CHAPTER 3

A COMPREHENSIVE PROBABILISTIC RELATIONAL DATABASE MANAGEMENT SYSTEM

3.1 INTRODUCTION

Since there is a need to extend the relational model to handle and manipulate uncertain information, a probabilistic relational database management system is being proposed and a theoretical framework is provided in which such a proposed system can be developed[89]. To obtain an exact prototype of the proposed system, probabilistic integrity constraints have been defined. The relational algebra is extended to handle probabilistic data and it has been named as probabilistic relational algebra. The probabilistic normal forms are also described likewise.

3.2 STRUCTURE OF PROBABILISTIC RELATIONAL DATA BASE MANAGEMENT SYSTEM:

A block diagram of the proposed system is shown in Fig. 3.1

![Fig. 3.1](image-url)
User can interact with the system with the help of either PDDL (Probabilistic Data Definition Language) or PDML (Probabilistic Data Manipulation Language). The statements of PDDL are subjected to a PDDL interpreter which will convert them into probabilistic relations to be stored in database. PDML statements are subjected to a PDML compiler and after optimizing the query, results can be obtained through a query evaluation engine. The results are to be obtained through file manager which takes help of certain operating system services and satisfies the requests for retrieving or putting the data in the probabilistic database.

### 3.2.1 Probabilistic Relations: their structure and meaning:

Relation can be obtained by attaching a probability stamp to each tuple of a classical relation. The probability stamp refers to the probability that the said tuple belongs to the relation. This probability stamp is a joint probability of the given realizations of all the attributes taken together for a particular tuple. The probabilities of the individual attributes can be derived by marginalizing the said distribution.

Let R(A1, A2, ..., An) be a relation schema and Dom(Ai) represents the domain of attribute Ai for i=1 to n. Then a probabilistic relation R can be defined as

\[ R \text{ is subset of } \{ t. p(t) | t=(t1,t2,..tn) \in \text{Dom}(A1)\times\text{Dom}(A2)\times...\times\text{Dom}(An) \} \]

And \( p(t) \in [0,1] \), \( t. p(t) \) refers to concatenation of t with \( p(t) \) and is called a probabilistic tuple.

The probabilistic relation is a combination of such probabilistic tuples and probabilistic database is a combination of such probabilistic relations.

An example of classical (Deterministic) and Probabilistic Relation is given below along with their interpretations:

### Table 3.1: (Classical Relation)

<table>
<thead>
<tr>
<th>FAC_ID</th>
<th>Name</th>
<th>Dept</th>
<th>Office</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S K Sharma</td>
<td>MBA</td>
<td>Delhi</td>
<td>2477001</td>
</tr>
<tr>
<td>2</td>
<td>C L Mittal</td>
<td>E &amp; C</td>
<td>Delhi</td>
<td>2438002</td>
</tr>
<tr>
<td>3</td>
<td>A K Sharma</td>
<td>CE</td>
<td>Mumbai</td>
<td>2459921</td>
</tr>
<tr>
<td>4</td>
<td>S K Gupta</td>
<td>IT</td>
<td>Kolkata</td>
<td>2467734</td>
</tr>
</tbody>
</table>
Its probabilistic counterpart can be obtained as follows:

<table>
<thead>
<tr>
<th>FAC_ID</th>
<th>Name</th>
<th>Dept</th>
<th>Office</th>
<th>Phone</th>
<th>pS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S K Sharma</td>
<td>MBA</td>
<td>Delhi</td>
<td>2477001</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>S K Sharma</td>
<td>Mktg</td>
<td>Chandigarh</td>
<td>2477001</td>
<td>0.3</td>
</tr>
<tr>
<td>1</td>
<td>S K Sharma</td>
<td>Mktg</td>
<td>Chandigarh</td>
<td>2477001</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>C L Mittal</td>
<td>E &amp; C</td>
<td>Delhi</td>
<td>2438002</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>C L Mittal</td>
<td>E &amp; I</td>
<td>Delhi</td>
<td>2438002</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>A K Sharma</td>
<td>CE</td>
<td>Mumbai</td>
<td>2459921</td>
<td>1.0</td>
</tr>
<tr>
<td>4</td>
<td>S K Gupta</td>
<td>IT</td>
<td>Kolkata</td>
<td>2467734</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The semantics of the tuple for the said Faculty_P relation is as follows:

\[
\text{Prob[ Faculty}_P (1, \text{S K Sharma, MBA, Delhi, 2477001}])=0.6
\]

It means that although faculty member with ID 1 is known to exist with certainty, its certain attribute values may not be known with certainty. The probability of this faculty member being in MBA department is 0.6 and belonging to marketing department is 0.4.

### 3.3. METHODS FOR OBTAINING PROBABILISTIC RELATIONS AND THEIR COMPARISONS:

Consider the scenario in which an Institute wants to celebrate its Diamond Jubilee. It involves inviting alumni that have graduated from the Institute. To send invitation letters, it requires one to have particulars of the alumni. The record of the alumni may be found either at central Registrar office of the Institute or the records can be had from department also. It might be the case that information at both the places is not same for a particular person. To handle such type of uncertainty, probability theory is a potential candidate. Let relational databases be used to represent the alumni information. For the sake of convenience, it is assumed that the information fields available at both the places are identical. However this may not be the only information contained in the tables, rather only the information that is common at both the places are depicted. There can be several conditions that will further complicate the consolidation of such data. For example, the tables might not hold the same type of information. Under such condition, it is difficult to determine what fields should be compared and how to assume that fields are equal or
comparable? It is difficult to determine when two persons are equal. For the purpose of obtaining practical results it is assumed that relations at both places have same schema. Let the following information at two places is given in Table 3.3 and Table 3.4:

**Table 3.3 (Office record)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Employer</th>
<th>Branch</th>
<th>Batch</th>
<th>Tel-No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ram</td>
<td>Delhi</td>
<td>Hughes</td>
<td>Computer</td>
<td>1960</td>
<td>011-12345</td>
</tr>
<tr>
<td>2</td>
<td>Shyam</td>
<td>Mumbai</td>
<td>Alcatel</td>
<td>Computer</td>
<td>1970</td>
<td>022-12345</td>
</tr>
<tr>
<td>3</td>
<td>Laxman</td>
<td>Kolkata</td>
<td>Oracle</td>
<td>Computer</td>
<td>1975</td>
<td>033-12345</td>
</tr>
</tbody>
</table>

**Table 3.4 (Department Record)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Employer</th>
<th>Branch</th>
<th>Batch</th>
<th>Tel-No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ram</td>
<td>Delhi</td>
<td>Hughes</td>
<td>Computer</td>
<td>1960</td>
<td>011-12345</td>
</tr>
<tr>
<td>2</td>
<td>Bharat</td>
<td>Mumbai</td>
<td>Alcatel</td>
<td>Computer</td>
<td>1970</td>
<td>022-12345</td>
</tr>
<tr>
<td>3</td>
<td>Laxman</td>
<td>Kolkata</td>
<td>IBM</td>
<td>Computer</td>
<td>1975</td>
<td>033-12345</td>
</tr>
</tbody>
</table>

To send proper invitation letters to the persons concerned there is need to consolidate the information in one table and at one place. In consolidating the information in one table, following probabilistic table (Table 3.5) is obtained whose structure and interpretation is identical to the one defined in[89].

**Table 3.5 (Probabilistic Relation)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Address</th>
<th>Employer</th>
<th>Branch</th>
<th>Batch</th>
<th>Tel-No</th>
<th>ps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ram</td>
<td>Delhi</td>
<td>Hughes</td>
<td>Computer</td>
<td>1960</td>
<td>011-12345</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>Shyam</td>
<td>Mumbai</td>
<td>Alcatel</td>
<td>Computer</td>
<td>1970</td>
<td>022-12345</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>Bharat</td>
<td>Mumbai</td>
<td>Alcatel</td>
<td>Computer</td>
<td>1970</td>
<td>022-12345</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>Laxman</td>
<td>Kolkata</td>
<td>Oracle</td>
<td>Computer</td>
<td>1975</td>
<td>033-12345</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>Laxman</td>
<td>Kolkata</td>
<td>IBM</td>
<td>Computer</td>
<td>1975</td>
<td>033-12345</td>
<td>0.5</td>
</tr>
</tbody>
</table>

It may be noted here that the last attribute in the above table represents the probability that the tuple belongs to the relation. The probabilistic stamp, ps, is assigned on the basis of the assumption that records at both the places are equally credible. Another alternative representation of the same information could be in the form of attaching appropriate probabilities at the attribute level instead of tuple level. In this manner the representation would be as depicted in table 3.6:
The numeric value associated with each data item is the probability of particular data item with respect to the information available with us. In table 3.5, the information can be interpreted as the probability of existence of an entity and in table 3.6 it can be interpreted as the probability of occurrence of various fields of the entity.

Relations in the form of Table 3 are better as compared to relations in the form of Table 3.5, i.e. representing uncertainty at tuple level is better than representation at attribute level because of following two reasons:

(i) At attribute level, it gives us non 1 NF relations which are difficult to implement.
(ii) It may result in loss of information or wrong information may be interpreted which may not be there in the original relations because of the fact that attributes are independent of each other.

Therefore, relations in the form of Table 3.5 are choosen as the representation for uncertain information.

### 3.4 PROBABILISTIC INTEGRITY CONSTRAINTS:

In typical relational model, two integrity constraints namely entity integrity and referential integrity are specified. Entity integrity refers to the primary key value should not be null and it should be unique also. Whereas, referential integrity refers to the concept of a foreign key and that the tuple of a relation should refer to an existing tuple of another relation so that relationship between two relations can be specified. The same constraints can also be applicable on probabilistic relations albeit in somewhat different way. The Entity Integrity will refer to the scenario in which the entity assumed to be existing with certainty should add up the probability stamps to one and entities whose existence is uncertain, their probabilities should add up to less than one. The primary key
and probability stamps should not be null, being parallel to classical relational model. The changed referential integrity constraint would be that probability stamp for the foreign key should be less than the probability stamp of the referred entity.

3.5 PROBABILISTIC RELATIONAL ALGEBRA OPERATIONS:

Basic extensions of relational algebra operations can also be obtained for the probabilistic relations which are simple generalizations of classical relational algebra operations:

3.5.1 Probabilistic set-theoretic operations:

The operations have been borrowed from set-theory. There are four operations namely-union, intersection, difference and cartesian product. The first three of the operators require the operand relations to be union-compatible. Mathematically,

**Union:** Let \( r_1 \) and \( r_2 \) be two union compatible probabilistic relations. Then \( r_1 \cup r_2 \) can be defined as follows:

\[
R = r_1 \cup r_2 = \{ t.p(t) \mid t.p(t) \in r_1 \text{ or } t.p(t) \in r_2 \}
\]

**Intersection:** Let \( r_1 \) and \( r_2 \) be two union compatible probabilistic relations. Then \( r_1 \cap r_2 \) can be defined as follows:

\[
R = r_1 \cap r_2 = \{ t.p(t) \mid t.p(t) \in r_1 \text{ and } t.p(t) \in r_2 \text{ and } p(t) = p_1(t) \cdot p_2(t) \}
\]

**Difference:** Let \( r_1 \) and \( r_2 \) be two union compatible probabilistic relations. Then \( r_1 - r_2 \) can be defined as follows:

\[
R = r_1 - r_2 = \{ t.p(t) \mid t.p(t) \in r_1 \text{ and } t.p(t) \in r_2 \text{ and } p(t) = p_1(t) \cdot [1 - p_2(t)] \}
\]

**Cartesian Product:** Let \( r_1 \) and \( r_2 \) be two probabilistic relations then \( r_1 \times r_2 \) can be defined as:

\[
R = r_1 \times r_2 = \{ t_1.t_2, p(t_1.t_2) \mid t_1.p(t_1) \in r_1 \text{ and } t_2.p(t_2) \in r_2 \text{ and } p(t_1.t_2) = p_1(t_1) \cdot p_2(t_2) \}
\]

3.5.2 Probabilistic relation-oriented operators:

**Selection:** Depending upon the selection condition the tuples are chosen for which the selection condition is true.

\[
R = \sigma_{\text{cond}}(r_1) = \{ t.p(t) \mid t.p(t) \in r_1 \text{ and } \text{cond} \text{ is true} \}
\]

**Projection:** Let \( R(A,B) \) be a probabilistic relation and let \( A \) be the candidate key, then projection of \( R \) on \( A \) will be equal to \( A \) alongwith the associated probability stamps of attributes. The duplicates may be eliminated.

\[
\pi_A(R) = \{ a.p(a) \}
\]
Join: Let \( r_1(a,b) \) and \( r_2(b,c) \) be two probabilistic relations, then join of \( r_1 \) with \( r_2 \) is equivalent to:

\[
\{abc.p(abc) \mid ab.p(ab) \in r_1 \text{ and } bc.p(bc) \in r_2 \text{ and } p(abc) = p(ab)p(bc)\}
\]

### 3.6 PROBABILISTIC NORMAL FORMS:

The issue of redundancy is very important from the viewpoint of database design. A redundant database not only requires more storage space but also poses problems with respect to data integrity, quality and consistency. Therefore, once a probabilistic relation is obtained from the source data, it is necessary to examine the dependencies across its attributes and thereafter normalize the relation[24]. The issue of normalization is basically a trade-off between data redundancy and query performance. By normalizing the relations to a higher level, one usually attains less data redundancy and therefore, superior data integrity. On the other hand, answering queries may involve time-consuming join operations on several of the decomposed relations.

With reference to conventional deterministic relations, a functional dependency is described as a constraint between two sets of attributes from the database. If \( R \) is a relation schema and let \( X, Y \) are a subset of \( R \), then \( R \) satisfies a functional dependency \( X \rightarrow Y \) if and only if , for any relation \( r \) on \( R \), whenever two tuples agree on all the attributes in \( X \), they also agree on all the attributes in \( Y \)[24]. A functional dependency is a property of semantics of the real-world situation.

In case of probabilistic relations, these functional dependencies must be generalized to stochastic dependencies, and probabilistic normal forms be defined in a manner that preserves these stochastic dependencies. For deterministic relations, a functional dependency can be derived by considering the semantics of sets of attributes in a pairwise fashion. However, when defining stochastic dependencies, the mere specification of pairwise dependence/ independence is often not enough, one needs to examine the joint probability distribution that exists across all the attributes to obtain a clear picture of how one attribute influences the distribution of another. Bayesian networks can be used to represent such interdependence among attributes[25]. There are three major reasons for using this representational scheme as below:
1. A Bayesian networks represents dependencies among attributes in a graphical format that is easy to interpret.

2. These networks enable one to capture the influence of one attribute on another in a rigorous manner[25].

3. Bayesian network representation does not depend on the underlying relational structure and subsequently the theoretical foundation is general enough to be applied to other probabilistic structures as well.

3.6.1 Bayesian Networks:

A bayesian network captures the stochastic dependencies among attributes of a probabilistic relation in the form of a directed graph with attributes represented as nodes of the graph. A direct influence of one attribute on another is represented by a directed edge between the attributes. Indirect influence is obtained by paths containing two or more edges. It is formally defined as follows:

Let $U$ be a set of random variables with a joint probability distribution $\mathcal{P}$. Let $G=(U,E)$ be a directed acyclic graph. Let $P_{\alpha}$ be the set of parents for $\alpha \in U$, i.e., $\beta \in P_{\alpha}$ if and only if there is an edge from $\beta$ to $\alpha$. $G$ is a Bayesian network representing the distribution $\mathcal{P}$ if each node $\alpha \in U$ is conditionally independent of all its nondescendants, given its parent $P_{\alpha}$ and no proper subset of $P_{\alpha}$ has this property.

Primary key can be used to determine the distribution of all the attributes, therefore primary key attributes have only edges directed away from them. The dependency structure of the probabilistic relation will have the following important characteristics:

1. Every node in the network represents an explicit attribute of $R$.
2. Every edge $e(\alpha, \beta)$ represents a direct influence of $\alpha$ on $\beta$.
3. An indirect influence of $\alpha$ on $\beta$ is captured by a path that is defined to be a set of connected edges from $\alpha$ to $\beta$.

Consider the table given below:

<table>
<thead>
<tr>
<th>EMP PROJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp#</td>
</tr>
</tbody>
</table>

The primary key for the relation is $\{\text{Emp#}, \text{Proj#}\}$. 
The dependency structure of the above relation is shown in fig. 3.2:

![Dependency Structure Diagram]

Fig. 3.2

### 3.6.2 Stochastic dependency:

Two types of stochastic dependencies can be defined based on the structure of Bayesian network as depicted above namely direct dependency and indirect dependency. Let $R$ be a probabilistic relation and let $G$ be the Bayesian network representing the dependency structure of $R$. An attribute is said to be directly dependent on all supersets of its parent attribute. Indirect dependency is similar to a transitive dependency and can be inferred from the direct dependency in a recursive fashion.

The probabilistic normal forms can be defined in a similar fashion based on the concept of stochastic dependency.

### 3.6.3 Normal Forms:

Issue of normalization is closely related to the database designer’s objective to reduce data redundancy in a database, thereby reducing the chances of insertion, deletion and modification anomalies. In deterministic relations, functional dependencies play a vital role in deriving normalized relations.

The definition for probabilistic First Normal Form (1PNF) is the same as in deterministic case ensuring that there exists a single value at the intersection of any row and a column in the probabilistic relation. Second Probabilistic Normal Form is concerned with partial stochastic dependency. Let $R$ be a relation schema and $K$ be the primary key of $R$. A set of attributes $X$ which is a subset of $(R - K)$ is called partially stochastically dependent on the key attribute if there exists a proper subset $K'$ of $K$ such that $K' \rightarrow X$ (Indirect dependency). $R$ is in second probabilistic normal form (2PNF) if no set of nonkey attributes in $R$ is partially stochastically dependent on the key $K$. 
The definition of the third normal form is also given. Let R be a relation scheme with K as the primary key. R is said to be in third probabilistic normal form if, whenever the dependency \( X \rightarrow \alpha \) is satisfied by R for some \( \alpha \in (R - K) \), at least one of the following holds:

(i) K is a subset of X 
(ii) \( X \rightarrow \alpha \) (a functional dependency) and X is a superkey or
(iii) \( X \rightarrow \alpha \) (a functional dependency) and \( \alpha \in K \).

Similarly, Boyce-Codd probabilistic Normal Form is also defined. Let R be a relation schema with K as the primary key. R is said to be in Boyce-Codd probabilistic normal form if, whenever the dependency \( X \rightarrow \alpha \) (Direct dependency) is satisfied by R for some \( \alpha \in (R - X) \), either

(i) K is a subset of X or
(ii) \( X \rightarrow \alpha \) (a functional dependency) and X is a superkey.

### 3.7 PROBABILISTIC STRUCTURED QUERY LANGUAGE (PSQL):

The name SQL is an acronym for Structured Query Language. Originally SQL was called SEQUEL (Structured English Query Language) and was designed and implemented at IBM Research as the interface for an experimental relational database system called SYSTEM R[15]. SQL has now become a standard language for commercial relational DBMSs. ANSI and ISO made joint efforts to evolve a standard version of SQL (ANSI 1986), called SQL-86 or SQL 1. A revised and expanded standard called SQL 2 (also referred to as SQL-92) was subsequently developed. To include object oriented concepts, SQL 2 has been extended and the version is called SQL 3.

A query language for the comprehensive probabilistic relational database management system is proposed called PSQL (Probabilistic Structured Query language) and the said language is patterned after SQL. Certain commands of PSQL are given in the following section:
PSQL: Certain commands and their purpose:

PDDL and PDML Commands:

1. create table and other commands:

The general syntax for create table command is as follows:

   Create table table-name
   (attribute-name data-type [not null]
   { attribute-name data-type [not null]}
   ps float not null,
   primary key( ),
   foreign key( ) references ( );

The general syntax for alter table command is as follows:

   Alter table table-name
   (add attribute-name data-type,
   );

2. Insert, Delete and Update commands:

The general syntax for various update operations is as follows:

   Insert into table-name
   [attribute-name values( )]
   delete from table-name
   [where condition]
   update table-name
   [set attribute-name to]

3. Extended select:

Select statement is formed from three clauses select, from and where having the following form

   Select<expression>
   From<table-list>
   Where<condition>

The difference in SQL and PSQL expressions is that latter can contain a probabilistic expression. In BNF notation, an expression might be of the form

   Expr ::= arith_exp| const_exp| prob_exp;
Where arith\_exp represents some arithmetic expression, const\_exp refers some constant expression and prob\_exp represents some probabilistic expression. Expressions could be expressed recursively in terms of attribute-name or lists of attribute names, embedded within some aggregate functions and connected via some arithmetic or relational operator. The table list essentially contains probabilistic relations and condition is basically consisting of predicates which might be connected via logical operators. Therefore the syntax of select of SQL is identical to the syntax of select of PSQL. If some query refers to two or more attributes with the same name, then we can qualify the attribute name with the relation name to avoid ambiguity.

The example of select command could be:

```
Select name from Faculty_P
where dept= "MBA" and
p[FAC_ID,dept]>0.5
```

### 3.7.1 Embedded PSQL:

PSQL statements might be embedded in a host language and PSQL structures may be called embedded PSQL. The code for embedded PSQL must also be subjected to a DML precompiler so that appropriate embedded PSQL requests are converted to normal procedure calls of the host language. The embedded PSQL statements may also be enclosed between EXEC SQL and END EXEC. To write a query, we use declare pcursor command and for communication between PSQL code and host language, we use open and fetch commands. For example the following is an embedded PSQL code:

```
EXEC SQL

Declare pcursor for
Select office
from FACULTY_P
Where name= "S K Sharma" and p[FAC_ID,dept] > 0.5

END EXEC
```
3.7.2 Syntax of PSQL:

BNF is widely used for specifying the syntax of a programming language. The advantage of BNF notation for specifying the syntax of a programming language is that it is quite easy to establish a correspondence between BNF definition of a programming language and the actions of a parser. Syntax definition of a language contains:
- symbols used for building blocks
- structure of the word
- structure of well formed phrases
- sentence structure

BNF is used to specify internal structure of the language. We specify syntax of PSQL by listing its syntax domains and BNF rules:

Q ε Query
D ε Declaration
C ε Command
E ε Expression
O ε Operator
N ε Numeric
relop ε Relational operator
L ε logical operators
A ε Attribute name
R ε relation name

BNF rules could be:

O::= +/-/*/*
N::=0/1/2/3/..../9
E::= arith_exp|const_exp|prob_exp
relop::= =/<>/</<=/>/>=
L::= not/and/or
C ::= create table | drop table | select | from | where | create index | declare
cursor | ....

Select_clause ::= select A | select E
From_clause ::= from R
Where_clause ::= where E
Query ::= select_clause. From_clause. Where_clause | ....

The parse tree for the select query in section 3.7 would be as follows:

```
Query
   |
   v
Where_clause
   |  
Select clause  From clause
   |        |   
Command  Command Command   Logical operator
Select name from faculty_P where dept="MBA" and p[FAC_ID, dept]=0.5
```

The next section gives certain example queries that can be executed on these probabilistic relations using the syntax rules defined above.

### 3.7.3 Example Queries:

#### Query 2:

What is the probability that S K Sharma is in Marketing department?

```
Select p[FAC_ID, Dept] from Faculty_P where name='S K Sharma' and Dept='Mktg'.
```

Here, the probability expression is used more directly to display the required marginal probability. It would be calculated in this case by adding the ps-values of the second and the third rows of the Faculty_P probabilistic relation. The result of the query would be \{0.4\}. 

37
**Query 3** : What is the probability distribution of the Department that S K Sharma works in?

```
Select dept,p[FAC_ID, Dept] from Faculty_P
Where name= ' S K Sharma'.
```

Here, the result would be \{ <MBA,0.6> , <Mktg , 0.4>\}

The next chapter provides an overview of the concepts for the probabilistic transaction and also enumerates the PLACID properties that the probabilistic transaction should possess.