Notes on Relational Theory

Ahmed S. Darwish[*]
darwish.07@gmail.com

July 16, 2010

0.1 The relational model

The relational model stresses on *separation of concerns* by means of data independence: application programs should not be logically impaired cause of growth in data types and changes in data representation. It provides means to describe data in its natural structure only, without imposing *any* additional structure for machine representation.[1]

This data model central concept is the *relation* in its mathematical set theory sense. Relations were informally described by De Morgain as

> When two objects, qualities, classes, or attributes, viewed together by the mind, are seen under some connexion, that connexion is called a relation.

where an example is the fatherhood relation between a person-male father, and a person child whether male or female. Such a relation can be represented as $F = \{\langle Adam,\ Jane\rangle,\ \langle Adam,\ Casey\rangle,\ \ldots\}$ where $\langle X, Y\rangle \in F$ implies person X as a father of Y in the miniworld the relation represents.

[*]These notes are based on the databases book cited at [4]. Excerpts from E. F. Codd's papers are included in the first section.

Mathematically, given sets $S_1, S_2, \ldots, S_n$ representing domains that are not necessarily distinct, R is called a relation on these n sets if it is a set of n-tuples[1] each of which has its first element from S_1, its second element from S_2, and so on. More concisely:

$$R \subseteq S_1 \times S_2 \times \ldots \times S_n$$

where S_j is referred as the jth domain of R, and '$\times$' is the cartesian product operator.

Relations are sets, but not all sets are relations. The elements of a relation of degree n are called n-tuples or tuples. A relation R has the following properties: (1) each row represents an n-tuple of R, (2) the ordering of rows is insignificant, (3) rows are distinct, (4) the ordering of columns is significant cause it corresponds to the ordering $S_1, S_2, \ldots, S_n$ of the domains[2]. Denoting $|X|$ as the number of elements in set X, we have:

$$|R| \leq |S_1| * |S_2| * \ldots * |S_n|$$

One domain (or combination of domains) of a given relation has values which uniquely identify each n-tuple element of that relation. Such a domain (or a combination) is called the *primary key.* A domain (or a domain combination) of relation R is a *foreign key* if it's not the primary key of R but its elements are values of the primary key of some relation S, where the possibility of $S \equiv R$ is not excluded.

The domain elements are *atomic* (nondecomposable) values within the system data model. A relation is in its *first normal form* if it has the property that none of its domains has elements which are themselves sets. An *unnormalized relation* is one which is not in its first normal

[1]An n-tuple is an ordered list of n elements.

[2]This property has been relaxed in further iterations of the model by mandating a distinct (column) name for each participating domain role.

form. The first normal form is sometimes called the *flat relational model*; much of the relational model theory was developed with this flat model in mind.

Further notation was eventually added to the relational model. A *relation schema* $R(A_1, A_2, \ldots, A_n)$ is made-up of relation name R and the list of attributes $A_1, \ldots, A_n$, where each attribute is a name of a *role*[3] describing a relation-participating domain set. A relation instance $r(R)$ is one of the possible n-tuples sets $r(R) = \{t_1, t_2, \ldots, t_m\}$ materializing the relation schema definition R. A value in tuple $t \in r(R)$, corresponding to an R's attribute A (or a combination), can be referred to as $t[A]$.[4]

There can be many *constraints* on the data-model imposed by the miniworld the database represents. *Domain constraints* specify that corresponding values for each attribute A must be an atomic element from the domain set: $\forall t \in r(R)\colon t[A] \in dom(A)$. The *key constraint* mandates that for any superkey subset of attributes SK, we have $\forall t_i \in r(R), t_j \in r(R), i \neq j\colon t_i[SK] \neq t_j[SK]$. Another constraint is the *not-null constraint* where some attributes are not allowed to have undefined (`NULL`) values.[4]

Since no positional identification of tuples exists, the *entity-integrity constraint* mandates that primary key values cannot be `NULL`: it would imply an inability to identify some tuples. Finally, the *referential integrity constraint* is used to maintain consistency among two relations tuples. A set of attributes FK in the relation R is a valid foreign key to relation S if (1) the attributes in FK have the same domain as the primary key attributes PK of S, and (2) $\forall t_i \in r(R)\colon t_i[FK] = \texttt{NULL} \vee \exists t_j \in s(S)\colon t_i[FK] = t_j[PK]$.

Finally, note that the terms "relation" and "table" are not synonymous. From their set bases, relations have no positional concepts. Thus, there's no "nextness" of rows. Similarly, and especially in later versions of the model, one may shuffle the columns without affecting the information content, providing the column heading is taken with each column. Thus, there's also no "nextness" of columns. Neither of these activities can occur with such immunity to arrays.[3]

0.2 Relational algebra

Beside the regular set theory operators, the relational algebra operators are introduced because of their key role in deriving relations from other relations. *Selection*: filter tuples that satisfies a certain condition. It can be visualized as a *horizontal partitioning* between two set of tuples, those tuples that satisfy the condition, and those that do not:

$$\sigma_{<selection\_condition>}(R)$$

where $\langle$selection_condition$\rangle$ = $\langle$attribute_name $A\rangle$ $\langle$comparison op$\rangle$ $\langle$value $\in dom(A)\rangle$ or $\langle$selection_condition$\rangle$ = $\langle$attribute_name $A_i\rangle$ $\langle$comparison op$\rangle$ $\langle$attribute_name $A_j\rangle$ where $dom(A_i) = dom(A_j)$. From definition, some of the selection operator properties include:

$$\begin{aligned}
&\sigma_{<c1>}(\sigma_{<c2>}(R)) = \sigma_{<c1 \textbf{ and } c2>}(R)\\
&\sigma_{<c1>}(\sigma_{<c2>}(\ldots(\sigma_{<cn>}(R))\ldots))\\
&\qquad \equiv \sigma_{<c1 \textbf{ and } c2 \textbf{ and } \ldots \textbf{ and } cn>}(R)\\
&\sigma_{<c1>}(\sigma_{<c2>}(R)) = \sigma_{<c2>}(\sigma_{<c1>}(R))\\
&|\sigma_{<c>}(R)| \leq |R|
\end{aligned}$$

Projection: selects certain columns of the relation (striking out the others), then removes from the resulting array any duplication of rows. The

[3]The *role* concept was originally created by Codd to distinguish equivalent domains in the same relation.

[4]For simplicity, the *notational* difference between a schema R and its instance $r(R)$ is sometimes ignored.

final array represents a relation which is said to be a projection of the given relation. This operator can be visualized as a *vertical partitioning* of the relation into two instances: one that does include the specified attributes, and another that does not:

$$\pi_{<attribute\_list>}(R)$$

where $\langle$attribute_list$\rangle$ is the desired list of attributes from relation R. If any attribute is in $\langle$attribute_list$\rangle$ but not in R, then the expression is incorrect. Some properties include:

$$\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$$
$$\pi_{<A1,A2,\ldots,An>}(\sigma_c(R)) = \sigma_c(\pi_{<A1,A2,\ldots,An>}(R))$$

where $\langle$list1$\rangle$ includes *all* the $\langle$list2$\rangle$ attributes and the selection condition c involves *only* attributes from the set $\{A_1, A_2, \ldots, A_n\}$.

Set operators: relations are sets, thus all of the usual set operations are applicable. Nevertheless, the result may *not* be a relation: for example, the union of a binary relation and a ternary relation is not one. Two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ are *union-compatible* if they have the same degree ($n = m$) and if $dom(A_i) = dom(B_i)$ for all $1 \leq i \leq n$.[5] Some of the set operators properties include:

$$R \cup S = S \cup R, \text{ and } R \cap S = S \cap R$$
$$R - S \neq S - R$$
$$\sigma_c(R\ \theta\ S) = \sigma_c(R)\ \theta\ \sigma_c(S)$$
$$\pi_L(R \cup S) = \pi_L(R) \cup \pi_L(S)$$

where θ is any of the set operators $\cup$, $\cap$, or $-$. Finally, note that projection can't be commuted with intersection or set difference ('$\cap$', '$-$') since it conflicts with these operators *column-sensitive* semantics.

Cartesian Product: the well-known set-theory operator used to combine every member from the first set with every member from the second set; it can be used between relations which are *not* union compatible. Given $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$, we have $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ of degree $n+m$ if $R \times S = Q$. Assuming $|R| = n_r$ and $|S| = n_s$, then $|Q| = n_r * n_s$. In general, the cartesian product is not very useful on its own, but by using selection afterwards, it can be exploited to select matching tuples from *any* two relations.

Some properties include: (1) if the selection condition c only involves attributes of R, we have $\sigma_c(R \times S) = \sigma_c(R) \times S$, (2) alternatively, if the condition c can be written as (c_1 **and** c_2) where condition c_1 only involves attributes from R and c_2 only involves attributes from S, we have:

$$\sigma_d(R) \times \sigma_e(S) = \sigma_{<d \textbf{ and } e>}(R \times S)$$
$$\sigma_c(R \times S) = \sigma_{c1}(R) \times \sigma_{c2}(S)$$

where d and e are selection conditions, (3) commutativity with projection: suppose $L = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ where A_i is an attribute of R and B_j is an attribute of S for all $1 \leq i \leq n$ and $1 \leq j \leq m$, we have $\pi_L(R \times S) \equiv \pi_{<A1,\ldots,An>}(R) \times \pi_{<B1,\ldots,Bm>}(S)$.

Join: one of the most important operators in relational algebra by allowing us to exploit the implicit relationships between relations: it combines related tuples from two relations into a single tuple. For $R(A_1, \ldots, A_n)$ and $S(B_1, \ldots, B_m)$, we have the join Q of degree $n+m$:

$$Q = R \bowtie_c S = \sigma_c(R \times S)$$

where condition c contains expressions in the form $A_i\ \theta\ B_j$ for $1 \leq i \leq n$ and $1 \leq j \leq m$, $dom(A_i) = dom(B_j)$, and $\theta \in \{<, \leq, =, >, \geq,$

[5]To let the set operators result in a relation, the result set – as in any relation – must be totally built from the same type of tuples. Two tuples are of the same type if they have the same length and include equivalent domains for each corresponding tuple attribute.

$\neq\}$. Finally assuming $|R| = n_r$, and $|S| = n_s$, we have $0 \leq |Q| \leq n_r * n_s$.[6] The referential integrity constraint is essential in having matching tuples in the resulting relation.

Similar to the cartesian product, a selection with condition c can be commuted with the join '$R \bowtie_d S$' if c can be expressed in the form $c_1 \wedge c_2$, where c_1 only involves attributes from R and c_2 only contains attributes from S:

$$\begin{aligned}
&\sigma_c(R \bowtie_d S)\\
&\quad = \sigma_c(\sigma_d(R \times S))\\
&\quad = \sigma_d(\sigma_c(R \times S))\\
&\quad = \sigma_d(\sigma_{c1}(R) \times \sigma_{c2}(S))\\
&\quad = \sigma_{c1}(R) \bowtie_d \sigma_{c2}(S)
\end{aligned}$$

by the join operator definition, commutativity of selection, and by commuting 'σ' with '$\times$'. Similarly, projection can be commuted with the join:

$$\begin{aligned}
&\pi_L(R \bowtie_d S)\\
&\quad = \pi_L(\sigma_d(R \times S))\\
&\quad = \sigma_d(\pi_L(R \times S)) && (*)\\
&\quad \equiv \sigma_d(\pi_{A1,\ldots,An}(R) \times \pi_{B1,\ldots,Bm}(S))\\
&\quad \equiv \pi_{A1,\ldots,An}(R) \bowtie_d \pi_{B1,\ldots,Bm}(S)
\end{aligned}$$

where the list of projection attributes $L = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ having $A_1, \ldots A_n$ as a subset of R's attributes and $B_1, \ldots, B_m$ as a subset of S's attributes, and where all attributes in the join condition d are covered in projection list L due to the transformation line marked by $(*)$.

Division: the algebraic counterpart of predicate logic's *universal quantifier*.[2] Let Z, X, Y be sets of relation attributes where $X \subseteq Z$, $Y = Z - X$, and thus $Z = X \cup Y$. Letting $R(Z)$, $S(X)$, and $T(Y)$ be relations on these attributes, we can define division as

$$T(Y) = R(Z) \div S(X) \tag{1}$$

where:

$$t_y \in T \iff (S \times \{t_y\}) \subseteq R \tag{2}$$

and $R = (S \times T) \cup D$, having D as the division remainder. Note that throughout this formalization, relations are considered domain-unordered: a *relationship*, using Codd's terminology. Thus, the cartesian product results in equation (2) are only *semantically* union-compatible with R.

As a moderate example, let $R(a, b) = \{\langle p, 1\rangle, \langle p, 2\rangle, \langle p, 3\rangle, \langle q, 1\rangle, \langle r, 1\rangle, \langle r, 3\rangle\}$ and $S(b) = \{\langle 1\rangle, \langle 3\rangle\}$, we have $T(a) = R(a, b) \div S(b) = \{\langle p\rangle, \langle r\rangle\}$, and the remainder $D(a, b) = \{\langle p, 2\rangle, \langle q, 1\rangle\}$.

The division operator is only a shorthand for the sequence π, $\times$, $-$ as follows:

$$\begin{aligned}
&T_1 \leftarrow \pi_Y(R)\\
&T_2 \leftarrow \pi_Y((S \times T_1) - R)\\
&T \leftarrow T_1 - T_2
\end{aligned}$$

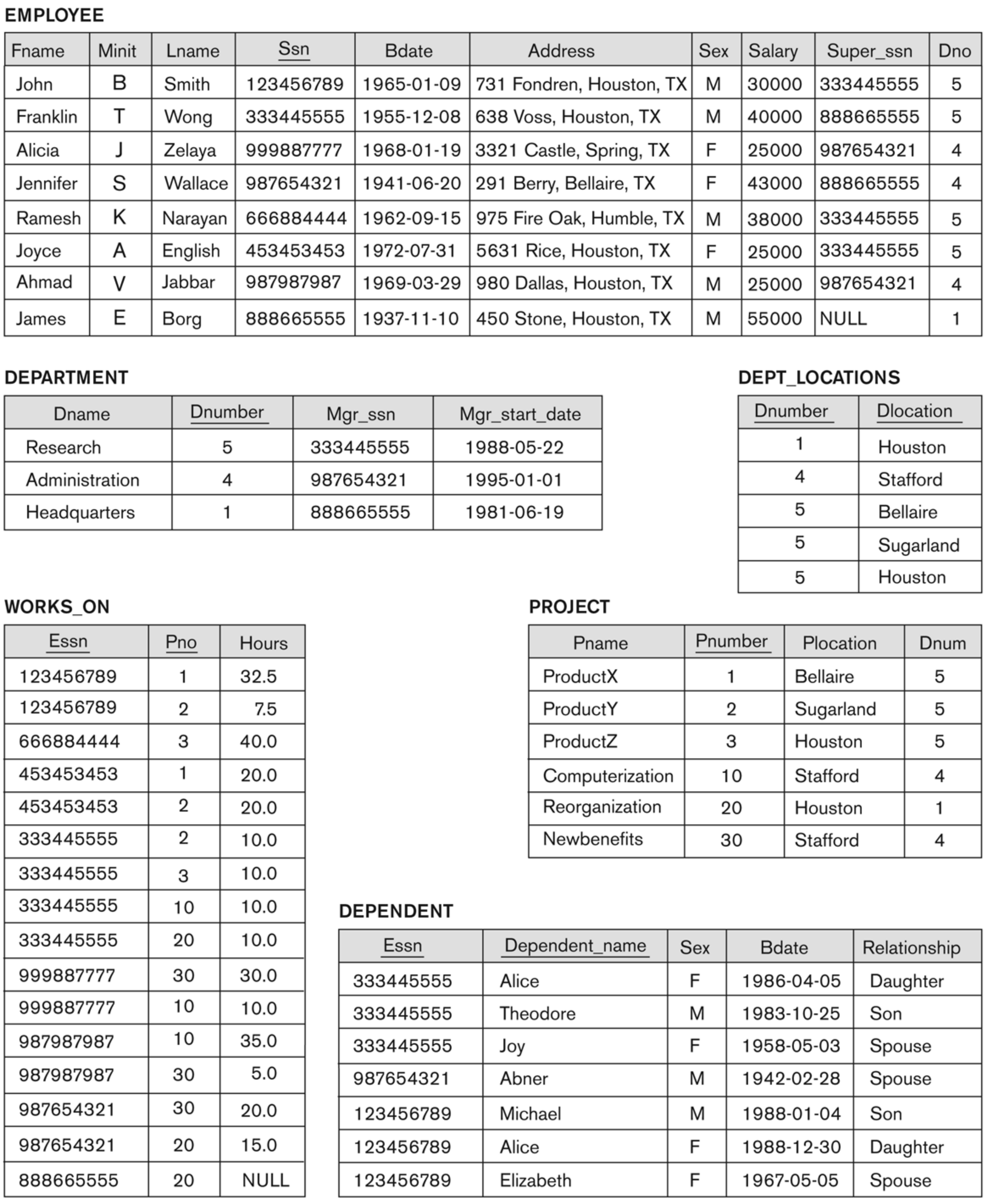
Why? Remembering that S is represented by the schema $S(X)$, T_1 by $T_1(Y)$, R by $R(Z)$ where $Z = X \cup Y$, and by definition of the cartesian product, $S \times T_1$ represents all possible tuples – taken combined, but not individually – that can satisfy the division at (1). If there's a tuple $t \in R$ where $t[Y] \notin T_2$, then *all* possible tuple combinations $t_r \in (S \times \{t[Y]\})$ are in R. Thus, by the division operator definition, $t[Y] \in T$. Namely:

$$\begin{aligned}
&\forall t \in R, t[Y] \notin T_2 \colon (S \times \{t[Y]\}) \subseteq R && (\dagger)\\
&\quad \Rightarrow \forall t \in \pi_Y(R), t \notin T_2 \colon (S \times \{t\}) \subseteq R\\
&\quad \Rightarrow \forall t \in (T_1 - T_2) \colon t \in T
\end{aligned}$$

Proving $(\dagger)$ by contradiction:[7] Let $t \in R$, $t[Y] \notin T_2$, and $(S \times \{t[Y]\}) \not\subseteq R$. From the first and third axiom, we have $t[Y] \in \pi_Y(R)$, $\exists t_x \colon t_x \in (S \times \{t[Y]\})$, $t_x \notin R$ which can be further inferred to $t_x[Y] \in T_2$. Since by t_x definition $t_x[Y] = t[Y]$, then $t[Y] \in T_2$, contradicting the second axiom '$t[Y] \notin T_2$' above. Q.E.D.

[6]Compare this with cartesian product's $|Q| = n_r * n_s$.

[7]$\neg(\text{if } a \wedge \neg b, \text{then } y) \Rightarrow \text{if } a \wedge \neg b, \text{then } \neg y$.



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Figure (1): *Courtesy of Elmari&Navathe's "Fundamentals of Database systems – 5th edition": relation schemas, representing a company business environment, and one of their possible conforming states.*

0.3 Example Queries

Using the relation schemas outlined in figure 1 page 5, build relational algebra queries satisfying the following requests[8]: (1) get employee tuples whose department number is 4.

$res \leftarrow \sigma_{Dno=4}(employee)$

(2) Select employees whose salary is greater than $30,000$\$.

$res \leftarrow \sigma_{Salary>30000}(employee)$

(3) Select employees who either work in department 4 and make over $25,000$\$ per year, or work in department 5.

$\sigma_{(Dno=4 \textbf{ and } Salary>25000) \textbf{ or } Dno=5}(employee)$

(4) List each employee's first and last name and salary.

$res \leftarrow \pi_{Fname,Lname,Salary}(employee)$

(5) Retrieve first and last name of employees who work on department 1.

$res \leftarrow \pi_{Fname,Lname}(\sigma_{Dno=1}(employee))$

(6‡) Retrieve the social security number of all employees who work on department 5 or who manage someone working on department 5.

$emp\_dep5 \leftarrow \sigma_{Dno=5}(employee)$
$emp\_ssn \leftarrow \pi_{Ssn}(emp\_dep5)$
$mgrs\_ssn \leftarrow \pi_{Super\_ssn}(emp\_dep5)$
$res \leftarrow emp\_ssn \cup mgrs\_ssn$

(7) List the name of female employees dependents.

$fem\_emps \leftarrow \sigma_{Sex=F}(employee)$
$deps \leftarrow fem\_emps \bowtie_{Essn=Ssn} dependent$
$res \leftarrow \pi_{Fname,Lname,Dependent\_name}(deps)$

(8) Retrieve the name of the manager of each department.

$dep\_mgr \leftarrow department \bowtie_{Mgr\_ssn=Ssn} employee$
$res \leftarrow \pi_{Dname,Fname,Lname}(dep\_mgr)$

(9) Retrieve the names of employees who work in *any* of the projects that "John Smith" works on.

$smith \leftarrow \sigma_{Fname=John \wedge Lname=Smith}(employee)$
$smith\_pnos \leftarrow \pi_{Pno}(smith \bowtie_{Ssn=Essn} works\_on)$
$emp\_ssn \leftarrow \pi_{Essn}(smith\_pnos * works\_on)$
$smith\_ssn \leftarrow \pi_{Ssn}(employee)$
$res\_ssn(Ssn) \leftarrow emp\_ssn - smith\_ssn$
$res \leftarrow \pi_{Fname,Lname}(employee * res\_ssn)$

(10‡) Retrieve the names of employees who work in *all* of the projects that "John Smith" works on.

$smith \leftarrow \sigma_{Fname=John \wedge Lname=Smith}(employee)$
$smith\_pnos \leftarrow \pi_{Pno}(smith \bowtie_{Ssn=Essn} works\_on)$
$all\_ssn\_pno \leftarrow \pi_{Essn,Pno}(works\_on)$
$emps\_ssn(Ssn) \leftarrow all\_ssn\_pno \div smith\_pnos$
$res \leftarrow \pi_{Fname,Lname}(emps\_ssn * employee)$

(11) For every male dependent, retrieve the dependent's name, and the dependee employee name and department.

$male\_deps \leftarrow \sigma_{Sex=M}(dependent)$
$deps\_e \leftarrow male\_deps \bowtie_{Essn=Ssn} employee$
$deps\_e\_d \leftarrow deps\_e \bowtie_{Dno=Dnumber} department$
$res \leftarrow \pi_{Dependent\_name,Fname,Lname,Dname}(deps\_e\_d)$

(12) For all departments with employees, retrieve each department number, the number of

[8]'‡'-marked requests were taken from [4] example queries; the rest are the book's end-of-chapter exercises.

employees in the department, and their average salary. Rename the result columns with meaningful names.

$tmp \leftarrow Dno \Im_{COUNT\ Ssn, AVERAGE\ Salary}(emp)$

$res(Dno, No\_of\_emps, avg\_salary) \leftarrow tmp$

(13‡) Select all of the employees who are either directly supervised by "James Borg" or by any of his supervisees.

$borg \leftarrow \sigma_{Fname=James \wedge Lname=Borg}(emp)$

$borg(Ssn0) \leftarrow \pi_{Ssn}(borg)$

$lvl1(Ssn1) \leftarrow \pi_{Ssn}(borg \bowtie_{Ssn0=Super\_ssn} emp)$

$lvl2(Ssn2) \leftarrow \pi_{Ssn}(lvl1 \bowtie_{Ssn1=Super\_ssn} emp)$

$res \leftarrow (\rho_{(Ssn)}(lvl1 \cup lvl2)) * employee$

(14) Select names of department managers with at least one female dependent.

$dept\_dep \leftarrow department \bowtie_{Mgr\_ssn=Essn} dependent$

$mgrs \leftarrow \sigma_{Sex=F}(dept\_dep)$

$res \leftarrow \pi_{Fname,Lname}(mgrs \bowtie_{Essn=Ssn} employee)$

(15) Find employees who work on *all* projects controlled by the "Jennifer Wallace"-managed department.

$jennifer \leftarrow \sigma_{Fname=Jennifer \wedge Lname=Wallace}(emp)$

$jen\_dep \leftarrow jennifer \bowtie_{Ssn=Mgr\_ssn} department$

$jen\_dep\_prj \leftarrow jen\_dep \bowtie_{Dnumber=Dnum} project$

$jen\_pnos(Pno) \leftarrow \pi_{Pnumber}(jen\_dep\_prj)$

$ssn\_pno(Ssn, Pno) \leftarrow \pi_{Essn,Pno}(works\_on)$

$emp\_ssn \leftarrow ssn\_pno \div jen\_pnos$

$res \leftarrow \pi_{Fname,Lname}(emp\_ssn * employee)$

(16) List names of employees working on two or more projects.

$tmp \leftarrow Essn \Im_{COUNT\ Pno}(works\_on)$

$prjs\_count(Ssn, Count) \leftarrow tmp$

$prjs\_2plus \leftarrow \sigma_{Count \geq 2}(prjs\_count)$

$res \leftarrow \pi_{Fname,Lname}(prjs\_2plus * employee)$

(17) Retrieve the names of all employees on department 5 who work more than 10 hours per week on the project 'ProductX'.

$e\_w \leftarrow employee \bowtie_{Ssn=Essn} works\_on$

$e\_w\_p \leftarrow e\_w \bowtie_{Pno=Pnumber} project$

$res \leftarrow \sigma_{Dno=5\ \wedge\ hours>10\ \wedge\ Pname=ProductX}(e\_w\_p)$

(18) List the names of all employees who have the same dependent name as themselves.

$e\_d \leftarrow (employee \bowtie_{Ssn=Essn} dependent)$

$emps \leftarrow \sigma_{Fname=Dependent\_name}(e\_d)$

$res \leftarrow \pi_{Fname,Lname}(emps)$

(19) For each project, list the project name and the total hours per week (by all employees) spent on the project.[9]

$tmp \leftarrow Pno \Im_{SUM\ Hours}(works\_on)$

$prj\_hours(Pnumber, Total) \leftarrow tmp$

$res \leftarrow \pi_{Pname,Total}(prj\_hours * project)$

(20) Retrieve the names of employees who work on *every* company project.

$all\_pnos(Pno) \leftarrow \pi_{Pnumber}(project)$

$ssn\_pnos(Ssn, Pno) \leftarrow \pi_{Essn,Pno}(works\_on)$

$ssn \leftarrow ssn\_pnos \div all\_pnos$

$res \leftarrow \pi_{Fname,Lname}(ssn * employee)$

(21) Retrieve the names of all employees who do not work on *any* project.

$all\_ssn \leftarrow \pi_{Ssn}(employee)$

$working\_ssn(Ssn) \leftarrow \pi_{Essn}(works\_on)$

$free\_ssn \leftarrow all\_ssn - working\_ssn$

$res \leftarrow \pi_{Fname,Lname}(free\_ssn * employee)$

[9] The answer assumes there's at least one employee per project.

(22) Retrieve the average salary of all female employees.

$res \leftarrow \Im_{AVERAGE\ Salary}(\sigma_{Sex=F}(employee))$

(23) For each department, retrieve the department name and the average salary of all of its employees.[10]

$tmp \leftarrow Dno\Im_{AVERAGE\ Salary}(employee)$
$deps\_avg(Dno, Avg\_salary) \leftarrow tmp$
$deps \leftarrow department \lhd_{Dnumber=Dno} deps\_avg$
$res \leftarrow \pi_{Dname,Avg\_salary}(deps)$

(24) List the names of all department managers who have no dependents.

$dept\_dep \leftarrow department \bowtie_{Mgr\_ssn=Essn} dependent$
$mgrs\_dep(Ssn) \leftarrow \pi_{Essn}(dept\_dep)$
$all\_mgrs(Ssn) \leftarrow \pi_{Mgr\_ssn}(department)$
$mgrs\_no\_dep \leftarrow all\_mgrs - mgrs\_dep$
$res \leftarrow \pi_{Fname,Lname}(mgrs\_no\_dep * employee)$

(25) Find the names and addresses of all employees who work on at least one project located in Houston, but whose department has no location in Houston.
Answer: the solution can be divided to three parts; i) employees who work on at least one project in Houston:

$e\_w \leftarrow employee \bowtie_{Ssn=Essn} works\_on$
$e\_w\_p \leftarrow e\_w \bowtie_{Pno=Pnumber} project$
$houston\_ssn \leftarrow \pi_{Ssn}(\sigma_{Plocation=Houston}(e\_w\_p))$

ii) employees whose department has no location in Houston:

$houston\_deps \leftarrow \sigma_{Dlocation=Houston}(dept\_locations)$
$houston\_dno(Dnum) \leftarrow \pi_{Dnumber}(houston\_deps)$
$all\_dno(Dnum) \leftarrow \pi_{Dnumber}(department)$
$no\_houston\_dno \leftarrow all\_dno - houston\_dno$
$tmp \leftarrow employee \bowtie_{Dno=Dnum} no\_houston\_dno$
$no\_houston\_dept\_ssn \leftarrow \pi_{Ssn}(tmp)$

iii) final result: common employees from the two queries above, satisfying the original request.

$common \leftarrow houston\_ssn \cap no\_houston\_dept\_ssn$
$res \leftarrow \pi_{Fname,Lname,Address}(common * employee)$

[10] Due to a typesetting limitation, we've used the non-standard symbol '$\lhd$' to denote a *left outer join.*

References

[1] E. F. Codd, *A Relational Model of Data for Large Shared Data Banks.* Communications of the ACM, Volume 13, Issue 6, June 1970.

[2] E. F. Codd, *Extending the data base relational model to capture more meaning.* ACM Transactions on Database Systems, Volume 4, Issue 4, December 1979.

[3] E. F. Codd, *Relational Model for Database Management – version 2.* Addison-Wesley, 1990.

[4] RA. Elmasri & S. B. Navathe, *Fundamentals of Database Systems – 5th Edition.* Addison-Wesley, 2007.

Typeset with LaTeX