An Introduction to Database Systems

Chapter 3. An Introduction to Relational Databases

spark@dblab.sogang.ac.kr

3.1 Introduction

- preliminary and informal introduction to the relational approach
- In Part II, Theoretical foundations

spark@dblab.sogang.ac.kr

3.2 Relational Systems(1/12)

- minimum definition
  1. Structural aspect :
     - The data is perceived by the user as tables and nothing but tables
  2. Integrity aspect :
     - Those tables satisfy certain integrity constraints
  3. Manipulation aspect :
     - The operators available to the user for manipulating those tables(e.g, for data retrieval) are :
       - operators that generate new tables from old
       - those operators include at least SELECT(RESTRICT), PROJECT, JOIN

spark@dblab.sogang.ac.kr

3.2 Relational Systems(2/12)

- loose definitions of those operations
  - SELECT(known as RESTRICT)
    - extracts specified rows from a table
  - PROJECT
    - extracts specified columns from a table
  - JOIN
    - joins together two tables on the basis of common values in a common column

spark@dblab.sogang.ac.kr

3.2 Relational Systems(3/12)

- the departments-and-employees databases(sample values)

DEPT

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M
D3	Research	5M

EMP

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Satio	D2	35K

spark@dblab.sogang.ac.kr

3.2 Relational Systems(4/12)

☐ RESTRICT, PROJECT(examples)

RESTRICT :

DEPTs where BUDGET > 8M

Result

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M

PROJECT :

DEPTs over DEPT#, BUDGET

Result

DEPT#	BUDGET
D1	10M
D2	12M
D3	5M

spark@dblab.sogang.ac.kr

3.2 Relational Systems(5/12)

- JOIN (example) :

DEPTs and EMPs over DEPT#

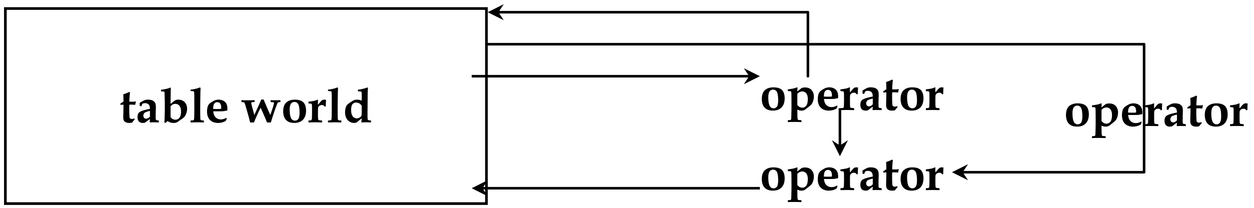
Result

DEPT#	DNAME	BUDGET	EMP#	ENAME	SALARY
D1	Marketing	10M	E1	Lopez	40K
D1	Marketing	10M	E2	Cheng	42K
D2	Development	12M	E3	Finzi	30K
D2	Development	12M	E4	Satio	35K

spark@dblab.sogang.ac.kr

3.2 Relational Systems(6/12)

- the relational property of ***closure***
  - the result of each of the three operation is another table
  - the output of any operation is the can become input to another
    - ex) a projection of a join, a join of two restrictions
    - nested expressions



spark@dblab.sogang.ac.kr

3.2 Relational Systems(7/12)

- **from a conceptual point of view,**
  - **the output from each operation is another table**
    - **does not materialize the result of every individual operation in entirety.**
    - **If intermediate results are fully materialized → materialized evaluation**
    - **If intermediate results are passed to subsequent operations →pipelined evaluation**
- **the operations are**
  - **all set-at-a-time, not row(record)-at-a-time**
  - **operands and results are table, not single row**

spark@dblab.sogang.ac.kr

3.2 Relational Systems(8/12)

- set processing capability
  - a major distinguishing characteristic of relational system systems
  - in non-relational systems, the operations are row- or record-at-a-time level

spark@dblab.sogang.ac.kr

3.2 Relational Systems(9/12)

- ***Figure 3.1***

1) Logical structure **:** *perceived by the user as* ***tables***

  - **are logical structure, not the physical structure**
  - **at physical level,**
    - **be free to use any or all of storage structure**
    - **e.g, sequential files, indexing, hashing, pointer chains, compression, etc..**
  - **abstraction of way the data is physically stored**
  - **i.e, all hidden from the user**

spark@dblab.sogang.ac.kr

3.2 Relational Systems(10/ 12)

2) ***The information Principle*** **: The entire information content of the database is represented in one and only one way, namely as *explicit data values***

- **only method available in a relational database.**
- **no pointers connecting one table to another**
- **ex) connection between the D1 row of table DEPT and the E1 row of table EMP (connection is represented by appearance of the value D1, not by a pointer)**
- **in nonrelational system, connection information is represented by some kind of pointer that is explictly visible to the user**

spark@dblab.sogang.ac.kr

3.2 Relational Systems(11/ 12)

- **Integrity aspect :**
  - Database might be required to satisfy any number of integrity constraints
    - Ex) 25K ≤ salary ≤ 95K, 1M ≤ budget ≤ 15M
  - Each row in table DEPT must include a unique DEPT# value, likewise, each row in table EMP must include a unique EMP# value
  - Each DEPT# value in table EMP must exist as a DEPT# value in table DEPT(to reflect the fact that every employee must be assigned to an existing department)
- **Primary key :**
  - columns DEPT# in table DEPT
  - columns EMP# in table EMP
- **Foreign key**
  - Column DEPT# in table EMP

spark@dblab.sogang.ac.kr

3.2 Relational Systems(12/12)

- The relational model consists of the five components
  1. scalar types
  2. a relation type generator
  3. relation variables
  4. relational assignment operation
  5. relational operators

spark@dblab.sogang.ac.kr

3.3 Relations and RELVARS (1/3)

- Relational Systems are based on the relational model
  - an abstraction theory of data
  - set theory and predicate logic
- E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", CACM, 1970 June.
- a wide-ranging influence
  - database technology, AI, natural-language proecssing,
    HW system design

spark@dblab.sogang.ac.kr

3.3 Relations and RELVARS (2/3)

- many of the terms are fuzzy
  - ex) record(logical, physical, stored, virtual)
- terminology
  - relation vs. table
  - tuple vs. record, row
  - attribute vs. column

spark@dblab.sogang.ac.kr

3.3 Relations and RELVARS (3/3)

- relation variables (relvar)
  - variables whose values are relation values
    - different relation values at different times

DELETE EMP WHERE EMP# = 'E4';

EMP

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K

EMP := EMP MINUS (EMP WHERE EMP# = 'E4');

Old relation value of EMP has been replaced by an entirely new relation value

spark@dblab.sogang.ac.kr

3.4 What Relations Mean (1/3)

- Columns has associated data types
- Users will be able to define their own types
  - TYPE EMP# … ;
  - TYPE NAME … ;
  - TYPE DEPT# … ;
  - TYPE MONEY … ;
- Every relation (value) has two parts
  - (heading) column-name : type-name pair
  - (body) a set of rows that conform to that heading

EMP

EMP# : EMP#	ENAME : NAME	DEPT# : DEPT#	SALARY : MONEY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

spark@dblab.sogang.ac.kr

3.4 What Relations Mean (2/3)

- A way of thinking about relations
  1. Given a relation r, the heading of r denotes a certain predicate or truth-valued function
  2. Each row in the body of r denotes a certain true proposition, obtained from the predicate by substituting certain argument values of the appropriate type the for the placeholders(instantiating predicate)
- Predicate
  - 'Employee EMP# is named ENAME, works in department DEPT#, and earns salary SALARY'
- True proposition
  - 'Employee E1 is named Lopez, works in department D1, and earns salary 40K'
- Types are (set of) things we can talk about

spark@dblab.sogang.ac.kr

3.4 What Relations Mean (3/3)

- Types are (set of) things we can talk about;
- Relations are (set of) things we say about the things we can talk about.

cf) Types are to relations as nouns are to sentences

1. Types and Relations are both necessary (without types, we have nothing to talk about; without relations, we cannot say anything)
2. Types and Relations are sufficient as well as necessary

- Types and Relations are not the same thing

- Every relation has an associated predicate

spark@dblab.sogang.ac.kr

3.5 Optimization (1/4)

- SQL : set-level operations (select, project, join)
  - non-procedural language
  - user specify what, not how
- automatic navigation system
  - the process of *"navigating"* is performed automically by the system, *not manually by the user*
  - in nonrelational system, manual navigation
  - Fig 3.5 automatic vs. manual navigation

spark@dblab.sogang.ac.kr

❑Fig 3.5 automatic vs. manual navigation

```
INSERT INTO SP ( S#, P#, QTY )
VALUES ( 'S4', 'P3', 1000 ) ;
```

```
MOVE 'S4' TO S# IN S
FIND CALC S
ACCEPT S-SP-ADDR FROM S-SP CURRENCY
 FIND LAST SP WITHIN S-SP
While SP found PERFORM
   ACCEPT S-SP-ADDR FROM S-SP CURRENCY
   FIND OWNER WITHIN P-SP
   GET P
   IF P# IN P < 'P3'
      leave loop
   END-IF
   FIND PRIOR SP WITHIN S-SP
END-PERFORM
MOVE 'P3' TO P# IN P
FIND CALC P
ACCEPT P-SP-ADDR FROM P-SP CURRENCY
FIND LAST SP WITHIN P-SP
```

```
While SP found PERFORM
   ACCEPT P-SP-ADDR FROM P-SP CURRENCY
   FIND OWNER WITHIN S-SP
   GET S
    IF S# IN S < 'S4'
      leave loop
   END-IF
   FIND PRIOR SP WITHIN P-SP
END-PERFORM
MOVE 1000 TO QTY IN SP
FIND DB-KEY IS S-SP-ADDR
FIND DB-KEY IS P-SP-ADDR
STORE SP
CONNECT SP TO S-SP
CONNECT SP TO P-SP
```


spark@dblab.sogang.ac.kr

3.5 Optimization (2/4)

- relational languages such as SQL at a higher level of abstraction than C and Cobol
- Optimizer ; DBMS component
  - how to perform the automatic navigation
  - choose efficient way to implement user request

spark@dblab.sogang.ac.kr

3.5 Optimization (3/4)

- Let us suppose the user request
  - RESULT := (EMP Where EMP# = 'E4') [SALARY]
- two ways of performing the necessary data access
  - By doing a physical sequential scan of(the stored version of) table EMP until the required record is found
  - If there is an index on(stored version of) the EMP# column of that table, then by using that index and thus going directly to the E4 data.
    - EMP# is primary key
    - Most systems require an index on the primary key

spark@dblab.sogang.ac.kr

3.5 Optimization (4/4)

- Basis of strategy
  - which tables are referenced in the request
    - (there may be more than one if, e.g., there are any joins involved)
  - How bigs those tables are
  - What indexes exist
  - How selective those indexes are
  - How the data is physically clustered on the disk
  - What relational operations are involved

spark@dblab.sogang.ac.kr

3.6 The Catalog (1/5)

- catalog or dictionary function
  - the place where
    - all of the various schema(external, conceptual, internal)
    - all of the corresponding mappings (external/conceptual, conceptual/internal)
  - detailed informations(descriptor information or metadata) regarding the various objects to the system itself
    - such as tables, indexes, users, integrity rules, security rules,...

spark@dblab.sogang.ac.kr

3.6 The Catalog (2/5)

- **examples**
  - **the optimizer uses catalog information about indexes**
  - **the security subsystem uses catalog information about users and security rules**
- ***the catalog itself consists of tables***
  - **system tables**

spark@dblab.sogang.ac.kr

3.6 The Catalog (3/5)

- Users can interrogate the catalog in exactly the same way as they interrogate their own data
- two system tables
  - TABLE, COLUMN
  - named tables known to the system
  - self-describing(includes catalog tables)

spark@dblab.sogang.ac.kr

3.6 The Catalog (4/5)

- **TABLE and COLUMN of departments-and-employees database**

TABLE

TABNAME	COLCOUNT	ROWCOUNT	
DEPT	3	3	
EMP	4	4	
...	...	...	
TABLES			
COLUMNS			

COLUMN

TABNAME	COLNAME	
DEPT	DEPT#	
DEPT	DNAME	
DEPT	BUDGET	
EMP	EMP#	
EMP	ENAME	
EMP	DEPT#	
EMP	SALARY	
TABLES	TABNAME	
	COLCOUNT	
	ROWCOUNT	

spark@dblab.sogang.ac.kr

3.6 The Catalog (5/5)

- Suppose some user wants to to know what column the DEPT table contains.

  (COLUMN WHERE TABNAME = 'DEPT') [COLNAME]

- "Which tables include a column called EMP# ? "

  (COLUMN WHERE COLNAME = 'EMP#') [TABNAME]

- exercise : what does the following do ?

  ((TABLE JOIN COLUMN)

  WHERE COLCOUNT < 5) [TABNAME, COLNAME]

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (1/7)

- base relvar
  - named relvar
  - independent existence
- derived relvar
  - obtained from base relvars
  - depend on base relvars
  - defined in terms of other relvars

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (2/7)

- a means for creating the base relvars in SQL
  - CREATE TABLE statements
- derived relvars are not named
  - one particular kind of derived relvars : view have a name
  - but, does not have an independent existence
  - ex) CREATE VIEW TOPEMPS AS
    (EMP WHERE SALARY > 33K) [EMP#, ENAME, SALARY]

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (3/7)

- TOPEMPs as a view

TOPEMPS

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D12	42K
E3	Finzi	D2	30K
E4	Satio	D2	35K

- virtual relvar
  - the table that would result if the view-defining expression were actually evaluated

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (4/7)

- view is effectively just a window into the underlying relvar EMP(not a seperate copy of data)
- example of a query involving view TOPEMPS
  - (TOPEMPS WHERE SALARY < 42K) [EMP#, SALARY]

EMP#	SALARY
E1	40K
E4	35K

  - operation against a view : replacing references to the view by the expression that defines the view
  - the expression is saved in the catalog

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (5/7)

- the expression

  (TOPEMPS WHERE SALARY < 42K) [EMP#, SALARY]

is modified by the system

  ((EMP WHERE SALARY > 33K) [EMP#,ENAME, SALARY]) WHERE SALARY < 42K) [EMP#, SALARY]

after rearrangement

  (EMP WHERE SALARY >33K AND SALARY <42K) [EMP#, SALARY]

- The original operation against the view is effectively converted into an equivalent operation against the underlying base relvar

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (6/7)

- view definition can be of arbitrary complexity
  - TOPEMPS - a row-and-column subset
  - ex) view definition includes a join of two base relvars

- View has a specific meaning in relational contexts that is not identical to ANSI/SPARC
  - In relational systems, a view is a named, derived, virtual relvar
  - the relational analog of an ANSI/SPARC "external view" is a collection of several relvars

spark@dblab.sogang.ac.kr

3.7 Base Relvars and Views (7/7)

- base relvar vs. view distinction is
  - Base relvars "really exist", in the sense that they represent data that is actually stored in the database;
  - Views, by contrast, do not "really exist" but merely provide different ways of looking at the "real" data.

spark@dblab.sogang.ac.kr

3.8 Transactions

- A transaction is a logical unit of work
- The user needs to be able to inform the system when distinct operations are part of the same transaction
- BEGIN TRANSACTION, COMMIT, ROLLBACK operations

```
BEGIN TRASACTION ; /* move $$$ from account A to account B */
UPDATE account A ;        /* withdrawal      */
UPDATE account B ;        /* deposit         */
IF everything worked fine
    THEN COMMIT;          /* normal end      */
    ELSE ROLLBACK;        /* abnormal end    */
END IF ;
```

- Points
  1. atomicity
  2. durability
  3. isolation
  4. consistency
  5. serializability

spark@dblab.sogang.ac.kr

3.9 Suppliers-and-Parts Database (1/3)

S

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

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

spark@dblab.sogang.ac.kr

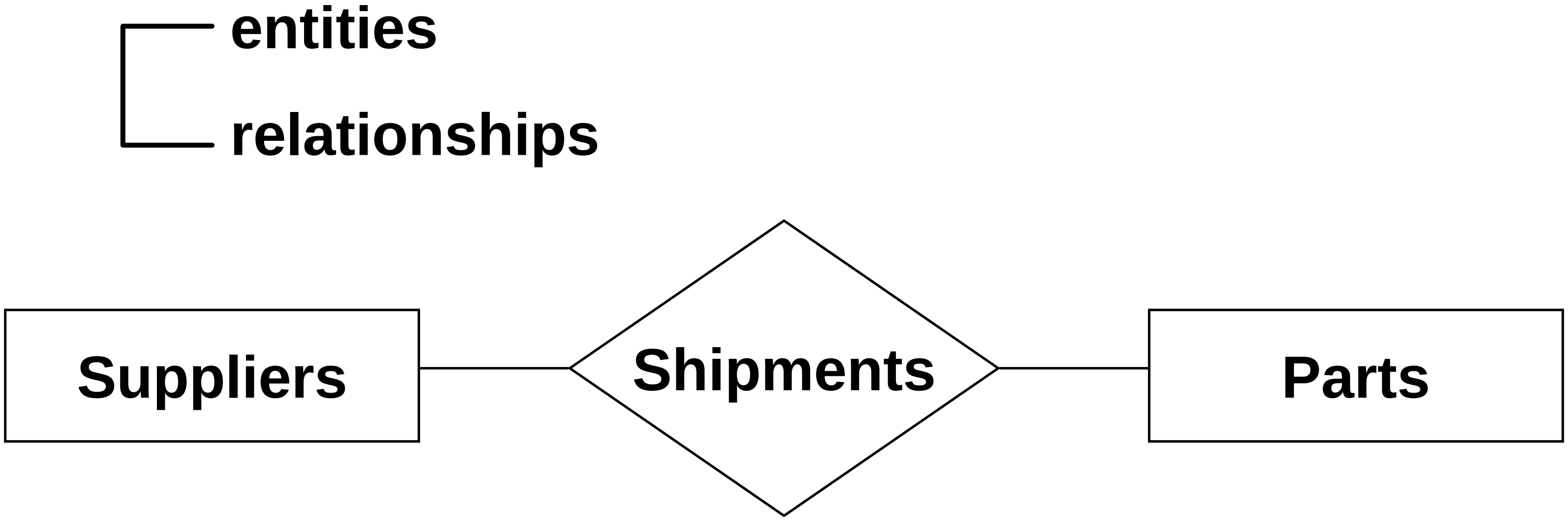
3.9 Suppliers-and-Parts Database (2/3)

```
TYPE S# ... ;
TYPE NAME ... ;
TYPE P# ... ;
TYPE COLOR ... ;
TYPE WHIGHT ... ;
TYPE QYT ... ;

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

VAR P BASE RELATION
        { P#          P#,
          PNAME       NAME,
          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 ;
```


spark@dblab.sogang.ac.kr

3.9 Suppliers-and-Parts Database (3/3)



spark@dblab.sogang.ac.kr