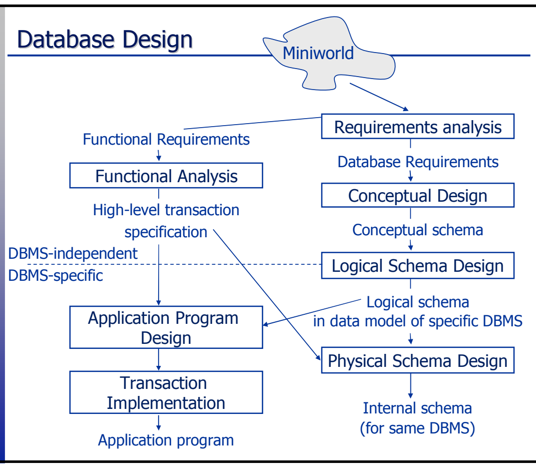
Relational Languages:
Relational Algebra & Relational calculus

Relational Algebra
Relational Calculus Languages
  Tuple Calculus
  Domain Calculus
Relational Languages Equivalence

Database Design



Relational Languages

- Data model:
  - Collection of concepts to describe structure of a DB ✓
  - Basic operations for specifying retrieval and update
- Formal Languages for handling data
  - Relational Algebra: procedural language, specifies *how* to evaluate queries
  - Relation calculus: predicate logic interpretation of data and queries, declarative language, specifies *which* data to select
  - Both are closed languages: results of queries on relations are relations

---

Relational Languages: Example Queries

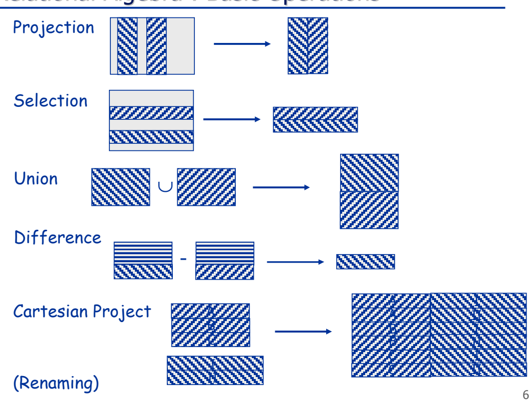
- Queries at Video shop DB

1. Names of all customers
2. All customers named Anna
3. All movies by George Lucas from 1999 or later
4. All tapes and their corresponding movie
5. All customers who have rented at least one science fiction film
6. All customers whose rented movies all have category "suspense"
7. Customers that had rented all movies
8. All movies no copy of which are currently on loan
9. Number of tapes for each movie
10. Total receipts of each movie within the last year

Relational Algebra: Basics

- Procedural query language
- Consider two relations
  $R(A_1:C_1, A_2:C_2, \ldots, A_r:C_r)$, degree r
  $S(B_1:D_1, B_2:D_2, \ldots, B_s:D_s)$, degree s
- 5 (+1) Basic Operations
- Other operations can be derived
- Schema-compatible relations
  = identical schemas (with renaming)
  - same degree of relations [r=s]
  - domains of R(R) and R(S) are pair-wise compatible
    [$\exists$ Permutation φ of indices {1, ..., r}: $\forall$ i, $1 \le i \le r$: $C_i = D_{\varphi(i)}$]

Relational Algebra : Basic Operations



Relational Algebra : Basic Operations

- **Projection:** $\Pi_{\langle \text{List of attributes} \rangle}(\text{Relation})$ — Important concept
  - Reduction of relation schema to selected attributes
  - Reduction of relation tuple to respective attributes
  - Removal of duplicate tuples (set condition!)
  - $\Pi_B(R) = \{t[B] \mid t \in R\}, B \subseteq R(R)$

Example: Names of all customers

Customer

mem_no	last_name	first_name	address	telephone
001	Müller	Tina	...	....
007	Katz	Anna	...	...
002	Maus	Carla	...	...
011	Hinz	Fritz	...	...
023	Kunz	Anna	...	...
111	Müller	Bert	...	...
....	....	....	...	...

Π_{last_name} (Customer)

last_name
Müller
Katz
Maus
Hinz
Kunz
....

Wrong number of customers!

Relational Algebra: Basic Operations

Example: Names of all customers

Include key attribute into projection:

Customer

mem_no	last_name	first_name	address	telephone
001	Müller	Tina	...	....
007	Katz	Anna	...	...
002	Maus	Carla	...	...
011	Hinz	Fritz	...	...
023	Kunz	Anna	...	...
111	Müller	Bert	...	...
....	....	....	...	...

mem_no	last_name
001	Müller
007	Katz
002	Maus
011	Hinz
023	Kunz
111	Müller
....	....

$\Pi_{mem\_no,\, last\_name}$ (Customer)

Relational Algebra: Basic Operations

- **Selection:** $\sigma_{\langle \text{condition} \rangle}(\text{Relation})$ — Important concept
  - New relation schema = old relation schema
  - All tuples $t \in R$ that satisfy condition
  - Condition: qualificator-free Boolean predicate P
  - $\sigma_P(R) = \{ t \mid P(t) \text{ true}, t \in R\}$

Example: All customers named Anna

Customer

mem_no	last_name	first_name	address	telephone
001	Müller	Tina	...	....
007	Katz	Anna	...	...
002	Maus	Carla	...	...
011	Hinz	Fritz	...	...
023	Kunz	Anna	...	...
111	Bla	Bert	...	...
....	....	....	...	...

mem_no	last_name	first_name	address	telephone
007	Katz	Anna	...	...
023	Kunz	Anna	...	...
...	...	...	...	...

$\sigma_{first\_name=\text{"Anna"}}$ (Customer)

Relational Algebra: Selection predicates

- **Primitive Predicates**
  - Compare an attribute and a value or two attributes
  - Operands: Attribute names, constants
  - Operators: $=, \neq, \leq, \geq, <, >$
- **Boolean row predicates**
  - Combine primitive predicates by AND, OR, NOT and parenthesis
  - Boolean operators: $\wedge, \vee, \neg$
  - Operator preference and brackets as usual
- **No "second order" predicates**

  Example: "Rows which have NULL as value of all attributes"

Relational Algebra: Basic Operations

Example: All movies by Lucas from 1999 or later

Movie

id	title	category	year	director	price	length
095	Psycho	suspense	1960	Hitchcock	2.00	...
112	ET	comedy	1982	Spielberg	1.50	...
345	Star Wars I	SciFi	1999	Lucas	2.00	...
222	Psycho	suspense	1998	Van Sant	2.20	...
290	Star Wars IV	ScFi	1997	Lucas	2.00	...
100	Jaws	horror	1975	Spielberg	1.50	...
...	...	...	...	...	...	...

$\sigma_{director=\text{"Lucas"} \wedge year>=1999}$ (Customer)

id	title	category	year	director	price	length
345	Star Wars I	SciFi	1999	Lucas	2.00	...
...	...	...	...	...	...	...

Relational Algebra: Basic Operations

- **Selection properties**
  - $\sigma_P(\sigma_Q(R)) = \sigma_Q(\sigma_P(R))$
  - $\sigma_P(\sigma_P(R)) = \sigma_P(R)$
  - $\sigma_{Q \wedge P}(R) = \sigma_Q(\sigma_P(R))$
  - $\sigma_{Q \vee P}(R) = \sigma_Q(R) \cup \sigma_P(R)$
  - $\sigma_{\neg P}(R) = R - \sigma_P(R)$
- **Projection properties**
  - $X \subseteq Y \subseteq R(R) \rightarrow \Pi_X(\Pi_Y(R)) = \Pi_X(R)$
  - $X, Y \subseteq R(R) \rightarrow \Pi_X(\Pi_Y(R)) = \Pi_{X \cap Y}(R) = \Pi_Y(\Pi_X(R))$
- **Selection and Projection Property**
  - $attr(P) \subseteq X \subseteq R(R) \rightarrow \Pi_X(\sigma_P(R)) = \sigma_P(\Pi_X(R))$

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

Relational Algebra: Basic Operations

- Union
  - R and S are schema-compatible
  - R ∪ S = {t | t ∈ R ∨ t ∈ S}
- Difference
  - R and S are schema-compatible
  - R - S = {t | t ∈ R ∧ t ∉ S} = R \ S

R

A	B
a	c
b	c
a	d
b	e
a	a
b	d
a	e

\-

S

A	B
a	c
b	c
a	a

=

A	B
a	e
b	d
a	d
b	e

Relational Algebra: Basic Operations

- Cartesian Product T= R x S

Important concept

  - $T(A_1:C_1, A_2:C_2, \ldots, A_r:C_r, B_1:D_1, B_2:D_2, \ldots, B_s:D_s)$
  - R x S = {t•u | t ∈ R, u ∈ S}
  - Tupel concatenation of $t = (v_1, \ldots, v_r)$, $u = (w_1, \ldots, w_s)$
    t•u := $(v_1, \ldots, v_r, w_1, \ldots, w_s)$, degree r + s
- Rename:
  - Changes Schema, no new relation
  - Necessary for using relation or attribute more than once in one query
  - Rename relation R in S $\rho_S(R)$
  - Rename attribute B in A $\rho_{A \leftarrow B}(R)$

Relational Algebra: Basic Operations

Example: All Tapes and their corresponding movie

Tape

id	format	movie_id
0001	DVD	095
0002	DVD	112
0003	VHS	222
0004	DVD	345
0005	VHS	345
0009	VHS	345
....	....	....

Movie

id	title	category	year	director	price	length
095	Psycho	suspense	1960	Hitchcock	2.00	...
112	ET	comedy	1982	Spielberg	1.50	...
345	Star Wars I	SciFi	1999	Lucas	2.00	...
222	Psycho	suspense	1998	Van Sant	2.20	...
290	Star Wars IV	ScFi	1997	Lucas	2.00	...
100	Jaws	horror	1975	Spielberg	1.50	...
...	...	...	...	...	...	...

$\sigma_{Tape.movie\_id=Movie.id}(Tape \times Movie) =$

$\sigma_{t.movie\_id=m.id}(\rho_t(Tape) \times \rho_m(Movie))$

t.id	t.format	t.movie_id	m.id	m.title	m.category	m.year	m.director	m.price	m.length
0001	DVD	095	095	Psycho	suspense	1960	...	2.00	...
0002	DVD	112	112	ET	comedy	1982	...	1.50	...
0003	VHS	222	222	Psycho	suspense	1998	...	2.20	...
0004	DVD	345	345	Star Wars I	ScFi	1999	...	2.00	...
0005	VHS	345	345	Star Wars I	ScFi	1999	...	2.00	...
0009	VHS	345	345	Star Wars I	SciFi	1999	...	2.00	...
....	....	....	...	...	...	...	...	...	...

Relational Algebra: Additional Operations

- Intersection R ∩ S
  - R and S are schema-compatible
  - R ∩ S = R- (R-S)
- Division R ÷S
  - For queries with all-quantifier
  - $R(A_1, \ldots, A_r, B_1, \ldots, B_k)$, $S(B_1, \ldots, B_k)$
  - T= R ÷S, $T(A_1, \ldots, A_r)$
  - $R \div S = \Pi_{R-S}(R) - \Pi_{R-S}((\Pi_{R-S}(R) \times S) - R)$

R

A	B
a	c
b	c
a	**d**
b	**e**
a	a
b	**d**
a	c

÷

S

B
d
e

=

A
b

Relational Algebra: Additional Operations

- Theta- Join: $R \bowtie_\theta S$

Important concept

  - Consider $R(A_1, \ldots, A_r)$, $S(B_1, \ldots, B_s)$
  - $T = R \bowtie_\theta S = \{(A_1, \ldots A_r, B_1, \ldots B_s) \mid \theta(A_1, \ldots A_r, B_1, \ldots B_s) \text{ true}\}$
  - Boolean predicate θ formed of primitive predicates a op b, a ∈ R (R), b ∈ R (S), op ∈ {=, ≠, ≤, ≥ , <, >}
  - $R \bowtie_\theta S == \sigma_\theta(R \times S)$
- Equijoin: theta-join form $R \bowtie_{R.Ai=S.Bj} S$

Relational Algebra: Additional Operations

Example: All Tapes and their corresponding movie

$Tape \bowtie_{Tape.movie\_id=Movie.id} Movie$

tape.id	tape.format	tape.movie_id	movie.id	movie.title	movie.category	movie.year	movie.director	movie.price	movie.length
0001	DVD	095	095	Psycho	suspense	1960	...	2.00	...
0002	DVD	112	112	ET	comedy	1982	...	1.50	...
0003	VHS	222	222	Psycho	suspense	1998	...	2.20	...
0004	DVD	345	345	Star Wars I	ScFi	1999	...	2.00	...
0005	VHS	345	345	Star Wars I	ScFi	1999	...	2.00	...
0009	VHS	345	345	Star Wars I	SciFi	1999	...	2.00	...
....	....	....	...	...	...	...	...	...	...

Relational Algebra: Additional Operations

- Natural join: R ⋈ S

Important concept

- Consider $R(A_1, \dots, A_r, C_1, \dots, C_k)$, $S(B_1, \dots, B_s, C_1, \dots, C_k)$
- $dom(C_l)$ equal in R and S, $\forall\ 1 \le l \le k$
- $\forall\ 1 \le i \le r\ \forall\ 1 \le j \le s : A_i \ne B_j$
- $R \bowtie S = \Pi_{A1,\dots,Ar,R.C1,\dots,R.Ck,B1,\dots,Bs}(\sigma_{R.C1=S.C1 \wedge \dots \wedge R.Ck=S.Ck}(R \times S))$
- $R \bowtie S$ has degree r + s + k

Relational Algebra: Additional Operations

Example: All Tapes and their corresponding movie

$$Tape \bowtie (\rho_{movie\_id \leftarrow id}(Movie))$$

id	format	movie_id	title	category	year	director	price	length
0001	DVD	095	Psycho	suspense	1960	...	2.00	...
0002	DVD	112	ET	comedy	1982	...	1.50	...
0003	VHS	222	Psycho	suspense	1998	...	2.20	...
0004	DVD	345	Star Wars I	ScFi	1999	...	2.00	...
0005	VHS	345	Star Wars I	ScFi	1999	...	2.00	...
0009	VHS	345	Star Wars I	SciFi	1999	...	2.00	...
....	....	....	...	...	...	...	...	...

Relational Algebra: Additional Operations

- Inner joins (Natural join, Theta-join)
  - Non-matching tuples are lost
  - Associative and commutative
- Outer join R ⟗ S
  - Non-matching tuples included
  - Right outer join: ⟖
    (r, NULL,....,NULL) - rows for each r ∈ R which does have join row in S – according to join predicate P

R

A1	A2
a	c
b	d

⟖ $_{R.A2=S.B1}$

S

B1	B2
d	f
e	g

=

A1	A2	B1	B2
b	d	d	f
-	-	e	g

Relational Algebra: Additional Operations

- Left outer join: ⟕
  (NULL,...NULL, s) – rows for each s ∈ S which does have join row in S – according to join predicate P

R

A1	A2
a	c
b	d

⟕ $_{R.A2=S.B1}$

S

B1	B2
d	f
e	g

=

A1	A2	B1	B2
a	c	-	-
b	d	d	f

- Full outer join: ⟗
  tupels of both R and S are included

R

A1	A2
a	c
b	d

⟗ $_{R.A2=S.B1}$

S

B1	B2
d	f
e	g

=

A1	A2	B1	B2
a	c	-	-
b	d	d	f
-	-	e	g

- Left and right outer join not commutative

Relational Algebra: Additional Operations

- Semi-Joins
  - $R \ltimes S = \Pi_{R(R)}(R \bowtie S) = S \rtimes R$
  - All tuples in R that have potential join partners in S (semi-join of R with S)

R

A1	A2
a	c
b	d

⋉ $_{R.A2=S.B1}$

S

B1	B2
d	f
e	g

=

A1	A2
b	d

R

A1	A2
a	c
b	d

⋊ $_{R.A2=S.B1}$

S

B1	B2
d	f
e	g

=

B1	B2
d	f

- Semi-join not commutative

Relational Algebra: Examples

All Customers who have rented at least one science fiction film

$$\Pi_{mem\_no,\ last\_name}(customer \bowtie_{member\_no=member} rental \bowtie_{tape\_id=id} tape \bowtie_{movie\_id=movie.id} (\sigma_{category=\text{"SciFi"}}(movie)))$$

All Customers whose rented movies all have category "suspense"

$$customer - (customer \ltimes_{member=mem\_no} (rental \ltimes_{Tape\_id=id} (tape \ltimes_{movie\_id=movie.id} (\sigma_{category \ne \text{"suspense"}}(movie)))))$$

Relational Algebra: Examples

Customers that had rented all movies

$(\Pi_{mem\_no,\ movie.id}$ (customer $\bowtie_{member\_no=member}$ rental $\bowtie_{tape\_id=id}$ tape $\bowtie_{movie\_id=movie.id}$ movie)) $\div\ \Pi_{movie.id}$ (movie)

All movies no copy of which are currently on loan

movie - (movie $\ltimes_{movie\_id=movie.id}$ $(\sigma_{until=\text{"NULL"}}$ (rental) $\bowtie_{tape\_id=id}$ tape))

Relational Algebra: Examples

Number of tapes for each movie

No count!

Total receipts of each movie within the last year

No count!

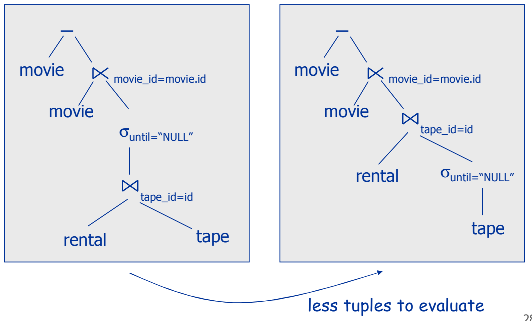
Relational Algebra: Operator Tree

Important concept

- Data structure representing RA expression
- More clearly structured
- Evaluation by recursive evaluation of the tree
- Basis for algebraic optimization
  - Implementation of Algebraic Optimization by transformation of operator tree
  - Interchange of operations according to the laws of RA e.g., $(R \bowtie \sigma_P(S))$ faster than $(\sigma_P\ (R \bowtie S))$
  - Reduction of number of tuples to evaluate
  - No change of time complexity in general

Relational Algebra: Operator Tree

Example: All movies no copy of which are currently on loan



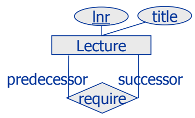
Relational Algebra: Operations

- Basic set of RA operations $\{\Pi, \sigma, \times, -, \cup\}$
  - RA expressions using different operators of RA can be expressed by the above operators only
- Missing operations in RA:
  - Transitive closure

    Example: Lecture(<u>lnr</u>, ...)

    Require(<u>preLNR</u>, succLNR)

    Query: <u>All</u> lectures required for lecture XYZ



  - Predicates on tables, e.g. arithmetic functions: SUM, AVERAGE, MAXIMUM, MINIMUM, or COUNT, grouping

Relational Algebra: Summary

- Relational Algebra:
  - Formal language for handling data in relational model
  - Procedural language, *how* to retrieve data
  - No practical relevance for querying DB
  - Formal basis for query optimization
- Important terms & concepts
  - Union $R \cup S$
  - Difference $R - S$
  - Selection $\sigma_{<predicate>}(R)$
  - Projection $\Pi_{<attribute\ list>}(R)$
  - Cartesian Product $R \times S$
  - Joins $R \bowtie_{<predicate>} S$
  - Operator tree

Relational Calculus: Languages

- Non-procedural, declarative query language
- Two types of languages
  - Tuple calculus
    Variables in expressions represent a row of a relation (tuple variable)

    Example: {c.last_name, c.first_name | c $\in$ Costumer}

  - Domain calculus
    Variables represent domain values of attributes of a relation of the DB (domain variables)

    Example: {[l,f] | $\exists$m,a,t([m,l,f,a,t] $\in$ Customer)}

Tuple Calculus: Introduction

Important concept

- Queries in tuple calculus
  - Form: {t | F(t)} with tupel variable t
  - F is formula
  - t is the only free variable of formula F
  - Answer of query: set of tuples t from DB with F(t) = TRUE
- Example:
  All customers named Anna
  {c | c $\in$ Costumer $\wedge$ c. first_name = "Anna"}
- Free variable: no existence- or all-quantor ($\exists$, $\forall$)
- Formulae made of atoms
- Atoms evaluate to TRUE or FALSE

Tuple Calculus: Atom Forms

- t$\in$ R
  - with relation name R and tuple variable t
  - Alternative notation: R(t)
  - Example: c $\in$ Costumer
- (t.A operator s.B)
  - t,s tuple variables
  - A,B attribute names of of relations on which t, s ranges
  - operator $\in$ {=, $\leq$, $\geq$, $\neq$, <, >}
  - Example: c.member_no=r.member
- (t.A operator c)
  - c constant value
  - Example: c. first_name = "Anna"

Relational Calculus: Atoms and Formulae

1. Every atom is a formula.
2. If F_1 is a formula so are $\neg F_1$ and (F_1)
3. If F_1, F_2 are formulae so are $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \Rightarrow F_2$
4. If F with t free is formula so are $\exists$t (F), $\forall$t (F)
5. Nothing else is a formula

Tuple Calculus: Examples

- Formula properties
  - $\forall$t $\in$ R(F(t)) = $\neg$($\exists$ t $\in$R($\neg$F(t))
  - $\exists$t $\in$ R(F(t)) = $\neg$($\forall$ t $\in$R($\neg$F(t))
- Tuple Calculus Examples:
  - Names of all customers
    {c.last_name, c.first_name | c $\in$ Costumer}
  - All customers named Anna
    {c | c $\in$ Costumer $\wedge$ c. first_name = „Anna"}
  - All movies by George Lucas from 1999 or later
    {m.id, m.title | m $\in$ Movie $\wedge$ m.director="Lucas"
    $\wedge$ m.year>=1999 }

Tuple Calculus: Examples

All Tapes and their corresponding movie

{t.id, m.title | t $\in$ Tape $\wedge$ m$\in$Movie $\wedge$ t. movie_id = m.id}

All Customers who have rented at least one science fiction film

{c | c $\in$ Costumer
$\wedge$ $\exists$ r $\in$ Rental(c.member_no=r.member
$\wedge$ $\exists$ t $\in$ Tape(r.tape_id=t.id
$\wedge$ $\exists$ m $\in$ Movie(t.movie_id=m.id
$\wedge$ m.category="SciFi")))}

Tuple Calculus: Examples

All Customers whose rented movies all have category "suspense"

$\{c \mid c \in \text{Costumer}$
$\wedge \exists\, r \in \text{Rental}(\text{c.member\_no=r.member}$
$\wedge \exists\, t \in \text{Tape}(\text{r.tape\_id=t.id}$
$\wedge \forall m \in \text{Movie}(\text{t.movie\_id=m.id} \Rightarrow$
$\text{m.category="suspense"})))\}$

$\{c \mid c \in \text{Costumer}$
$\wedge \exists\, r \in \text{Rental}(\text{c.member\_no=r.member}$
$\wedge \exists\, t \in \text{Tape}(\text{r.tape\_id=t.id}$
$\wedge \neg\exists m \in \text{Movie}(\text{t.movie\_id = m.id} \Rightarrow$
$\text{m.category} \neq \text{"suspense"})))\}$

Tuple Calculus: Examples

Customers that had rented all movies

$\{c \mid c \in \text{Customer}$
$\wedge \forall m \in \text{Movie}($
$\exists\, t \in \text{Tape}(\text{t.movie\_id=m.id}$
$\wedge \exists\, r \in \text{Rental}(\text{r.tape\_id=t.id}$
$\wedge \text{c.member\_no=r.member}))\}$

All movies no copy of which are currently on loan

$\{m \mid m \in \text{Movie}$
$\wedge \forall\, t \in \text{Tape}(\text{t.movie\_id=m.id}$
$\Rightarrow \neg\exists\, r \in \text{Rental}(\text{r.tape\_id=t.id}$
$\wedge \text{r.until='NULL'}))\}$

Tuple Calculus: Examples

Number of tapes for each movie

No count!

Total receipts of each movie within the last year

No count!

Tuple Calculus vs Relational Algebra

- Selection
  $\sigma_{\langle predicate \rangle}(R)$ equivalent to $\{ r \mid r \in R \wedge \langle predicate \rangle \}$
- Projection, cross product
  $\Pi_{R.a,\, S.b}(R \times S)$ equivalent to $\{ r.a, s.b \mid r \in R \wedge s \in S \}$
- Join
  $R \bowtie_P S$ equivalent to $\{ r \mid r \in R \wedge s \in S \wedge P \}$
- Union
  $R \cup S$ equivalent to $\{ t \mid t \in R \vee t \in S \}$
- Difference
  $R - S$ equivalent to $\{ t \mid t \in R \wedge \neg t \in S \}$

Tuple Calculus: Safe expression

Important concept

- Solution set for $\{ t \mid \neg t \in R \}$?
  - All tuples NOT belonging to R, infinite set?
- Formula domain:
  - all attribute data of referenced relations in DB, and
  - constants of the formula
- Safe expression:
  tuple calculus expression is safe if result is subset of domain
  - Idea: safe if all free tuple variables restricted in F
  - Example: $\{x \mid x \in T \wedge \neg x \in R \}$ safe

Tuple Calculus: Practical Use

- Tuple calculus basis for DB language QUEL
  - In 70s used in Ingres (University of Berkeley)
  - Commercial INGRES now SQL
- Examples:
  - All customers named Anna
    RANGE of c is Customer
    RETRIEVE (c.mem_no, c.last_name, c.first_name)
    WHERE c. first_name = "Anna"
  - All movies by George Lucas from 1999 or later
    RANGE of m is Movie
    RETRIEVE (m.title)
    WHERE m.director="Lucas" AND year>=1999

Domain Calculus: Introduction

- Queries in domain calculus
  - Form: {[a,b,c] | F([a,b,c])} with a,b,c domain variables
  - F is formula
  - a, b, c are the only free variables of formula F
  - Answer of query: set of tuples t from DB with F(t) = TRUE
- Example:

  All customers named Anna

  {[m,f,l] | ∃a,t ([m,f,l,a,t] ∈ Costumer ∧f= "Anna")}
- Domain variables represent sets of possible attribute values (domains)
- Formulae made of atoms
- Atoms evaluate to TRUE or FALSE

Domain Calculus: Atom Forms

- $[a_1,\ldots,a_n] \in R$
  - with relation R of grade n
  - Domain variables $a_1,\ldots,a_n$ according to order in schema of R
  - Example: [m,f,l,a,t] ∈ Costumer
- (a_i operator a_j)
  - operator ∈ {=, ≤, ≥, ≠, <, >}
  - a_i, a_j domain variables
  - Example: mn = r
- (a_i operator c)
  - c constant value
  - Example: f= "Anna"

Domain Calculus: Examples

Names of all customers

{[l,f] | ∃m,a,t ([m,f,l,a,t] ∈ Costumer)}

All customers named Anna

{[m,f,l] | ∃a,t ([m,f,l,a,t] ∈ Costumer ∧f= "Anna")}

All movies by George Lucas from 1999 or later

{[i,t] | ∃c,y,d,p,l ([i,t,c,y,d,p,l]∈ Movie ∧
d="Lucas"∧ y>=1999) }

All Tapes and their corresponding movie

{[ti,t] | ∃tf,mi ([ti,tf,mi]∈ Tape ∧
∃i,c,y,d,p,l ([i,t,c,y,d,p,l]∈ Movie ∧ mi = i))}

Domain Calculus: Examples

All Customers who have rented at least one science fiction film

{[m,f,l] | ∃a,t ([m,f,l,a,t] ∈ Costumer
∧ ∃ rti,rm,rf,ru ([rti,rm,rf,ru] ∈Rental ∧ m=rm
∧ ∃ ti,tf,tmi ([ti,tf,tmi] ∈ Tape ∧ rti=ti
∧ ∃ mi,mt,mc,my,md,mp,ml ([mi,mt,mc,my,md,mp,ml]∈ Movie
∧ tmi=mi ∧ mc="SciFi")))))}

All Customers whose rented movies all have category "suspense"

{[m,f,l] | ∃a,t ([m,f,l,a,t] ∈ Costumer
∧ ∃ rti,rm,rf,ru ([rti,rm,rf,ru] ∈Rental ∧ m=rm
∧ ∃ ti,tf,tmi ([ti,tf,tmi] ∈ Tape ∧ rti=ti
∧ ∀ mi,mt,mc,my,md,mp,ml ([mi,mt,mc,my,md,mp,ml]∈
Movie
∧ tmi=mi ⇒ mc="suspense")))))}

Domain Calculus: Examples

Customers that had rented all movies

{[m,f,l] | ∃a,t ([m,f,l,a,t] ∈ Costumer
∧ ∀ mi,mt,mc,my,md,mp,ml ([mi,mt,mc,my,md,mp,ml]∈ Movie
∧ ∃ ti,tf,tmi ([ti,tf,tmi] ∈ Tape
∧ tmi=mi
∧ ∃ rti,rm,rf,ru ([rti,rm,rf,ru] ∈Rental ∧ m=rm
∧ rti=ti))))}

All movies no copy of which are currently on loan

{[i,t,d] | ∃ c,y,p,l ([i,t,c,y,d,p,l]∈ Movie
∧ ∀ ti,tf,tmi ([ti,tf,tmi] ∈ Tape ∧ tmi=i ⇒
¬∃ rti,rm,rf,ru ([rti,rm,rf,ru] ∈Rental ∧ rti=ti)))}

Domain Calculus: Examples

Number of tapes for each movie

No count!

Total receipts of each movie within the last year

No count!

Domain Calculus

- Safe expression $\{[a_1,\dots,a_n] \mid F(a_1,\dots,a_n)\}$:
  1. Constants $c_i (1 \le i \le n)$ in domain of F if $[c_1,\dots,c_n]$ in solution
  2. For all $\exists a_1(F(a_1))$ F true only for elements of domain of F
  3. For all $\forall a_1(F(a_1))$ F true only iff true for all elements of domain of F

  2.+3. necessary since domain variables not bound to relations

- Formula domain:
  - all attribute data of referenced relations in DB, and
  - constants of the formula

Domain Calculus: Practical Use

- Domain calculus basis for DB language QBE
  - QBE = query by example
  - One of the first graphical query languages
  - Interface option for DB2 (IBM)
  - templates of relations on screen
  - Users fill in constants (...), examples (_...), output (P. ...)
  - Example: All movies by George Lucas from 1999 or later

Movie	id	title	category	year	director	price	length
	P.1	P.bla		≥1999	Lucas		

each column in template as implicit domain variable

$\{[i,t] \mid \exists c,y,d,p,l\ ([i,t,c,y,d,p,l] \in \text{Movie} \wedge d=\text{"Lucas"} \wedge y>=1999)\}$

Relational Calculus: Summary

- Relational Calculus:
  - Formal languages for handling data in relational model
  - Declarative language, *which* data to retrieve
  - Basis for QUEL, QBE, SQL

- Important terms & concepts
  - Tuple Calculus
  - Domain Calculus
  - Bound and free variables
  - Safe expression
  - Relational Completeness

Relational Languages: Conclusion

Important concept

- Relational completeness:
  - query language for relational model is complete if at least as expressive as relational algebra
  - base line for DB query languages: every query language should be as expressive as relational algebra

- Equivalent expressivenesses:
  - Relational Algebra
  - Tuple calculus restricted to safe expressions
  - Domain calculus restricted to safe expressions
  - Proof using induction:
    RA expressions → TC expressions →
    DC expressions → RA expressions