DATE('1999', 'yyyy')` ∨ `cat = 'entertainment' )`

Propositional semantics

Semantics of predicates:

Let
a, b be attributes of table R, r ∈ R,
P the predicate a θ v , Q is the predicate a θ b
r(a) : value for attribute a of tuple r

Then
P(r) := r(a) θ v
Q(r) := r(a) θ r(b)
S ≡ P ∧ Q : S(r,t) := P(r) ∧ Q(r) according to ∧ semantics

¬, ∨ and preference as usual in propositional logic
Frequently, θ is equality predicate (=)

Selection of rows

Selection σ

$\sigma_P (R) = \{r \mid r \in R \text{ and } P(r) = \text{TRUE}\}$
where P is a row predicate

Note:

- Selection operator selects the row with all attributes:
  $\Sigma(R) = \Sigma(\sigma_P(R))$
- size of result depends on selectivity of P
  selectivity $:= |\sigma_P(R)| \,/\, |R|$
  important for optimization

Relational Algebra Basic Operations

Example

"Movies directed by Spielberg produced 1997 or later ! "

`Movie(mId,title,...,director, year)`

σ_{P} (Movie)

where P = "director = 'Spielberg' and year >= 1997"

```
M_ID TITLE       CAT         YEAR    DIRECTOR  PRICE_DAY  LENGTH
---- -----    ---------- ------- ---------- -------- --------
2    Amistat          Drama      01.04.97 Spielberg    1        120
4    Minority Report  Drama      01.04.02 Spielberg    2        145

2 Zeilen ausgewählt.
```

Wrong result…? "Truth" defined relative to contents of DB

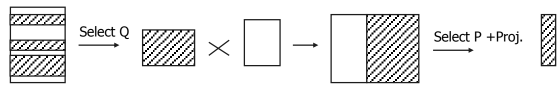
Closed world assumption: for all predicates P $r \notin R \Rightarrow P(r) = False$

Relational Algebra: combining operators

π_{title} (σ_{P}(Movie))

where P = "director = 'Spielberg' and year >= 1997"

Find the actors performing in movie directed by Spielberg



π_{stage_name} (σ_{P} (σ_{Q}(Movie) X Starring)

where P = "Movie.id = Starring.movieId "

Q = "director = 'Spielberg'

Renaming

Renaming of relations

Example: `Employee( id, name, boss, …)`

Find subordinates of 'Miller'

~~π_{name} (σ_{P} (σ_{Q} (Employee) X Employee)))~~

~~where P = "Employee.name = 'Miller' "~~

~~Q = "Employee.boss = Employee.id "~~

Renaming

$\rho_{\text{<newname>}}$ (<relname>)

Relation <relname> is renamed to <newname> in the context of expression

$\rho_{\text{<attrname>} \leftarrow \text{<newAttrname>}}$ (<relname>)

Attribute <attrname> of relation <relname> is renamed to <newAttrname> in the context of expression

$\pi_{Sub.name}$ (σ_{Q} (σ_{P} (Employee X (ρ_{Sub} (Employee)))))

where P = "Employee.name = 'Miller' "

Q = "Sub.boss = Employee.id "

Evalution example: one table – two roles

Employee

id	name	boss
001	Abel	NULL
002	Bebel	005
004	Cebel	005
005	Miller	001
006	Debel	001
		....

X

Sub

id	name	boss
001	Abel	NULL
002	Bebel	005
004	Cebel	005
005	Miller	001
006	Debel	001
		....

ρ_{Sub} (Employee)

Employee			Sub		
id	name	boss	id	name	boss
001	Abel	NULL	001	Abel	NULL
001	Abel	NULL	002	Bebel	004
...			...	...	
002	Bebel	005	001	Abel	NULL
002	Bebel	005	002	Bebel	005
...	...	...	...	...	...
005	Miller	001	001	Abel	Null
005	Miller	001	002	Bebel	005
005	Miller	001	004	Cebel	005
005	Miller	001	005	Miller	001
005	Miller	001	006	Debel	001
..	...	...	...	...	...
006	Debel	001	005	Miller	001
006	Debel	001	006	Debel	001

π_{name}

σ_{Q}

σ_{P}

6.3 Relational Algebra: Syntax and Semantics

Syntax of (simple) Relational Algebra defined inductively :

(1) Each table identifier is a RA expression

(2) ρ_{A}(B), $\rho_{s \leftarrow y}$(A) are RA expressions where A,B table identifiers, s, v attribute identifiers

(3) If E and F are RA expressions then

π_{D}(E), σ_{P}(E), E X F, $E \cup F$, E \ F are RA expressions (if union-compatible etc.)

where $D \subseteq \Sigma(E)$

(4) These are all RA expressions

Semantics of Relational Algebra

val is a function which assigns to each relational algebra expression a result table:

val ('R') = R

"The value of a relation name is the relation (table)"

val ('τ (E)') = τ (val (E))

where τ is some unary rel. Operation like π

"The value of an unary relational operator applied to an relational algebra expression E is the result of applying the operator to the value of E "

val ('E ω F') = val (E) ω val (F)

where ω is some binary operator like X

"The value of an unary relational operator applied to a relational algebra expression E is the result of applying the operator to the value of E"

6.4 Relational Algebra : More Operators

Some sequences of operations occur frequently like cartesian product followed by a select

⇨ Define compound operators

Join (θ-join)

R, S relations,

Important concept

$R \bowtie_P S$

$= \{(a_1, \dots a_n, b_1, \dots b_m) \mid P(a_1, \dots a_n, b_1, \dots b_m) \text{ is } \mathbf{true} \}$

$= \sigma_P (R \times S)$

where P is a (boolean) predicate composed of primitive predicates of the form

$a\,\theta\, b$, $a \in \Sigma(R)$, $b \in R\ \Sigma(S)$, $\theta \in \{ =, \neq, <, <=, > >=\}$

(Join predicate)

Relational Algebra Join

$R \bowtie_{R.a < S.c \wedge R.b=S.d} S$ =

1	A	2	1	3	A
2	A	2	1	3	A

The result usually does not have a name

R(a b c)

a	b	c
1	A	2
2	A	2
3	C	1

S(a c d)

a	c	d
1	3	A
2	2	B
1	2	C

R X S

1	A	2	1	3	A
1	A	2	2	2	B
1	A	2	1	2	C
2	A	2	1	3	A
2	A	2	2	2	B
2	A	2	1	2	C
3	C	1	1	3	A
3	C	1	2	2	B
3	C	1	1	2	C

Note: exactly the same as taking the set of all pairs of R and S rows and checking the predicate subsequently

Relational Algebra : more operators

Equijoin: equality comparison $\bowtie_P$

Important type of join: all primitive predicates in P compare equality of column values of two rows at a time : $P \equiv \wedge\, R.x_i = S.y_i$, $\{x_i\} \subseteq \Sigma(R)$, $\{y_i\} \subseteq \Sigma(S)$,

Implements the "values as pointers" concept of RDB for foreign keys, but is more general.

Example using foreign key: Find movie title on tape 27

π_{title} (Movie $\bowtie_{id=Tape.mid}$ $\sigma_{id=27}$ (Tape))

Example

Movie(mId,title,...,director, year)

mId	title	director	year
25	Amistad, . . .	Spielberg	1997
35	A.I.	Spielberg	2001
47	Matrix	Azzopardi	1993
55	Private Ryan	Spielberg	1998

Tape(id, acDate, format, movieId

id	acDate	format	movieId
11	9-3-98	VHS	25
23	4-6-01	DVD	47
17	1-3-99	DVD	25

$R \bowtie_{Movie.mId\ =\ Tape.movieId} S$ =

25	Amistad . .	.Spielberg	1997	11	9-3-98	VHS	25
25	Amistad . .	.Spielberg	1997	17	1-3-99	DVD	25
47	Matrix ...	Azzopardi	1993	23	4-6-01	DVD	47

Relational Algebra: more operators

- Renaming required, if identical column names
- No canonical projection of columns if columns are redundant

Example above: `mId` and `movieId`
Query with subsequent projection:

"Find title, tapeId and format for all movies"

$\pi_{title,\ id,\ format}$ (Movie $\bowtie_{Movie.id\ =\ Tape.movieId}$ Tape)

Result:

Amistad	11	VHS
Amistad	17	DVD
Matrix	23	DVD

Relational Algebra: Natural join

Natural Join R $\bowtie$ S :

equijoin over all literally identical column names of R and S and projection of redundant columns.

Join predicate omitted .

R(a b c)

a	b	c
1	A	2
2	A	2
3	C	1

S(a c d)

a	c	d
1	3	A
2	2	B
1	2	C

R $\bowtie$ S =

R.a	R.b	R.c	S.d
2	A	2	B
1	A	2	C

$$R \bowtie S = \pi_{\Sigma(R) \cup \Sigma(S)} (\sigma_P (R \times S))$$

where $P \equiv \wedge\, R.x = S.x, \quad x \in \Sigma(R) \cap \Sigma(S)$

Realtional algebra: outer join

Motivation: only tuples of S participate in a join R $\bowtie$ S, which have a "counterpart" in R.

```
Customer(mem_no,name,f_name, zip, city)
 Phones (phoneNo, mem_no)

"Print telephon list of customers"

  π name, phoneNo ( Customer ⋈ Phones)
```

Customers without phoneNo will not appear

Relational Algebra: outer join

Left outer join R $⟕_P$ S

Includes (r, NULL,…NULL) – if there is no join partner for $r \in R$

a	b	c
1	A	2
2	A	2
3	C	1

$⟕_{R.a < S.c \wedge R.b=S.d}$

a	c	d
1	3	A
2	2	B
1	2	C

=

1	A	2	1	3	A
2	A	2	1	3	A
3	C	1	-	-	-

R $⟕_P$ S =

R $\bowtie_P$ S $\cup$ { $(r_1,\ldots r_n$, NULL,…NULL)| $(r_1,\ldots r_n) \in R$, and for all $(s_1,\ldots,s_m) \in S$, $(r_1,\ldots r_n) \in R$: $P(r_1,\ldots r_n, s_1,\ldots s_m)$ =FALSE }

Outer join typically extension of equijoin

Relational Algebra: outer join

Right outer join R $⟖$ S

Includes (NULL,…NULL, s) – if there is no join partner for $s \in S$

a	b	c
1	A	2
2	A	2
3	C	1

$⟖_{R.a < S.c \wedge R.b=S.d}$

a	c	d
1	3	A
2	2	B
1	2	C

=

1	A	2	1	3	A
2	A	2	1	3	A
-	-	-	2	2	B
-	-	-	1	2	C

Full outer join: union of left and right outer join

1	A	2
2	A	2
3	C	1

$⟗_{R.a < S.c \wedge R.b=S.d}$

1	3	A
2	2	B
1	2	C

=

1	A	2	1	3	A
2	A	2	1	3	A
3	C	1	-	-	-
-	-	-	2	2	B
-	-	-	1	2	C

Relational Algebra More operators

Semjoin

R $⋉$ S = $\Pi_{\Sigma(R)}$ (R $\bowtie$ S)

Left Semijoin is the subset of R, each r of which has a corresponding tuple s from S in the join.

Typically extension of equijoin or natural join

R(a b c)

a	b	c
1	A	2
2	A	2
3	C	1

$⋉_{R.a = S.c \wedge R.b=S.d}$

S(a c d)

a	c	d
1	3	A
2	2	B
1	2	C

=

(a b c)

a	b	c
1	A	2

Right Semijoin defined symmetrically :

R $⋊$ S = $\Pi_{\Sigma(S)}$ (R $\bowtie$ S)

Relational Algebra: Base operators

Base

Set of operators which allow to express all other operators

e.g { $\wedge, \vee, \neg$ } in propositional logic

Relational operators

$\pi, \sigma, \times, \backslash$ and $\cup$ form a basis of relational algebra operators

Means: every RA expression may be expressed only with these operators

Example: $R \bowtie_P S = \sigma_P (R \times S)$

Relational Algebra: table predicates

Row predicates:

p defined over rows (or pairs of rows)

Table predicates

Example: find all movies which are available in all formats

Cannot be answered by comparing individual rows

Predicates with universal quantifier are table predicates

e.g. $P(x) \equiv \forall x \; \exists y \, (Q(x,y,m)$

"for all formats (*in the database*) x there exists a tape y with movie m"

Express table predicates with base operators?

Relational Algebra: Division

$T = \pi_{format,movieID}$ `Tape(id,format,movieId)` $F = \pi_{format}$ `Format(format,extra_Ch))`

1	VHS	7
2	DVD	7
4	VHS	55
5	VHS	1
8	HQ	7
11	VHS	25
17	DVD	25

VHS	0.0
DVD	1.0
HQ	1.5

Result:

1	7
2	7
8	7

Find movies which are available in all formats

Relational Division

Informally T . /. F is the set of all tuples r of T projected on attributes not belonging to F such that $\{(r)\} \times F \subseteq T$

Relational Algebra: an operator based on table predicates

Relational Division T . /. F

- Simulates universal quantifier for finite sets
- In order to divide T by F, the attributes of F must be a subset of the attributes of T: $\Sigma(F) \subset \Sigma(T)$
- Signature of T ./. F is $D = \Sigma(T) \setminus \Sigma(F)$

$T \,./.\, F = \{ t' \mid t' \in \pi_D\,(T) \;\wedge$
$(\forall s \in F)\,(\exists t \in T)\;\; \pi'_{\Sigma(F)}(t) = s \wedge \pi'_D\,(t) = t' \}$

π' denotes the projection of a row as opposed to π, which is defined on tables.

$\pi_D\,(T) = \{(7), (55), (1), (25)\}$

F = {VHS,DVD, HQ}

let t' be (55), for s = (DVD) there is no tuple (DVD, 55) in T. t' = (7) is the only one which qualifies

Relational Algebra Division

T ./. F may be defined in terms of other relational operators

$T \,./.\, F = \pi_D\,(T) \setminus \underbrace{(\pi_D\,\underbrace{(\pi_D\,(T) \times F) \setminus T}_{\text{The "missing" tuples of T}})}_{\text{Building the complement}}$

$\boxed{D = \Sigma(T) \setminus \Sigma(F)}$

Proof: Assignment

Property of relational division:

Let $D = \Sigma(T) \setminus \Sigma(F)$,

if D contains the key of T and $|F| > 1$ then $T \,./.\, F = \varnothing$

Relational Algebra RA expression examples

Examples find algebraic expressions

`C(mem_No,name,..,)` `T(id, m_Id, f)` `R(tape_Id,from, mem_No,..)` `M(id,title,...)`

0. Movies (m_Id) and its formats $\pi_{m\_id,\, format}\,(T)$

1. Tapes loaned by 'Abel' $\pi_{tape\_Id}\,(R \bowtie \sigma_{name='Abel'}\,(C))$

2. List of films that are currently available (i.e. not rented by anyone)

$\pi_{title}\,(M) \setminus \pi_{title}\,(R \bowtie_{tape\_Id=id} T \bowtie_{m\_Id=M.id} M))$

3. First name, last name of customers who rented “To be or not to be” ….
4. List of customers and the films they have currently rented …
5. Has 'Bebel' loaned a tape? *Cannot be formluated, why?*
6. Find the films which a available in all formats …

6.5 Special Topics of RA

6.5.1 RA operators in SQL/DML

- Transformation rule: for every relational algebra expression with join, project, cartesian product and select operations there is an equivalent expression of the form:

$\pi \; \ldots (\sigma_P \; (R_1 \times R_2 \times \ldots \times R_n))$

Simple SQL (Sequel) block:

```
SELECT DISTINCT a,b,.....
FROM   R1 ,..., Rn
WHERE  < predicate P >
```

projection

cartesian product

predicate

DISTINCT : Elimination of duplicates

Relational Algebra and SQL

"Find title, tapeId and format for all movies"

```
…
SELECT DISTINCT m.title, t.id,
t.format
FROM Movie m, Tape t
WHERE m.mId(+) = t.movieId
ORDER BY title;
```

Old Notation for left outer join,

```
SELECT DISTINCT m.title, t.id, tt.format
FROM Movie m LEFT OUTER JOIN  Tape t
     ON m.mID = t.movieID
ORDER BY title;
```

New notation

```
Movies                   TapeNO FORMA
--------------- ----------- -----
Amistad                      11 VHS
A.I.                         17 DVD
Matrix                       23 DVD
Private Ryan
```


6.5.2 Relational completeness

- Completeness
  - A DB language L is called relational complete, if every RA expression can be expressed in L
  - Are there any operations on relations, which cannot be expressed by a finite RA expression (select, project, product or join; SPJ) ?
  - Yes: transitive closure of a relation cannot be expressed in this way

Pred	Descend
Paul	Mary
Mary	Peter
John	Bill
Peter	George

No RA expression to find all decendents of 'Paul'.

Recursion is missing!

6.5.3 What is missing in RA

- Arithmetic operators,
- many practically important operators like grouping of results
  "find movies together and their number of copies"

Title	copyCount
Amistad	2
To be or ..	3
Private Ryan	1
Marnie	1
The Kid	2

- More Predicates on tables (not rows)

Anyway relational algebra important conceptual basis for query languages and query evaluation

6.5.4 Relational Algebra operator trees

Algebraic Optimization

- Evaluation of RA expressions in canonical form
  π .... (σ_P (R_1 X R_2 X … X R_n))
  is very inefficient
- How to speed up evaluation of RA (and SQL) expressions?
- Example: Two tables R and S with n and m tuples
  Worst case complexity of :
  $\sigma_P (R \bowtie S)$
  is O(m*n)
- Interchange of select and join may result in O(n+m)
  time σ_P (R) $\bowtie$ S depending on the join algorithm

Some rewrite rules for RA

Properties of selection and projection

$\sigma_P (\sigma_Q (R)) = \sigma_Q (\sigma_P (R))$

$\sigma_P (\sigma_P (R)) = \sigma_P (R)$

$\sigma_{Q \wedge P} (R) = \sigma_Q (\sigma_P (R)) = \sigma_Q (R) \cap \sigma_P (R)$

$\sigma_{Q \vee P} (R) = \sigma_Q (R) \cup \sigma_P (R)$

$\sigma_{\neg P} (R) = R \setminus \sigma_P (R)$

if $X \subseteq Y \subseteq \Sigma(R)$ then $\pi_X(\pi_Y(R)) = \pi_X(R)$

if $X, Y \subseteq \Sigma(R)$ then $\pi_X(\pi_Y(R)) = \pi_{X \cap Y}(R) = \pi_Y(\pi_X(R))$

$attr(P) \subseteq X \subseteq \Sigma(R)$ then $\pi_X(\sigma_P(R)) = \sigma_P(\pi_X(R))$

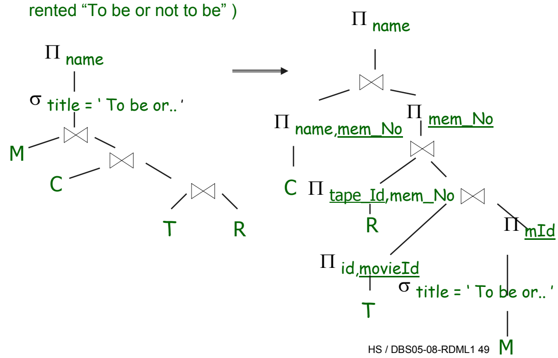
where attr(P) denotes the set of attributes used in P

Relational Algebra Using RA for opitmization

- An relational algebra operator tree is the data structure representing a RA expression
  Compare with operator trees for arithmetic expressions
- Algebraic optimization: systematic interchange of operation according to the laws of RA
- Does not change time complexity in general, but "makes n small".
- Implementation of Algebraic Optimization by transformation of the operator tree
- Evaluation by recursive evaluation of the tree
- Systematic treatment of different optimization techniques
  → Part II (Implementation)

RA Operator tree transformation: example

Operator tree: example "Last name of customers who rented "To be or not to be")



6.5.5 Multivalued dependencies and 4NF

Multiple values: example (left over from ch. 5: Normal Forms)

Person (name, affiliation, hobbies)

Meier	FUB	skiing
Müller	TUB	trekking
Meier	FUB	trekking
Schulze	HU	skating

Redundancy introduced by multiple values

'hobbies' is multivalued dependent on name

Definition for single attribute multi valued (MV) dependencies:

Let R = (a, y, b),

b is multivalued dependent on a (a ->>b) if for each value v of a $\{v\} \times (\pi_y(\sigma_{a=v} R)) \times (\pi_b(\sigma_{a=v} R)) \subseteq R$

Example: {'Meier'} X {'FU'} X {'skiing', 'trekking'} $\subseteq$ Person

{'Müller'} X {'TU'} X {'trekking'} $\subseteq$ Person

{'Schulze'} X {'HU'} X {'skating'} $\subseteq$ Person

Fourth Normal Form

A relation R is in Fourth Normal Form

if for every MVD A ->> B

- $B \subseteq A$ or
- $B = \Sigma(R) \setminus A$ or
- A contains a key

May be easily calculated by splitting up a relation R with a MVD A->> B into R1 and R2 such that

$\Sigma(R1) = A \cup B$, $\Sigma(R2) = \Sigma(R) \setminus \Sigma(R1) \cup A$

Müller	TUB
Meier	FUB
Schulze	HU

Müller	trekking
Meier	trekking
Meier	skiing
Schulze	skating

Better to have multi valued attributes?

Summary

- Relational algebra: algebra on tables
- Operators: project, select, cartesian product, union, set difference, (rename)
- Several compound operators : join, outer join, semi-join, division
- Serves as a basis for relational DB languages
- No recursion ⇨ not computationally complete
- Base of SQL
- Used for optimization by operator tree transformation