

| Basic concepts of the relational model 4 | Basic concepts of the relational model 5 |
|---|--|
| Constraints | Constraints |
| Referential constraints | Integrity constraints |
| Intension (indirectly) gives a specification of foreign keys in a relation (as in the supplier-parts relation, with tuples of the form (S#, P#, QTY)) | Certain constraints are imposed by the semantics of the data. E.g. person's height is positive, date-of-birth won't normally be a future date etc |
| The use of keys for supplier and parts in this way independently constrains the S# and P# attributes to values that are either null or designate uniquely identified entities | Real-world constraints can be too rich to express:hard to capture type of real-world observableshave data dependent constraints, motivating triggers |
| 4/20/2005 5 CS319 Theory of Databases | 4/20/2005 6 CS319 Theory of Databases |
| | |
| Summary: Basic concepts of relational model | Query Languages for Relational Databases 1 |
| Summary: Basic concepts of relational model Relation: relation, attribute, tuple | Query Languages for Relational Databases 1 Issue: how do we model data extraction formally? |
| Relation: relation, attribute, tuple Relation as analogue of file: cf. file, field type, record | |
| Relation: relation, attribute, tuple Relation as analogue of file: cf. file, field type, record Relational scheme for a database: cf. file system | Issue: how do we model data extraction formally? E.F. ("Ted") Codd is the pioneer of relational DBs |
| | Issue: how do we model data extraction formally? E.F. ("Ted") Codd is the pioneer of relational DBs Early papers: 1969, 70, 73, 75 Two classes of query language: algebra / logic 1. Algebraic languages |
| Relation: relation, attribute, tuple Relation as analogue of file: cf. file, field type, record Relational scheme for a database: cf. file system - Degree and cardinality of a relation | Issue: how do we model data extraction formally? E.F. ("Ted") Codd is the pioneer of relational DBs Early papers: 1969, 70, 73, 75 Two classes of query language: algebra / logic |
| Relation: relation, attribute, tuple Relation as analogue of file: cf. file, field type, record Relational scheme for a database: cf. file system - Degree and cardinality of a relation - Intensional & extensional views of a relational scheme | Issue: how do we model data extraction formally? E.F. ("Ted") Codd is the pioneer of relational DBs Early papers: 1969, 70, 73, 75 Two classes of query language: algebra / logic 1. Algebraic languages |

| Query Lan | guages for Relational Databases 2 |
|--------------------------------|--|
| Issue: how do | we model data extraction formally? |
| | Calculus languages inding values satisfying predicate |
| Two kinds of | predicate calculus language |
| , i | ive objects) tuples xor domain values: |
| • tuples \Rightarrow tu | ple relational calculus |
| domain val | ues \Rightarrow domain relational calculus |
| 4/20/2005 | 9 CS319 Theory of Databases |

Query Languages for Relational Databases 4

Issue: how are query languages to be compared?

Answer (Codd)

Can formulate a notion of **completeness**, and show that the core queries in these languages have equivalent expressive power

- mathematical notion, based on relational algebra
- in practice, is a *basic* measure of expressive power: practical query languages are 'more than complete'

4/20/2005

CS319 Theory of Databases

Query Languages for Relational Databases 3

Examples of Query Languages

algebraic: ISBL - Information System Base Language

tuple relational calculus: QUEL, SQL

domain relational calculus: QBE - Query by Example

Issue: how are these languages to be compared?

10

4/20/2005

CS319 Theory of Databases

Relational Algebra 1

Relational Algebra

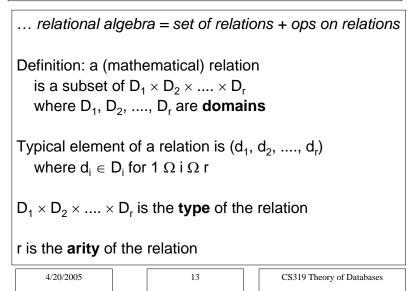
algebra = underlying set with operations on it

elements of the underlying set are referred to as "elements of the algebra"

relational algebra = set of relations + ops on relations

cf set of polynomials with addition and multiplication





Relational Algebra 4

Basic algebraic operations on relations

1. Union

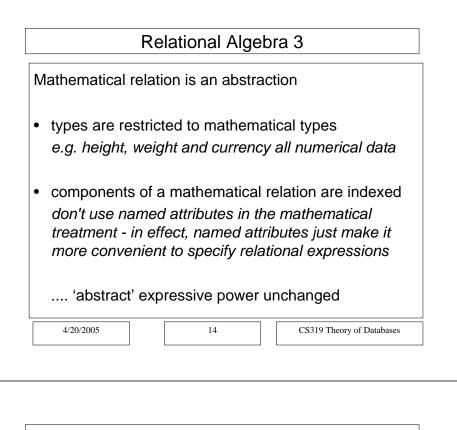
 $R \cup S$ defined when R and S have same type $R \cup S$ = union of the sets of tuples in R and S

2. Set Difference

 $\mathsf{R}-\mathsf{S}$ defined when R and S have same type

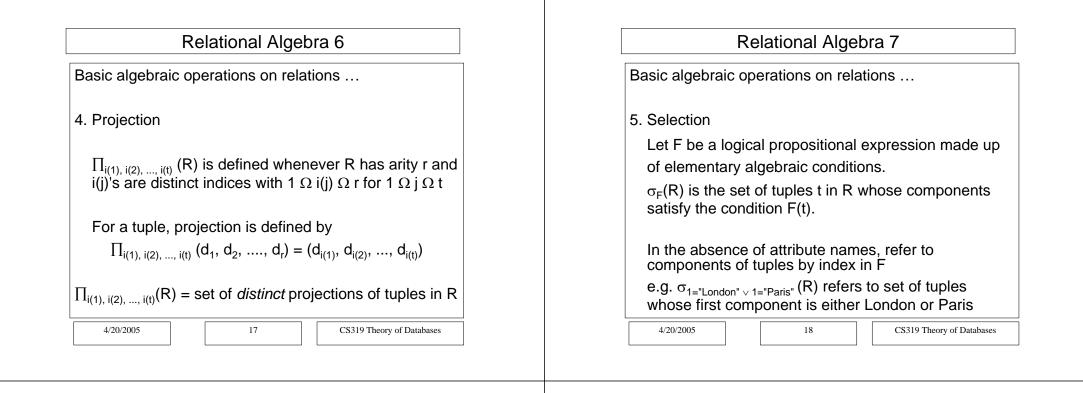
15

R - S is the set of tuples in R but not in S



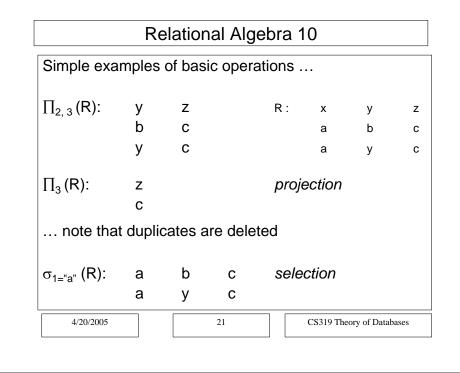
Relational Algebra 5

Basic algebraic operations on relations ...3. Cartesian ProductR of type $D_1 \times D_2 \times \dots \times D_r$
S of type $E_1 \times E_2 \times \dots \times E_s$ R × S is of type $D_1 \times D_2 \times \dots \times D_r \times E_1 \times E_2 \times \dots \times E_s$ R × S is the set of tuples of the form
 $(d_1, d_2, \dots, d_r, e_1, e_2, \dots, e_s)$
where $(d_1, d_2, \dots, d_r) \in R$, $(e_1, e_2, \dots, e_s) \in S$ 4/20/2005



| Simple R : x |
|------------------|
| R: x |
| |
| a |
| а |
| $R \cup S$: |
| х |
| а |
| a |
| Х |
| 4/20/2 |
| x a a x |

| | R | elatio | nal Alg | gebra | 9 | | |
|------------|-------|--------|---------|------------|-----------|---------------|-------|
| Simple exa | mples | of bas | ic oper | ations | | | |
| R : | x | у | z | S : | x | у | t |
| | а | b | с | | а | b | с |
| | а | У | С | | | | |
| R × S : | | | | car | tesian p | product | Ļ |
| x | У | Z | Х | У | t | | |
| x | У | Z | а | b | С | | |
| а | b | С | Х | у | t | | |
| а | b | С | а | b | С | | |
| а | У | С | Х | у | t | | |
| а | У | С | а | b | С | | |
| 4/20/2005 | | | 20 | | CS319 The | eory of Datab | bases |



Relational Algebra 12

Use of attribute names

In practical use of query languages, commonly use attribute names to define operations, e.g.

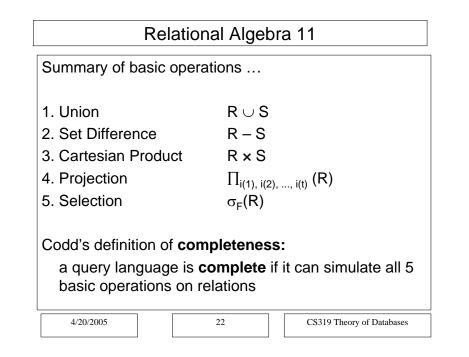
- projection onto specific attribute names
- identification of components in selection
- making distinctions between domains
- forming natural joins

Claim:

none of these devices specifies operations that can't be derived from the basic ones

23

4/20/2005



Relational Algebra 13

Definition:

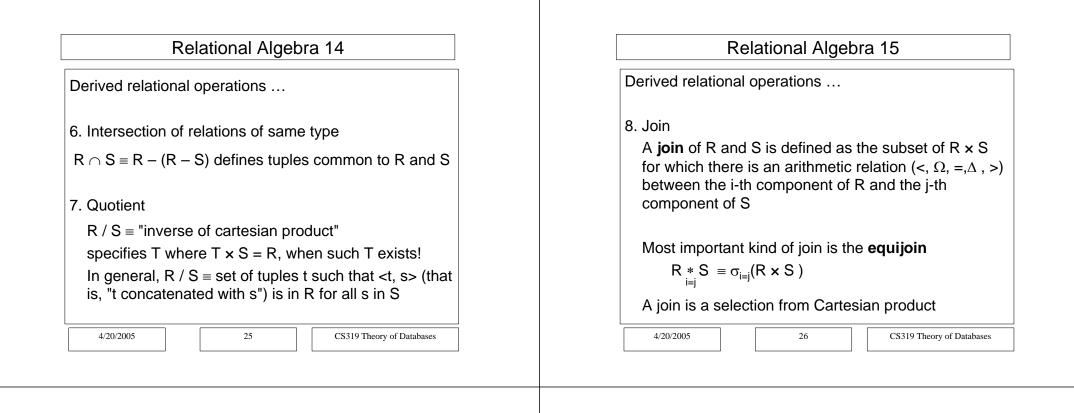
a **derived operation** in an algebraic system is an operation that is expressible in terms of standard operations of the algebra

e.g. sq() is derived from * via sq(x)=x*x

Derived operations on relations include

- intersection
- quotient
- join
- natural join

4/20/2005



Relational Algebra 16

Derived relational operations ...

In practice, Cartesian product often generates relations that are too large to be computed efficiently

More practical operation to join relations is natural join.

Definition of natural join refers to equality of domains \Rightarrow simplest to describe w.r.t. named attributes

natural join = "equijoin without duplicate columns"

27

CS319 Theory of Databases

Relational Algebra 17

Derived relational operations ...

9. The Natural Join

Derive the natural join R * S by

- forming product R × S
- selecting those tuples (r,s) where r and s have same values for all common attributes
- making a projection to remove duplicate columns that correspond to these common attributes

 $\mathsf{R} * \mathsf{S} = \prod_{i(1), i(2), \dots, i(m)} \sigma_{\Lambda(r.x=s.x)}(\mathsf{R} \times \mathsf{S})$

with an appropriate choice of indices i(j) & attributes x

CS319 Theory of Databases

| Primitive operations:1. Union | $R \cup S$ | ISBL - Information System Base Language |
|--|--|---|
| 2. Set Difference 3. Cartesian Product 4. Projection 5. Selection | $ \begin{array}{l} R - S \\ R \times S \\ \prod_{i(1), i(2), \dots, i(t)} (R) \\ \sigma_{F}(R) \end{array} $ | Devised by Todd in 1976 IBM Peterlee Relational Test Vehicle (PRTV) PL/1 environment with query language ISBL |
| Derived operations: intersect Codd's definition of complet | | One of the first relational query languages closely based on relational algebra |
| a query language is comp basic operations on relation | blete if it can simulate all 5 | The six basic operations in ISBL are union, difference intersection, natural join, projection and selection |
| 4/20/2005 29 | CS319 Theory of Databases | 4/20/2005 30 CS319 Theory of Database |

ISBL: A Relational Algebra Query Language 2 Operators in ISBL are '+', '-', '%', ':' and '*'.

- R+S union of relations
- R S difference operation with extended semantics
- R % A, B, ... , Z projection onto named attributes
- R: F selection of tuples subject to boolean formula F
- R.S intersection
- R * S natural join
- R S is defined whenever R and S have some attribute names in common: delete tuples from R that agree with S on all common attributes.

31

4/20/2005

ISBL: A Relational Algebra Query Language 3

Comparison: Relational Algebra vs ISBL

| $R \cup S$ | R+S |
|---------------------------------|-----------------------|
| R – S | R-S subsumes |
| R × S | no direct counterpart |
| $\prod_{i(1), i(2),, i(t)} (R)$ | R % A, B, , Z |
| σ _F (R) | R : F |
| contrived derived op | R * S |

To prove completeness of ISBL, enough to show that can express Cartesian product using the ISBL operators - return to this issue later

4/20/2005

| ISBL: A Relati | ional Algebra Query Language 4 |
|------------------------------------|---|
| ISBL as a query I | anguage |
| Two types of stat | ement in ISBL |
| LIST <exp></exp> | print the value of exp |
| R = <exp></exp> | assign value of exp to relation R |
| In this context, R | is a variable whose value is a relation |
| Notation: use R(A attributes A, B, | A,B,,Z) to refer to a relation with |
| 4/20/2005 | 33 CS319 Theory of Databases |

ISBL: A Relational Algebra Query Language 6

Assignment and call-by-value

After the assignment

RCS = (R * S) : B=C % A, D

the variable RCS retains its assigned value whatever happens to the values of R and S

Hence all subsequent "LIST RCS" requests obtain same value until reassignment

cf call-by-value parameter passing mechanisms

35

4/20/2005

ISBL: A Relational Algebra Query Language 5 **Example ISBL query** to specify the composition of two binary relations R(A,B) and S(C,D) where A,B,C,D are attributes defined over the same domain X (as when defining composition of functions X□X):

Specify composition of R and S as RCS, where RCS = (R * S) : B=C % A, D In this case: R * S = R × S because attribute names (A, B), (C, D) are disjoint [cf. completeness of ISBL]

Illustrates archetypal form of query definition:

projection of selection of join

34

4/20/2005

CS319 Theory of Databases

ISBL: A Relational Algebra Query Language 7

Delayed evaluation and call-by-name

- have a delayed evaluation mechanism to change the semantics of assignment cf. a "definitive notation" or a spreadsheet definition
- to delay the evaluation of the relation named R in an expression, use N!R in place of R

RCS = (N!R * N!S) : B=C % A, D

- this means that the variable RCS is evaluated on a call-by-name basis: i.e. it's value is computed as required using the current values of R and S
- whenever the user invokes "LIST RCS" in this case, the value of RCS is re-computed

ISBL: A Relational Algebra Query Language 8 ISBL: A Relational Algebra Query Language 9 Renaming Uses for delayed evaluation For union & intersection, attribute names must match definition of views is facilitated e.g. R(A,B) + S(A,C) is undefined etc. allows incremental definition of complex expressions: To overcome this can rename attributes of R by use sub-expressions with temporary names, supply $(R\%A, B \rightarrow C)$ extensional part later This project-and-rename creates relation R(A,C). • useful for optimisation: assignment means immediate Can use this to make attributes of R & S disjoint, so that computation at every step, delayed evaluation allows $R * S = R \times S$. intelligent updating of values proving that ISBL is a complete query language 4/20/2005 4/20/2005 38 37 CS319 Theory of Databases CS319 Theory of Databases ISBL: A Relational Algebra Query Language 10 Tensions between theory and practice in ISBL Limitations of ISBL Mathematical relations abstract away certain characteristics of data that are important to the human interpreter – e.g. types, order for table ISBL is complete, but lacks features of QUEL, SQL etc inspection

e.g. no aggregate operators

programming language PL/1

4/20/2005

Primarily a declarative query language

no insertion, deletion and modification

Address these issues in the PRTV environment - user can also access relations via the general-purpose

40

CS319 Theory of Databases

- · Certain activities that are an essential part of data processing, such as updating relations, forming aggregates etc are not easy to describe formally
- Classical algebra uses homogeneous data types, doesn't deal elegantly with exceptions 3/0 = ? etc

39

| Illustrative examples of ISBL use | Illustrative examples of ISBL use |
|--|--|
| | MEMBERS(NAME, ADDRESS, BALANCE) |
| Refer to the Happy Valley Food Company [Ullman 82] | ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) |
| Relations in this DB are: | 1. Print the names of members in the red: |
| MEMBERS(NAME, ADDRESS, BALANCE) | LIST MEMBERS : BALANCE < 0 % NAME |
| ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) | |
| SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) | i.e. select members with negative balance and project out their names |
| 4/20/2005 41 CS319 Theory of Databases | 4/20/2005 42 CS319 Theory of Databases |
| | |
| ISBL: A Relational Algebra Query Language 13 | ISBL: A Relational Algebra Query Language 14 |
| | ISBL: A Relational Algebra Query Language 14 |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) |
| Illustrative examples of ISBL use | |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. Print the supplier names, items & prices for suppliers | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: SUPPLIERS required for supplier details |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: SUPPLIERS required for supplier details ORDERS to know what Brooks has ordered |
| ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. Print the supplier names, items & prices for suppliers who supply at least one item ordered by Brooks | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: SUPPLIERS required for supplier details ORDERS to know what Brooks has ordered The join OS holds tuples where item field contains item |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. Print the supplier names, items & prices for suppliers who supply at least one item ordered by Brooks OS = ORDERS * SUPPLIERS | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: SUPPLIERS required for supplier details ORDERS to know what Brooks has ordered The join OS holds tuples where item field contains item <i>"ordered with associated order info"</i> and |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) Print the supplier names, items & prices for suppliers who supply at least one item ordered by Brooks | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: SUPPLIERS required for supplier details ORDERS to know what Brooks has ordered The join OS holds tuples where item field contains item "ordered with associated order info" and "supplied by supplier with assoc supplier info" |
| Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. Print the supplier names, items & prices for suppliers who supply at least one item ordered by Brooks OS = ORDERS * SUPPLIERS | Illustrative examples of ISBL use MEMBERS(NAME, ADDRESS, BALANCE) ORDERS(ORDER_NO, NAME, ITEM, QUANTITY) SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE) 2. (commentary on answer) Need two of the relations: SUPPLIERS required for supplier details ORDERS to know what Brooks has ordered The join OS holds tuples where item field contains item <i>"ordered with associated order info"</i> and |

| 3. Print suppliers who supply <i>every</i> item ordered by | 3 suppliers supplying <i>every</i> item ordered by Brooks |
|---|--|
| Brooks | S = SUPPLIERS % SNAME |
| "Every item" is universal quantification | I = SUPPLIERS % ITEM |
| Every item is universal quantification | NS = (S * I) - (SUPPLIERS % SNAME, ITEM) |
| Strategy: translate $(\forall x)(p(x))$ to $\neg(\exists x)(\neg p(x))$ | S records all supplier names, and I all items supplied |
| find suppliers who don't supply at least one of the items that is ordered by Brooks, and take the | NS is the "does not supply" relation: all supplier-item pairs with pairs such that s supplies i eliminated |
| complement of this set of suppliers | Now specify items ordered by Brooks |
| Notation: \forall is "for all", \exists is "there exists", \neg is "not" | B = ORDERS : NAME="Brooks" % ITEM |
| 4/20/2005 45 CS319 Theory of Databases | 4/20/2005 46 CS319 Theory of Databases |
| | |
| ISBL: A Relational Algebra Query Language 17 | |
| | |
| | To follow |
| 3 suppliers supplying <i>every</i> item ordered by Brooks NS = "doesn't supply" relation | To follow Relational Theory: Algebra and Calculus SQL review |
| 3 suppliers supplying <i>every</i> item ordered by Brooks NS = "doesn't supply" relation B = "items ordered by Brooks" find suppliers who don't supply at least one item in B | Relational Theory: Algebra and Calculus |
| 3 suppliers supplying <i>every</i> item ordered by Brooks NS = "doesn't supply" relation B = "items ordered by Brooks" find suppliers who don't supply at least one item in B NSB = NS.(S * B) set of (supplier, item) pairs such s doesn't supply i | Relational Theory: Algebra and Calculus |
| 3 suppliers supplying <i>every</i> item ordered by Brooks NS = "doesn't supply" relation B = "items ordered by Brooks" find suppliers who don't supply at least one item in B NSB = NS.(S * B) set of (supplier, item) pairs such s doesn't supply i and Brooks ordered i. | Relational Theory: Algebra and Calculus |