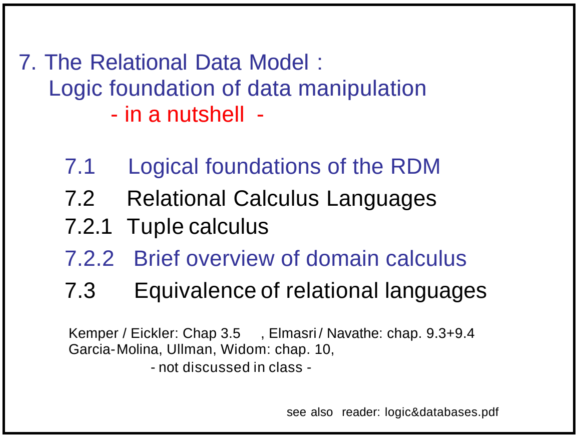


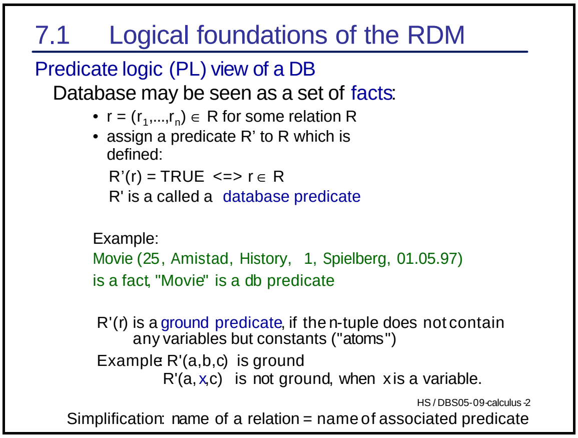
7. The Relational Data Model : Logic foundation of data manipulation

- in a nutshell -

7.1 Logical foundations of the RDM
7.2 Relational Calculus Languages
7.2.1 Tuple calculus
7.2.2 Brief overview of domain calculus
7.3 Equivalence of relational languages

Kemper / Eickler: Chap 3.5 , Elmasri / Navathe: chap. 9.3+9.4
Garcia-Molina, Ullman, Widom: chap. 10,
- not discussed in class -

see also reader: logic&databases.pdf



7.1 Logical foundations of the RDM

Predicate logic (PL) view of a DB

Database may be seen as a set of facts:

- $r = (r_1,\ldots,r_n) \in R$ for some relation R
- assign a predicate R’ to R which is defined:

  R’(r) = TRUE <=> $r \in R$

  R' is a called a database predicate

Example:

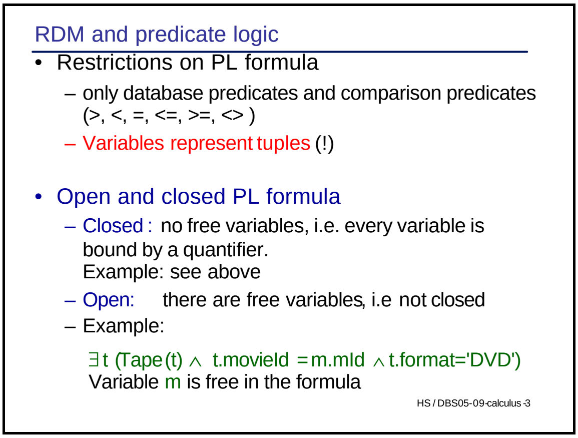
Movie (25, Amistad, History, 1, Spielberg, 01.05.97)

is a fact, "Movie" is a db predicate

R'(r) is a ground predicate, if the n-tuple does not contain any variables but constants ("atoms")

Example: R'(a,b,c) is ground
R'(a, x,c) is not ground, when x is a variable.

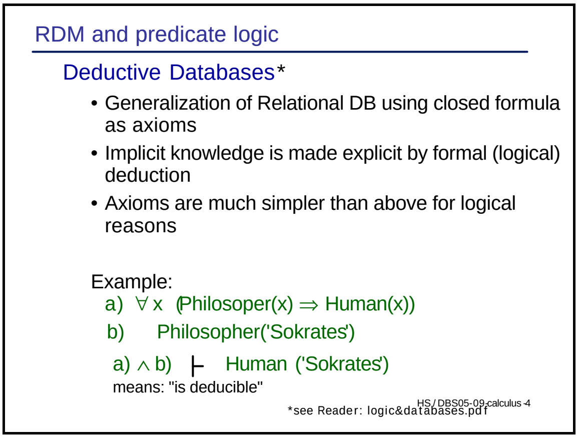
Simplification: name of a relation = name of associated predicate



RDM and predicate logic

- Restrictions on PL formula
  - only database predicates and comparison predicates (>, <, =, <=, >=, <>)
  - Variables represent tuples (!)
- Open and closed PL formula
  - Closed : no free variables, i.e. every variable is bound by a quantifier.
    Example: see above
  - Open: there are free variables, i.e not closed
  - Example:

    $\exists t\ (\text{Tape}(t) \wedge t.\text{movieId} = m.\text{mId} \wedge t.\text{format}='\text{DVD}')$
    Variable m is free in the formula



RDM and predicate logic

Deductive Databases*

- Generalization of Relational DB using closed formula as axioms
- Implicit knowledge is made explicit by formal (logical) deduction
- Axioms are much simpler than above for logical reasons

Example:

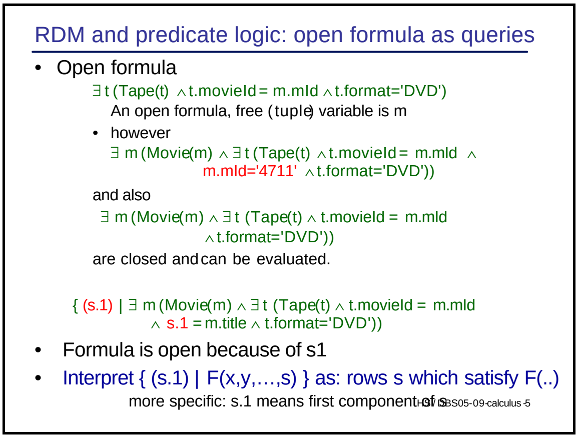
a) $\forall x\ (\text{Philosoper}(x) \Rightarrow \text{Human}(x))$

b) Philosopher('Sokrates')

a) $\wedge$ b) $\vdash$ Human ('Sokrates')

means: "is deducible"

*see Reader: logic&databases.pdf



RDM and predicate logic: open formula as queries

- Open formula

  $\exists t\ (\text{Tape}(t) \wedge t.\text{movieId} = m.\text{mId} \wedge t.\text{format}='\text{DVD}')$

  An open formula, free (tuple) variable is m

  - however

    $\exists m\ (\text{Movie}(m) \wedge \exists t\ (\text{Tape}(t) \wedge t.\text{movieId} = m.\text{mId} \wedge m.\text{mId}='4711' \wedge t.\text{format}='\text{DVD}'))$

  and also

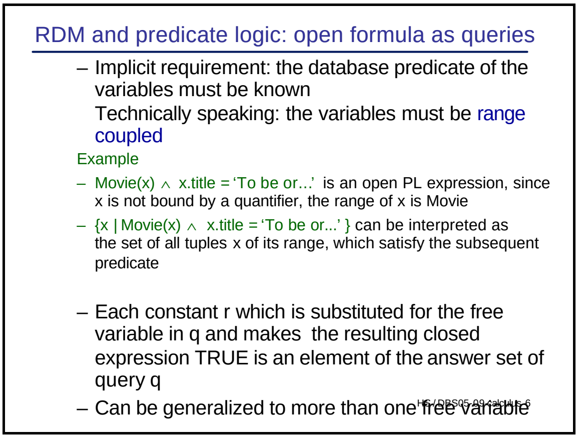
  $\exists m\ (\text{Movie}(m) \wedge \exists t\ (\text{Tape}(t) \wedge t.\text{movieId} = m.\text{mId} \wedge t.\text{format}='\text{DVD}'))$

  are closed and can be evaluated.

$\{ (s.1) \mid \exists m\ (\text{Movie}(m) \wedge \exists t\ (\text{Tape}(t) \wedge t.\text{movieId} = m.\text{mId} \wedge s.1 = m.\text{title} \wedge t.\text{format}='\text{DVD}'))$

- Formula is open because of s1
- Interpret { (s.1) | F(x,y,…,s) } as: rows s which satisfy F(..)

  more specific: s.1 means first component of s



RDM and predicate logic: open formula as queries

- Implicit requirement: the database predicate of the variables must be known

  Technically speaking: the variables must be range coupled

Example

- Movie(x) $\wedge$ x.title = ‘To be or...’ is an open PL expression, since x is not bound by a quantifier, the range of x is Movie
- {x | Movie(x) $\wedge$ x.title = ‘To be or...’ } can be interpreted as the set of all tuples x of its range, which satisfy the subsequent predicate

- Each constant r which is substituted for the free variable in q and makes the resulting closed expression TRUE is an element of the answer set of query q
- Can be generalized to more than one free variable

7.2 Calculus Languages

- Predicate logic as a query language

Called "calculus" languages for historical reasons
Two types of languages

Domain calculus
All variables represent typed values (or domains) of the relations of the DB (domain variables)

Tuple calculus
The variables in expressions represent a row of a relation (tuple variable)

7.2.1 Tuple Calculus

- Tuple Calculus language
  - Defined over
    - predicates R, S, T,... which correspond to database relations
    - set A of attribute names { a, b,.. }
    - values from the domains of A
    - tuple variables r,s,t,...
  - A tuple calculus expression (with one free variable) has the form
    $\{s \mid F(s)\}$
    where s is a tuple variable and F(s) a formula in which s occurs free

Example:
$\{(s.1,s.2) \mid \exists m\ (Movie(m)$
$\wedge\ m.title = s.1 \wedge \text{and } m.year > \text{'1992'} \wedge m.year=s.2)\}$
i.e. all movie and year of production titles produced after 1999

Relational Calculus Tuple calculus

- s is called the target list
- s is not range coupled and its components are denoted
  s.1, s.2,….s.k, if there are k output components
- s.i has to be connected in F to some range coupled variables

Example:
$\{(s.1,s.2) \mid \exists m\ (Movie(m)$
$\wedge\ m.title = s.1 \wedge \text{and } m.year> \text{'1992'} \wedge m.year=s.2)\}$
i.e. all movie and year of production titles produced after 1999

Tuple calculus

- Simplification
  $\{(s.1,s.2) \mid \exists m\ (Movie(m)$
  $\wedge\ m.title = s.1 \wedge \text{and } m.year> \text{'1992'} \wedge m.year=s.2)\}$
- Substitute "logical variables" s.i by row variables
  $\{(m.title,m.year) \mid \cancel{\exists m}\ (Movie(m)\ \wedge \text{and } m.year> \text{'1992'}\ )\}$
- Eliminate existential quantifiers : "all free variables are existentially quantified" … at least those in target list.
  $\{(m.title,m.year) \mid\ Movie(m) \wedge \text{and } m.year> \text{'1992'}\ \}$

Range coupling of (row) variable m

Tuple calculus and relational algebra

Tuple calculus expression for algebra operators

- Projection, cross product
  $\pi_{a,b}\ (R \times S) \equiv$
  $\{\ (x.a,\ y.b) \mid R(x) \wedge S(y)\}$
- Join $R \underset{P}{\bowtie} S$
  $\{r.e,\dots,t.k \mid R(r) \wedge S(t) \wedge P\}$
  $\{(t.t\_id,\ t.m\_id) \mid\ Tape(t) \wedge Rental(r) \wedge r.tape\_id = t.t\_id$
  $\wedge\ t.format = \text{'DVD'}\}$
- Selection
  e.g. $\sigma_{(s.a = v \vee s.a = w)\ \wedge\ s.b = s.c}\ (R)$
  $\{(s.a,\dots,s.k) \mid R(s) \wedge\ (s.a = v \vee s.a = w)\ \wedge\ s.b = s.c\ \}$

More examples in the class

Tuple calculus

Examples

Movies (title) all copies of which are on loan
$\{m.title \mid Movie(m) \wedge \forall t\ (Tape(t) \wedge m.m\_id = t.m\_Id$
$\Rightarrow \exists x\ (\ Rental(x) \wedge x.t\_id = t.t\_id))\ \}$

Find movietitles available in all formats (in the DB)
"… for all formats there exists a tape with this format and this movie"
$\{m.title \mid Movie(m) \wedge \forall f\ (Format(f)\ \Rightarrow\ \exists x (Tape(x) \wedge$
$f.format = x.format \wedge x.m\_id = m.m\_id))\}$

Customer names and titles of movies, they have lent
$\{c.name,\ m.title \mid Customer(c) \wedge Movie(m) \wedge$
$\exists t\ (\exists r\ (\ (Tape(t) \wedge Rental(r) \wedge t.id = r.t\_id \wedge r.mem\_no =$
$c.mem\_no \wedge t.m\_id = m.m\_id)))\ \}$

Tuple calculus used in Ingres / UCB as data handling language QUEL

Tuple calculus

… example: more than one variable for the same relation

Find actors who played together in the same movie.

'There exists an actor and another actor and two different "starring" entries, such that the movie-attributes of both entries are the same and the actor attribute values are the foreign key values for these two actors '

$$\{(a1.stage\_name, a2.stage\_name) \mid Actor(a1) \wedge Actor(a2) \wedge \exists\, s1\ (Starring(s1) \wedge \exists\, s2\ (Starring(s2) \wedge s1.actor\_name = a1.stage\_name \wedge s2.actor\_name = a2.stage\_name \wedge s1.movie\_id = s2.movie\_id \text{ and } s1 <> s2 \}$$

Limitations of RCalc and safe expressions

- Limitations, extensions and issues
  - Difference to first order predicate logic (FOL)
    - no functions $\forall x\ (x > 1 \Rightarrow square(x) > x)$ not allowed
    - FOL interested in formula valid for all domains
      e.g. $\forall x\ P(x) \vee \neg P(x)$
      Interpretation of tuple calculus expressions over the DB
  - What does $\{x \mid \neg R(x)\} = \{x \mid \neg \exists t\ (R(t) \wedge x = t\}$ mean?
    All tuples NOT belonging to R is a lot, may not even be a finite set
  - A tuple calculus expression is called safe, if the result is finite
    - Unfortunately safety property is not decidable
    - Roughly speaking (syntactically), expressions are safe, if no range variable occurs negated outside an expression which restricts the result set otherwise
      e.g. $\{x \mid R(x)\}$ and $\{x \mid T(x) \wedge \neg R(x)\}$ are safe,

7.3 Relational completeness

Relational Algebra and calculus are equivalent

- For each RA expression there is an equivalent safe tuple calculus expression
- For each safe tuple calculus expression there is an equivalent safe domain calculus expression
- For each safe domain calculus expression there is an equivalent RA expression

  Equivalent means: results are the same when evaluated over the same DB
- This property of relational languages is called

  relational completeness

  Relational complete does not mean computational complete. Remember: transitive closure cannot be expressed in RA, which basically means that recursion is missing

Relational completeness

- Has been considered as the base line for database query languages: every query language should be as expressive as relational algebra
- SQL is in this tradition, but has introduced many concepts which are difficult or impossible to express in RA
  - grouping and predicates over sets
    e.g. find those movies having the maximal number of copies (tapes)
  - arithmetic in expressions, e.g. find cheapest product prices including taxes (in an appropriate DB)
  - partial matches of attribute values, e.g. find movies the titles of which are LIKE 'To be$ '
  - application specific comparison functions (and types), e.g: find those customers whose names sound like "Maia"

Relational Languages Summary

- Relational Algebra
  - Applicative language on tables for specifying result tables
  - Base for SQL (partially) and query optimization
- Relational Calculus:
  - Formal languages for handling data in relational model
  - Declarative language, specify *which*, not *how*, data to retrieve
  - Basis for QUEL, QBE, SQL (partially)
- Important terms & concepts
  - Operator tree (Rel. Algebra)
  - Implicit / explicit representation of knowledge ("intensional" vs. "extensional")
  - Tuple Calculus
  - Domain Calculus
  - Safe expression
  - Relational Completeness

Relational Calculus Practice (home work)

Examples (find tuple calculus expressions)

C(mem_No, name,..,)

25	Abel	…
47	Bebel	…
55	Abel	…

T(id, m_Id,f)

1	7
3	7
4	55
5	1
11	25
17	25

R(tape_Id, from,mem_No,..)

1	30.4.03	55
4	2.5.03	47

M(id,title,...)

7	"To be or not to Be"
55	"Private Ryan"
1	"Marnie"
2	"The Kid"
25	"Amistad"

T = Tape
M = Movie
C = Customer
R = Rental
H = Has

0. Movies (mId) and the format.
1. Tapes loaned by Abel
2. List of films that are currently available (i.e. not rented by anyone)
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?
6. Find the films which a available in exactly one formats