Relational Languages: Relational Algebra & Relational calculus

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

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
      \[ \exists \text{ Permutation } \phi \text{ of indices } \{1, \ldots, r\} : \forall i, 1 \leq i \leq r : C_i = D_{\phi(i)} \]

Relational Languages: Database Design
- Minimalworld
- Requirements analysis
- Database Requirements
- Conceptual Design
- Logical Schema Design
- In data model of specific DBMS
- Physical Schema Design
- Internal schema (for same DBMS)

Relational Algebra: Basic Operations
- Projection
- Selection
- Union
- Difference
- Cartesian Project
- (Renaming)
Relational Algebra: Basic Operations

Projection: $\Pi_{	ext{List of attributes}}(\text{Relation})$
- Reduction of relation schema to selected attributes
- Reduction of relation tuple to respective attributes
- Removal of duplicate tuples (set condition)
- $\Pi_B(R) = \{ \{ B \} \mid t \in R \}$, $B \subseteq R$(R)

Example: Names of all customers

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Anna</td>
<td>Müller</td>
</tr>
<tr>
<td>2</td>
<td>Carla</td>
<td>Maus</td>
</tr>
<tr>
<td>3</td>
<td>Fritz</td>
<td>Kunz</td>
</tr>
<tr>
<td>4</td>
<td>Bert</td>
<td>Hinz</td>
</tr>
<tr>
<td>5</td>
<td>Tina</td>
<td>Katz</td>
</tr>
</tbody>
</table>

Wrong number of customers!

Selection and Projection

Example: All customers named Anna

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Anna</td>
<td>Müller</td>
</tr>
<tr>
<td>2</td>
<td>Carla</td>
<td>Maus</td>
</tr>
<tr>
<td>3</td>
<td>Fritz</td>
<td>Kunz</td>
</tr>
<tr>
<td>4</td>
<td>Bert</td>
<td>Hinz</td>
</tr>
<tr>
<td>5</td>
<td>Tina</td>
<td>Katz</td>
</tr>
</tbody>
</table>

Relational Algebra: Selection predicates

Primitive Predicates
- Compare an attribute and a value or two attributes
- Operands: Attribute names, constants
- Operators: $\neg$, $=$, $\neq$, $<$, $>$

Boolean row predicates
- Combine primitive predicates by AND, OR, NOT and parenthesis
- Boolean operators: $\land$, $\lor$, $\neg$
- Operator preference and brackets as usual

No "second order" predicates
Example: "Rows which have NULL as value of all attributes"
Relational Algebra: Basic Operations

- Union
  - $R$ and $S$ are schema-compatible
  - $R \cup S = \{ t \mid t \in R \lor t \in S \}$

- Difference
  - $R$ and $S$ are schema-compatible
  - $R - S = \{ t \mid t \in R \land t \notin S \}$

$$
\begin{array}{c|c|c|c|c|c|c}
\text{R} & \text{S} & \text{DIFFERENCE} \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
$$

Relational Algebra: Additional Operations

- Intersection $R \cap S$
  - $R$ and $S$ are schema-compatible
  - $R \cap S = R - (R - S)$

- Division $R \div S$
  - For queries with all-quantifier
    - $R(A_1, \ldots, A_i, B_1, \ldots, B_j)$
    - $T = R \div S, \text{relation}s \in (R \div S)$
  - $R \div S = \{ t \in R \mid \forall s \in S (t[A_1, \ldots, A_i] = s[A_1, \ldots, A_i]) \}$

Relational Algebra: Basic Operations

- Cartesian Product $T = R \times S$
  - $T(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$
  - $R \times S = \{ t \mid t \in R \land u \in S \}$
  - Tuple concatenation of $t = (v_1, \ldots, v_n)$, $u = (w_1, \ldots, w_m)$
    - $t \cup u = (v_1, \ldots, v_n, w_1, \ldots, w_m)$, 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 $S \rho R(R)$
  - Rename attribute $B$ in $A \rho_{A\rightarrow B}(R)$

Example: All Tapes and their corresponding movie

<table>
<thead>
<tr>
<th>Tape</th>
<th>Movie</th>
<th>Format</th>
<th>Box _</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVD</td>
<td>Psycho</td>
<td>DVD</td>
<td>1986</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>Star Wars</td>
<td>VHS</td>
<td>1977</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>Star Wars</td>
<td>VHS</td>
<td>1977</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>Star Wars</td>
<td>Dvd</td>
<td>1999</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>Star Wars</td>
<td>Dvd</td>
<td>1999</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Example: All Tapes and their corresponding movie

<table>
<thead>
<tr>
<th>Tape</th>
<th>b4</th>
<th>Movie</th>
<th>Tape.movie_id</th>
<th>Movie.id</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVD</td>
<td>345</td>
<td>Psycho</td>
<td>1986</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>346</td>
<td>Star Wars</td>
<td>1977</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>347</td>
<td>Star Wars</td>
<td>1999</td>
<td>2.0</td>
</tr>
<tr>
<td>VHS</td>
<td>348</td>
<td>Star Wars</td>
<td>1999</td>
<td>2.0</td>
</tr>
</tbody>
</table>
Relational Algebra: Additional Operations

- **Natural join:** $R \bowtie S$
  - Consider $R(A_1, \ldots, A_i, C_i, \ldots, C_n)$, $S(B_1, \ldots, B_i, C_i, \ldots, C_n)$
  - $\text{dom}(C_i)$ equal in $R$ and $S$, $\forall$ $i$ such that $A_i = B_j$
  - $R \bowtie S = \{ r \mid r[A_1\ldots A_i] \in R \land r[B_{i+1}\ldots B_n] \in S \}$
  - $R \bowtie S$ has degree $r \times s = k$

Relational Algebra: Additional Operations

- **Examples:**
  - All Tapes and their corresponding movie
    - Tape $\bowtie$ $\pi_{\text{tapedate}}(\text{Movie})$

<table>
<thead>
<tr>
<th>Tape_id</th>
<th>Title</th>
<th>Category</th>
<th>Year</th>
<th>Tape_price</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Psycho</td>
<td>Suspense</td>
<td>1997</td>
<td>1.20</td>
</tr>
<tr>
<td>002</td>
<td>Psycho</td>
<td>Suspense</td>
<td>1998</td>
<td>1.20</td>
</tr>
<tr>
<td>003</td>
<td>Star Wars</td>
<td>SciFi</td>
<td>1977</td>
<td>2.00</td>
</tr>
<tr>
<td>004</td>
<td>Star Wars</td>
<td>SciFi</td>
<td>1978</td>
<td>2.00</td>
</tr>
<tr>
<td>005</td>
<td>Star Wars</td>
<td>SciFi</td>
<td>1979</td>
<td>2.00</td>
</tr>
</tbody>
</table>

- **Semi-Joins**
  - $R \bowtie S = \{ \alpha \mid \exists \beta \in S : \alpha \bowtie \beta \}$
  - All tuples in $R$ that have potential join partners in $S$

Relational Algebra: Additional Operations

- **Join:** $R \bowtie S$
  - Rows $r \in R$ and $s \in S$ with $r[A_1\ldots A_i] = s[B_{i+1}\ldots B_n]$
  - $R \bowtie S = \{ (r, s) \mid r[A_1\ldots A_i] = s[B_{i+1}\ldots B_n] \}$

- **Left outer join:** $R \bowtie S$
  - Rows for each $s \in S$ which does not have a join row in $R$

- **Full outer join:** $R \bowtie S$
  - Tuples of both $R$ and $S$ are included

- **Left and right outer join not commutative**
Relational Algebra: Examples

Customers that had rented all movies

\[ \{\text{customer number, movie} : \text{rental} \} \cap \{\text{movie} : \text{tape} \} \] \[ \cap \{\text{movie} \} \] \[ \cap \{\text{tape} \} \]

All movies no copy of which are currently on loan

\[ \text{movie} \cap \{\text{movie} : \text{tape} \} \sigma_{\text{until} = \text{NULL}} \]

Relational Algebra: Operator Tree

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

Relational Algebra: Operations

- Basic set of RA operations (\{\cup, \sigma, \pi, -, \},) RA expressions using different operators of RA can be expressed by the above operators only
- Missing operations in RA: Transitive closure
  - Example: Lecture(R, ..., ) Requires(preLNR, succLNR) predecessor successor
  
    Query: All lectures required for lecture XYZ

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

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_{\text{predicate}}(R)
  - Projection \Pi_{\text{attribute list}}(R)
  - Cartesian Product R \times S
  - Joins R \bowtie_{\text{predicate}} S
  - 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_m(S)) faster than (\sigma_{(R \bowtie_m S)})
- Reduction of number of tuples to evaluate
- No change of time complexity in general
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: \( (c, \text{last\_name}, c, \text{first\_name} | c \in \text{Customer}) \)
  - Domain calculus
    - Variables represent domain values of attributes of a relation of the DB (domain variables)
    - Example: \( (l, f) | \exists m, a, t (m, l, f, a, t \in \text{Customer}) \)

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 \( F_1 \lor F_2, F_1 \land F_2 \)
4. If \( F \) with \( t \) free is formula so are \( \exists t (F), \forall t (F) \)
5. Nothing else is a formula

Tuple Calculus: Introduction

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

Tuple Calculus: Atom Forms

- \( t \in R \)
  - with relation name \( R \) and tuple variable \( t \)
  - Alternative notation: \( R(t) \)
  - Example: \( c \in \text{Customer} \)
- \( (t.A \text{ operator } s.B) \)
  - \( t, s \) tuple variables
  - \( A, B \) attributes names of of relations on which \( t, s \) ranges
  - Operator \( \in \{ =, \leq, \geq, \lt, \gt \} \)
  - Example: \( c, \text{member\_no}=r, \text{member} \)
- \( (t.A \text{ operator } c) \)
  - \( c \) constant value
  - Example: \( c, \text{first\_name} = \text{"Anna"} \)

Tuple Calculus: Examples

- Formula properties
  - \( \forall t (R(t)) = \neg \exists t (\neg R(t)) \)
  - \( \exists t (R(t)) = \neg \forall t (\neg R(t)) \)

- Tuple Calculus Examples:
  - Names of all customers
    \( \{ c, \text{last\_name}, c, \text{first\_name} | c \in \text{Customer} \} \)
  - All customers named Anna
    \( \{ c | c \in \text{Customer} \land c, \text{first\_name} = \text{"Anna"} \} \)
  - All movies by George Lucas from 1999 or later
    \( \{ m, \text{id}, m, \text{title} | m \in \text{Movie} \land m, \text{director}=\text{"Lucas"} \land m, \text{year} \geq 1999 \} \)

- All Tapes and their corresponding movie
  \( \{ t, \text{id}, m, \text{title} | t \in \text{Tape} \land m \in \text{Movie} \land t, \text{movie\_id} = m, \text{id} \} \)

- All Customers who have rented at least one science Fiction film
  \( \{ c | c \in \text{Customer} \land \exists t \in \text{Rental} (c, \text{member\_no}=t, \text{member} \land \exists m \in \text{Movie} (t, \text{movie\_id}=m, \text{id} \land m, \text{category} = \text{"SciFi"}))) \} \)
Tuple Calculus: Examples

All customers whose rented movies all have category “suspense”

\{c \mid c \in \text{Costumer} \land \exists r \in \text{Rental}(c.\text{member_no}=r.\text{member} \\
\land \exists t \in \text{Tape}(r.\text{tape_id}=t.\text{id} \\
\land \forall m \in \text{Movie}(t.\text{movie_id}=m.\text{id} \Rightarrow m.\text{category} = “suspense” )\)\}

Customers that had rented all movies

\{c \mid c \in \text{Customer} \\
\land \forall m \in \text{Movie(} \\
\land \exists t \in \text{Tape(t.movie_id}=m.\text{id} \\
\land \exists r \in \text{Rental(r.tape_id}=t.\text{id} \\
\land c.\text{member_no}=r.\text{member})))\}

All movies no copy of which are currently on loan

\{m \mid m \in \text{Movie} \\
\land \forall t \in \text{Tape(t.movie_id}=m.\text{id} \\
\land \exists r \in \text{Rental(r.tape_id}=t.\text{id} \\
\land r.\text{until}=\text{NULL})\}

Tuple Calculus vs Relational Algebra

- Selection
  \sigma_{\text{predicate}}(R) \text{ equivalent to } \{ r \mid r \in R \land \text{predicate}\}

- Projection, cross product
  \Pi_{a,b} (R \times S) \text{ equivalent to } \{ r.a, s.b \mid r \in R \land s \in S \}

- Join
  \text{Join } R \ast S \text{ equivalent to } \{ t \mid t \in R \land t \in S \land P\}

- Union
  \text{Union } R \lor S \text{ equivalent to } \{ t \mid t \in R \lor t \in S\}

- Difference
  \text{Difference } R - S \text{ equivalent to } \{ t \mid t \in R \land t \notin S\}

Tuple Calculus: Safe expression

- 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 \land \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: $\langle a,b,c \rangle \cdot F(a,b,c)$ 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 $F(t) = \text{TRUE}$

- Example:
  - All customers named Anna
    $\langle m,f,l \rangle \mid \exists a,t \ (\langle m,f,l,a,t \rangle \in \text{Costumer} \land f = \text{"Anna"})$

- Domain variables represent sets of possible attribute values (domains)
- Formulae made of atoms
- Atoms evaluate to TRUE or FALSE

Domain Calculus: Atom Forms

- $\langle a_1, \ldots, a_n \rangle \in R$
  - with relation $R$ of grade $n$
  - Domain variables $a_1, \ldots, a_n$ according to order in schema of $R$
- Example: $\langle m,f,a,t \rangle \in \text{Costumer}$

- $\langle a, \text{ operator } a \rangle$
  - $\text{ operator } \in \{=, \leq, \geq, \neq, <, >\}$
  - $a, a$ domain variables
- Example: $\text{m} = r$

- $\langle a, \text{ operator } c \rangle$
  - $c$ constant value
- Example: $f = \text{"Anna"}$

Domain Calculus: Examples

- Names of all customers
  $\langle i,t \mid \exists m,a,t \ (\langle m,f,a,t \rangle \in \text{Costumer}) \rangle$

- All customers named Anna
  $\langle m,f,a,t \rangle \mid \exists a,t \ (\langle m,f,a,t \rangle \in \text{Costumer} \land f = \text{"Anna"})$

- Total receipts of each movie within the last year
  No count!
Domain Calculus

- **Safe expression** \([a_1, \ldots, a_n] \mid F(a_1, \ldots, a_n)\):
  1. Constants \(c_1, \ldots, c_k\) in domain of \(F\) if \(c_1, \ldots, c_k\) in solution
  2. For all \(\exists a_i(F(a_i))\) \(F\) true only for elements of domain of \(F\)
  3. For all \(\forall a_i(F(a_i))\) \(F\) true only if \(F\) 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

Relational Languages: Conclusion

- **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 expressiveness**:
  - Relational Algebra
  - Tuple calculus restricted to safe expressions
  - Domain calculus restricted to safe expressions

  - Proof using induction:
    - RA expressions \(\rightarrow\) TC expressions \(\rightarrow\)
    - DC expressions \(\rightarrow\) RA expressions

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

<table>
<thead>
<tr>
<th>Movie</th>
<th>id</th>
<th>Title</th>
<th>Category</th>
<th>year</th>
<th>Director</th>
<th>since</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.I</td>
<td>P. Mia</td>
<td>1999</td>
<td>Lucas</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  each column in template as implicit domain variable

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