An Introduction to Database Systems

chapter 8. Relational Calculus

spark@dblab.sogang.ac.kr

8.1 Introduction (1/4)

- relational algebra
  - provides a collection of a explicit operations
    - join, union, projection, etc. , ..
  - how to construct from some desired relation in terms of the given relations
- relational calculus
  - provides a notation for stating the definition of that desired relation in terms of given relations

spark@dblab.sogang.ac.kr

8.1 Introduction (2/4)

- **Query : "Get supplier numbers and cities for suppliers who supply part P2"**
- **algebraic formulation**
  - 1) join relations S and SP over S#
  - 2) restrict the result of that join to tuples for part P2
  - 3) project the result of that restriction over S# and CITY
- **calculus formulation**
  - Get S# and CITY for suppliers such that there exists a shipment SP with the same S# value and with P# value P2
- **defining characteristics of the desired relation**

spark@dblab.sogang.ac.kr

8.1 Introduction (3/4)

- **Relational algebra and relational calculus**

relational algebra	relational calculus
- prescriptive prescribes a procedures for solving that problem - procedural - like a programming language	- descriptive describes what the problem is - non-procedural - like a natural language

- **The algebra and the calculus are logically equivalent**
  - **An algebraic expression $\leftrightarrow$ an equivalent calculus expression**

spark@dblab.sogang.ac.kr

8.1 Introduction (4/4)

- fundamentals of the calculus
  - Predicate calculus
    - ALPHA
    - QUEL (INGRES)
  - tuple variable(range variable)
    - variable that "ranges over" some relation
    - variable whose only permitted values are tuples of that relation
  - tuple calculus
    - based on tuple variables
  - domain calculus
    - in which tuple variables are replaced by domain variables (elements)
    - domain variables that ranges over a domain instead tuple of a relation

spark@dblab.sogang.ac.kr

Tuple Calculus vs. Domain Calculus

- **example**
  "Get supplier numbers for suppliers in London"

QUEL

RANGE OF SX IS S

RETRIEVE (SX.S#) WHERE SX.CITY = 'London'

QBE

S	S#	SNAME	STATUS	CITY
	P.SX			London

8.2 Tuple Calculus (1/11)

- **Syntax**

<relational expression>

::= RELATION { <tuple expression commalist> }

<relvar name>

<relational operation>

(<relational expression>)

<range var definition>

::= ***RANGEVAR <range var name>***

RANGES OVER <relational expression commalist> ;

<range attribute reference>

::= <range var name> . <attribute reference> [AS <attribute name>]

<boolean expression>

::= ***… all the usual possibilities, together with:***

| ***<quantified boolean expression>***

<quantified boolean expression>

::= EXISTS <range var name> (<boolean expression>)

| FORALL <range var name> (<boolean expression>)

<relational operation>

::= <proto tuple> [WHERE <boolean expression>]

<proto tuple>

::= <tuple expression>

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (2/11)

- **Conditions**
  - **x Θ y**

    where Θ is any one of =, <, <=, >, >=, ≠ and at least one of x and y is an expression of T.A and the other is either a similar expression or a constant.

- **Well-formed formulas(WFFs)**
  - **constructed from conditions, Boolean operators(and, or, not), and quantifiers(∃, ∀) according to rules F1-F5**

F1. Every condition is a WFF

F2. If f is a WFF, then so are (f) and NOT(f)

F3. If f and g are WFFs, then so are (f AND g) and (f OR g)

F4. If f is a WFF in which T occurs as a free variable, then

∃T(f) and ∀T(f) are WFFs

F5. Nothing else is a WFF

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (3/11)

- **tuple variables**
  - **Range of T is X_1, X_2 , ... X_n ;**
    - **T : tuple variable**
    - **X_i : either a relation name or a tuple calculus expression**
    - **T.A : A is an attribute of the relation over which T ranges**
- **Free and Bound Variables**
  - **within a simple comparison such as T.A < U.A,**
    - **all tuple variable occurrences are free**
  - **Tuple variable occurrences in the WFFs(f) and NOT f**
    - **free or bound according as they are free or bound in f**
  - **Tuple variable occurrences in the WFFs f AND g and f OR g**
    - **free of bound according as they are free or bound in f or g**
  - **Occurrences of T that are in f are bound**
    - **in the WFFs EXISTS T(f) and FORALL T(f)**

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (4/11)

❑ Range Variables

```
RANGEVAR    SX    RANGES  OVER    S ;
RANGEVAR    SY    RANGES  OVER    S ;
RANGEVAR    SPX   RANGES  OVER    SP ;
RANGEVAR    SPY   RANGES  OVER    SP ;
RANGEVAR    PX    RANGES  OVER    P ;

RANGEVAR    SU    RANGES  OVER
        (   SX  WHERE  SX.CITY = ‘London’ ) ,
        (   SX  WHERE  EXISTS  SPX  (SPX.S# = SX.S#  AND
                                   SPX.P#  SPX = P# (‘P1’) ) ) ;
```

- SU is defined to range over the union of the set of supplier tuples for suppliers who are located in London and the set of supplier tuples for suppliers who supply part P1

8.2 Tuple Calculus (5/11)

- **Free and Bound Variable References**
  - **Every references to a range variable is either free or bound**
  - **Let V be a range variable**
    - **References to V in the WFF "NOT p" are free or bound within that WFF according as they are free or bound in p. References to V in the WFFs "(p AND q)" and "(p OR q)" are free or bound in those WFFs according as they are free or bound in p or q, as applicable**
    - **References to V that are free in the WFF "p" are bound in the WFFs "EXISTS V(p)" and "FORALL V(p)." Other references to range variables in "p" are free or bound in the WFFs "EXISTS V(p)" and "FORALL V(p)" according as they are free or bound in "p"**
    - **The sole reference to V in the <range var name> "V" is free within that <range var name>**
    - **The sole reference to V in the <range attribute reference> "V.A" is free within that <range attribute reference>**
    - **If a reference to V is free in some expression exp, that reference is also free in any expression exp' that immediately contains exp as a subexpression, unless exp' introduces a quantifier that makes the reference bound instead**

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (6/11)

Free and Bound Variable References

- **Examples**
  - **Simple comparisons :**

    SX.S# = S# (‘S1’)

    SX.S# = SPX.S#

    SPX.P# ≠ PX.P#

  - **Boolean combinations of simple comparisons :**

    PX.WEIGHT < WEIGHT (15.5) OR PX.CITY = ‘Rome’

    NOT (SX.CITY = ‘London’)

    SX.S# = SPX.S# AND SPX.P# ≠ PX.P#

    PX.COLOR = COLOR (‘Red’) OR PX.CITY = ‘London’

  - **Quantified WFFs :**

    EXISTS SPX (SPX.S# = SX.S# AND SPX.P# = P# (‘P2’))

    FORALL PX (PX.COLOR = COLOR (‘Red’))

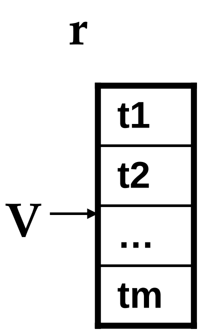
spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (7/11)

- **Quantifiers : EXISTS and FORALL**
  - **If p is a WFF in which V is free, then EXISTS V(p) and FORALL V(p) are both legal WFFs and V is bound in both of them**
  - **EXISTS**

ex) EXISTS SPX (SPX.S# = SX.S# AND SPX.P# = P# ('P2'))

(SPX : bound, SX : free)



- **Existential quantifier**
- **There exists at least one value of V that makes p evaluate to true**
- **Single occurrence of variable v is true**
- **An iterated OR :**

If (a) r is a relation with tuples, t1, t2, …, tm,

(b) V is a range variable that ranges over r, and

(c) p(V) is a WFF in which V occurs as a free variable,

Then the WFF *EXISTS V(p(V))* is defined to be equivalent to the WFF *False OR p(t1) OR … OR p(tm)*

8.2 Tuple Calculus (8/11)

- **EXISTS**
  - **Example**

r

A	B	C
(1,	2,	3)
(1,	2,	4)
(1,	3,	4)

EXISTS V (V.C > 1) : true

EXISTS V (V.B > 3) : false

EXISTT V (V.A > 1 OR V.C = 4) : true

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (9/11)

- **Quantifiers : EXISTS and FORALL**
  - **FORALL**

    ex) FORALL PX (PX.COLOR = COLOR('Red'))

    (PX : bound)

    - **Universal quantifier**
    - **For all values of V, p evaluates to true**
    - **Every occurrence of variable V is true**
    - **An iterated AND :**

      If r, V, and p(V) are as in discussion of EXISTS, then the WFF *FORALL V(p(V))* is defined to be equivalent to the WFF *True AND p(t1) AND ... AND p(tm)*

      (- true if R is empty)

    - **Examples**

FORALL V	(V.C > 1)	: false
FORALL V	(V.B > 1)	: true
FORALL V	(V.A = 1 OR V.C > 2)	: true

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (10/11)

- **identity**
  - **FORALL V(p) = NOT EXISTS V (NOT p)**
  - ***"all V's satisfy p" is the same as " no V's do not satisfy p"***
  - ***"For all integers x, there exists an integer y such that y > x"***
  - ***"There does not exist an integer x such that there does not exist an integer y such that y > x"***

- **Free and Bound Variable References revisited**

EXISTS	**x**	**(x > 3)**		
$\equiv$ **EXISTS**	**y**	**(y > 3)**		
EXISTS	**x**	**(x > 3)**	**AND**	**x < 0**
$\equiv$ **EXISTS**	**y**	**(y > 3)**	**AND**	**x < 0**
$\not\equiv$ **EXISTS**	**y**	**(y > 3)**	**AND**	**y < 0**

  - **Closed WFF vs Open WFF**

spark@dblab.sogang.ac.kr

8.2 Tuple Calculus (11/11)

- **Relational Operations**
  - **Syntax**

    <relational operation>

    ::= <proto tuple> [WHERE <boolean expression>]

    <proto tuple>

    ::= <tuple expression>

    - **First, all references to range variables in the "proto tuple" must be free within that "proto tuple"**
    - **Second, a reference to a range variable in the WHERE clause can be free only if a reference to that very same range variable (necessarily free) appears in the corresponding "proto tuple"**
  - **Examples**
    - **" Get supplier numbers for suppliers in London"**

      SX.S# WHERE SX.CITY = 'London'

    - **" Get supplier names for suppliers who supply part P2"**

      SX.SNAME WHERE EXISTS SPX (SPX.S# = SX.S# AND SPX.P# = P# ('P2'))

spark@dblab.sogang.ac.kr

8.3 Examples (1/3)

- **8.3.1 Get supplier numbers and status for suppliers in Paris with status > 20**

  (SX.S#, SX.STATUS) WHERE SX.CITY = ‘Paris’ AND SX.STATUS > 20

- **8.3.2 Get all pairs of supplier number such that the two suppliers are located in the same city**

  (SX.S# AS SA, SY.S# AS SB)

  WHERE SX.CITY = SY.CITY AND SX.S# < SY.S#

- **8.3.3 Get full supplier information for suppliers who supply part P2**

  SX WHERE EXISTS SPX (SPX.S# = SX.S# AND

  SPX.P# = P#(‘P2’))

spark@dblab.sogang.ac.kr

8.3 Examples (2/3)

- **8.3.4 Get supplier names for suppliers who supply at least one red part**

  SX.SNAME WHERE EXISTS SPX (SX.S# = SPX.S# AND
  EXISTS PX (PX.P# = SPX.P#
  AND PX.COLOR = COLOR (‘Red’))

  prenex normal form, in which all quantifiers appear at the front of the WFF

  SX.SNAME WHERE EXISTS SPX (EXISTS PX (SX.S# = SPX.S# AND
  SPX.P# = PX.P# AND
  PX.COLOR = COLOR(‘Red’))

- **8.3.5 Get supplier names for suppliers who supply at least one part supplied by supplier S2**

  SX.SNAME WHERE EXISTS SPX (EXISTS SPY
  (SX.S# = SPX.S# AND
  SPX.P# = SPY.P# AND
  SPY.S# = S# (‘S2’))

- **8.3.6 Get supplier names for suppliers who supply all parts**

  SX.SNAME WHERE FORALL PX (EXISTS SPX (SPX.S# = SX.S# AND
  SPX.P# = PX.P#))

  without using FORALL

  SX.SNAME WHERE NOT EXISTS PX (NOT EXISTS SPX (SPX.S# = SX.S# AND
  SPX.P# = PX.P#))

spark@dblab.sogang.ac.kr

8.3 Examples (3/3)

- **8.3.7 Get supplier names for suppliers who do not supply part P2**

  SX.SNAME WHERE NOT EXISTS SPX
  (SPX.S# = SX.S# AND SPX.P# = P# (‘P2’))

- **8.3.8 Get supplier numbers for suppliers who supply at least all those parts supplied by supplier S2**

  SX.S# WHERE FORALL SPY (SPY.S# ≠ S#(‘S2’) OR
  EXISTS SPY (SPY.S# = SX.S# AND
  SPY.P# = SPX.P#))

  Logical implication : IF p THEN q END IF ≡ (NOT p) OR q

  SX.S# WHERE FORALL SPX (IF SPX.S# = S# (‘S2’) TEHN EXISTS SPY (SPY.S# = SX.S# AND SPY.P# = SPX.P#) END IF)

- **8.3.9 Get part numbers for parts that either weigh more than 16 pounds or are supplied by supplier S2, or both**

  RANGEVAR PU RANGES OVER
  (PX.P# WHERE PX.WEIGHT > WEIGHT (16.0)),
  (SPX.P# WHERE SPX.S# = S# (‘S2’)) ;

  PU.P#

spark@dblab.sogang.ac.kr

8.4 Relational Calculus vs. Relational Algebra (1/6)

- fundamentally equivalent to each other
- the algebra is at least as powerful as the calculus
- Codd's reduction algorithm
  - an arbitrary expression of the calculus could be reduced to a semantically equivalent expression of the algebra
- Example
  - Q : “ Get names and cities for suppliers who supply at least one Athens project with at least 50 of very part”

(SX.NAME, SX.CITY)

WHERE EXISTS JX FORALL PX EXISTS SPJX

(JX.CITY = 'ATHENES' AND

JX.J# = SPJX.J# AND PX.P# = SPJX.P# AND

SX.S# = SPJX.S# AND SPJX.QTY ≥ QTY(50))

spark@dblab.sogang.ac.kr

8.4 Relational Calculus vs. Relational Algebra (2/6)

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athenes

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

J

J#	JNAME	CITY
J1	Sorter	Paris
J2	Display	Rome
J3	OCR	Athenes
J4	Console	Athenes
J5	RAID	London
J6	EDS	Oslo
J7	Tape	London

SPJ

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

spark@dblab.sogang.ac.kr

8.4 Relational Calculus vs. Relational Algebra (3/6)

- **step 1**
  - for each tuple variable, retrieve the range(i.e., set of possible values for that variable), restricted if possible
  - SX : all tuples of S 5 tuples
  - PX : all tuples of P 6 tuples
  - JX : tuples of J where CITY = 'Athenes' 2 tuples
  - SPJX : tuples of SPJ where QTY ≥ QTY(50) 24 tuples
- **step 2**
  - construct the Cartesian product of the ranges retrieved in step 1
  - 5 * 6 * 2 * 24 = 1440 tuples

8.4 Relational Calculus vs. Relational Algebra (4/6)

- step 3
  - restrict the Cartesian product constructed in step 2 in accordance with the "join condition" portion of the WHERE clause
- step 4
  - apply the quantifiers from right to left as follows
  - for the quantifiers EXISTS RX
    - project the current result to eliminate all attributes of relation R
  - for the quantifiers FORALL RX
    - divide the current result by the "restricted range" relation associated with RX as retrieved in step 1

spark@dblab.sogang.ac.kr

8.4 Relational Calculus vs. Relational Algebra (5/6)

- **step 4 (continue)**
  - **(EXISTS SPJX) : project away the attributes of SPJ**

S#	SN	STATUS	CITY	P#	PN	COLOR	WEIGHT	CITY	J#	JN	CITY
S1	Sm	20	Lon	P1	Nt	Red	12.0	Lon	J4	Cn	Ath
S2	Jo	10	Par	P3	Sc	Blue	17.0	Rom	J3	OR	Ath
S2	Jo	10	Lon	P3	Sc	Blue	17.0	Rom	J4	Cn	Ath
S4	Cl	20	Ath	P6	Cg	Red	19.0	Lon	J3	OR	Ath
S5	Ad	30	Ath	P2	Bt	Green	17.0	Par	J4	Cn	Ath
S5	Ad	30	Ath	P1	Nt	Red	12.0	Lon	J4	Cn	Ath
S5	Ad	30	Ath	P3	Sc	Blue	17.0	Rom	J4	Cn	Ath
S5	Ad	30	Ath	P4	Sc	Red	14.0	Lon	J4	Cn	Ath
S5	Ad	30	Ath	P5	Cm	Blue	12.0	Par	J4	Cn	Ath
S5	Ad	30	Ath	P6	Cg	Red	19.0	Lon	J4	Cn	Ath

  - **(FORALL PX) : divide by relation P**

S#	SNAME	STATUS	CITY	J#	JNAME	CITY
S5	Adams	30	Athens	J4	Console	Athens

spark@dblab.sogang.ac.kr

8.4 Relational Calculus vs. Relational Algebra (6/6)

- step 4 (continue)
  - (EXISTS JX) project away the attributes of J

S#	SNAME	STATUS	CITY
S5	Adams	30	Athens

- step 5
  - project the result of step 4 in accordance with the specifications in the target item commalist
  - target item commalist : SX.SNAME, SX.CITY

SNAME	CITY
Adams	Athens

spark@dblab.sogang.ac.kr

8.5 Computational Capabilities (1/3)

- **syntax for *aggregate function reference***

  aggregate function (expression [, attribute])

  - **aggregate function : COUNT, SUM, AVG, MAX, MIN**
  - **expression : expression of the tuple calculus**
  - **attribute : attribute of that result relation over which the aggregation is to be done**

- **aggregate function**
  - **act as a new kind of quantifier**

    aggregate function ((target-commalist (WHERE f [, attribute])

- ***expression* and *attributes***
  - **avoid the need for SQL's *ad hoc* trick of using a DISTINCT operator to eliminate duplicates**

spark@dblab.sogang.ac.kr

8.5 Computational Capabilities (2/3)

- **8.5.1 Get the part number and the weight in grams for each part with weight > 10000 gram**

  (PX.P#, PX.WEIGHT * 454 AS GMWT)

  WHERE PX.WEIGHT* 454 > WEIGHT(10000.0)

- **8.5.2 Get all suppliers and tag each one with the literal value "Supplier"**

  (SX, 'Supplier' AS TAG)

- **8.5.3 For each shipment, get full shipment details, including total shipment weight**

  (SPX, PX.WEIGHT * SPX.QTY) AS SHIPWT

  WHERE PX.P# = SPX.P#

spark@dblab.sogang.ac.kr

8.5 Computational Capabilities (3/3)

- **8.5.4 For each part, get the part number and the total shipment quantity**

  (PX.P#, SUM (SPX WHERE SPX.P# = PX.P#, QTY) AS TOTQTY)

- **8.5.5 Get the total shipment quantity**

  SUM (SPX, QTY) AS GRANDTOTAL

- **8.5.6 For each supplier, get the supplier number and the total number of parts supplied**

  (SX.S#, COUNT (SPX WHERE SPX.S# = SX.S#) AS #_OF_PARTS)

- **8.5.7 Get part cities that store more than five red parts**

  ```
  RANGEVAR   PY   RANGES   OVER  P ;
  PX.CITY  WHERE  COUNT ( PY WHERE PY.CITY = PX.CITY
                                AND   PY.COLOR = COLOR('Red') ) > 5
  ```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (1/20)

- **8.6.1 Get color and city for "nonParis" parts with weight greater than ten pounds.**

```
SELECT   PX.COLOR, PX.CITY
FROM     P AS PX
WHERE    PX.CITY <> 'Paris'
AND      PX.WEIGHT > 10.0 ;
```

  - **Note the use of the comparison operator “<>”(not equals)**
  - **Note also the specification “ P AS PX” in the FROM clause**
  - **SQL also supports the notion of implicit range variables, according to which the query at hand might equally well have been expressed as follows:**

```
SELECT   P.COLOR, P.CITY
FROM     P
WHERE    P.CITY <> 'Paris'
AND      P.WEIGHT > 10.0 ;
```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (2/20)

- **Unqualified column names throughout this example**

```
SELECT    COLOR, CITY
FROM      P
WHERE     CITY <> 'Paris'
AND       WEIGHT > 10.0 ;
```

- **ORDER BY clause can also be used in interactive SQL queries**

```
SELECT    P.COLOR, P.CITY
FROM      P
WHERE     P.CITY <> 'Paris'
AND       P.WEIGHT > 10.0
ORDER  BY CITY DESC;
```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (3/20)

- **“SELECT * “ is shorthand for a commalist of all column names in the table(s) referenced in the FROM clause**

```
SELECT    *
FROM      P
WHERE     PX.CITY <> 'Paris'
AND       PX.WEIGHT > 10.0 ;
```

- **SQL does not eliminate redundant duplicate rows from a query result unless the user explicitly requests it to do so via keyword DISTINCT**

```
SELECT    DISTINCT P.COLOR, P.CITY
FROM      P
WHERE     PX.CITY <> 'Paris'
AND       PX.WEIGHT > 10.0 ;
```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (4/20)

- **8.6.2 For all parts, get the part number and the weight of that part in grams**

  SELECT P.P#, P.WEIGHT * 454 AS GMWT

  FROM P ;

  - **if the AS GMWT is omitted, the result column would have been unnamed**

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (5/20)

- **8.6.3 Get all combinations of supplier and part information such that the supplier and part in question are colocated**
  - **many different ways of formulating this query**

```
1) SELECT   S.*, P.P#, P.PNAME, P.COLOR, P.WEIGHT
   FROM     S, P
   WHERE    S.CITY = P.CITY;
```

```
2) S JOIN P USING CITY;  (JOIN support was added in SQL/92)
```

```
3) S NATURAL JOIN P;
```

- **First, the FROM clause is executed, to yield the Cartesian product S TIMES SP**
- **Next, the WHERE clause is executed, to yield a restriction of that product in which the two CITY values in each row are equal**
- **Finally, the SELECT clause is executed, to yield a project of that restriction over the columns specified in the SELECT clause**

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (6/20)

- **8.6.4 Get all pairs of city names such that a supplier located in the first city supplies a part stored in the second city**

```
SELECT    DISTINCT S.CITY AS SCITY, P.CITY AS PCITY
FROM      S JOIN SP USING S# JOIN P USING P# ;
```

/* incorrect version */

```
SELECT    DISTINCT S.CITY AS SCITY, P.CITY AS PCITY
FROM      S NATURAL JOIN SP NATURAL JOIN P ;
```

  - **because it includes CITY as a joining column in the second join**

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (7/20)

- **8.6.5 Get all pairs of supplier numbers such that the two suppliers concerned are colocated**

```
SELECT    A.S# AS SA, B.S# AS SB
FROM      S AS A, S AS B
WHERE     A.CITY = B.CITY
AND       A.S# < B.S# ;
```

  - **explicit range variables *FIRST and SECOND***

8.6 SQL Facilities (8/20)

- **8.6.6 Get the total number of suppliers**

```
SELECT    COUNT(*) AS N
FROM      S ;
```

- **usual set of aggregate functions(COUNT, SUM,AVG, MAX and MIN)**
- **optionally, DISTINCT**
- **MAX, MIN : DISTINCT has no effect**
- **COUNT(*) :**
  - **DISTINCT not allowed**
  - **all rows in a table without any duplicate elimination**
  - **except for the case of COUNT(*), any nulls are not eliminated**

8.6 SQL Facilities (9/20)

- **8.6.7 Get the maximum and minimum quantity for part P2**

```
SELECT MAX(SP.QTY) AS MAXQ, MIN (SP.QTY) AS MINQ
FROM      SP
WHERE     SP.P# = 'P2' ;
```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (10/20)

- **8.6.8 For each part supplied, get the part number and the total shipment quantity**

```
SELECT      SP.P#, SUM(SP.QTY) AS TOTQTY
FROM        SP
GROUP       BY  SP.P# ;
```

  - **if the GROUP BY clause is specified, expressions in the SELECT clause must be *<u>single-valued per group</u>***
  - **alternative formulation(*nested exression* to represent *scalar items* was added in *SQL*/92)**

```
SELECT P.P#,        (SELECT SUM (SP.QTY)
                     FROM    SP
                     WHERE SP.P#=P.P# ) AS TOTQTY
FROM  P ;
```

8.6 SQL Facilities (11/20)

- **8.6.9 Get part numbers for all parts supplied by more than one supplier**

```
SELECT     SP.P#
FROM       SP
GROUP BY   SP.P#
HAVING     COUNT(SP.S#) > 1 ;
```

  - **HAVING clause is to groups what the WHERE clause is to rows**

8.6 SQL Facilities (12/20)

- **8.6.10 Get supplier names for suppliers who supply part P2**

```
SELECT      DISTINCT S.SNAME
FROM        S
WHERE       S.S#   IN
            (SELECT SP.S#
             FROM   SP
             WHERE  SP.P# = 'P2' ) ;
```

is equivalent to

SELECT DISTINCT S.SNAME FROM S WHERE S.S# IN ('S1','S2','S3','S4') ;	SELECT DISTINCT S.SNAME FROM S, SP WHERE S.S# = SP.S# AND SP.P# = 'P2' ;

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (13/20)

- **8.6.11 Get supplier names for suppliers who supply at least one red part**

```
SELECT    DISTINCT S.SNAME
FROM      S
WHERE     S.S#   IN
          (SELECT SP.S#
           FROM   SP
           WHERE SP.P# IN
                  (SELECT P.P#
                   FROM   P
                  WHERE P.COLOR = 'Red' ) ) ;
```

  - **EXERCISE : give some equivalent join formulations of this query**

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (14/20)

- **8.6.12 Get supplier numbers for suppliers with status less than the current maximum status in the S table**

```
SELECT      S.S#
FROM        S
WHERE       S.STATUS <
            ( SELECT MAX ( S.STATUS )
              FROM   S ) ;
```

  - **two distinct implicit range variables, both denoted by the same symbol 'S' and both ranging over the S table**

8.6 SQL Facilities (15/20)

- **8.6.13 Get the supplier names for suppliers who supply part P2**

```
SELECT DISTINCT    S.SNAME
FROM               S
WHRE               EXISTS
                  ( SELECT  *
                    FROM SP
                   WHERE SP.S# = S.S#
                   AND     SP.P#  = 'P2' ) ;
```

  - **SQL EXISTS function : existential quantifier of relational calculus**
  - **"EXIST" evaluates to true if and only if the result of evaluating the " SELECT .. FROM .." is not empty**

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (16/20)

- **8.6.14 Get supplier names for suppliers who do not supply part P2**

```
SELECT     DISTINCT S.SNAME
FROM       S
WHERE      NOT EXISTS
           ( SELECT *
             FROM  SP
            WHERE  SP.S# = S.S#
            AND       SP.P# = 'P2' ) ;
```

- **alternatively**

```
SELECT DISTINCT S.SNAME
FROM       S
WHERE      S.S#  NOT  IN
           ( SELECT SP.S#
            FROM   SP
            WHERE SP.P# = 'P2' ) ;
```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (17/20)

- **8.6.15 Get supplier names for suppliers who supply all parts**

```
SELECT DISTINCT S.SNAME
FROM        S
WHERE       NOT EXISTS
            ( SELECT *
              FROM  P
             WHERE NOT EXISTS
                        (SELECT  *
                         FROM   SP
                        WHERE SP.S# = S.S#
                        AND      SP.P# = P.P# ) ) ;
```


spark@dblab.sogang.ac.kr

8.6 SQL Facilities (18/20)

- universal quantifier FORALL is not supported directly
- to be expressed in terms of existential quantifiers and double negation

```
SELECT      DISTINCT  S.SNAME
FROM        S
WHERE       ( SELECT COUNT (SP.P#)
             FROM   SP
            WHERE SP.S# = S.S# ) =
            ( SELECT COUNT(P.P#)
              FROM P ) ;
```

  - **Equivalent only because a certain integrity constraint is in effect**
  - **formulation to compare two counts was not supported in SQL but was added in SQL/92**
  - **What we would really like to do is compare two tables**

8.6 SQL Facilities (19/20)

```
SELECT      DISTINCT  S.SNAME
FROM        S
WHERE       ( SELECT SP.P#
              FROM   SP
            WHERE SP.S# = S.S# ) =
            ( SELECT P.P#
              FROM P ) ;
```

- **SQL does not directly support comparisons between tables, however, and so we have to resort to the trick of comparing table cardinalities instead**

8.6 SQL Facilities (20/20)

- **8.6.16 Get part number for parts that either weigh more than 16 pounds or are supplied by supplier S2, or both**

```
SELECT   P.P#
FROM     P
WHERE    P.WEIGHT > 16.0
UNION
SELECT   SP.P#
FROM     SP
WHERE    SP.S# = 'S2' ;
```

  - **redundant duplicate rows are always eliminated from the result of unqualified UNION, INTERSECT, EXCEPT(MINUS)**
  - **qualified form : UNION ALL, INTERSECT ALL, EXCEPT ALL**

spark@dblab.sogang.ac.kr

8.6 SQL Facilities (20/20)

- **8.6.17 Get part number and the weight in grams for each part with weight > 10,000**

```
SELECT      P.P# , P.WEIGHT * 454 AS GMWT
FROM        P
WHERE       P.WEIGHT * 454 > WEIGHT ( 10000.0 )

WITH T1 AS ( SELECT P.P#, P.WEIGHT * 454 AS GMWT
             FROM P )
     SELECT  T1.P#, T1.GMWT
     FROM    T1
     WHERE   T1.GMWT > WEIGHT ( 10000.0 )  ;
```

  - **Avoid having to write the expression P.WEIGHT * 454 out twice**

spark@dblab.sogang.ac.kr

8.7 Domain Calculus (1/5)

- domain calculus expression
  - domain variables D, E, F, ...
  - conditions
    - X * Y where x and y are domain variables
    - membership conditions

      R(pair, pair, ...) where R is a relational name and “pair” is of the form A:v

      A : is an attribute of R

      v : is either a domain variable or a literal

      - TRUE iff there exists a tuple in relation R having the specified values for the specified attributes
  - WFFs
    - F1 ~ F5
  - Expressions

    D,E, ... , F [where f]

    where D, E, .., F : domain variables

    f : WFF containing exactly D, E, ... F as free variables

spark@dblab.sogang.ac.kr

8.7 Domain Calculus (2/5)

- **Examples of domain calculus expressions**

(SX)

(SX) WHERE S (S#:SX)

(SX) WHERE S (S#:SX, CITY:'London')

(SX, CITYX) WHERE S (S#:SX, CITY:CITYX)
AND SP (S#:SX, P#:P#('P2'))

(SX , PX) WHERE S (S#:SX , CITY.CITYX)
AND P (P#:PX, CITY.CITYY)
AND CITYX $\neq$ CITYY

spark@dblab.sogang.ac.kr

8.7 Domain Calculus (3/5)

- **8.7.1 Get supplier numbers for suppliers in Paris with status > 20**

  SX WHERE EXISTS STATUSX

  (STATUSX > 20 AND S (S#:SX, STATUS:STATUSX, CITY:'Paris'))

- **8.7.2 Get all pairs of supplier numbers such that the two suppliers are colocated**

  (SX AS SA, SY AS SB)

  WHERE EXISTS CITYZ

  (S (S#:SX, CITY:CITYZ) AND

  S (S#:SY, CITY:CITYZ) AND

  SX < SY)

- **8.7.3 Get supplier names for suppliers who supply at least one red part**

  NAMEX WHERE EXISTS SX EXISTS PX

  (S (S#:SX, SNAME:NAMEX)

  AND SP (S#:SX, P#:PX) AND P (P#:PX, COLOR:COLOR('Red')))

spark@dblab.sogang.ac.kr

8.7 Domain Calculus (4/5)

- **8.7.4 Get supplier names for suppliers who supply at least one part supplied by supplier S2**

```
NAMEX WHERE EXISTS SX EXISTS PX
        (S (S#:SX, SNAME:NAMEX)
         AND  SP ( S#:SX, P#:PX ) AND SP (S#:S#('S2'), P#:PX) )
```

- **8.7.5 Get supplier names for suppliers who supply all parts**

```
NAMEX WHERE EXISTS SX ( S (S#:SX, SNAME:NAMEX )
        AND FORALL PX (  IF P ( P#:PX )
                         THEN SP ( S#:SX, P#:PX )
                       END IF )
```

- **8.7.6 Get supplier names for suppliers who do not supply part P2**

```
NAMEX WHERE EXISTS SX ( S ( S#:SX, SNAME:NAMEX )
                        AND NOT SP ( S#:SX, P#:P#('P2' ) ) )
```


spark@dblab.sogang.ac.kr

8.7 Domain Calculus (5/5)

- **8.7.7 Get supplier numbers for suppliers who supply at least all those parts supplied by supplier S2**

```
SX WHERE FORALL PX ( IF SP ( S#:S#('S2'), P#:PX )
                     THEN SP ( S#:SX, P#:PX )
                  END IF )
```

- **8.7.8 Get part numbers for parts that either weigh more than 16 pounds or are supplied by supplier S2, or both**

```
PX WHERE EXISTS WEIGHTX
      ( P ( P#:PX, WEIGHT:WEIGHTX )
          AND WEIGHTX > WEIGHT(16.0) )
      OR SP ( S#:S#('S2'), P#:PX )
```


spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (1/10)

- **Query-By-Example (QBE)**

  The best-known example of a language based on the domain calculus.
- **Example**
  - **Get supplier names for suppliers who supply at least one part supplied by supplier S2**

S	S#	SNAME
	_SX	P._NX

SP	S#	P#
	_SX	_PX

SP	S#	P#
	S2	_PX

  - **Explanation**

    The user is asking the system to *present* (“P.”) supplier names (_NX).

    If the supplier number is _SX, then supplier _SX supplies some part _PX, and part _PX in turn is supplied by supplier S2.

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (2/10)

- 8.8.1 Get supplier numbers for suppliers in Paris with status >20

S	S#	SNAME	STATUS	CITY
	P.		> 20	Paris

- It is also possible to specify "P." against the entire row.

S	S#	SNAME	STATUS	CITY
P.			> 20	Paris

- This example is equivalent to specifying "P." in every column position in the row.

S	S#	SNAME	STATUS	CITY
	P.	P.	P. > 20	P.Paris

- The system will provide facilities to allow black tables to be edited on the screen by the addition or removal of columns and rows.

S	S#	STATUS	CITY
	P.	> 20	Paris

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (3/10)

- **8.8.2 Get part numbers for all parts supplied, with redundant duplicates eliminated**

SP	S#	P#	QTY
UNQ.		P.	

  - UNQ. Stands for unique (it corresponds to DISTINCT in SQL).

- **8.8.3 Get supplier numbers and status for suppliers in Paris, in ascending supplier number order within descending status order**

S	S#	STATUS	CITY
	P.AO(2).	P.DO(1).	Paris

  - “AO.” stands for ascending order, “DO.” for descending order.
  - The integers in parentheses indicate the major-to-minor sequence for ordering columns. Ex) STATUS is major, S# is the minor column.

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (4/10)

- **8.8.4 Get supplier numbers and status for suppliers who either are located in Paris or have status > 20, or both** (modified version of 8.8.1)
  - To “OR” two conditions, they must be specified in different rows.

S	S#	STATUS	CITY
	P.		Paris
	P.	> 20	

  - Another approach to this query makes use of what is known as a ***condition box***.

S	S#	STATUS	CITY
	P.	_ST	_SC

CONDITIONS
_SC = Paris OR _ST > 20

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (5/10)

- 8.8.5 Get parts whose weight is in the range 16 to 19 inclusive

P	P#	WEIGHT	WEIGHT
	P.	>= 16.0	<= 19.0

- 8.8.6 For all parts, get the part number and the weight of the part in grams

P	P#	WEIGHT	GMWT
	P.	_PW	P. _PW * 454

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (6/10)

- **8.8.7 Get supplier names for suppliers who supply part P2**

S	S#	SNAME
	_SX	P.

SP	S#	P#
	_SX	P2

- The query can be paraphrased:

  Get supplier names for suppliers SX such that there exists a shipment showing supplier SX supplying part P2.

- QBE does implicitly support EXISTS. However, it does not support NOT EXISTS.

  ex) "Get supplier names for suppliers who supply all parts" cannot be expressed in QBE, and QBE is not relationally complete.

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (7/10)

- **8.8.8 Get all supplier-number/part-number pairs such that the supplier and part concerned are "colocated"**

S	S#	CITY
	_SX	_CX

P	P#	CITY
	_PX	_CX

P.	_SX	_PX

  - Three blank tables are needed for this query, one each for S and P and one for the result.
  - The entire query can be paraphrased:

    Get supplier-number/part-number pairs, SX and PX say, such that SX and PX are both located in the same city CX.

- **8.8.9 Get all pairs of supplier numbers such that the suppliers concerned are colocated**

S	S#	CITY
	_SX	_CZ
	_SY	_CZ

P.	_SX	_SY

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (8/10)

- **8.8.10 Get the total quantity of part P2 supplied**

SP	S#	P#	QTY	
		P2	_QX	P.SUM._QX

  - QBE supports the usual aggregate operators.

- **8.8.11 For each part supplied, get the part number and the total shipment quantity**

SP	S#	P#	QTY	
		G.P.	_QY	P.SUM._QY

  - “G.” causes grouping (it corresponds to GROUP BY in SQL).

- **8.8.12 Get part numbers for all parts supplied by more than one supplier**

SP	S#	P#
	_SX	G.P.

CONDITIONS
CNT._SX > 1

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (9/10)

- 8.8.13 Get part numbers for parts that either weigh more than 16 pounds or are supplied by supplier S2, or both

P	P#	WEIGHT
	_PX	> 16.0

SP	S#	P#
	S2	_PY

P.	_PX
P.	_PY

- 8.8.14 Insert part P7 (city Athens, weight 24, name and color at present unknown) into table P

P	P#	PNAME	COLOR	WEIGHT	CITY
I.	P7			24.0	Athens

  - “I.” applies to the entire row and so appears beneath the table name.

spark@dblab.sogang.ac.kr

8.8 QUERY-BY-EXAMPLE (10/10)

- **8.8.15 Delete all shipments with quantity greater than 300**

SP	S#	P#	QTY
D.			> 300

  - The "D." appears beneath the table name.

- **8.8.16 Change the color of part P2 to yellow, increase the weight by 5, and set the city to Oslo**

P	P#	PNAME	COLOR	WEIGHT	WEIGHT	CITY
	P2		U.Yellow	_WT	U._WT + 5	U.Oslo

- **8.8.17 Set the shipment quantity to five for all suppliers in London**

SP	S#	QTY
	_SX	U. 5

S	S#	CITY
	_SX	London

spark@dblab.sogang.ac.kr