Information Systems

Relational Databases

Temur Kutsia

Research Institute for Symbolic Computation
Johannes Kepler University of Linz, Austria
kutsia@risc.uni-linz.ac.at

Outline

The Relational Model (Continues from the Previous Lecture)
- Data Structure. Types and Relations
- Data Manipulation. Relational Algebra

Outline

The Relational Model (Continues from the Previous Lecture)
- Data Structure. Types and Relations
- Data Manipulation. Relational Algebra

Relations

- Up to now we discussed type, values, and variables in general.
- Now: Relations types, values, and variables in particular.
- Since relations are built out of tuples, we examine tuple types, values, and variables.

Tuples

Tuple

- Given a collection of (not necessarily distinct) types $T_i, 1 \leq i \leq n$, a tuple value (or tuple) t on those types is a set of ordered triples of the form $\langle A_i, T_i, v_i \rangle$, where
  - A_i is an attribute name, T_i is a type name, v_i is a value of type T_i.
  - The value n is the degree or arity of t.
  - The ordered triple $\langle A_i, T_i, v_i \rangle$ is a component of t.
  - The ordered pair $\langle A_i, T_i \rangle$ is an attribute of t and is uniquely identified by A_i. (A_i's are all distinct.)
  - v_i is the attribute value for A_i.
  - T_i is the attribute type for A_i.
  - The complete set of attributes is the heading of t.
  - The tuple type of t is determined by the heading of t. The tuple type name is precisely

$$\text{TUPLE}\ \{\ A_1\ T_1, A_2\ T_2, \ldots, A_n\ T_n\ \}.$$

Tuple

Example

Sample tuple:

$$\{\langle \text{MAJOR\_P\#}, \text{P\#}, \text{P2} \rangle, \langle \text{MINOR\_P\#}, \text{P\#}, \text{P4} \rangle, \langle \text{QTY}, \text{QTY}, 7 \rangle\}$$

MAJOR_P# : P#	MINOR_P# : P#	QTY : QTY
P2	P4	7

- Attribute names: MAJOR_P#, MINOR_P#, QTY.
- The corresponding type names: P#, P#, and QTY.
- The corresponding values: P2, P4, 7.
- The degree of the tuple is three.
- The heading:

MAJOR_P# : P#	MINOR_P# : P#	QTY : QTY

- The type: TUPLE { MAJOR_P# P#, MINOR_P# P#, QTY QTY}

Tuple

- In informal contexts type names are often omitted from a tuple heading, showing just the attribute names.

For instance, writing

MAJOR_P#	MINOR_P#	QTY
P2	P4	7

instead of

MAJOR_P# : P#	MINOR_P# : P#	QTY : QTY
P2	P4	7

Tuple Properties

- Every tuple contains exactly one value for each attribute.
- The order of components of a tuple does not matter.
- Every subset (including the empty subset) of a tuple is a tuple.

Tuple Type Generators

- Example:
  ```
  VAR ADDR TUPLE {
      STREET CHAR,
      CITY CHAR,
      STATE CHAR,
      ZIP CHAR } ;
  ```
- Defines the variable ADDR to be of type
  ```
  TUPLE { STREET CHAR, CITY CHAR,
          STATE CHAR, ZIP CHAR }
  ```
- Tuple selector operator:
  ```
  TUPLE { STREET '1600 Penn. Ave.', CITY 'Washington',
          STATE 'DC', ZIP '20500' }
  ```

Operations on Tuples

Tuple equality:

- Tuples t_1 and t_2 are equal ($t_1 = t_2$) iff
  1. they have the same attributes $Attr_1, \ldots, Attr_n$, and
  2. the value v_i of $Attr_i$ in t_1 is equal to the value v_i of $Attr_i$ in t_2.

Operations on Tuples

Assume the current value of the ADDR variable is
TUPLE { STREET '1600 Penn. Ave.', CITY 'Washington',
STATE 'DC', ZIP '20500' }

- Tuple projection: ADDR { CITY, ZIP } denotes the tuple TUPLE { CITY 'Washington', ZIP '20500' }.
- Extraction: ZIP FROM ADDR denotes '20500'.
- Tuple type inference: Tuple type of the result of ADDR { CITY, ZIP } is TUPLE { CITY CHAR, ZIP CHAR }.

Operations on Tuples

WRAP and UNWRAP:

- Consider the tuple types:
  TT1: TUPLE { NAME NAME, ADDR TUPLE {
  STREET CHAR, CITY CHAR,
  STATE CHAR, ZIP CHAR } }.
  TT2: TUPLE { NAME NAME,
  STREET CHAR, CITY CHAR,
  STATE CHAR, ZIP CHAR }.
- NADDR1, NADDR2: The variables of types TT1, TT2, resp.
- The expression
  NADDR2 WRAP {STREET, CITY, STATE, ZIP} AS ADDR
  takes the current value of NADDR2 and wraps STREET, CITY, STATE, ZIP components into a single tuple-valued ADDR component. The result is of of type TT1.
- The expression NADDR1 UNWRAP ADDR takes the current value of NADDR1 and unwraps ADDR into four separate components. The result is of type TT2.

Relations

Relation

- A relation value (or relation) r consists of a heading and a body, where
  - The heading of r is a tuple heading. Relation r has the same attributes and the same degree as that heading does.
  - The body of r is the set of tuples, all having that same heading; the cardinality of that set is said to be the cardinality of r.

Relation type

- A relation type of r is determined by the heading of r.
- It has the same attributes (and hence attribute names and types) and degree as that heading does.
- The relation type name is
  RELATION { A1 T1, . . . , An Tn }

Relations

Example

MAJOR_P# : P#	MINOR_P# : P#	QTY : QTY
P1	P2	5
P1	P3	3
P2	P3	2
P2	P4	7
P3	P5	4
P4	P6	8

Type:
RELATION { MAJOR_P# : P#, MINOR_P# : P#, QTY : QTY }

Relations

- n-ary relation: relation of degree n.
- Every subset of a heading is a heading.
- Every subset of a body is a body.

The RELATION Type Generator

Example:

- VAR PART_STRUCTURE . . .
  RELATION { MAJOR_P# : P#, MINOR_P# : P#, QTY : QTY }
- PART_STRUCTURE: relation variable (relvar)
- RELATION { MAJOR_P# : P#, MINOR_P# : P#, QTY : QTY }: Invocation of the RELATION type generator, gives a generated type.

Relation Properties

Within the same relation

- every tuple contains exactly one value for each attribute,
- no left-to-right ordering to the attributes,
- no top-to-bottom ordering to the tuples,
- no duplicate tuples.

Relations with No Attributes

- Every relation has a set of attributes.
- This set, in particular, can be empty: No attributes at all.
- Does not mean the empty relation!
- Empty relation: relation with the empty body.
- Relation with no attributes: relation with the empty heading.

Relations with No Attributes

- Relation with no attributes can contain at most one tuple, the 0-tuple.
- The 0-tuple contains no components.
- Hence, two relations of degree 0: one that contains just one tuple, and one that contains no tuples at all.
- Names: TABLE_DEE and TABLE_DUM, respectively.

Operators on Relations

Comparisons:

- $=, \neq, \subseteq, \subset, \supseteq, \supset,$ IS_EMPTY.
- They can appear whenever a boolean expression is expected.
- Example: S { CITY } = P { CITY }: Is the projection of suppliers over CITY equal to the projection of parts over city?

Operators on Relations

Other operators:

- Test whether the given tuple t appears in a given relation r: $t \in r$.
- Extracting the single tuple from a relation of cardinality one: TUPLE FROM r
- Other operators like join, restrict, project, etc. Considered in the relational algebra part.

Operators on Relations

Relation type inference:

- Given the suppliers relvar S, the expression S { S#, CITY } yields a relation whose type is
  RELATION { S# S#, CITY CHAR }

Relation Variables

- Base and derived relvars.
- Base and derived relations.
- Other name of derived relvars: views

Relation Variables

- Base and derived relvars.
- Base and derived relations.
- Other name of derived relvars: views

Relation Variables

Example

Defining base relvars S, P, and SP:

```
VAR S BASE RELATION
  { S#          S#,
    SNAME       NAME,
    STATUS      INTEGER,
    CITY        CHAR }
  PRIMARY KEY { S# } ;
```

```
VAR P BASE RELATION
  { P#          P#,
    PNAME       NAME,
    COLOR       COLOR,
    WEIGHT      WEIGHT,
    CITY        CHAR }
  PRIMARY KEY { P# } ;
```

```
VAR SP BASE RELATION
  { S#      S#,
    P#      P#,
    QTY     QTY }
  PRIMARY KEY { S#, P# }
  FOREIGN KEY { S# } REFERENCES S
  FOREIGN KEY { P# } REFERENCES P
```

Explanation

- The relation type of the relvar S is
  RELATION {S# S#, SNAME NAME, STATUS INTEGER, CITY CHAR }
- The terms heading, body, attributes, tuple, degree, etc. are interpreted to apply to relvars.
- When a base relvar is defined, it is given an initial value that is the empty relation of appropriate type.
- It is assumed that a means is available for specifying default values to some attributes of base relvars.

Updating Relvars

- Assume S' and SP' are base relvars.
- The type of S' is the same as the type of S.
- The type of SP' is the same as the type of SP.
- Some valid examples of relation assignment:
  1. S' := S, SP' := SP;
  2. S' := S WHERE CITY = 'London'
  3. S := S WHERE NOT (CITY = 'Paris')
- Each assignment
  (a) retrieves the relation specified on the right hand side and
  (b) updates the relvar specified on the left hand side.

Outline

The Relational Model (Continues from the Previous Lecture)
- Data Structure. Types and Relations
- Data Manipulation. Relational Algebra

Relational Algebra

- Theoretical basis for database query languages.
- Attracted attention after Edgar F. Codd introduced the relational model in 1970-ies.
- Formal system for manipulating relations:
  - Operands: relations.
  - Operators: union, intersection, difference, Cartesian product, restrict, project, join, divide, rename.
  - Operations operate on relations and produce relations (closure).

Rename

- Purpose: Rename attributes within a specified relation.
- Action: Takes a given relation and returns another one that is identical to the given one except that one of its attributes has a different name.
- Example:

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

S RENAME CITY AS SCITY

S#	SNAME	STATUS	SCITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

Union

- Specification: Given two relations *a* and *b* of the same type, *a* UNION *b* is a relation of the same type, with body consisting of all tuples *t* such that *t* appears in *a* or in *b* or both.
- Example:

A

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris

A UNION B

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London
S2	Jones	10	Paris

Intersection

- Given two relations *a* and *b* of the same type, *a* INTERSECT *b* is a relation of the same type, with body consisting of all tuples *t* such that *t* appears in both *a* and *b*.
- Example:

A

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris

A INTERSECT B

S#	SNAME	STATUS	CITY
S1	Smith	20	London

Difference

- Given two relations *a* and *b* of the same type, *a* MINUS *b* is a relation of the same type, with body consisting of all tuples *t* such that *t* appears *a* and not in *b*.
- Example:

A

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris

A MINUS B

S#	SNAME	STATUS	CITY
S4	Clark	20	London

B MINUS A

S#	SNAME	STATUS	CITY
S2	Jones	10	Paris

Cartesian Product

- Given two relations *a* and *b* without common attribute names, *a* TIMES *b* is a relation with a heading that is the (set theoretic) union of the heading of *a* and *b* and with the body consisting of the set of all tuples *t* such that *t* is a (set theoretic) union of a tuple appearing in *a* and a tuple appearing in *b*.
- Example:

A

S#
S1
S2

B

P#
P1
P2
P3

A TIMES B

S#	P#
S1	P1
S1	P2
S1	P3
S2	P1
S2	P2
S2	P3

Restriction

- Given a relation *a* with attributes $X, Y, \ldots, Z$ and a truth-valued function *p* whose parameters are some subset of $X, Y, \ldots, Z$, the restriction of *a* according to *p*, *a* WHERE *p*, is a relation with the same heading as *a* and with body consisting of all those tuples in *a* on which *p* evaluates to TRUE.
- Example:

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

S WHERE CITY = 'London'

S#	SNAME	STATUS	CITY
S1	Smith	20	London

Restriction

- Given a relation *a* with attributes $X, Y, \ldots, Z$ and a truth-valued function *p* whose parameters are some subset of $X, Y, \ldots, Z$, the restriction of *a* according to *p*, *a* WHERE *p*, is a relation with the same heading as *a* and with body consisting of all those tuples in *a* on which *p* evaluates to TRUE.
- Example:

P

P#	PN	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Oslo
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris

P WHERE WEIGHT < WEIGHT (14.0)

P#	PN	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P5	Cam	Blue	12.0	Paris

Restriction

- Given a relation *a* with attributes $X, Y, \ldots, Z$ and a truth-valued function *p* whose parameters are some subset of $X, Y, \ldots, Z$, the restriction of *a* according to *p*, *a* WHERE *p*, is a relation with the same heading as *a* and with body consisting of all those tuples in *a* on which *p* evaluates to TRUE.
- Example:

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S2	P1	400
S2	P2	100

SP WHERE S# = S# ('S3') or P# = P# ('P4')

S#	P#	QTY

Projection

- Given a relation a with attributes $X, Y, \ldots, Z$, the projection of a according on $X, Y, \ldots, Z$, written $a\{X, Y, \ldots, Z\}$, is a relation with
  - a heading derived from the heading of a by removing all attributes that are not among $X, Y, \ldots, Z$;
  - a body consisting of all tuples $\{X\ x, Y\ y, \ldots, Z\ z\}$ such that the tuple appears in a with X value x, Y value $y, \ldots,$ and Z value z.
- Example:

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

S { CITY }

CITY
London
Paris

Projection

- Given a relation a with attributes $X, Y, \ldots, Z$, the projection of a according on $X, Y, \ldots, Z$, written $a\{X, Y, \ldots, Z\}$, is a relation with
  - a heading derived from the heading of a by removing all attributes that are not among $X, Y, \ldots, Z$;
  - a body consisting of all tuples $\{X\ x, Y\ y, \ldots, Z\ z\}$ such that the tuple appears in a with X value x, Y value y,..., and Z value z.
- Example:

P

P#	PN	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Oslo
P4	Screw	Red	14.0	London

P {COLOR, CITY }

COLOR	CITY
Red	London
Green	Paris
Blue	Oslo

Projection

- Given a relation a with attributes $X, Y, \ldots, Z$, the projection of a according on $X, Y, \ldots, Z$, written $a\{X, Y, \ldots, Z\}$, is a relation with
  - a heading derived from the heading of a by removing all attributes that are not among $X, Y, \ldots, Z$;
  - a body consisting of all tuples $\{X\,x, Y\,y, \ldots, Z\,z\}$ such that the tuple appears in a with X value x, Y value y, ..., and Z value z.
- Example:

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

(S WHERE CITY = 'Paris') { S# }

S#
S2
S3

Join

- Let a relation a have attributes $X_1, \ldots, X_m$, $Y_1, \ldots Y_n$, and b have the attributes $Y_1, \ldots Y_n, Z_1, \ldots, Z_p$.
- The (natural) join of a and b, denoted a JOIN b is a relation with heading $X_1, \ldots, X_m$, $Y_1, \ldots, Y_n, Z_1, \ldots, Z_p$ and body consisting of all tuples
$X_1\, x_1, \ldots, X_m\, x_m$, $Y_1\, y_1, \ldots, Y_n\, y_n, Z_1\, z_1, \ldots, Z_p\, z_p$ such that
  - a tuple appears in a with X_i value x_i, and Y_j value y_j for all $1 \leq i \leq m$ and $1 \leq j \leq n$, and
  - a tuple appears in b with Y_j value y_j and Z_k value z_k for all $1 \leq j \leq n$ and $1 \leq k \leq p$.

Join. Example

S

S#	SNAME	ST	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

P

P#	PN	COLOR	WGT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Oslo
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris

S JOIN P

S#	SNAME	ST	CITY	P#	PN	COLOR	WGT
S1	Smith	20	London	P1	Nut	Red	12.0
S1	Smith	20	London	P4	Screw	Red	14.0
S2	Jones	10	Paris	P2	Bolt	Green	17.0
S2	Jones	10	Paris	P5	Cam	Blue	12.0
S3	Blake	30	Paris	P2	Bolt	Green	17.0
S3	Blake	30	Paris	P5	Cam	Blue	12.0

Divide

- Let a relation a have attributes $X_1, \ldots, X_m$ and b have the attributes $Y_1, \ldots Y_n$ such that no X_i has the same name as any Y_j, and
- let a relation c have the attributes $X_1, \ldots, X_m, Y_1, \ldots Y_n$.
- The division of a by b per c (a dividend, b divisor, c mediator), denoted a DIVIDEBY b PER c is a relation with heading $X_1, \ldots, X_m$ and body consisting of all tuples $X_1\ x_1, \ldots, X_m\ x_m$ appearing in a such that a tuple $X_1\ x_1, \ldots, X_m\ x_m, Y_1\ y_1, \ldots, Y_n\ y_n$ appears in c for all tuples $Y_1\ y_1, \ldots, Y_n\ y_n$ appearing in b.

Divide. Example

DEND

S#
S1
S2
S3
S4
S5

MED

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
..	..
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

DOR

P#
P1

DOR

P#
P2
P4

DOR

P#
P1
P2
P3
P4
P5
P6

DEND DIVEDEBY DOR PER MED

S#
S1
S2

S#
S1
S4

S#
S1

Examples. Supplier-and-Parts

- Get supplier names for suppliers who supply part P2.
- ((SP JOIN S) WHERE P# = P# ('P2')) { SNAME }
- SP JOIN S extends which SP tuple with the corresponding supplier information (SNAME, STATUS, CITY values). The result is restricted to just those tuples for part P2. The restriction is projected over SNAME

Examples. Supplier-and-Parts

- Get supplier names for suppliers who supply at least one red part.
- (((P WHERE COLOR = COLOR('Red'))
  JOIN SP) { S# } JOIN S){ SNAME }

Examples. Supplier-and-Parts

- Get supplier names for suppliers who supply all parts.
- ((S { S#} DIVIDEBY P { P#} PER SP { S#, P# })
  JOIN S) { SNAME }

Examples. Supplier-and-Parts

- Get supplier numbers for suppliers who supply at least all those parts supplied by supplier S2.
- S { S# } DIVIDEBY (SP WHERE S# = S# ('S2')) {P#}
  PER SP {S#, P#})

Examples. Supplier-and-Parts

- Get all pairs of supplier numbers such that the suppliers are located in the same city.
- (((S RENAME S# AS SA) { SA, CITY } JOIN
  (S RENAME S# AS SB) { SB, CITY })
  WHERE SA < SB { SA, SB }

Examples. Supplier-and-Parts

- Get supplier names for suppliers who do not supply part P2.
- ((S { S# } MINUS (SP WHERE P# = P# ('P2')) {S#})
  JOIN S){ SNAME }