Organization and Analysis of Access Facilities for Set-valued Objects Based on Signature Files

July 1985

Yoshiharu Ishikawa

Organization and Analysis of Access Facilities for Set-valued Objects Based on Signature Files

by

Yoshiharu Ishikawa

A Dissertation Presented to
the Faculty of University of Tsukuba
in Candidacy for the Degree of
Doctor of Engineering

July, 1995

Acknowledgments

First and foremost, I am grateful to my adviser Associate Professor Hiroyuki Kitagawa, provided encouragement, numerous suggestions, and guidance on this research.

I would like to thank to Associate Professor Nobuo Ohbo who motivated me to become involved in research in this area. The constant encouragement of Professor Yuzuru Fujiwara and Professor Isao Suzuki is gratefully acknowledged.

Mr. Yoshiaki Fukushima of NEC and Mr. Noriyasu Watanabe collaborated on the research of set-based signature files. I wish to thank them for their technical assistance and discussions.

Thanks to Associate Professor Kazunori Yamaguchi of University of Tokyo, As-

sistant Professor Jeffrey X. Yu of Australian National University, Dr. Kazutaka Furuse of RICOH Corp., Mr. Takayuki Suzuki, and members of database laboratory of University of Tsukuba for their helpful suggestions.

I have moved to Nara Institute of Science and Technology (NAIST) in 1994. I indebted to Professor Shunsuke Uemura and Associate Professor Masatoshi Yoshikawa of NAIST for their thoughtful and valuable comments. And thanks to all members of Uemura Laboratory.

Finally, I am grateful to Processor Seiichi Nishihara and Associate Professor Yasushi Kiyoki of University Tsukuba for their comments to improve the thesis.

Abstract

Set is a fundamental data structure and plays important role in data modeling not only for traditional database applications but for advanced ones. The main issue of this dissertation is *indexing methods for set-valued objects*. Set has several inherent comparison operators such as inclusion ($\supseteq$) and membership ($\ni$). Therefore, indexing methods for set-valued objects must support efficient retrieval of objects under such set retrieval conditions.

In this dissertation, superimposed-coded *signature files*, popular methods in text retrieval area, are proposed as promising indexing methods for set-valued objects. Such signature files are called *set-based signature files*. In this dissertation, several issues of set-based signature files, such as organization schemes, retrieval/update algorithms, and availability, are investigated.

As queries, four kinds of set retrieval conditions are considered: *has-subset* ($T \supseteq Q$), *is-subset* ($T \subseteq Q$), *has-intersection* ($T \sqcap Q$), and *is-equal* ($T \equiv Q$). For each query, false drop probability formulas, important measures to estimate the performance of signature files, are derived.

The first main issue of this dissertation is *set retrieval of non-nested objects.* For signature files, two representative file organization schemes, the *Sequential Signature File (SSF)* and the *Bit-Sliced Signature File (BSSF)*, are considered. In addition to these organization schemes, *compressed BSSF* (*BSSFcmpr*) is proposed as another candidate for the organization scheme of set-based signature files. For the comparison purpose, the *nested index (NIX)*, an well-known indexing method for nested objects, is also examined as an alternative set access facility. They are compared in terms of retrieval cost, storage cost, and update cost for small-scale databases and medium-scale databases. Based on the analysis, query evaluation strategies (*smart retrieval strategies*) are proposed to improve the retrieval costs of these set retrieval facilities.

The second main issue is *set retrieval of nested objects.* The target is extended to multi-level nested objects with set-valued attributes. By combinating BSSF and the nested index, four candidate set access facilities, $\mathcal{I}_{\text{BSSF}}$, $\mathcal{I}_{\text{NIX}}$, $\mathcal{I}_{\text{BSSF-NIX}}$, and $\mathcal{I}_{\text{NIX-NIX}}$ are proposed. For each set access facility, retrieval, insertion, and deletion algorithms are described. Then, cost formulas including the navigation costs in nested objects are derived, and the retrieval, storage, insertion, and deletion costs are evaluated under some parameter settings.

The analyses in this dissertation clarify the advantages and disadvantages of set-based signature files.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Set is a fundamental data structure and plays important role in data modeling. The relational data model, proposed by E. F. Codd and has become the basis of many commercial and prototype DBMSs, is constructed based on the mathematical notion of set. However, the relational data model lacks of the ability to explicitly handle set values; it can only represent relations, namely, sets of flat tuples. The ability to directly represent and manipulate complex data including set values were required in advanced database application areas (e.g., CAD/CAM, CASE). To meet this requirement, advanced data models with more expressive power, such as nested

relational models and object-oriented database models, have been proposed and investigated. In these data models, some kinds of constructs to represent set-valued objects (e.g., set constructor) are generally provided.

In implementing such advanced data models, there still remain many issues; indexing method is an important one of them. The main issue of this dissertation is *indexing methods for set-valued objects.* Set has several inherent comparison operators such as inclusion ($\supseteq$) and membership ($\ni$). Therefore, indexing methods for set-valued objects must support efficient retrieval of objects under such set retrieval conditions. So far, a lot of indexing schemes for complex objects were proposed. For example, for nested objects, some basic index structures (e.g., the nested index) were proposed and their derivatives were intensively studied [BK89, Ber90, Ber91, Ber93, Ber94, CBBC94, KM94b, KKD90, KP92, LL92b, MS86, PY94, SHH+95, Sai95, SM91]. However, they are not designed to fully support set manipulation in general.

Set-based Signature Files In this dissertation, superimposed-coded *signature files* are proposed as promising indexing methods for set-valued objects. Such signature files are called *set-based signature files.* Although the use of signature files for traditional record retrieval has been discussed by some researchers [CYKL93, Fal85a, Fal90, LL92a, Lin91, PF94, SK86, Sta90, TR92], study on their capabilities from the standpoint of set retrieval has not been reported. In this dissertation, several issues of set-based signature files, such as organization schemes,

retrieval/update algorithms, and availability, are investigated.

As queries, four kinds of set retrieval conditions are considered: *has-subset* ($T \supseteq Q$), *is-subset* ($T \subseteq Q$), *has-intersection* ($T \sqcap Q$), and *is-equal* ($T \equiv Q$). T and Q stand for a set value in the database (*target set*) and the set value in the query condition (*query set*), respectively. $T \equiv Q$ represents retrieval conditions based on set equality and $T \sqcap Q$ means $T \cap Q \neq \emptyset$. Membership $T \ni q$ is a special case of $T \supseteq Q$.

Retrieval with signature files is always accompanied by mismatches called *false drops*. The number of false drops has a direct effect on the performance of signature files. Therefore, it is important to estimate the false drops for the cost estimation and the signature file design. The frequency of false drop is usually measured in *false drop probability*. In this dissertation, formulas estimating false drop probabilities for the four types of retrieval conditions and their derivation are shown.

Set Retrieval of Non-Nested Objects The first main issue of this dissertation is set retrieval of non-nested objects. For signature files, a lot of physical organization schemes have been proposed. In this dissertation, two representative ones, the *Sequential Signature File (SSF)* and the *Bit-Sliced Signature File (BSSF)*, are considered. In addition to these organization schemes, *compressed BSSF* (*BSSFcmpr*) is proposed as another candidate for the organization scheme of set-based signature files. For the comparison purpose, the *nested index (NIX)*, an well-known indexing

method for complex objects, is also examined as an alternative set access facility.

SSF, BSSF, BSSFcmpr, and NIX are compared in terms of retrieval cost, storage cost, and update cost for small-scale and medium-scale databases. A cost model is developed and retrieval costs of the four set retrieval facilities for $T \supseteq Q$ and $T \subseteq Q$ queries are analyzed in detail. Based on the analysis, novel query evaluation strategies to improve the retrieval costs of BSSF and NIX, called *smart object retrieval strategies*, is developed, and further the retrieval costs under the strategies are compared.

Set Retrieval of Nested Objects The second main issue of this dissertation is set retrieval of nested objects. Nested objects frequently appear in databases for advanced application areas and may contain set values in their attributes. Therefore, efficient indexing methods for set retrieval are also required for nested objects. In this chapter, the target is extended to multilevel nested objects with set-valued attributes.

By combinating the signature file method and the nested index (NIX), four candidate set access facilities are proposed. $\mathcal{I}_{\text{BSSF}}$ is the direct extension of BSSF to set retrieval of nested objects. $\mathcal{I}_{\text{NIX}}$ is also direct extension of NIX. $\mathcal{I}_{\text{BSSF-NIX}}$ consists of a BSSF file and an NIX file, and $\mathcal{I}_{\text{NIX-NIX}}$ consists of two NIX files.

The target queries for comparing their retrieval costs are root-level object retrieval with a leaf-level set comparison predicate. As the set comparison condition, $T \supseteq Q$ and $T \subseteq Q$ are considered. For each set access facility and query type, the retrieval algorithm is described in detail. In compared with the non-nested object case, retrieval algorithms for nested objects become more complicated. Similarly, insertion/deletion algorithms are also specified. Based on these algorithms, cost formulas for retrieval, insertion, and deletion are derived and evaluated by simulations.

Outline of the Dissertation The remainder of the dissertation is organized as follows. In Chapter 2, the background of the research – the signature file method and complex object management – is described. Some basic concepts of the signature file method also hold for set-based signature files. In Chapter 3, the notion of set retrieval and set-based signature file is introduced. Then, false drop probability formulas, important measures to estimate performance of signature files, are derived in a generic manner. In Chapter 4 and 5, set retrieval of non-nested objects and nested objects are discussed respectively. Chapter 6 is the final discussion and the conclusion.

Chapter 2

Background

2.1 Signature Files

Signature files were originally proposed for text retrieval [CYKL93, Fal85a, Fal90, LL92a, Lin91, PF94, SK86, Sta90, TR92]. Generally speaking, they require much smaller storage space than inverted files, and can handle update easily. A *signature*

is a bit pattern formed for each data object and stored in the *signature file*. In text retrieval, data objects are usually *logical blocks* (text blocks) consisting of text data including a predefined number of words. In addition to the signature file, there exists an *OID file* (or pointer file, TID file, etc.), which relates each signature to the ID of its corresponding data object.

The *superimposed coding method* is often used to form a signature [Fal85a]. In text retrieval, a *word signature* is first created for each word in a logical block. All word signatures have F-bit length and contain m "1"'s and $F - m$ "0"'s. The number of "1"'s in a signature is called the *weight*, so that the weight of an element signature is m. Then, a *block signature* is composed by OR-ing (superimposing) word signatures for each logical block and stored in the signature file. F and m are design parameters tuned based on performance perspective.

Basic structure of signature file is shown in Figure 2.1. Suppose that the document to be indexed contains a sentence '`signature files were originally proposed for text retrieval`'. In the first step of insertion, the document are divided into logical blocks that contain predefined number of words. In general, *stop words* – words that frequently occur in usual documents such as '`were`' and '`for`' – are eliminated from the logical blocks. For example, three words are to be contained in a block, two logical blocks '`signature files originally`' and '`proposed text retrieval`' may be generated. For these blocks, block signatures are made and inserted into the signature file. Each block signature is accompanied with its block id to specify the location of the block in the document.

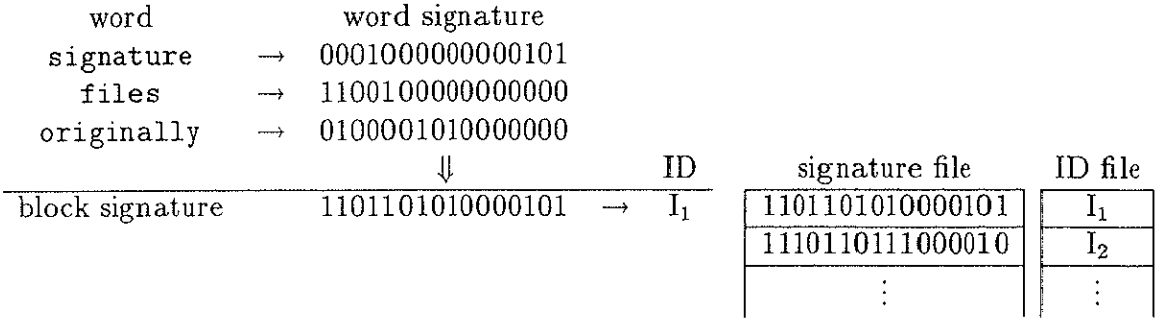

Figure 2.1: Organization of Signature File ($F = 16$, $m = 3$)

Typical query processing for signature files is processed as follows. When a query (one or more words) is given, a *query signature* is formed from the query as the same way to block signatures. Then each signature in the signature file is examined over the query signature for potential match. If the signature satisfies a predefined condition implied by the query condition, the corresponding data object becomes a candidate that may satisfy the query. Such a data object is called a *drop*. The last step is the *false drop resolution*, and each drop is accessed and examined as to whether it actually satisfies the query condition. Drops that fail the test are called *false drops*, while the qualified data objects are called *actual drops*. False drops occur due to the following reasons:

1. Generation of word signatures by hashing.

2. Generation of block signatures based on the superimposed coding.

The number of false drops has a direct effect on the number of disk page accesses. Therefore, it is important to estimate the false drops and to properly control them in the design of signature files. *False drop probability* Fd is an important measure to estimate the performance of signature files and given by the following formula [FC84, FC88]:

$$Fd = \frac{\text{false drops}}{\text{total number of objects} - \text{actual drops}}.$$

Next, representative physical organization schemes for signature files are introduced. There are a number of choices in physical signature file organization. *Sequential signature file (SSF)* and *bit-sliced signature file (BSSF)* are representative and basic ones. Figure 2.2 illustrates the file structures of SSF and BSSF. In this dissertation, they are applied to set retrieval as candidates for set-based signature file organization.

SSF is the simplest organization and directly implements the concept of signature file. SSF is easy to implement and requires low storage space and low update cost. Signatures are stored sequentially in the signature file. When a query is given, a full scan of the signature file is required. Therefore, it is generally slow in retrieval. On the other hand, BSSF stores signatures in a columnar manner. Thus, F files (called *bit-slice files*), one per each bit position of signatures, are used. In retrieval, only a part of the F bit-slice files have to be scanned, so that the search cost is lower than that of SSF. However, update cost becomes larger. For example, an insertion of a new signature typically requires about F disk accesses, one for each bit-slice file.

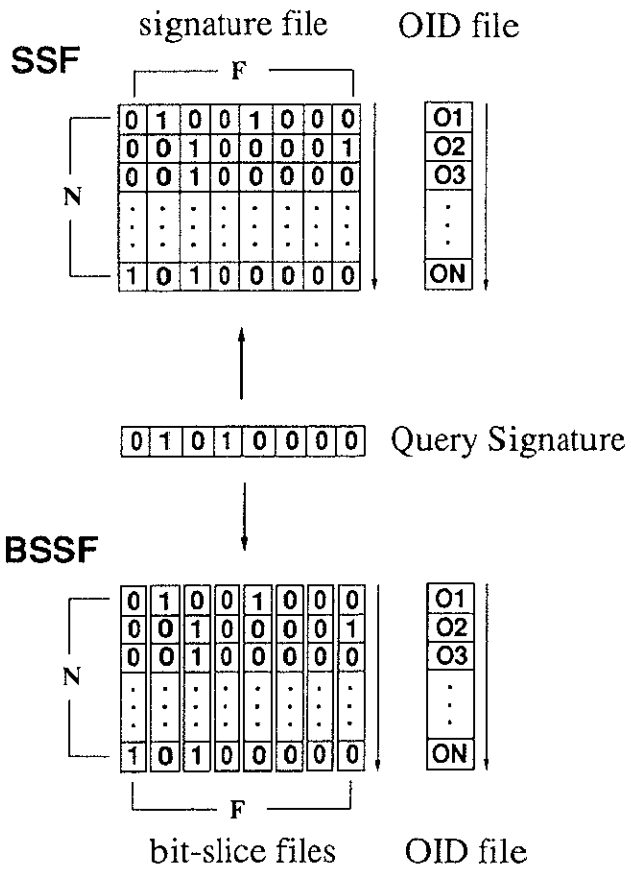

Figure 2.2: SSF and BSSF

Application areas of signature file are not limited to text retrieval. For other areas, the applications have been proposed:

- multikey access, formatted record retrieval [CS89, CL89, CL92, CYL92, Fal88, PBC80, Rob79, Sti60, YCLK93]
- Prolog system [BCH87, CJ86, RS86, TRN86, WW91]
- office information system, office filing [BRG88, CF84, CHT86, DGMS89, FC87, TC83]
- image database [LYC92, RS91, SDR83, SD85]
- document ranking [WLO+85]
- object-oriented database systems [LL92b, YLK94]
- transitive closure query [Teu94]
- image retrieval [RS91, LYC92]
- spatial databases [CJ94]

In each area, signature file organization scheme is devised considering the requirements and performance.

In this dissertation, the signature file method is applied to set retrieval. Application of signature files to set retrieval has not been reported before by other

researchers.

2.2 Complex Object Management

Complex objects frequently appear in advanced database application areas. To represent and manipulate complex objects, many data models have been proposed. Especially, *nested relational models* are closely related to set retrieval because they directly represent and manipulate set values. In the following, the notion of nested relational models is briefly introduced.

The relational model have influenced the construction of the database theory because of its simplicity and rigorous mathematical foundation. However, advanced database applications required more powerful modeling power to manipulate complex data structure in a more natural manner. Therefore, powerful data modeling abilities are required.

Based on such requirements, *nested relational models* were proposed and have been intensively studied. In 1977, Makinouchi proposed to relax the relational model by removing the first normal form assumption [Mak82]. Jaeschke and Schek also generalized the relational model by allowing relations with set-valued attributes and

two operators (nest and unnest) [JS82]. Thomas and Fischer generalized Jaeschke and Schek's model and allowed nested relations of arbitrary depth [TF86]. Many proposals for nested relational models and nested relational algebras were appeared [AB84, AB86, AFS89, Bid87, Col89, Col90, FT83, FSTG85, GF88, GG88, KK89, KKO91, LL90b, Ozs88, RKS88, RKS89, SS86, Sch86, SPS87, TF86]. Also, several prototypes implementing nested relational models were proposed [DKA+86, Dad88, DL89, DG88, HHR93, HSR91, Lin87, PSSD87, PT85, PA86, PD89, SS89, SPSW90, SPS87, SAB+89, VER86]. These nested relational models explicitly handle set-valued attributes or relation-valued attributes. Therefore, set access facilities will be very important in these models.

Other representative data models supporting complex objects are *object-oriented data models*. For object-oriented data models and languages, too many proposals have been studied and cannot cover them here. Research surveys and text books for object-oriented database systems are found in [BM93, Cat91, Dit90, Dit91, Hug91, KM94a, Kho93, KL89, Kim90a, Kim90b, Mas91, Mak91, Tan91, Yos91, ZM90]. These object-oriented data models usually support set-valued objects, and set retrieval should also be important.

Chapter 3

Set-based Signature Files

3.1 Notion of Set Retrieval

In this dissertation, data retrieval based on conditions containing set comparison operators (e.g., $\supseteq$) is called *set retrieval*. In this section, the notion of set retrieval is introduced, and it is shown that the notion is applicable to database query processing

in various situations.

There exist several set comparison operators used in query evaluation. To examine as to whether two sets satisfy subset relationship or not, the inclusion operators ($\supset$, $\supseteq$) are used. The membership operator ($\ni$) is a special case of the inclusion operators to compare a primitive value with a set value. Moreover, the equality operator ($\equiv$) for the set equality and the overlap operator ($\sqcap$), which checks as to whether two sets intersect or not, also appear in queries.

To illustrate examples, let us consider a sample database. The schema of the sample database (Figure 3.1) consists of three class definitions. "$[\cdots]$" denotes the tuple constructor, and "$\{\cdots\}$" denotes the set constructor. The structure of a class is defined by combining these constructors. For example, **Student** class has a primitive attribute *name* and two set-valued attributes *courses* and *hobbies*. The *courses* attribute takes a set of OIDs of **Course** objects as a value, and the *hobbies* attribute takes a set of strings. This convention for schema description is used throughout the dissertation.

Student = [name: *string*, courses: {**Course**}, hobbies: {*string*}]
Course = [name: *string*, category: *string*, teacher: **Teacher**]
Teacher = [name: *string*, ...]

Figure 3.1: Example Schema

Example objects for these classes are shown in Figure 3.2. In the figure, s1 and s2 denote OIDs of **Student** objects, c1 to c6 denote OIDs of **Course** objects, and

t1 and t2 denote OIDs of **Teacher** objects.

```
s1:[name:"Jeff", courses:{c1, c3, c4},
    hobbies:{"baseball", "fishing", "tennis"}]
s2:[name:"Mike", courses:{c1, c3, c5, c6},
    hobbies:{"baseball", "music"}]
...
c1:[name:"Database Theory", category:"database",
    teacher:t1]
c2:[name:"Linear Algebra", category:"math", teacher:t2]
...
```

Figure 3.2: Example Objects

Let us consider the following query $\mathbf{Q}_1$ for this sample database:

Query $\mathbf{Q}_1$ ($T \supseteq Q$): *Find all Students whose hobbies contains both "baseball" and "fishing".*

select name

from Student

where hobbies $\supseteq$ {"baseball", "fishing"}

The set {"baseball", "fishing"} is called the *query set* and represented by Q. On the other hand, each set that resides in the database and becomes the target of the query is called a *target set* and represented by T. In the above database instance, {"baseball", "fishing", "tennis"} is a target set. This kind of query is called "$T \supseteq Q$" (*has-subset*). Membership query ($T \ni q$) is a special case of $T \supseteq Q$.

Let us illustrate other kinds of queries.

Query Q_2 ($T \subseteq Q$, *is-subset*): *Find all Students whose hobbies do not contain other than "baseball", "fishing", "tennis", and "jogging".*

select name
from Student
where hobbies $\subseteq$ {"baseball", "fishing", "tennis", "jogging"}

Query Q_3 ($T \sqcap Q$, *has-intersection*): *Find all Students whose hobbies contains "tennis", or "jogging".*

select name
from Student
where hobbies $\sqcap$ {"tennis", "jogging"}

Namely, $T \sqcap Q$ means $T \cap Q \neq \emptyset$.

Query Q_4 ($T \equiv Q$, *is-equal*): *Find all Students whose hobbies contains only "baseball" and "fishing".*

select name

from Student

where hobbies $\equiv$ {"baseball", "fishing"}

In summary, these four kinds of queries are considered as set retrieval.

1. $\boldsymbol{T} \supseteq \boldsymbol{Q}$ ($\boldsymbol{T} \ni q$): has-subset (membership)
2. $\boldsymbol{T} \subseteq \boldsymbol{Q}$: is-subset
3. $\boldsymbol{T} \sqcap \boldsymbol{Q}$: has-intersection
4. $\boldsymbol{T} \equiv \boldsymbol{Q}$: is-equal

In the next paragraph, some application examples of set retrieval in object-oriented databases are shown.

Application to Object-Oriented Databases In object-oriented databases, multilevel nested objects are frequently appear. For such databases, similar set-retrievals are also considered. Consider the following query $\mathbf{Q}_5$:

Query $\mathbf{Q}_5$ ($T \subseteq Q$): *Find all Students who take only lectures in database, math, or network category.*

select name

from Student

where courses.category $\subseteq$ {"database", "math", "network"}

However, for multilevel nested objects, there are other kinds of queries. Three sample queries $\mathbf{Q}_6$, $\mathbf{Q}_7$, and $\mathbf{Q}_8$ are given below.

Query $\mathbf{Q}_6$:

Find all students who take all of the lectures in the "database" category.

For this kind of query, the following query processing scheme can be used:

1. Retrieve the OIDs of **Course** objects which satisfy the condition '**Course**.*category* = "database"' into a set. Let this set be S_{OID}.
2. Retrieve **Student** objects which satisfy the condition '**Student**.*courses* $\supseteq$ S_{OID}'.

Query $\mathbf{Q}_7$:

Find all students who take only the lectures in the "database" category.

If the search condition '**Student**.*courses* $\supseteq S_{\text{OID}}$' in the above query processing scheme is replaced with '**Student**.*courses* $\subseteq S_{\text{OID}}$', this query can be processed in a similar manner.

Query Q_8:

Find all students who take lecture(s) in the "database" category.

In this case, set comparison operator "$\sqcap$" is used. This query is very popular in complex object databases and can be expressed simply as follows:

select name
from Student
where courses.category = "database"

Query Q_8 can be processed with existing indexing methods for nested objects (e.g., the nested index) [Ber93]. However, for Q_6 and Q_7, such indexing methods cannot be applied directly.

As exemplified above, facilities to efficiently evaluate set predicates are also very important in query processing in object-oriented databases. In general, commercial OODBMSs and object-relational DBMSs support set classes (types) and membership conditions ($T \ni q$) over the classes. For example, the `OC_Set` class of ONTOS OODBMS [ONT] has a member function `OC_Boolean isMember(OC_Argument element)`, and ObjectStore OODBMS [Obj] has a member function `os_int32 contains((E)const)`. UniSQL [Uni], an object-relational DBMS, provide three collection types (`set`, `multiset`, and `sequence`), and some set comparison operators over the set type. For example, $T \ni q$ query can be expressed as

```
select name
from student
where 'tennis' in hobbies;
```

Query $\mathbf{Q}_1$ ($T \supseteq Q$) can be represented by the `superseteq` (or `superset`) operator.

```
select name
from student
where hobbies superseteq {'baseball', 'fishing'};
```

As other set comparison operators, UniSQL have `subset`, `subseteq` ($T \subseteq Q$), `seteq` ($T \equiv Q$), and `setneq`. However, UniSQL does not support indexes over the set type. In UniSQL, indexes can be created on attributes of all data types *except* `set`, `multiset`, and `sequence` [Uni].

3.2 Set-based Signature Files

In this dissertation, the superimposed coding-based signature file technique is applied to efficiently process set retrievals. To facilitate searching qualified set attribute values, *set-based signature files* are created and used for query processing. Such attributes are called *indexed set attributes*. For each indexed set attribute value, the superimposed coding method is used to generate its *set signature*. First, each element in a set value is hashed into a binary bit pattern called an *element signature*. All element signatures have F-bit length, and m bits are set to "1". Then, a set signature is obtained by bit-wise OR-ing (*superimposed coding*) element signatures of all the elements in the set. An example of the element signature generation is shown in Figure 3.3. Each set signature made from an indexed set attribute value is called a *target signature*. Pairs of such a target signature and the OID of the object including the target set are stored in the set-based signature file as shown in Figure 3.4.

set element	element signature	
baseball	0001000000000101	
fishing	1100100000000000	
tennis	0100001010000000	OID
set signature →	1101101010000101	s1

Figure 3.3: Generation of a Set Signature ($F = 16$, $m = 3$)

{"baseball", "fishing", "tennis"}	→	1101101010000101	s1
{"baseball", "music"}	→	1110110111000010	s2
		⋮	⋮

Figure 3.4: Set-based Signature File

When a query is given, a *query signature* is generated from the query set Q and then the signature file is examined. If a target signature satisfies the following condition implied by the set predicate in the query, the corresponding data object becomes a candidate which may satisfy the query.

$T \supseteq Q$: *query signature* $\wedge$ *target signature* = *query signature*

$T \subseteq Q$: *query signature* $\wedge$ *target signature* = *target signature*

$T \sqcap Q$: $weight(\textit{query signature} \wedge \textit{target signature}) \geq m$

$T \equiv Q$: *query signature* = *target signature*

where '$\wedge$' stands for bit-wise AND operation. Then, the drops are retrieved and

checked as to whether they actually satisfy the query condition (false drop resolution).

Figure 3.5 illustrates the query processing of query $\mathbf{Q}_1$ ($T \supseteq Q$). For the two elements in the query set {"baseball", "fishing"}, element signatures are created respectively. Then, a query signature is composed by bit-wise OR-ing the element signatures and target signatures satisfying the above condition for $T \supseteq Q$ are searched. The object with the OID s1 becomes a drop because its target signature satisfies the condition. In this case, the object s1 has an indexed set value {"baseball", "fishing", "tennis"} and actually satisfies the query condition. Therefore, the object s1 becomes an actual drop.

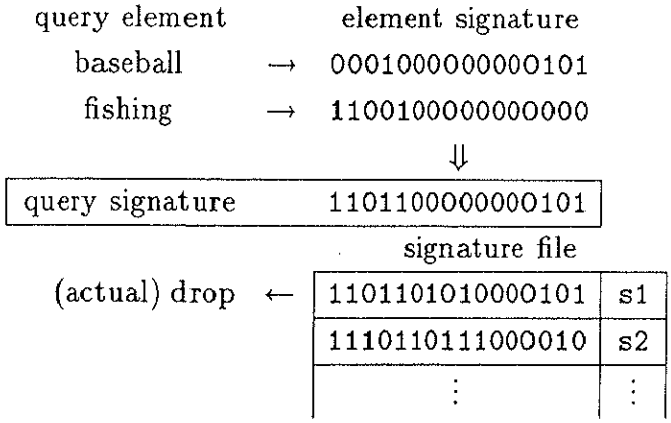

Figure 3.5: Query Processing of $\mathbf{Q}_1$ ($T \supseteq Q$)

Figure 3.6 shows a case where false drops occur. The query with the condition $T \supseteq$ {"skiing", "music"} is processed in this example. While this target set does not satisfy the query condition, the target signature satisfies the above search condition. Therefore, a false drop s2 occurs. This is due to hash collisions and the superimposed

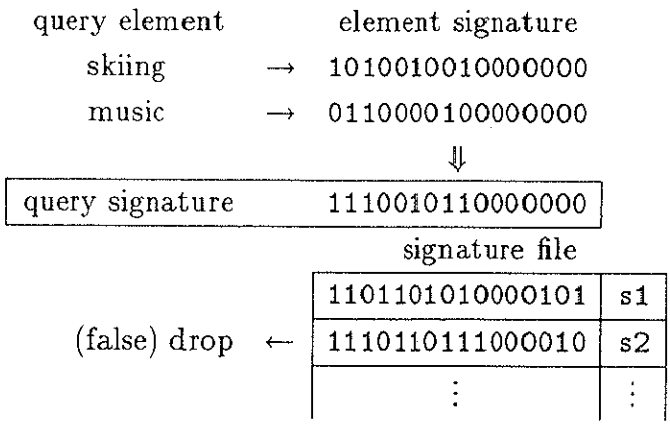

Figure 3.6: False Drop

coding method.

Figure 3.7 illustrates the query processing of query $\mathbf{Q}_2$. In this case, the object s1 becomes a drop and actually satisfies the query (actual drop). Other two queries ($T \sqcap Q$, $T \equiv Q$) are processed in a similar manner.

The query $T \sqcap Q$ could be processed following the above procedure. However, it has been clarified in [KFIO93] that this processing scheme for $T \sqcap Q$ is sometimes undesirable because the number of false drops is rather large. An alternative processing scheme for $T \sqcap Q$ to resolve this problem is shown below:

1) An element signature is generated for each element in the query set Q.

2) The signature file is examined. Each target set becomes a drop if any element

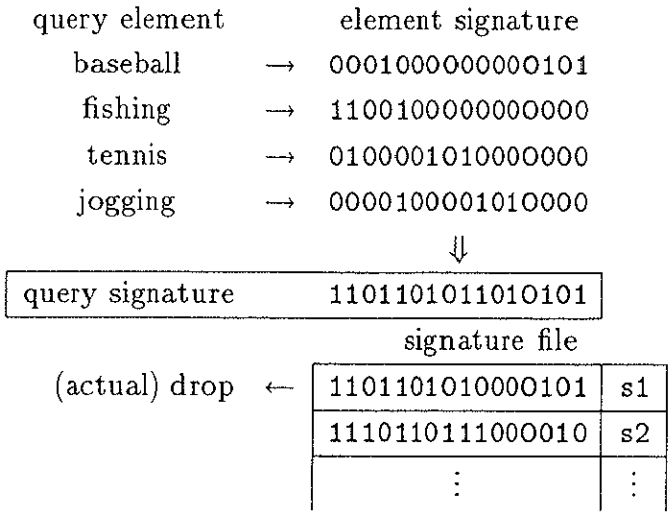

Figure 3.7: Query Processing of $\mathbf{Q}_2$ ($T \subseteq Q$)

signature generated in step 1) satisfies the following condition.

$$element\ signature \wedge target\ signature = element\ signature.$$

3) The drops in step 2) are retrieved and checked whether they actually satisfy the query condition (false drop resolution).

Figure 3.8 illustrates query processing of the query $\mathbf{Q}_3$ under this scheme. In the following discussion, the query $T \sqcap Q$ processed under the first scheme is denoted by $T \sqcap_1 Q$, and that processed under the second scheme is denoted by $T \sqcap_2 Q$.

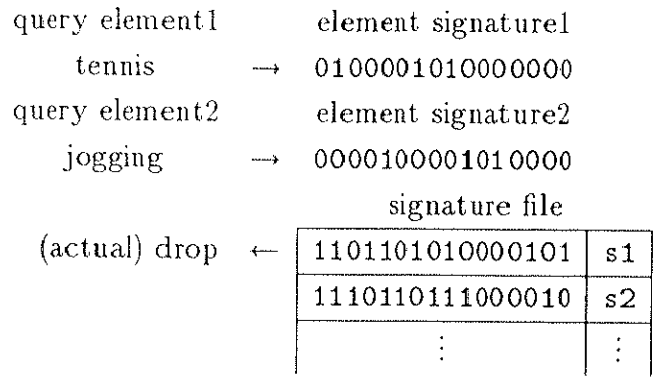

Figure 3.8: Query Processing of $\mathbf{Q}_3$

3.3 False Drop Analysis

3.3.1 Basic Considerations

Table 3.1 shows symbols used in our analysis. For the analysis, following assumptions are made:

1. The weight of an element signature is very small compared with the signature size ($m \ll F$).

2. The "1"'s are uniformly distributed in an element signature. Therefore, each bit position is set to "1" with the same probability.

From the assumption 2, each bit position in an element signature is set with the probability m/F. Therefore, the probability that a bit position b_t of a target signature is set to "1" is given by

$$p(b_t) = 1 - \left(1 - \frac{m}{F}\right)^{D_t} \approx 1 - e^{-\frac{mD_t}{F}} \quad (m \ll F). \tag{3.1}$$

Similarly, the probability that a bit position b_q of a query signature is set to "1" is given by

$$p(b_q) = 1 - \left(1 - \frac{m}{F}\right)^{D_q} \approx 1 - e^{-\frac{mD_q}{F}} \quad (m \ll F). \tag{3.2}$$

Table 3.1: Symbols

symbol	definition
F	Signature size in bits
m	Element signature weight
D_t	Cardinality of a target set T
D_q	Cardinality of a query set Q
N	Total number of target sets in the database
V	Cardinality of the set element domain

The following formula giving the false drop probability for $T \ni q$ was derived by Faloutsos and Christodoulakis [FC84]:

$$Fd_{\{T \ni q\}} = (p(b_t))^m \approx \left(1 - e^{-\frac{mD_t}{F}}\right)^m. \tag{3.3}$$

Stiassny derived the optimal m-value that minimize Eq. (3.3) for m:

$$m_{\text{opt}} = \frac{F \ln 2}{D_t}, \tag{3.4}$$

[Sti60], where ln stands for the natural logarithm. In addition to the assumption 1, he used the following assumption to derive m_{opt}:

$$\frac{V}{\binom{F}{m}} \ll 1,$$

where V is the vocabulary size, namely, the cardinality of the set element domain. Note that m_{opt} does not depend on V. In the text retrieval area, signature files with the superimposed coding often use m_{opt} as m-value [FC84, Fal85a, Fal85b, FC87, FC88].

In this section, false drop probability formulas for $T \supseteq Q$, $T \subseteq Q$, $T \sqcap Q$, and $T \equiv Q$ are derived taking these considerations as a starting basis.

3.3.2 Generic Formulas

First, the case where all target sets have the same cardinality D_{t} is considered. Let b_{t}^j $(1 \leq j \leq F)$ be the j-th bit position of the target signature and b_{q}^j $(1 \leq j \leq F)$ be j-th bit position of the query signature. For each i $(1 \leq i \leq F - m)$, the following equations hold:

$$\mathrm{Prob}\{b_{\mathrm{t}}^1 = 0 \wedge \cdots \wedge b_{\mathrm{t}}^i = 0\} \quad = \quad \left(\frac{\binom{F-i}{m}}{\binom{F}{m}} \right)^{D_{\mathrm{t}}}$$

$$= \left(\frac{(F-m)(F-m-1)\cdots(F-m-i+1)}{F(F-1)\cdots(F-i+1)}\right)^{D_t}$$
$$= \prod_{k=1}^{i}\left(1-\frac{m}{F-k+1}\right)^{D_t}. \quad (3.5)$$

If $\frac{m}{F-k+1} \ll 1$ is satisfied for $1 \le k \le i$,

$$\begin{aligned}\mathrm{Prob}\{b_t^1 = 0 \wedge \cdots \wedge b_t^i = 0\} &= \left(1-\frac{m}{F}\right)^{D_t}\left(1-\frac{m}{F-1}\right)^{D_t}\cdots\left(1-\frac{m}{F-i+1}\right)^{D_t} \\ &\approx \left(1-\frac{1}{F}\right)^{mD_t}\left(1-\frac{1}{F-1}\right)^{mD_t}\cdots\left(1-\frac{1}{F-i+1}\right)^{mD_t} \\ &= \left(\frac{(F-1)(F-2)\cdots(F-i)}{F(F-1)\cdots(F-i+1)}\right)^{mD_t} \\ &= \left(1-\frac{i}{F}\right)^{mD_t}.\end{aligned}$$

The above approximate condition is equivalent to the condition $\frac{m}{F-i+1} \ll 1$. Since $1 \le m \ll F$ (assumption 2), $\frac{m}{F-i+1} \simeq \frac{m}{F-i}$. Therefore, the approximate condition is expressed as $\frac{m}{F-i} \ll 1$. That is, if $m \ll F-i$ is satisfied, Eq. (3.5) can be approximated as

$$\mathrm{Prob}\{b_t^1 = 0 \wedge \cdots \wedge b_t^i = 0\} \simeq \left(1-\frac{i}{F}\right)^{mD_t}. \quad (3.6)$$

Similarly,

$$\mathrm{Prob}\{b_q^1 = 0 \wedge \cdots \wedge b_q^i = 0\} = \prod_{k=1}^{i}\left(1-\frac{m}{F-k+1}\right)^{D_q}. \quad (3.7)$$

If $m \ll F-i$ is satisfied for $1 \le k \le i$,

$$\mathrm{Prob}\{b_q^1 = 0 \wedge \cdots \wedge b_q^i = 0\} \approx \left(1-\frac{i}{F}\right)^{mD_q}. \quad (3.8)$$

In the following, generic false drop probability formulas for $T \supseteq Q$, $T \subseteq Q$, $T \sqcap Q$, and $T \equiv Q$ are derived. The probability that the target signature weight is i is denoted by $p_t(i)$, and the probability that the query signature weight is i is denoted by $p_q(i)$.

1) $\boldsymbol{T \supseteq Q}$: A false drop occurs when the following condition holds for every bit position j $(1 \leq j \leq F)$:

$$b_t^j = 0 \quad \Rightarrow \quad b_q^j = 0.$$

If the target signature weight is i, the number of "0"'s in the target signature is $F - i$. Therefore, the probability $f_{\{T \supseteq Q\}}(i)$ that the target set becomes a false drop is derived from Eq. (3.7) as follows:

$$f_{\{T \supseteq Q\}}(i) = \mathrm{Prob}\{b_q^1 = 0 \wedge \cdots \wedge b_q^{F-i} = 0\} = \prod_{k=1}^{F-i} \left(1 - \frac{m}{F-k+1}\right)^{D_q}.$$

If $m \ll F - i$ holds,

$$f_{\{T \supseteq Q\}}(i) \approx \left(1 - \frac{F-i}{F}\right)^{mD_q} = \left(\frac{i}{F}\right)^{mD_q}$$

is derived from Eq. (3.8). As distribution of the target signature weight is determined by $p_t(i)$, the false drop probability for $T \supseteq Q$ is given by

$$Fd_{\{T \supseteq Q\}} = \sum_{i=0}^{F} p_t(i) \left(\frac{i}{F}\right)^{mD_q}. \tag{3.9}$$

In case $D_q = 1$, this formula gives the false drop probability for $T \ni q$.

2) $T \subseteq Q$: A false drop occurs when the following condition holds for every bit position j $(1 \leq j \leq F)$:

$$b_q^j = 0 \quad \Rightarrow \quad b_t^j = 0.$$

If the query signature weight is i, the probability $f_{\{T \subseteq Q\}}(i)$ that the target set becomes a false drop is derived from Eq. (3.5) as follows:

$$f_{\{T \subseteq Q\}}(i) = \text{Prob}\{b_t^1 = 0 \wedge \cdots \wedge b_t^{F-i} = 0\} = \prod_{k=1}^{F-i} \left(1 - \frac{m}{F-k+1}\right)^{D_t}.$$

If $m \ll F - i$ holds,

$$f_{\{T \subseteq Q\}}(i) \approx \left(1 - \frac{F-i}{F}\right)^{mD_t} = \left(\frac{i}{F}\right)^{mD_t}$$

is obtained from Eq. (3.6). As distribution of the query signature weight is determined by $p_q(i)$, the false drop probability for $T \subseteq Q$ is given by

$$Fd_{\{T \subseteq Q\}} = \sum_{i=0}^{F} p_q(i) \left(\frac{i}{F}\right)^{mD_t}. \tag{3.10}$$

3) $T \sqcap Q$

3.1) $T \sqcap_1 Q$: A false drop occurs when the target signature weight and the query signature weight are at least m, and they have m or more bit intersection.

Therefore,

$$Fd_{\{T \sqcap_1 Q\}} = \sum_{i=m}^{F} p_t(i) \sum_{j=m}^{F} p_q(j) \sum_{k=\max(m, i+j-F)}^{\min(i,j)} \frac{\binom{i}{k}\binom{F-i}{j-k}}{\binom{F}{j}}. \tag{3.11}$$

is obtained. Here, $\sum_{k=\max(m,i+j-F)}^{\min(i,j)} \frac{\binom{i}{k}\binom{F-i}{j-k}}{\binom{F}{j}}$ is the probability that the target signature and the query signature have m or more bit intersection when their weights are $p_t(i)$ and $p_q(j)$, respectively.

3.2) $\boldsymbol{T \sqcap_2 Q}$: The false drop probability for $T \sqcap_2 Q$ is simply expressed with $Fd_{\{T \ni q\}}$ as follows:

$$Fd_{\{T \sqcap_2 Q\}} = \sum_{i=1}^{D_q} Fd_{\{T \ni q\}} \times (1 - Fd_{\{T \ni q\}})^{i-1} = 1 - (1 - Fd_{\{T \ni q\}})^{D_q}. \tag{3.12}$$

4) $\boldsymbol{T \equiv Q}$: A false drop occurs when both the target signature weight and the query signature weight take the same value i, and the target signature is equal to the query signature. Therefore, the false drop probability for $T \equiv Q$ is given by

$$Fd_{\{T \equiv Q\}} = \sum_{i=0}^{F} p_t(i) p_q(i) \frac{1}{\binom{F}{i}}. \tag{3.13}$$

3.3.3 False Drop Probability Formulas

As shown in Subsection 3.3.2, probability distributions of the target and query signature weights denoted by $p_t(i)$ and $p_q(i)$, respectively, play an important role in estimating the false drops. In this subsection, three sets of formulas by estimating $p_t(i)$ and $p_q(i)$ values are derived in the following three different approaches.

Formulas F1

Here, an assumption is made: the target and query signature weights are equal to their expected values m_t and m_q, respectively, given as follows:

$$\overline{m}_t = F \times p(b_t) \approx F(1 - e^{-\frac{mD_t}{F}}) \tag{3.14}$$

$$\overline{m}_q = F \times p(b_q) \approx F(1 - e^{-\frac{mD_q}{F}}). \tag{3.15}$$

Therefore,

$$p_t(i) = \begin{cases} 1 & \text{if } i = F(1 - e^{-\frac{mD_t}{F}}) \\ 0 & \text{otherwise} \end{cases}$$

and

$$p_q(i) = \begin{cases} 1 & \text{if } i = F(1 - e^{-\frac{mD_q}{F}}) \\ 0 & \text{otherwise.} \end{cases}$$

The false drop probability formulas based on these $p_t(i)$ and $p_q(i)$ values are as follows:

1) $T \supseteq Q$:

$$Fd_{\{T \supseteq Q\},\mathrm{F1}} = (1 - e^{-\frac{mD_t}{F}})^{mD_q}. \tag{3.16}$$

2) $T \subseteq Q$:

$$Fd_{\{T \subseteq Q\},\mathrm{F1}} = (1 - e^{-\frac{mD_q}{F}})^{mD_t}. \tag{3.17}$$

3) $T \sqcap Q$:

3.1) $T \sqcap_1 Q$:

$$Fd_{\{T \sqcap_1 Q\},\mathrm{F1}} = \sum_{k=\max(m,\, F(1-e^{-\frac{mD_t}{F}})+F(1-e^{-\frac{mD_q}{F}})-F)}^{\min(F(1-e^{-\frac{mD_t}{F}}),\, F(1-e^{-\frac{mD_q}{F}}))} \frac{\binom{F(1-e^{-\frac{mD_t}{F}})}{k}\binom{F-F(1-e^{-\frac{mD_t}{F}})}{F(1-e^{-\frac{mD_q}{F}})-k}}{\binom{F}{F(1-e^{-\frac{mD_q}{F}})}}. \tag{3.18}$$

3.2) $T \sqcap_2 Q$:

$$Fd_{\{T \sqcap_2 Q\},\mathrm{F1}} = 1 - \left(1 - (1 - e^{-\frac{mD_t}{F}})^m\right)^{D_q}. \tag{3.19}$$

4) $T \equiv Q$:

$$Fd_{\{T \equiv Q\},\mathrm{F1}} = \frac{1}{\binom{F}{\overline{m}}}, \tag{3.20}$$

where $\overline{m} = F(1 - e^{-\frac{mD_t}{F}}) = F(1 - e^{-\frac{mD_q}{F}})$.

Note that Eq. (3.20) for $T \equiv Q$ is applicable only when $D_t = D_q$. Also, note that Eq. (3.16) becomes Eq. (3.3) in case $D_q = 1$.

In Appendix A, some useful properties of $Fd_{\{T \supseteq Q\},\mathrm{F1}}$ and $Fd_{\{T \subseteq Q\},\mathrm{F1}}$ are derived.

Formulas F2

In this approach, we assume that each bit position in the target signature and the query signature is set to "1" with probabilities $p(b_t)$ and $p(b_q)$ (given in Subsection 3.3.1), respectively, independently of other bit positions. Then, the distribution of the target and query signature weights follow the binomial distribution, and we get

$$\begin{aligned} p_t(i) &= \binom{F}{i} p(b_t)^i (1 - p(b_t))^{F-i} \\ p_q(i) &= \binom{F}{i} p(b_q)^i (1 - p(b_q))^{F-i}. \end{aligned}$$

The false drop probability formulas based on these $p_t(i)$ and $p_q(i)$ values are as follows:

1) $T \supseteq Q$:

$$Fd_{\{T \supseteq Q\},\mathrm{F2}} = \sum_{i=0}^{F} \binom{F}{i} (1 - e^{-\frac{mD_t}{F}})^i e^{-\frac{mD_t}{F}(F-i)} \left(\frac{i}{F}\right)^{mD_q}. \tag{3.21}$$

2) $T \subseteq Q$:

$$Fd_{\{T \subseteq Q\},\mathrm{F2}} = \sum_{i=0}^{F} \binom{F}{i} (1 - e^{-\frac{mD_q}{F}})^i e^{-\frac{mD_q}{F}(F-i)} \left(\frac{i}{F}\right)^{mD_t}. \tag{3.22}$$

3) $T \sqcap Q$:

3.1) $T \sqcap_1 Q$:

$$\begin{aligned} & Fd_{\{T \sqcap Q\},\mathrm{F2}} \\ = & \sum_{i=m}^{F} \binom{F}{i} p(b_t)^i (1 - p(b_t))^{F-i} \sum_{j=m}^{F} \binom{F}{j} p(b_q)^j (1 - p(b_q))^{F-j} \sum_{k=\max(m, i+j-F)}^{\min(i,j)} \frac{\binom{i}{k}\binom{F-i}{j-k}}{\binom{F}{j}}. \end{aligned} \tag{3.23}$$

3.2) $T \sqcap_2 Q$:

$$Fd_{\{T \sqcap Q\},\mathrm{F2}} = 1 - \left(1 - \sum_{i=0}^{F} \binom{F}{i} (1 - e^{-\frac{mD_t}{F}})^i e^{-\frac{mD_t}{F}(F-i)} \left(\frac{i}{F}\right)^{m}\right)^{D_q}. \tag{3.24}$$

4) $T \equiv Q$:

$$Fd_{\{T \equiv Q\},\mathrm{F2}} = \sum_{i=0}^{F} \binom{F}{i} (1 - e^{-\frac{mD_t}{F}})^i (1 - e^{-\frac{mD_q}{F}})^i e^{-\frac{m(D_t+D_q)}{F}(F-i)}. \tag{3.25}$$

Formulas F3

In deriving the formulas F1 and F2, two different approaches are employed to estimate the target and query signature weights. Murphree and Aktug derived a more strict mathematical formula giving the probability distribution of the set signature generated by superimposed coding [MA]. Murphree and Aktug considered superimposed coding of D element signatures as a Markov process consisting of D stages. Let m_i $(1 \leq i \leq D)$ be the weight of the i-th element signature, Y_i be the weight of the set signature after the stage i, and W be the final signature weight. Then, $m_1 = Y_1 \leq Y_2 \leq \cdots \leq Y_D = W$ holds. They derived the following probability distribution formula for the set signature weight by analyzing this Markov chain:

$$\text{Prob}\{W = w\} = \sum_{j=0}^{w-m_1} \binom{F-m_1}{j} \binom{F-m_1-j}{w-m_1-j} (-1)^{w-m_1+j} \prod_{r=2}^{D} \frac{\binom{m_1+j}{m_r}}{\binom{F}{m_r}}, \quad (3.26)$$

where $m_1 \leq w \leq \min(F, m_1 + \cdots + m_D)$. When we apply this formula to our context, we get the following formulas:

$$p_t(i) = \begin{cases} \displaystyle\sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_t-1} & \text{if } m \leq i \leq mD_t, \\ 0 & \text{otherwise,} \end{cases}$$

$$p_q(i) = \begin{cases} \displaystyle\sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_q-1} & \text{if } m \leq i \leq mD_q, \\ 0 & \text{otherwise.} \end{cases}$$

The false drop probability formulas based on these $p_t(i)$ and $p_q(i)$ values are as follows:

1) $T \supseteq Q$:

$$Fd_{\{T\supseteq Q\},\mathrm{F3}} = \sum_{i=m}^{\min(F,mD_t)} \sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_t-1} \left(\frac{i}{F}\right)^{mD_q}. \tag{3.27}$$

2) $T \subseteq Q$:

$$Fd_{\{T\subseteq Q\},\mathrm{F3}} = \sum_{i=m}^{\min(F,mD_q)} \sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_q-1} \left(\frac{i}{F}\right)^{mD_t}. \tag{3.28}$$

3) $T \sqcap Q$:

3.1) $T \sqcap_1 Q$:

$$Fd_{\{T\sqcap_1 Q\},\mathrm{F3}} = \sum_{i=m}^{\min(F,mD_t)} p_t(i) \sum_{j=m}^{\min(F,mD_q)} p_q(j) \sum_{k=\max(m,i+j-F)}^{\min(i,j)} \frac{\binom{i}{k}\binom{F-i}{j-k}}{\binom{F}{j}}. \tag{3.29}$$

3.2) $T \sqcap_2 Q$:

$$
\begin{aligned}
& Fd_{\{T \sqcap_2 Q\},\mathrm{F3}} \\
= \; & 1 - \left(1 - \sum_{i=m}^{\min(F,\, D_t)} \sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m_j}{m}}{\binom{F}{m}} \right]^{D_t - 1} \left(\frac{i}{F} \right)^m \right)^{D_q}.
\end{aligned}
\tag{3.30}
$$

4) $T \equiv Q$:

$$
\begin{aligned}
& Fd_{\{T \equiv Q\},\mathrm{F3}} \\
= \; & \sum_{i=m}^{\min(F,\, mD_t,\, mD_q)} \left[\left(\sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_t - 1} \right) \right. \\
& \left. \times \left(\sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_q - 1} \right) \frac{1}{\binom{F}{i}} \right].
\end{aligned}
\tag{3.31}
$$

3.3.4 Varying Target Cardinality

Hitherto, the case where all target sets have the same cardinality D_t has been focused. It is not difficult to extend the above study to the case where the cardinality of the target set varies. Let a *set element domain* be a set from which each target set element is taken, and its cardinality be V. If we do not consider the case that the target set is an empty set, the target set cardinality varies from 1 to V. Suppose the probability that the target set cardinality is D_t is given by the function $P(D_t)$ $(1 \leq D_t \leq V)$. Then, the false drop probability is derived with the following formula

for each case of $T \supseteq Q$, $T \subseteq Q$, $T \sqcap Q$, and $T \equiv Q$:

$$Fd_{\mathrm{VTC},c,f} = \sum_{D_t=1}^{V} P(D_t) Fd_{c,f}, \tag{3.32}$$

where c and f are parameters indicating one of $\{T \supseteq Q\}$, $\{T \subseteq Q\}$, $\{T \sqcap_1 Q\}$, $\{T \sqcap_2 Q\}$, $\{T \equiv Q\}$, and one of F1, F2, F3, respectively. $Fd_{c,f}$ is the false drop probability formula derived in Subsection 3.3 for each combination. For example, the false drop probability for $T \supseteq Q$ based on F3 (Eq. (3.27)) is given as follows:

$$\begin{aligned} & Fd_{\mathrm{VTC},\{T \supseteq Q\},\mathrm{F3}} \\ = & \sum_{D_t=1}^{V} P(D_t) \sum_{i=m}^{\min(F, mD_t)} \sum_{j=0}^{i-m} \binom{F-m}{j} \binom{F-m-j}{i-m-j} (-1)^{i-m+j} \left[\frac{\binom{m+j}{m}}{\binom{F}{m}} \right]^{D_t - 1} \left(\frac{i}{F} \right)^{mD_q}. \end{aligned}$$

Chapter 4

Set Retrieval of Non-Nested Objects

In this chapter, set retrieval of non-nested objects are discussed. SSF (sequential signature file) and BSSF (bit-sliced signature file) are employed as candidates of physical organization schemes for set-based signature files, and their storage cost, update cost, and storage cost are compared. For BSSF, compressed organization (BSSFcmpr) is proposed for medium- and large-scale databases. In addition to these files, the nested index is also examined as a possible index structure for set retrieval.

For signature files, file organization and query evaluation schemes are further devised based on the result of the performance evaluation and the analysis.

4.1 Set Access Facilities for Non-Nested Objects

As described in Chapter 2, there exist various organization schemes for signature files. But most of them are designed for text retrieval, and there have been no proposals for set retrieval. The performance of signature files depend on physical organization scheme. For text retrieval, a lot of file organization schemes are proposed but it is not clear which organization is best suited to set retrieval.

In this chapter, SSF and BSSF, representative signature file organization schemes, are considered as candidates for the organization schemes of set-based signature files. For set retrieval, the basic structure of SSF and BSSF are not different from that in text retrieval. SSF stores target signatures sequentially and BSSF stores them in a columnar manner. However, pattern matching conditions between a query signature and target signatures are different from those in text retrieval applications. For usual text retrieval, one or more keywords are specified and documents are retrieved based on the keywords. Such a query corresponds to $T \supseteq Q$ $(T \ni q)$ query in set retrieval. In addition to this query, there are other kinds of queries $T \subseteq Q$, $T \sqcap Q$, and $T \equiv Q$ for set retrieval and the pattern matching conditions are

different each other. For each query, the advantage and disadvantage of SSF and BSSF should be examined. In addition to SSF and BSSF, compressed BSSF (abbreviated as *BSSFcmpr*) is proposed especially for medium- and large-scale databases, and compared with other facilities.

For a comparative study, the *nested index* (abbreviated as *NIX*) [BK89, Ber93], proposed as an index method for nested objects with B^+-tree-like index structure, is employed as other candidate for set retrieval facility. In a leaf page, NIX stores index entries composed of a key value and the list of OIDs for objects that have the key value in the indexed nested attribute. The format of a nonleaf node is similar to that of B^+-tree. For example, suppose that an object with OID O_1 has a value $\{a, b, c\}$ in its indexed set attribute. In this case, three pairs $\langle a, O_1 \rangle$, $\langle b, O_1 \rangle$, and $\langle c, O_1 \rangle$ are inserted into the B^+-tree-like index file.

4.2 Cost Model

In order to evaluate the performance of set-based signature files and NIX, it is necessary to construct a precise cost model integrating various factors in set retrieval. For set-based signature files, there exist design parameters such as F (signature size) and m (weight in an element signature) as described in Chapter 3. These parameter settings have serious influence on the retrieval, update, and storage cost.

In addition to these design parameters, parameters for the target database also effect on the retrieval cost. For example, the number of entries of indexed objects (N), the distribution of the cardinalities of the target sets, the characteristics of the set domain (e.g., cardinality, distribution), the access costs for objects in the false drop resolution step, the kind of query issued by the user (e.g., $T \supseteq Q$), the cardinality of the query set (D_q), and so on.

In the previous chapter, approximate formulas for false drop probabilities are derived. These formulas are depend on parameters F, m, D_t, D_q and contain exponential expressions. Generally speaking, false drop probabilities dynamically change as the parameter values change. If false drop probability is small, the number of false drops is also small so that the cost of false drop resolution will be small. However, parameter settings that minimize false drop probabilities are not necessarily optimal when total retrieval costs are considered. For example, to minimize the false drop probability for $T \supseteq Q$, F should be infinitely large (property 4 in Appendix A) and m should be m_{opt} (property 1 in Appendix A). But such parameter settings will increase the scan cost for the signature file, then the total retrieval cost become worse.

Many studies on signature files for text retrieval employed the parameter settings that minimize false drop probabilities. If F is fixed, $m = m_{opt}$ (Eq. (3.4)) is such an optimal setting. One of this reason is the query pattern in text retrieval. In text retrieval, $T \ni q$ retrieval accompanying many false drops frequently occurs so that the false drop resolution cost generally dominates the total retrieval cost. However,

in set retrieval, there are some situations such that the false drop probability is so small that the scan cost of signature files rather than the false drop resolution cost dominates the total retrieval cost. To evaluate and analyze the retrieval cost of set-based signature files, the analysis of false drop probability alone is insufficient and a cost model considering the total retrieval cost should be constructed and used.

Table 4.1 lists values for the symbols used in the following analysis. To construct the cost model, some assumptions are made for simplification:

1. Each indexed set attribute value has the same cardinality D_t. Namely, D_t elements are contained in each indexed set attribute value. In the following analysis, two cases of $D_t = 10$ and $D_t = 100$ are considered.

2. The D_t elements of a target set are randomly selected from the set domain with the cardinality V.

3. The D_q elements of a query set are randomly selected from the set domain with the cardinality V.

4. Each object has an unique OID and can be accessed with a constant cost using its OID. For successful search, namely, when the indexed set attribute value of the object satisfies the query condition, the access cost is P_s. For unsuccessful search, the access cost is P_u. These access costs are used to estimate the false drop resolution costs.

Table 4.1: Symbols and Their Values

symbol	definition and value
F	Signature size in bits
m	Number of "1"'s (weight) in an element signature
$Fd_{\{c\}}$	False drop probability
D_t	Cardinality of a target set
D_q	Cardinality of a query set
N	Number of objects (= 32,000 or 320,000)
V	Cardinality of the set domain (= 13,000)
P	Size of a disk page (= 4096 bytes)
b	Number of bits per byte (= 8)
oid	Size of an OID (= 8 bytes)
P_s	Number of page accesses to fetch an object on successful retrieval (= 1 page)
P_u	Number of page accesses to fetch an object on unsuccessful retrieval (= 1 page)
RC	Retrieval cost (pages)
SC	Storage cost (pages)
IC	Insert cost for one set value (pages)
DC	Deletion cost for one set value (pages)
$LC_{\mathrm{OID}\{c\}}(N)$	Access cost for the OID file for N objects (pages)
$SC_{\mathrm{sig}}(N)$	Storage cost of signature file for N objects (pages) (SSF only)
$SC_{\mathrm{bsf}}(N)$	Storage cost of a bit-slice file for N objects (pages) (BSSF only)
$M_{\{c\}}$	Number of bit-slice files to be retrieved
N_{oid}	Number of OIDs in a disk page (= $\lfloor P/oid \rfloor$ = 512)
$SC_{\mathrm{OID}}(N)$	Size of an OID file in pages (= $\lceil N/N_{\mathrm{oid}} \rceil$ pages)
$A_{\{c\}}(N)$	Number of actual drops for N objects

In the following four subsections, the retrieval cost, the storage cost, and the update cost are estimated for each set retrieval facility. Since the performance of these facilities mainly depends on the I/O cost, the costs are estimated in terms of the number of page accesses. Since some of the cost formulas are also used in Chapter 5, cost formulas are derived in rather general forms.

4.2.1 Cost Estimation of SSF

SSF consists of two files: a signature file and an OID file (Figure 2.2). Set signatures and OIDs are sequentially stored in each file.

Retrieval Cost

Set retrieval using SSF is processed as follows.

1. A query signature is formed from the given query set value.
2. SSF is scanned sequentially to examine each target signature. If a target signature satisfies the pattern match condition explained in Section 3.2, the corresponding OID is retrieved. The set of retrieved OIDs is called S_{OID}.

3. For each OID entry in S_{OID}, the object is retrieved and checked as to whether it actually satisfies the query condition. This step is the false drop resolution. Qualified objects are returned to the user.

Based on the discussion in [FC88], the retrieval cost of SSF for N entries is derived as

$$\begin{aligned} RC_{\text{SSF}\{c\}}(N) &= SC_{\text{sig}}(N) + LC_{OID\{c\}}(N) \\ &\quad + P_s A_{\{c\}}(N) + P_u F d_{\{c\}}(N - A_{\{c\}}(N)), \end{aligned} \tag{4.1}$$

where $SC_{\text{sig}}(N)$ is the storage cost of the signature file itself:

$$SC_{\text{sig}}(N) = \left\lceil \frac{NF}{Pb} \right\rceil. \tag{4.2}$$

Since SSF requires a full scan over the signature file, $SC_{\text{sig}}(N)$ is so large that it directly influences the total retrieval cost RC. $LC_{\text{OID}\{c\}}(N)$ is the lookup cost of the OID file and its value is given below. $P_s A_{\{c\}}(N) + P_u F d_{\{c\}}(N - A_{\{c\}}(N))$ is the cost for the false drop resolution step. $P_s A_{\{c\}}(N)$ is the object retrieval cost for actual drops. Similarly, $P_u F d_{\{c\}}(N - A_{\{c\}}(N))$ is the object retrieval cost for false drops. $F d_{\{c\}}(N - A_{\{c\}}(N))$ is the number of false drops.

The lookup cost of the OID file is [1]

$$LC_{\mathrm{OID}\{c\}}(N) = npa(A_{\{c\}}(N) + Fd_{\{c\}}(N - A_{\{c\}}(N)),\ N,\ SC_{\mathrm{OID}}(N)), \tag{4.3}$$

where $SC_{\mathrm{OID}}(N)$ is the storage cost of the OID file, and npa is the formula of Yao to estimate the number of page accesses [Yao77]; To retrieve t records from n records stored on p pages, the number of page accesses is estimated by

$$npa(t,\ n,\ p) = p\left(1 - \prod_{i=1}^{t} \frac{n(1 - 1/p) - i + 1}{n - i + 1}\right).$$

Storage Cost

The storage cost of SSF is given as

$$SC_{\mathrm{SSF}}(N) = SC_{\mathrm{sig}}(N) + SC_{\mathrm{OID}}(N). \tag{4.4}$$

[1] For $LC_{\mathrm{OID}\{c\}}(N)$, other formula

$$LC_{\mathrm{OID}\{c\}}(N) = npa(A_{\{c\}}(N) + Fd_{\{c\}}(N - A_{\{c\}}(N)),\ N_{\mathrm{oid}} SC_{\mathrm{OID}}(N),\ SC_{\mathrm{OID}}(N))$$

can also be considered, but they are not different so much.

Update Cost

The update cost of SSF is as follows:

$$IC_{\mathrm{SSF}} = 4 \tag{4.5}$$

$$DC_{\mathrm{SSF}}(N) = \frac{SC_{\mathrm{OID}}(N)}{2} + 1. \tag{4.6}$$

Insertion of a set value requires two page access (read/write) both to the signature file and the OID file to append the information at the end of the files. In deleting a set value, a delete flag is set in the OID file. To set the delete flag, the corresponding entry is first searched by the given OID. Therefore, $\frac{SC_{\mathrm{OID}}(N)}{2}$ page accesses is required. When the delete flag is set, the page is written back. Therefore, one page access is needed.

4.2.2 Cost Estimation of BSSF

BSSF consists of F different bit-slice files and an OID file as illustrated in Figure 2.2.

Retrieval Cost

Set retrieval using BSSF slightly differs for $T \supseteq Q$ and $T \subseteq Q$, and is described as follows:

1. A query signature is formed from the given query set value.
2. If the query is $T \supseteq Q$,
   (a) For each bit position which is set to "1" in the query signature, the corresponding bit-slice file is retrieved. Therefore, the expected number of bit-slice files to be retrieved is equal to m_q, the weight of the query signature. The expected value of m_q is given as $\overline{m}_q \simeq F(1 - e^{-\frac{m}{F}D_q})$ (Eq. (3.15)).
   (b) These bit-slice files are AND-ed together. For each entry where "1" is set in the resulting AND-ed bit-slice file, the corresponding OID is retrieved from the OID file into a set S_{OID}.
3. If the query is $T \subseteq Q$,
   (a) For each bit position which is set to "0" in the query signature, the corresponding bit-slice file is retrieved. Therefore, the expected number of bit-slice files to be retrieved is $F - m_q$.
   (b) These bit-slice files are OR-ed together. For each entry where "0" is set in the resulting OR-ed bit-slice file, the corresponding OID is retrieved from the OID file into a set S_{OID}.

4. For each OID entry in S_{OID}, the object is retrieved and checked as to whether it actually satisfies the query condition. Qualified objects are returned to the user.

The retrieval cost of BSSF is as follows:

$$\begin{aligned} RC_{\text{BSSF}\{c\}}(N) &= SC_{\text{bsf}}(N) \times M_{\{c\}} + LC_{\text{OID}\{c\}}(N) \\ &\quad + P_s A_{\{c\}}(N) + P_u F d_{\{c\}}(N - A_{\{c\}}(N)). \end{aligned} \tag{4.7}$$

The first term represents the retrieval cost for bit-slice files. $SC_{\text{bsf}}(N)$ is the storage cost for a bit-slice file and given by

$$SC_{\text{bsf}}(N) = \left\lceil \frac{N}{Pb} \right\rceil. \tag{4.8}$$

M is the expected number of bit-slice files to be retrieved and given by

$$M_{\{T \supseteq Q\}} = m_q \tag{4.9}$$

$$M_{\{T \subseteq Q\}} = F - m_q. \tag{4.10}$$

$LC_{\text{OID}\{c\}}(N)$ is same as SSF (Eq. (4.3)).

Storage Cost

The storage cost of BSSF is given as

$$SC_{\mathrm{BSSF}}(N) = SC_{\mathrm{bsf}}(N) \times F + SC_{\mathrm{OID}}(N). \tag{4.11}$$

Update Cost

To derive the update cost of BSSF, the following assumption is made:

1. At the creation time of a BSSF file, all bits of F bit-slice files are initialized to "0".
2. On the insertion of an entry, the signature is appended to the end of the BSSF file.
3. On the deletion of an entry, the signature for the entry is re-generated, and the signature and the OID for the entry are given to the deletion procedure.

Therefore, the insertion algorithm is given as follows:

1. For each bit position in the given signature, the bit-slice file page corresponding to the insertion position is retrieved, and the bit of the insertion position is turned to "1", then the page is written back. The expected number of bits to be turned to "1" is equal to m_t, the weight of target signature, and is estimated as $\overline{m}_t \simeq F\left(1 - e^{-\frac{mD_t}{F}}\right)$ (Eq. (3.14)).

2. The OID file page corresponding to the insertion position is read and updated, then written back.

Thus, the insertion cost of BSSF is

$$IC_{\text{BSSF}} = 2(\overline{m}_t + 1). \tag{4.12}$$

The deletion algorithm is as follows:

1. The OID file entry is searched by the given OID. When the OID is found, a delete flag is set at the deletion position in the OID file, and the page is written back.

2. For each bit position that is set to "1" in the given signature, the bit-slice file page corresponding to the deletion position is retrieved, and the bit of the deletion position is turned to "0", then the page is written back.

Therefore, the deletion cost is

$$DC_{\mathrm{BSSF}}(N) = \frac{SC_{\mathrm{OID}}(N)}{2} + 2\overline{m}_t + 1. \quad (4.13)$$

The first and the third terms are the update cost of the OID file and the second term is the update cost of the bit-slice files.

4.2.3 Cost Estimation of BSSFcmpr

As described in the following cost evaluation, the overall retrieval cost of BSSF in set retrieval become better when the parameter m is very small (e.g., $m = 2, 3$). In such a parameter setting, each bit-slice file is very sparse and many bits are set to "0"'s. Therefore, it would be considered that the retrieval cost and the storage cost of BSSF might be improved by the compression of bit-slice files. In this study, such a BSSF file is called *compressed BSSF* and abbreviated as *BSSFcmpr*. In the following, the bit-length of a bit-slice file with compression is estimated. As a compression scheme, the run-length coding method of [GV75] is assumed.

The retrieval algorithm for BSSFcmpr is not different from that of BSSF. Only one exception is that BSSFcmpr requires decoding of compressed bit-slice files. As this cost model only considering I/O cost, the cost for decoding is not included in

the retrieval cost. The update algorithms for BSSFcmpr are also similar to those of BSSF. But the update costs of BSSFcmpr may become larger because BSSFcmpr does not permit direct access to a bit in a bit-slice file; decoding of the entire bit-slice file is required to change only one bit value. To reduce update costs, it would be better to divide a bit-slice file into some segments and to compress the segments independently. Although the worst naive update costs can be derived, the derivation of update costs for the improved scheme is rather difficult. Therefore, the update costs for BSSFcmpr are not derived in this analysis; more efficient update algorithms and their analysis are leaved as a future research theme.

The retrieval cost and storage cost of BSSFcmpr are expressed by Eq. (4.7) and Eq. (4.11), respectively. But $SC_{\mathrm{bsf}}(N)$ in these formulas is differ from the case of BSSF. Now the expected size $SC_{\mathrm{bsf}}(N)$ for BSSFcmpr is estimated. Let θ be the probability that "0" occurs in a bit in a bit-slice file and let l be an integer that satisfies

$$\theta^l + \theta^{l+1} \leq 1 < \theta^l + \theta^{l-1}.$$

The expected bit-length of a code with the optimal encoding, $\overline{n}$, is given as

$$\overline{n} = \lceil \log_2 l \rceil + 1 + \frac{\theta^k}{1-\theta^l}, \tag{4.14}$$

where $k = 2^{\lfloor \log_2 l \rfloor + 1} - l$ [GV75].

Since the probability that a bit in a signature stored in a BSSF file is "1" is given

by $p(b_t) \approx 1 - e^{-\frac{mD_t}{F}}$ (Eq. (3.1)), θ is derived as

$$\theta = 1 - p(b_t) = e^{-\frac{mD_t}{F}}. \tag{4.15}$$

After all, the expected size of a bit-slice file with compression is

$$SC_{\mathrm{bsf}}(N) = \left\lceil \frac{Np(b_t)\overline{n}}{Pb} \right\rceil, \tag{4.16}$$

where $Np(b_t)$ is the expected number of bits with the value "1". This is equal to the number of sequences of "0"'s in a bit-slice file.

4.2.4 Cost Estimation of NIX

Bertino and Kim compared NIX with the path index and the multi-index in the context of navigation queries in OODBs [BK89]. In the following, the costs of NIX for the queries $T \supseteq Q$ and $T \subseteq Q$ are estimated by extending the cost model introduced in [BK89, CBBC94]. Symbols for NIX are listed in Table 4.2. Other parameters are same as the case of signature files.

Table 4.2: Symbols for NIX

symbol	definition and value
x	Total number of keys
y	Average number of objects whose indexed set attribute includes a given set element value
z	Number of given key values
n_k	Number of records at level k
p_k	Number of pages at level k
$ll(y)$	Size of a leaf-node index entry for NIX in bytes
kl	Size of a key value for NIX (= 8 bytes)
$noid$	Size of a field which specifies the number of OID entries (= 2 bytes)
f	Average fanout from a nonleaf-node (= 218)
rc	Retrieval cost per index entry (pages)
$ds(y)$	Size of a directory in bytes
pp	Size of a page pointer (= 4 bytes)

NIX Configuration

The cost formulas for NIX will also be used in Chapter 5. Therefore, generic cost formulas are derived here.

Let x be the total number of keys, y be the number of entries corresponding to a key value in a leaf-node page of NIX, n_k and p_k be the numbers of records and pages at level k of NIX, and h be the height of NIX. The cost of NIX depends on the

condition that a leaf entry is larger than one page or not. The size of a leaf entry is

$$ll(y) = \begin{cases} y \times oid + kl + noid & \text{if } ll(y) \leq P \\ y \times oid + kl + noid + ds & \text{if } ll(y) > P, \end{cases} \tag{4.17}$$

where $ds(y)$ is the size of the directory created when a leaf entry exceeds one page, and given by

$$ds(y) = \left\lceil \frac{y \times oid + kl + noid}{P} \right\rceil \times (oid + pp), \tag{4.18}$$

where pp is the size of a page pointer. Therefore, the total number of leaf pages is given by

$$p_h = \begin{cases} \left\lceil \frac{x}{\lfloor P/ll(y) \rfloor} \right\rceil & \text{if } ll(y) \leq P \\ x \times \left\lceil \frac{ll(y)}{P} \right\rceil & \text{if } ll(y) > P. \end{cases} \tag{4.19}$$

The number of entries in a nonleaf-node page in level k $(1 \leq k \leq h-1)$ is

$$h_k = f, \tag{4.20}$$

where f is the *fanout* of NIX. The number of nonleaf-node pages is

$$p_k = \left\lceil \frac{p_{k+1}}{f} \right\rceil. \tag{4.21}$$

When z index keys are given in a query, the retrieval cost of NIX is [2]

$$rc_{\mathrm{NIX}}(x,\ y,\ z) = \begin{cases} \sum_{k=1}^{h} npa(t_k,\ n_k,\ p_k) & \text{if } z \geq 2 \\ h & \text{if } z = 1 \end{cases} \tag{4.22}$$

[CBBC94], where $t_h = z$ and $t_{k-1} = npa(t_k,\ n_k,\ p_k)$.

In the cost analyses in Section 4.3 and Section 4.4, there are four cases of combinations of D_t- and N-values. NIX configuration for each combination is shown in Table 4.3. $N = 32,000$ assumes small-scale databases and $N = 320,000$ assumes medium-scale databases. For only the case of $D_t = 100$ and $N = 320,000$, $ll(y)$ exceeds one page and the height of NIX becomes 3. For other cases, the height of NIX is 2.

Table 4.3: Configuration of NIX

N and D_t		p_h (pages)	nonleaf pages	h
$N = 32,000$	$D_t = 10$	650	4	3
$N = 32,000$	$D_t = 100$	6,500	29	3
$N = 32,0000$	$D_t = 10$	6,500	29	3
$N = 32,0000$	$D_t = 100$	65,000	280	4

[2] Note that this retrieval is not set retrieval but usual one.

Retrieval Cost

Set retrieval algorithms using NIX differ for $T \supseteq Q$ and $T \subseteq Q$, and are described as follows:

$T \supseteq Q$

1. For each element in the query set, the OIDs corresponding to the element is retrieved using NIX. As a result, D_q sets of OIDs are obtained.
2. The intersection of the D_q sets is taken. For each OID in the resulting set, the object is retrieved and returned to the user.

$T \subseteq Q$

1. For each element in the query set, the OIDs corresponding to the element is retrieved using NIX. As a result, D_q sets of OIDs are obtained.
2. The union of the D_q sets is taken. For each OID in the resulting set, the object is retrieved and checked as to whether it actually satisfies the query condition. Qualified objects are returned to the user. Note that the element in the union does not necessarily satisfies the query condition. Therefore, a processes corresponding to the false drop resolution in signature files is required.

The retrieval cost of NIX is given as follows:

$T \supseteq Q$

$$RC_{\mathrm{NIX}\{T \supseteq Q\}}(N) = rc_{\mathrm{NIX}}\left(V, \frac{D_t N}{V}, D_q\right) + P_s A_{\{T \supseteq Q\}}(N) \tag{4.23}$$

$T \subseteq Q$

$$\begin{aligned} RC_{\mathrm{NIX}\{T \subseteq Q\}} &= rc_{\mathrm{NIX}}\left(V, \tfrac{D_t N}{V}, D_q\right) + P_u N \frac{\sum_{i=1}^{D_t-1} \binom{D_q}{i}\binom{V-D_q}{D_t-i}}{\binom{V}{D_t}} \\ &\quad + P_s N \frac{\binom{D_q}{D_t}}{\binom{V}{D_t}} \end{aligned} \tag{4.24}$$

The derivation of the second and the third terms for $T \supseteq Q$ is explained as follows; After D_q index lookups, the number of objects to be accessed is given by

$$N \frac{\sum_{i=1}^{D_t-1} \binom{D_q}{i}\binom{V-D_q}{D_t-i}}{\binom{V}{D_t}} + N \frac{\binom{D_q}{D_t}}{\binom{V}{D_t}},$$

where the first term expresses the number of objects which do not satisfy the query condition, and the second term represents the number of objects which actually satisfy the query condition. Therefore, Eq. (4.24) is obtained.

Storage Cost

The generic formula for the storage cost of NIX is given by

$$SC_{\mathrm{NIX}}(x, y) = \sum_{k=1}^{h} p_k. \tag{4.25}$$

This formula is used in Chapter 5. For the following analyses, parameters x and y are set to V and $\frac{D_t N}{V}$, respectively. Therefore, the storage cost of NIX for the analyses in this chapter is

$$SC_{\mathrm{NIX}}\left(V, \frac{D_t N}{V}\right). \tag{4.26}$$

Update Cost

The generic insertion and the deletion costs based on z key values are

$$\begin{aligned} IC_{\mathrm{NIX}}(x, y, z) &= DC_{\mathrm{NIX}}(x, y, z) \\ &= \begin{cases} \sum_{k=1}^{h} npa(t_k, n_k, p_k) + npa(t_h, n_h, p_h) & (z \geq 2) \\ h + 1 & (z = 1). \end{cases} \end{aligned} \tag{4.27}$$

The second term for each formula represents the rewrite cost. On insertion and deletion of a target set, D_t elements are inserted or deleted. Therefore, the insertion/deletion formula of NIX for the analyses in this chapter is given as

$$IC_{\mathrm{NIX}}\left(V, \frac{D_t N}{V}, D_t\right) \tag{4.28}$$

$$DC_{\mathrm{NIX}}\left(V, \frac{D_t N}{V}, D_t\right). \tag{4.29}$$

4.2.5 Actual Drops

False drop probability formulas are derived in Section 3.3. False drops occur only in signature files. However, actual drops are related to all access facilities. Here, actual drop formulas for $T \supseteq Q$ and $T \subseteq Q$ are derived.

$T \supseteq Q$

In this case, we assume that $D_t \geq D_q$. Since the cardinality of the set domain is V, the probability that a query set value becomes a subset of a target set is

$$\frac{\binom{V-D_q}{D_t-D_q}}{\binom{V}{D_t}}.$$

The actual drop A is the expected number of such target sets. Therefore, we have

$$A_{\{T \supseteq Q\}}(N) = N \frac{\binom{V-D_q}{D_t-D_q}}{\binom{V}{D_t}}. \tag{4.30}$$

$T \subseteq Q$

We assume that $D_q \geq D_t$. The probability that a query set value becomes a superset of a target set is

$$\frac{\binom{D_q}{D_t}}{\binom{V}{D_t}}.$$

Therefore, we have

$$A_{\{T \subseteq Q\}}(N) = N \frac{\binom{D_q}{D_t}}{\binom{V}{D_t}}. \tag{4.31}$$

This actual drop value is almost negligible for probable values of D_t and D_q.

4.3 Cost Analysis for Small-scale Databases

In this section, the retrieval costs for SSF, BSSF, and NIX for queries $T \supseteq Q$ and $T \subseteq Q$ are compared in the context of small-scale databases ($N = 32,000$). As D_q-values, two cases of $D_t = 10$ and $D_t = 100$ are considered. In this section and the following section, F-values are set in a manner that the storage costs of signature files are less or at most equal to that of NIX. As false drop probabilities, formulas F1 in Section 3.3, simple but moderately accurate formulas

$$
\begin{aligned}
Fd_{\{T \supseteq Q\}} &= Fd_{\{T \supseteq Q\},\mathrm{F1}} \\
&= \left(1 - e^{-\frac{mD_t}{F}}\right)^{mD_q} \qquad \text{(Eq. (3.16))} \\
Fd_{\{T \subseteq Q\}} &= Fd_{\{T \subseteq Q\},\mathrm{F1}} \\
&= \left(1 - e^{-\frac{mD_q}{F}}\right)^{mD_t} \qquad \text{(Eq. (3.17))}
\end{aligned}
$$

are used.

4.3.1 Retrieval Cost for $T \supseteq Q$

Overall Trend of Retrieval Cost

First, the retrieval costs of SSF, BSSF, and NIX for $T \supseteq Q$ in small-scale databases ($N = 32,000$) are presented. Figure 4.1 shows the retrieval costs of three set access facilities under $D_t = 10$ and $N = 32,000$. D_q varies from 1 to 10. For signature files, there exist two design parameters F and m. In this case, two signature sizes $F = 250$ (bits) and $F = 500$ (bits) are used, and m_{opt} (Eq. (3.4)) is used as the m-value. In the context of text retrieval, m_{opt} is a typical m-value. When $m = m_{opt}$, the false drop probability is given by $Fd_{\{T \supseteq Q\}} \simeq \left(\frac{1}{2}\right)^{\frac{D_q}{D_t} F \ln 2}$ (see Eq. (A.1) in Appendix A). This false drop probability is almost negligible under the parameters of this figure.

In Figure 4.1, the retrieval costs of SSF and BSSF are higher than that of NIX. Since SSF requires a full scan over the signature file in retrieval, $SC_{sig}(N)$, the storage cost of the signature file, directly influences the retrieval cost. If smaller signature size F is chosen, the storage cost might decrease. However, the false drop probability will increase – this is a dilemma of SSF. On the other hand, the retrieval cost of BSSF does not depend on the storage cost but on $\overline{m}_q$, the expected number of "1"'s in the query signature. Since $\overline{m}_q$ increases as D_q increases, the retrieval cost of BSSF becomes higher for larger D_q as shown in Figure 4.1. However, the retrieval cost of BSSF can be improved. In the following subsection, the tuning of the parameter m of BSSF is discussed.

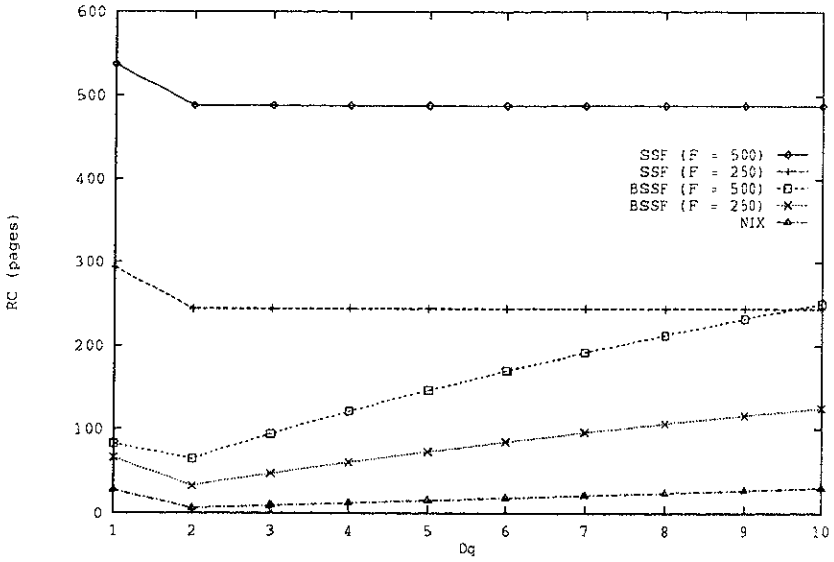

Figure 4.1: Retrieval Cost ($D_t = 10$, $N = 32,000$)

For three set access facilities, similar properties are also observed for $D_t = 100$.

Tuning m-value for BSSF

The retrieval cost of BSSF is directly influenced by the m-value. Although m_{opt} gives the minimum false drop probability, it is not necessarily the optimal choice for the total retrieval cost. If smaller m-value is taken, the total retrieval cost would reduce, even if the false drop probability becomes larger. However, if m is too small, the total cost increases drastically because the false drop probability becomes intolerable.

For $D_t = 10$ and $D_t = 100$, the retrieval cost of BSSF with small m-values is compared with that of NIX. Figure 4.2 shows the retrieval cost for $D_t = 10$ and $F = 500$. In this case, m ranges from 1 to 4. As shown in the figure, when $D_q = 1$, BSSF is inferior to NIX; For small D_q-values ($D_q = 1, 2$), the number of actual drops (Eq. (4.30)) is large but this is common to BSSF and NIX. In addition to actual drops, BSSF suffers from the overhead of false drops especially for small D_q-values and small m-values. However, for other D_q-values, the retrieval cost of BSSF with small m-values is comparable to or lower than that of NIX. This is because actual drops and false drops are turned to be negligible. In other cases (e.g., $D_t = 100$, $F = 2500$), similar results are obtained so that the advantage of BSSF with a small m-value is realized.

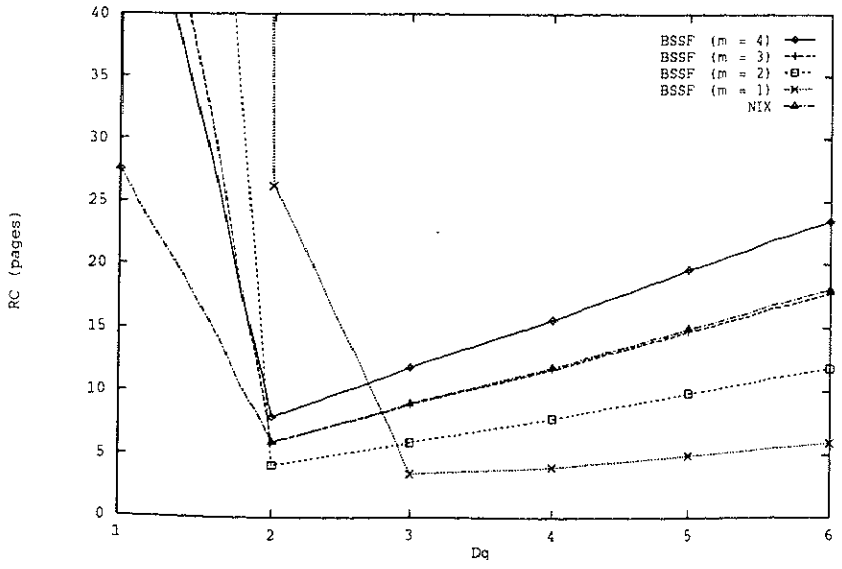

Figure 4.2: Retrieval Cost with Small m-value ($D_t = 10$, $F = 500$, $N = 32,000$)

Smart Retrieval Strategy

If Figure 4.2 is observed carefully, it can be seen that there exist efficient object retrieval strategies for BSSF and NIX which reduce the expected number of page accesses.

Suppose that a query with $D_q = 3$ is issued by the user to BSSF with $m = 2$. If the object retrieval is done as described in Subsection 4.2.2, it costs 6.0 page accesses as shown in Figure 4.2. However, note that it is not necessary to lookup all the $\overline{m}_q$ bit-slice files to answer the query. Only part of the $\overline{m}_q$ bit-slice files are sufficient in selecting candidate objects, since the final qualification of the candidates is done at the false drop resolution step in any case. In this example, if only two elements in the query set are used to form a query signature, the total cost decreases to 4.0 pages, as indicated by the page accesses for $D_q = 2$ and $m = 2$. Although the increase of D_q leads the reduction of false drops and actual drops, it also brings about the increase of $\overline{m}_q$. Therefore, it is not wise to stick to looking up all the $\overline{m}_q$ bit-slice files, as exemplified by the observation here.

The above discussion suggests the following *smart retrieval strategy* for BSSF ($m = 2$ in Figure 4.2):

1. If $D_q = 1$ or $D_q = 2$, lookup BSSF as described in Subsection 4.2.2.

2. If $D_q \geq 3$, form a query signature from only two arbitrary elements in the query set and lookup BSSF using this query signature.

Under this smart strategy, the retrieval cost become constant for $D_q \geq 2$.

Figure 4.2 indicates that the smart retrieval strategy can be applied to NIX in a similar way. The smart strategy for NIX is as follows:

1. If $D_q = 1$ or $D_q = 2$, use NIX as described in Subsection 4.2.4. D_q index lookups are needed.

2. If $D_q \geq 3$, lookup NIX only two times for two arbitrary elements in the query set. Then, take the intersection of the two sets. Finally, for each OID in the resulting set, retrieve the object and check as to whether it actually satisfies the query condition.

The retrieval costs under these smart retrieval strategies are shown in Figure 4.3 ($D_t = 10$) and Figure 4.4 ($D_t = 100$). As shown in these figures, NIX has an advantage only for $D_q = 1$. BSSF shows almost equal or lower retrieval cost for $D_q \geq 2$ in Figure 4.3 and $D_q \geq 3$ in Figure 4.4. Therefore, BSSF is comparable to NIX for $T \supseteq Q$.

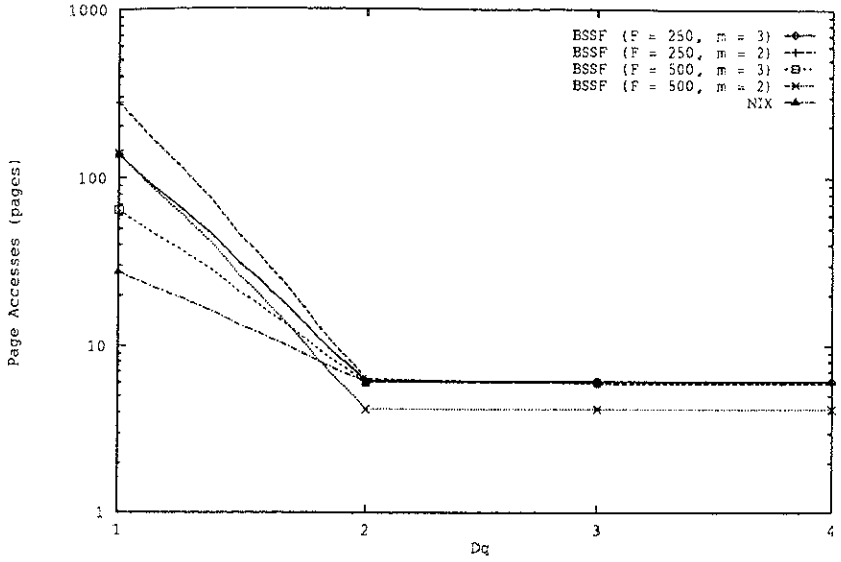

Figure 4.3: Smart Retrieval Cost ($D_t = 10$, $N = 32,000$)

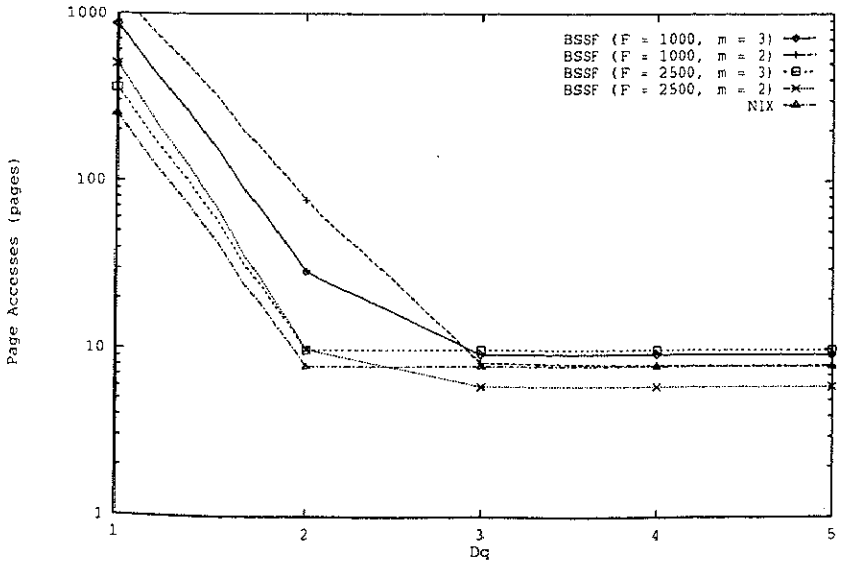

Figure 4.4: Smart Retrieval Cost ($D_t = 100$, $N = 32,000$)

4.3.2 Retrieval Cost for $T \subseteq Q$

Overall Trend of Retrieval Cost

Now the retrieval costs for $T \subseteq Q$ are analyzed for two D_t values $D_t = 10$ and $D_t = 100$. First, let us observe the overall trend of the retrieval costs of SSF, BSSF and NIX.

Figure 4.5 shows the retrieval costs of the three access facilities for $D_t = 10$ and $F = 500$. D_q varies from 10 $(= D_t)$ to 1000.

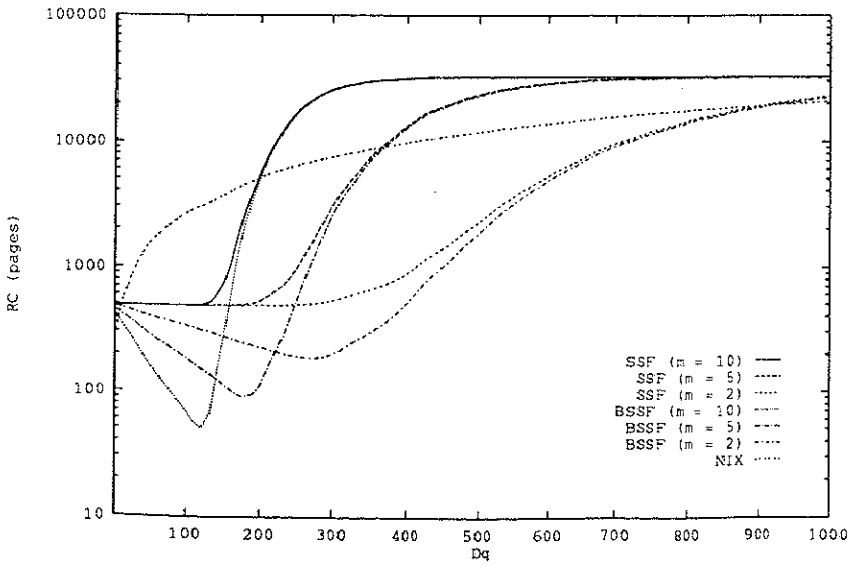

Figure 4.5: Retrieval Cost ($D_t = 10, N = 32,000$)

At first, SSF and BSSF are compared.. For large D_q values, the retrieval costs of SSF and BSSF are high and increase monotonically as D_q becomes larger. Both the retrieval costs of SSF and BSSF, given by Eq. (4.1) and Eq. (4.7), come close to $P_u N$ for large D_q values. The reason is that the false drop probability is almost 1 for large D_q-values [3]. In these situations, most of the data objects have to be accessed to answer the query. For small D_q-values, the retrieval cost of BSSF is superior to that of SSF. This is due to the difference of the file organizations; SSF requires a full scan over the signature file, whereas BSSF only requires accesses to $F - \overline{m}_q$ bit-slice files. For all D_q values, Figure 4.5 shows superiority of BSSF over SSF under the same m-value settings.

The retrieval cost of NIX increases monotonically as D_q increases. As described in Subsection 4.2.4, D_q lookups of NIX is performed for $T \subseteq Q$, and then the union of the retrieved sets of OIDs is taken. As D_q increases, the cardinality of the resulting set comes close to N. For other values of F and D_t, similar results are observed.

Smart Object Retrieval

As $T \supseteq Q$, the retrieval strategy of BSSF for $T \subseteq Q$ can also be improved. In the following, the smart retrieval strategy for $T \subseteq Q$ is described and the retrieval cost of BSSF with the smart strategy is compared with that of NIX.

[3] Actual drops $A_{\{T \subseteq Q\}}(N)$ is negligible in $T \subseteq Q$ queries for probable D_q-values.

Note that BSSF with $m = 2$ in Figure 4.5 takes the minimum value when $D_q \simeq 300$. Considering the object retrieval costs for $D_q = 100$ and $D_q = 300$, the difference is mainly due to the number of bit-slice files to be retrieved (i.e., $\left\lceil \frac{N}{Pb} \right\rceil (F - \overline{m}_q)$). It costs 335 pages for $D_q = 100$ and 150 pages for $D_q = 300$. Note that, for $D_q \leq 300$, the numbers of false drops and actual drops are almost 0. Therefore, $335 - 150 = 185$ (pages) are the page accesses for the useless bit-slice files. Namely, even if such bit-slice files are lookup-ed to lessen the number of drops, it does not pay because the number of drops is already almost 0.

Based on the above observation, the smart retrieval strategy for $T \subseteq Q$ is given as follows:

1. If $D_q \leq D_q^{opt}$, lookup only $Fe^{-\frac{m}{F}D_q^{opt}}$ files out of $F - m_q$ bit-slice files which correspond to the bit positions set to "0" in the query signature. These $Fe^{-\frac{m}{F}D_q^{opt}}$ bit-slice files can be selected arbitrarily. The remaining object access process is similar to that described in Subsection 4.2.2.

2. If $D_q > D_q^{opt}$, retrieve objects as described in Subsection 4.2.2.

Here, D_q^{opt} is the D_q-value that minimizes the total retrieval cost of BSSF and roughly estimated as

$$D_q^{opt} \approx -\frac{F}{m} \ln \left[1 - \left(\frac{F}{(1 + P_u) m D_t P b} \right)^{\frac{1}{m D_t}} \right].$$

The derivation of D_q^{opt} is in Appendix B.

The retrieval costs of BSSF under the smart object retrieval strategy are shown in Figure 4.6 ($D_t = 10$) and Figure 4.7 ($D_t = 100$). As m-values, small values are used for consistency with the analysis in Subsection 4.3.1. For comparison, the retrieval cost of NIX is also plotted. The retrieval cost becomes constant for $D_q \leq D_q^{opt}$. This is the effect of the smart retrieval strategy. Since D_q^{opt} is relatively larger than D_t, the retrieval cost of BSSF is considered to be a constant value for probable values of D_q. On the other hand, the retrieval cost of NIX is large even with the smaller values of D_q. The analysis show that BSSF is the most efficient set access facility for $T \subseteq Q$.

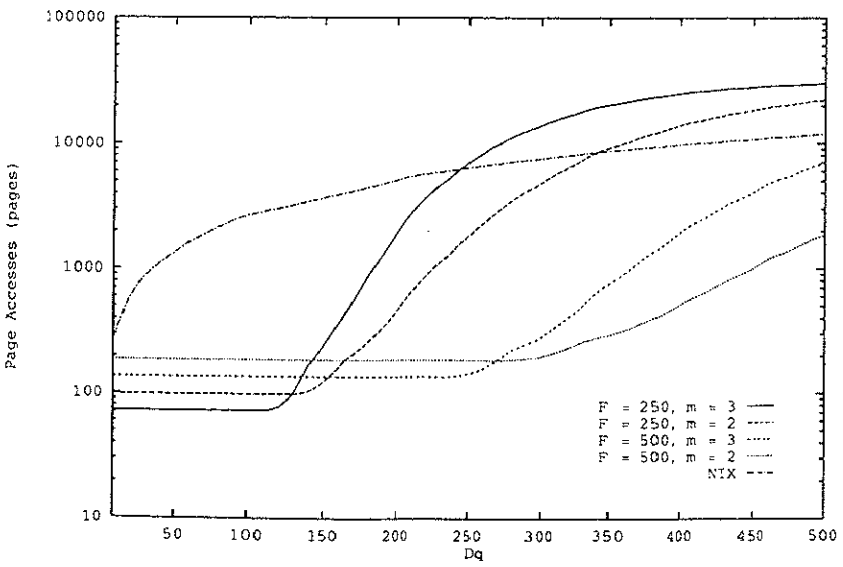

Figure 4.6: Smart Retrieval Cost ($D_t = 10$, $N = 32,000$)

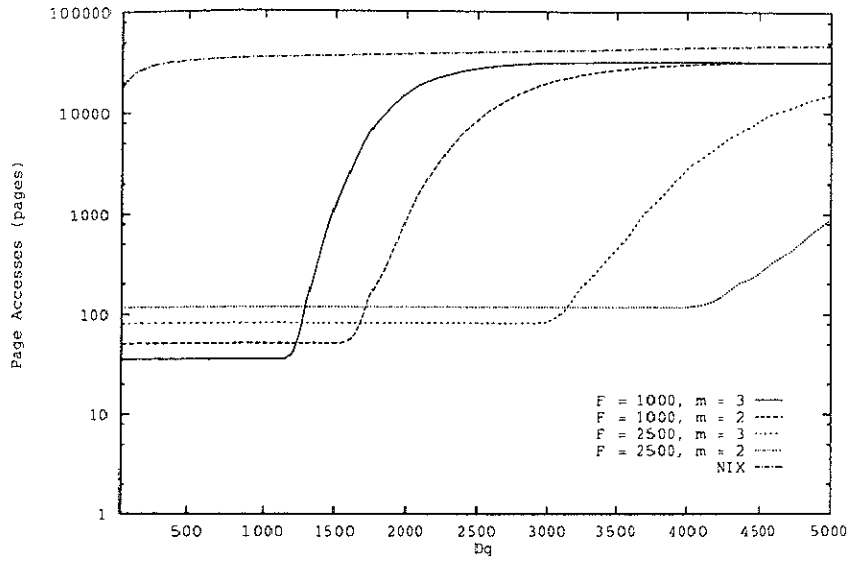

Figure 4.7: Smart Retrieval Cost ($D_t = 100$, $N = 32{,}000$)

4.3.3 Storage Cost

Table 4.4 shows the storage costs of the three set access facilities for several parameter values. The storage costs of SSF, BSSF, and NIX become higher in this order. Although SSF and BSSF are almost comparable, the storage cost of NIX is much higher than that of BSSF. For $D_t = 10$, two F values $F = 250$ and $F = 500$ are used. The storage costs of BSSF are about 45% ($F = 250$) and 80% ($F = 500$) of that of NIX. For $D_t = 100$, two F values $F = 1000$ and $F = 2500$ are used. Their storage costs are about 16% and 38% of that of NIX, respectively.

Table 4.4: Storage Cost

$D_t = 10$

file	SC (pages)
SSF ($F = 250$)	307
SSF ($F = 500$)	551
BSSF ($F = 250$)	313
BSSF ($F = 500$)	563
NIX	654

$D_t = 100$

file	SC (pages)
SSF ($F = 1000$)	1040
SSF ($F = 2500$)	2504
SSF ($F = 5000$)	4946
BSSF ($F = 1000$)	1063
BSSF ($F = 2500$)	2563
NIX	6529

4.3.4 Update Cost

Table 4.5 shows the update costs IC and DC for the three access facilities. The update cost of SSF is quite low in comparison with those of BSSF and NIX and constant with respect to D_t. The update cost of BSSF is the worst because it requires updates for bit-slice files.

4.4 Cost Analysis for Medium-scale Databases

In this section, the costs of BSSF, BSSFcmpr, and NIX for $T \supseteq Q$ and $T \subseteq Q$ in medium-scale databases ($N = 320,000$) are compared. BSSFcmpr is the devised

Table 4.5: Update Cost

D_t	file	IC (pages)	DC (pages)
10	SSF	4	32
	BSSF ($F = 250$, $m = 2$)	40	71
	BSSF ($F = 250$, $m = 3$)	59	89
	BSSF ($F = 500$, $m = 2$)	41	71
	BSSF ($F = 500$, $m = 3$)	60	90
	NIX	24	24
100	SSF	4	32
	BSSF ($F = 1000$, $m = 2$)	365	395
	BSSF ($F = 1000$, $m = 3$)	520	550
	BSSF ($F = 2500$, $m = 2$)	386	417
	BSSF ($F = 2500$, $m = 3$)	567	598
	NIX	229	229

version of BSSF and especially targeted for medium-scale databases. The cost of SSF is not considered here because of the poor performance is revealed in the previous section.

4.4.1 Retrieval Cost for $T \supseteq Q$

Figure 4.8 shows the case of $D_t = 100$ and $N = 320,000$ with the smart retrieval strategy. The retrieval cost of BSSF is obviously inferior to that of NIX. For example, consider BSSF with $F = 2500$ and $m = 2$. For small-scale databases (Figure 4.4),

the retrieval cost resides in same order to NIX. However, in Figure 4.8, the retrieval cost of BSSF is quite larger than that of NIX. This is because the size of a bit-slice file increase in proportion to N so that the retrieval cost of BSSF is also proportional to N. On the other hand, the effect of N to NIX is rather small because the height of NIX increase in the log order. For $D_t = 10$, similar characteristic is observed. In conclusion, the retrieval cost of BSSF for $T \supseteq Q$ is worse than NIX for medium-scale databases.

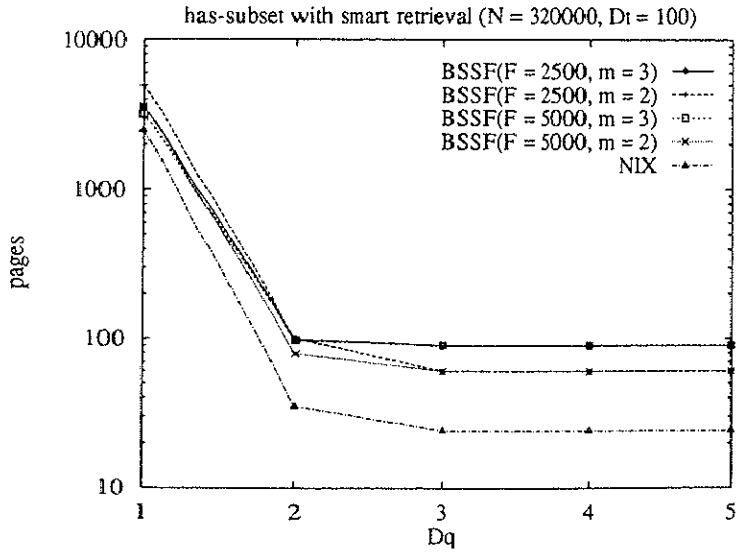

Figure 4.8: Smart Retrieval Cost ($D_t = 100$, $N = 320,000$)

Improvement of Retrieval Cost with Compression

To improve the retrieval cost of BSSF for medium-scale databases, the compressed BSSF (BSSFcmpr) is considered. The length of a compressed bit-slice file is given by $SC_{\mathrm{bsf}}(N) = \left\lceil \frac{Np(b_t)\overline{n}}{Pb} \right\rceil$ (Eq. (4.16)). As examples, several values of $\frac{Np(b_t)\overline{n}}{Pb}$ are derived

in Table 4.6. Under this parameter setting ($N = 320,000$), the size of a bit-slice file without compression is 10 pages. The use of small m-values for the smart retrieval also produces good results in terms of compression.

Table 4.6: Values of $\frac{Np(b_t)\overline{n}}{Pb}$ (pages) ($N = 320,000$, $D_t = 100$)

F	$m = 1$	$m = 2$	$m = 3$	$m = 4$
2500	2.72	4.59	6.11	5.94
5000	1.57	2.72	3.72	4.59
10000	0.885	1.57	2.17	2.72

The retrieval cost of BSSFcmpr is compared with that of NIX in Figure 4.9. It is observed that as the density of "1"'s in a bit-slice file becomes smaller, the improvement of retrieval performance of BSSFcmpr get larger. If BSSF and BSSFcmpr is compared under the same storage cost, BSSFcmpr can set larger F-value so that is advantageous because the false drop probability becomes small. The retrieval cost for $D_t = 10$ is shown in Figure 4.10. For other parameter settings, similar costs are obtained.

4.4.2 Retrieval Cost for $T \subseteq Q$

For BSSFcmpr, D_q^{opt}-values for $N = 320,000$ and $D_t = 100$ are shown in Table 4.7 with the retrieval cost at the point $D_q = D_q^{opt}$. The retrieval cost is proportional to N (Eq. (B.4) in Appendix B), therefore 10 times larger than that of small-scale

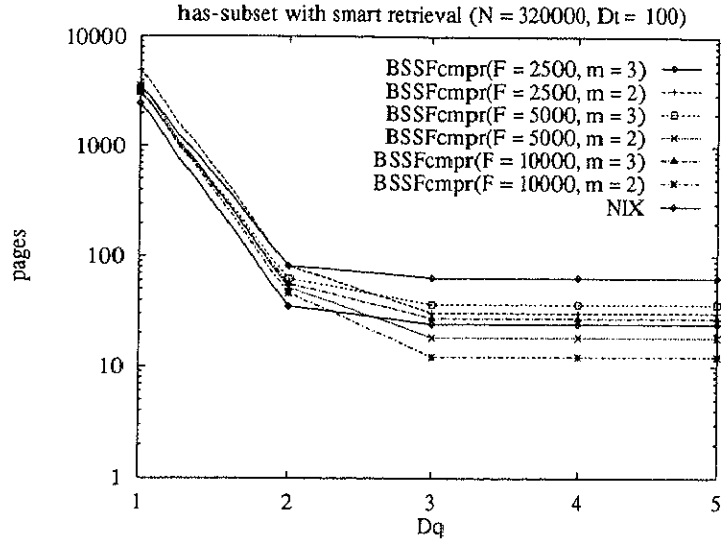

Figure 4.9: Smart Retrieval Cost of Compressed BSSF ($D_t = 100,\ N = 320,000$)

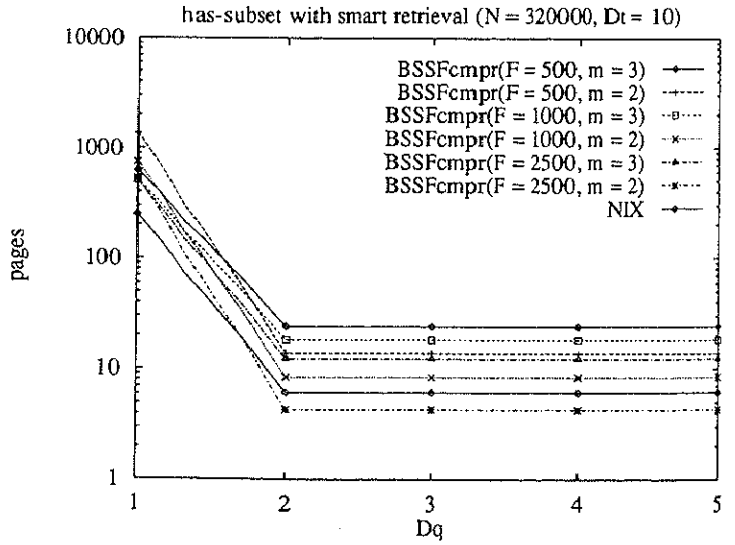

Figure 4.10: Smart Retrieval Cost of Compressed BSSF ($D_t = 10,\ N = 320,000$)

databases. On the contrary, the cost of NIX is very poor. For example, the retrieval cost of NIX with $D_t = 100$ is about 35,000 pages at the point of $D_q = 100$ and still increases as D_q increases. For $D_t = 10$, similar tendency holds.

Table 4.7: Retrieval Cost for Medium-scale Database (pages)

($T \subseteq Q$, $N = 320,000$, $D_t = 100$)

F and m	D_q^{opt}	retrieval cost (pages)
$F = 2500$, $m = 2$	3960	1170
$F = 2500$, $m = 3$	2940	818
$F = 5000$, $m = 2$	8130	2173
$F = 5000$, $m = 3$	6010	1523

Note that large F-values are not necessarily contribute to the improvement of the retrieval performance; This is differ from $T \supseteq Q$. For $T \subseteq Q$, bit-slice files corresponding to "0"'s in the query signature are retrieved, but the number of "0"'s increase as F increases so that the number of bit-slice files to be retrieved also increases.

Improvement of Retrieval Cost with Compression

When compression is used, the approximate formula for D_q^{opt} (Eq. (B.3)) cannot be applied. However, such optimal D_q values can be found if we actually set the parameters and look for the minimal point. Table 4.7 indicates such D_q-values (denoted as $D_q^{opt'}$) when $D_t = 10$, and the retrieval costs at those points are shown.

Table 4.8: Retrieval Cost of Compressed BSSF

($T \subseteq Q$, $N = 320,000$, $D_t = 100$)

F and m	$D_q^{opt'}$	retrieval cost (pages)
$F = 2500$, $m = 2$	3870	626
$F = 2500$, $m = 3$	2910	543
$F = 5000$, $m = 2$	7780	739
$F = 5000$, $m = 3$	5830	669
$F = 10000$, $m = 2$	11820	1926
$F = 10000$, $m = 3$	11660	983

By Table 4.7 and Table 4.8, it is concluded that BSSFcmpr is superior to BSSF for same parameter values F and m. The improvement of the retrieval cost with compression is also observed for $D_t = 10$.

4.4.3 Storage Cost

Table 4.9 indicates storage costs of BSSF and NIX. The storage costs of BSSF is about half or equal to that of NIX, but BSSF is poor in terms of retrieval costs for $T \supseteq Q$.

Table 4.10 shows the storage costs for BSSFcmpr. In comparison with Table 4.9,

Table 4.9: Storage Cost of BSSF and NIX

D_t	file	SC (pages)
10	BSSF ($F = 250$)	3125
	BSSF ($F = 500$)	5625
	NIX	6529
100	BSSF ($F = 2500$)	25625
	BSSF ($F = 5000$)	50625
	NIX	65280

the superiority of BSSFcmpr is obvious.

Table 4.10: Storage Cost of Compressed BSSF (pages) ($N = 320,000$)

D_t	F	$m = 2$ (pages)	$m = 3$ (pages)
10	500	1987	2483
	1000	2194	2799
	2500	2461	3204
100	2500	12099	15899
	5000	14242	19200
	10000	16311	22362

4.4.4 Update Cost

For $N = 320,000$ and $D_t = 100$, the update costs of BSSF and NIX for $D_t = 100$ are shown in Table 4.11. For insertion cost, BSSF is superior to that of NIX. For

deletion costs, NIX is superior to BSSF when $D_t = 10$. For $D_t = 100$, BSSF is better than NIX. The update cost of BSSFcmpr is a future work.

Table 4.11: Update Cost ($N = 320,000$)

D_t	file	IC (pages)	DC (pages)
10	BSSF ($F = 250$, $m = 2$)	40	352
	BSSF ($F = 250$, $m = 3$)	59	370
	BSSF ($F = 500$, $m = 2$)	41	353
	BSSF ($F = 500$, $m = 3$)	60	372
	NIX	229	229
100	BSSF ($F = 2500$, $m = 2$)	386	698
	BSSF ($F = 2500$, $m = 3$)	567	879
	BSSF ($F = 5000$, $m = 2$)	394	706
	BSSF ($F = 5000$, $m = 3$)	584	896
	NIX	307	307

4.5 Discussion

First, the case of small-scale databases is discussed. By the analyses of retrieval costs, it is shown that SSF is inferior to BSSF for both $T \supseteq Q$ and $T \subseteq Q$. SSF requires a full scan over the signature file, whereas only a part of the bit-slice files are accessed in BSSF. Moreover, there exist the smart retrieval strategy that improves the retrieval cost of BSSF. Therefore, as far as the retrieval cost is concerned, BSSF is more appropriate as a set-based signature file.

For $T \supseteq Q$, the retrieval cost of BSSF with a small m-value is almost equal to that of NIX except for $D_q = 1$. Under the smart retrieval strategies, their retrieval costs are constants for most of the D_q values, and the constant cost values are almost the same. However, for very small D_q values, the retrieval cost of BSSF becomes worse due to the false drops. In particular, for $D_q = 1$, NIX is more efficient than BSSF in all cases investigated. For $T \subseteq Q$, BSSF has small constant page accesses for probable values of D_q, and overwhelms NIX.

For storage costs, two signature files are at most equal to or smaller than that of NIX. For update costs, SSF has very small cost but BSSF is rather lager and the worst of the three indexes. To improve the deletion cost of BSSF, another algorithm that is similar to that of SSF may be used; When deleting an entry, BSSF simply set a delete flag at the deleting position and does not reuse the bit position. In this scheme, the deletion cost will become smaller, but the scheme is not suited to dynamic environments. The weak point that the deletion algorithm is not suited to dynamic environments also holds for SSF.

From these analyses, it is concluded that for small-scale databases, BSSF with a small m-value is rather stable set access facility for both $T \supseteq Q$ and $T \subseteq Q$. For example, in BSSF with $m = 2$ and $F = 250$ for $D_t = 10$, the storage cost of BSSF is half of that of NIX, and is almost same as that of SSF. For the retrieval cost for $T \supseteq Q$, BSSF is comparable to NIX except for $D_q = 1$ and is much superior than SSF. The retrieval cost of BSSF for $T \subseteq Q$ is much lower than those of SSF and NIX. The only problem with BSSF is that update costs are rather higher than those

of SSF and NIX. If $T \supseteq Q$ with small D_q-value (e.g., $T \ni q$) frequently occurs, two configurations are considered: 1) use an NIX file. In this case, $T \subseteq Q$ queries cannot be support very well. 2) use an NIX file for $T \supseteq Q$ queries with small D_q-values and use a BSSF file for other queries. Since NIX covers $T \supseteq Q$ with small D_q-values, the signature size F of BSSF can be set to smaller value and can achieve a compact organization.

In the context of text retrieval, m_{opt} (Eq. (3.4)), is usually used as the value of m-value. However, the analysis has clarified that we had better to set a far smaller value to m for efficient set retrieval.

Next, the case of medium-scale databases is discussed. For medium-scale databases, the costs of BSSF, BSSFcmpr, and NIX for $T \supseteq Q$ and $T \subseteq Q$ are compared. For $T \supseteq Q$, BSSF is inferior to NIX, but BSSFcmpr has almost similar retrieval costs except for $D_q = 1$. For $T \subseteq Q$, the retrieval cost of NIX is quite worse. In this case, BSSFcmpr also improved the cost of BSSF. For storage cost, compression achieves great reduction.

In conclusion, for these set access facilities, BSSF suited for small scale databases and BSSFcmpr suited be used for medium-scale databases. They attained almost equal and small cost for $T \supseteq Q$ (except for $D_q = 1$) and $T \subseteq Q$.

The estimated size of BSSFcmpr assumed an optimal encoding. Therefore, the

compression rate might become worse in the real situation. Efficient update algorithms for BSSFcmpr are remained as a research issue.

Chapter 5

Set Retrieval of Nested Objects

5.1 Introduction

Nested objects frequently appear in databases for advanced application areas. Many advanced database systems support some kind of construct to express and manipulate set values. Therefore, efficient indexing methods for set retrieval are also

required for nested objects. In order to support efficient retrieval of nested objects, several indexing methods such as the nested index, the path index, and the multi-index have been proposed [BK89, Ber93]. However, they are not designed to support set retrieval of nested objects.

In this chapter, the target is changed to multilevel nested objects with set-valued attributes and efficient set retrieval facilities are investigated. Four candidate set retrieval facilities are proposed by combining the signature file method and the nested index, and they are compared in terms of retrieval cost, storage cost, and update cost. Since it is difficult to construct a cost model for a general case, the following two independent cases are considered:

Case I : Nested objects do not have set attributes except for leaf-level attributes.

Case II : Nested objects may have set attributes in their nonleaf-level attributes, but two nested objects do not share their references.

So far, most performance analyses of indexes for nested objects have been performed assuming that nested objects do not have set attributes [BK89, Ber93]. Therefore, some of the results also contribute to analysis of indexes for nested objects in general as well as their set retrieval.

5.2 Preliminaries

In this section, the notion of nested objects is informally defined as the basis of the following discussion. Then a sample query is shown.

An *object* comprises tuple-structured data defined by the *tuple constructor* ([...]) and has one or more *attributes*. Each object is identified by its *object identifier* (*OID*). The structure of objects in a class is specified by the *class definition*. A set of class definitions is called a *schema*. There are two types of attributes: an *atomic attribute* takes a primitive value or an OID as its value, and a *set attribute* takes a set of primitive values or an OID set of objects in some class as its value. In a schema, a set attribute is specified by the *set constructor* ({...}). An example schema is shown in Figure 5.1.

{**Deptartment** = [dname: *string*, projects: {**Project**}, ...],
Project = [pname: *string*, *emps*: {**Employee**}, *leader*: **Employee**, ...],
Employee = [ename: *string*, *hobbies*: {*string*}, ...]}

Figure 5.1: An Example Schema

If an object O has an OID of some object O' as a primitive attribute value, or has an OID of some object O' in its set attribute value, we say that the object O *references* object O'. Next, assume that classes $C_1, C_2, \ldots, C_n$ are defined in a schema. A *path* P is defined as $P = C_1.A_1.A_2.\cdots.A_n$, where A_i $(1 \leq i \leq n-1)$ is an attribute of the class C_i and takes an OID of a C_{i+1} object or an OID set of

C_{i+1} objects as its value. A_n is an attribute of C_n and can take a primitive value, a set of primitive values, an OID, or an OID set as its value. An *instance* of the path P has the form $O_1.O_2.\cdots.O_n.X$, where O_1 is an OID of a C_1 object (this object is called the *root object*). If the attribute A_i $(1 \leq i \leq n-1)$ is a primitive attribute, O_{i+1} is an OID of a C_{i+1} object which appears as the A_i value of O_i. If A_i is a set attribute, the A_i value of O_i contains OID O_{i+1} as a set element. X is the A_n value of the object O_n.

In the derivation of retrieval costs, the following query form over the path $C_1.A_1.A_2.\cdots.A_n$ is assumed:

select ⟨attribute value(s) of C_1⟩
from C_1
where $A_1.A_2.\cdots.A_n$ ⟨op⟩ ⟨set value⟩,

where $A_1, A_2, \ldots, A_{n-1}$ are primitive or set attributes and A_n is a set attribute. The comparison operator ⟨op⟩ can be $\supseteq$ or $\subseteq$. An example of such a query is the following query $\mathbf{Q}_1$ based on the schema in Figure 5.1.

$\mathbf{Q}_1$: **select** dname
from Department
where projects.emps.hobbies $\supseteq$ {"baseball", "skiing"}

Q_1 retrieves department names such that hobbies of some employees in the department's projects include both "baseball"and "skiing". This kind of query is $T \supseteq Q$ (*has-subset*).

5.3 Set Access Facilities for Nested Objects

In this section, four set access facilities for nested objects, $\mathcal{I}_{\text{BSSF}}$, $\mathcal{I}_{\text{NIX}}$, $\mathcal{I}_{\text{BSSF-NIX}}$, and $\mathcal{I}_{\text{NIX-NIX}}$, are introduced.

5.3.1 File Structures of Set Access Facilities

File structures of four set access facilities are described here. $\mathcal{I}_{\text{BSSF}}$ is an extension of BSSF to facilitate set accesses to nested objects. $\mathcal{I}_{\text{NIX}}$ is based on the *nested index* (NIX) [BK89, Ber93], a B^+-tree-like indexing method proposed for nested objects. $\mathcal{I}_{\text{BSSF-NIX}}$ is a combination of the BSSF method and the NIX method, and $\mathcal{I}_{\text{NIX-NIX}}$ uses two NIX files.

An instance of a path $P = C_1.A_1.A_2.\cdots.A_n$ is expressed in the form

$O_1.O_2.\ldots.O_n.\{v_1, v_2, \ldots, v_m\}$, where $\{v_1, v_2, \ldots, v_m\}$ is a set of primitive values or an OID set. When an instance of P, $O_1.O_2.\cdots.O_n.\{v_1, v_2, \ldots, v_m\}$ is given, the following entries are inserted in each facility.

$\mathcal{I}_{\text{BSSF}}$: The set signature S created from the set $\{v_1, v_2, \ldots, v_m\}$ is paired with O_1, the OID of the root object, and the pair $\langle S, O_1 \rangle$ is stored in the BSSF file. If x instances of the path P are inserted, BSSF will have x entries.

$\mathcal{I}_{\text{NIX}}$: For each element of the set $\{v_1, v_2, \ldots, v_m\}$, the pair $\langle v_i, O_1 \rangle$ $(1 \leq i \leq m)$ is created and inserted into the NIX file. Since the format of a leaf-node entry of NIX is ⟨key value, OID set⟩, if the pair $\langle v_i, O_1 \rangle$ is inserted into NIX, the corresponding leaf-node entry becomes $\langle v_i, \{O_1, \ldots\} \rangle$.

$\mathcal{I}_{\text{BSSF-NIX}}$: The set signature S created from the set $\{v_1, v_2, \ldots, v_m\}$ is paired with O_n, the OID of the C_n object in the path P, and the pair $\langle S, O_n \rangle$ is inserted into the BSSF file. Next, the pair $\langle O_n, O_1 \rangle$ is inserted into the NIX file.

$\mathcal{I}_{\text{NIX-NIX}}$: For each element of the set $\{v_1, v_2, \ldots, v_m\}$, the pair $\langle v_i, O_n \rangle$ $(1 \leq i \leq m)$ is created and inserted into an NIX file. This NIX file is called NIX_1. Next, the pair $\langle O_n, O_1 \rangle$ is inserted into another NIX file. This NIX file is called NIX_2.

5.3.2 Query Processing Algorithms

In this subsection, query processing algorithms for the four set access facilities are described. Except for $\mathcal{I}_{\text{BSSF}}$, the query algorithms of each set access facility are not different for Case I and Case II. For Case II, it is assumed that two objects do not share their references. However, the algorithms described here are general and can cope with the case that two or more objects share their references.

$\mathcal{I}_{\text{BSSF}}$ Query algorithms for $\mathcal{I}_{\text{BSSF}}$ are slightly different for Case I and Case II.

For Case I, both $T \supseteq Q$ and $T \subseteq Q$ are processed as follows:

1. BSSF is searched based on the query condition and an OID set of C_1 objects is obtained. This set is called S_{OID}.
2. For each object in S_{OID}, a *forward traversal* [BK89, Ber93] is performed. Thus C_n objects are retrieved.
3. Each C_n object is examined as to whether it actually satisfies the query condition. If it satisfies the condition, the corresponding C_1 object is returned.

For Case II, both $T \supseteq Q$ and $T \subseteq Q$ are processed as follows.

1. BSSF is searched based on the query condition and an OID set of C_1 objects is obtained [1]. This set is called S_{OID}.

2. For each object in S_{OID}, a *forward traversal* [BK89, Ber93] is performed. When the forward traversal from $O_1 \in S_{\text{OID}}$ is performed, reachable C_n objects are retrieved. If at least one of the C_n objects satisfies the query condition ($T \supseteq Q$, $T \subseteq Q$), O_1 is included in the final result of this query and returned to the user.

$\mathcal{I}_{\text{NIX}}$

1. For each element in the query set, NIX is searched. Thus D_q OID sets of C_1 objects are obtained, where D_q is the cardinality of the query set.

2. For $T \supseteq Q$, the intersection of the D_q sets is taken. For $T \subseteq Q$, the union is taken.

3. For $T \supseteq Q$, C_1 objects are retrieved based on the OID set and returned.

3'. For $T \subseteq Q$, the following process is performed.

   (a) Forward traversals are performed for the OID set, and the corresponding C_n objects are checked as to whether they actually satisfy the query condition.

[1] When some objects share their references, a multiset of OIDs are generally obtained. In such a case, duplicates are immediately eliminated from the multiset.

(b) The root C_1 objects of the C_n objects which satisfy condition (a) are returned.

$\mathcal{I}_{\text{BSSF-NIX}}$ Both $T \supseteq Q$ and $T \subseteq Q$ are processed as follows.

1. BSSF is searched based on the query condition and an OID set of C_n objects is obtained [2].
2. Each C_n object in the OID set is retrieved and checked as to whether it actually satisfies the query condition.
3. For each C_n object that satisfies the condition, NIX is searched using its OID as a key value. As a result, an OID set of C_1 objects is obtained.
4. C_1 objects are retrieved based on the OID set and returned.

$\mathcal{I}_{\text{NIX-NIX}}$

1. For each element in the query set, NIX_1 is searched. Thus D_q OID sets of C_n objects are obtained.

[2] Actually, a duplicate elimination is also required for Case II. The reason is as follows. Let us assume that a C_1 object O_1 references two C_n objects O_n and O'_n and that OIDs O_n and O'_n are obtained in step 2. In this case, O_1 is retrieved two times so that two O_1's are obtained.

2. For $T \supseteq Q$, the intersection of the D_q sets is taken. For $T \subseteq Q$, the union is taken.

3. Only for $T \subseteq Q$, C_n objects are retrieved based on the OID set and checked as to whether they actually satisfy the query condition. Objects that do not satisfy the condition are removed from the OID set.

4. For each element of the OID set, NIX_2 is searched. Then an OID set of C_1 objects is obtained [3].

5. C_1 objects are retrieved based on the OID set.

5.3.3 Update Algorithms

Algorithms for inserting a new path instance $P = C_1.A_1.A_2.\cdots.A_n$ have already been described in Subsection 5.3.1. Therefore, only deletion algorithms are shown here for the case that a C_n object O_n is deleted from the database and that *backward references* are not supported. Assume that O_n is already retrieved into the memory and the A_n value of O_n is $\{v_1, v_2, \ldots, v_m\}$.

$\mathcal{I}_{\text{BSSF}}$

[3] In Case II, there may exist duplicates in the set. If duplicates are exist, they are eliminated here.

1. A query signature is generated from $\{v_1, v_2, \ldots, v_m\}$ and BSSF is searched based on the set equality condition ($T \equiv Q$). Namely, the target signatures which are equal to the query signature are searched. Thus an OID set of candidate C_1 objects is obtained.

2. For each object in the OID set, a forward traversal is performed. As a result, an OID set of C_1 objects that actually reference O_n are given. This set is called S_{OID}.

3. For each $O_1 \in S_{\text{OID}}$, the corresponding entry is deleted from the BSSF file [4].

$\mathcal{I}_{\text{NIX}}$ The update algorithms of $\mathcal{I}_{\text{NIX}}$ for Case I and Case II are different. The algorithm for Case I is easier and its cost will be small.

For Case I, the algorithm is as follows:

1. For each element v_i $(1 \leq i \leq m)$, NIX is searched. Thus m OID sets of C_1 objects are obtained. Next, the union is taken. This set is called S_{OID}.

2. For each object in S_{OID}, a forward traversal is performed and checked as to whether O_n is referenced. As a result, an OID set of C_1 objects which actually reference O_n is obtained. Let this set be S'_{OID}.

[4] When S_{OID} is obtained in step 1, the position of the BSSF entry for each $O_1 \in S_{\text{OID}}$ is temporarily memorized. Thus, the deletion process in step 3 will become more efficient.

3. For each object $O_1 \in S'_{\text{OID}}$, the corresponding NIX entry $\langle v_i, O_1 \rangle$ is deleted.

In Case II, since $A_1, \ldots, A_{n-1}$ may be set attributes, the deletion algorithm for NIX becomes complicated. A C_1 object O_1 may have multiple path instances to leaf-level objects. Therefore, O_1 has multiple set values corresponding to the multiple path instances, and they are not necessarily disjoint. Since NIX does not have counting information in its entries, we cannot delete $\langle v_i, O_1 \rangle$ from the NIX leaf-node entry immediately. It might be possible to settle this problem by modifying the file structure of NIX, but here an algorithm based on the normal NIX file structure is presented.

1. For each element v_i $(1 \leq i \leq m)$, NIX is searched. Thus m OID sets of C_1 objects are obtained. Next, the union is taken. This set is called S_{OID}.

2. For each object in S_{OID}, a forward traversal is performed and checked as to whether O_n is referenced. In this forward traversal, *all* reachable C_n objects are retrieved. As a result, an OID set of C_1 objects which actually reference O_n is obtained. Let this set be S'_{OID}.

3. For each object $O_1 \in S'_{\text{OID}}$, the referenced C_n objects are examined. If v_i $(1 \leq i \leq m)$ is not contained in the A_n value of any C_n objects other than O_n, the NIX entry $\langle v_i, O_1 \rangle$ is deleted.

$\mathcal{I}_{\text{BSSF-NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$ Algorithms for $\mathcal{I}_{\text{BSSF-NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$ are straightforward. For $\mathcal{I}_{\text{BSSF-NIX}}$, the entry of the BSSF file that corresponds to O_n is deleted and the entries in the NIX file that reference O_n are deleted. Deletions on $\mathcal{I}_{\text{NIX-NIX}}$ are processed in a similar manner.

5.4 Cost Models

In this section, the cost models for the four set access facilities are developed. The retrieval costs, storage costs, and update costs of the four set access facilities are derived for two cases (Case I and Case II). Only the number of page accesses will be taken into account as a cost factor. To simplify the estimation, it is assumed that all target sets have an equal cardinality D_t. In the following derivation, symbols used in Chapter 4 (Table 4.1) are also used. Additional symbols are shown in Table 5.1. In Chapter 4, the access cost to an object is represented by P_s (successful case) and P_u (unsuccessful case). However, they are unified to P_o to simplify the derivation. The V-value is changed from 13,000 to 10,000. To estimate costs, formulas for BSSF and NIX in Chapter 4 are used.

Table 5.1: Symbols and Their Values

Symbol	Definition and Value
PI	total number of the path instances
N_i	number of C_i objects
V	cardinality of the set domain of the attribute A_n $(= 10,000)$
$FT(x)$	forward traversal cost (x is the number of the root objects)
P_o	number of page accesses to fetch an object $(= 1)$

5.4.1 Configuration of Nested Objects

First, the configuration of nested objects for Case I (where $A_1, \ldots, A_{n-1}$ are primitive attributes) is considered. To derive retrieval costs, queries over the path $P = C_1.A_1.A_2.\cdots.A_n$ are considered and an assumption is made:

- The value of A_i $(1 \leq i \leq n-1)$ is not NULL.

Thus, the number of instances of the path P becomes

$$PI = N_1 \quad (= N_2 = \cdots = N_n). \tag{5.1}$$

Next, forward traversal cost is considered. If x C_1 objects are given and forward traversals are performed to retrieve C_n objects, the cost is simply given as

$$FT(x) = x \times n. \tag{5.2}$$

For Case II is considered, to simplify the derivation and analysis of costs, the following assumptions are made:

1. Each C_i object references fan_i^{i+1} C_{i+1} objects $(1 \leq i \leq n-1)$.
2. No two objects share their references.

fan_i^{i+1} is called the *fanout* from C_i to C_{i+1}. For classes C_i and C_j $(i < j)$, the fanout is defined as

$$fan_i^j = \prod_{k=i}^{j-1} fan_k^{k+1}. \tag{5.3}$$

Therefore,

$$PI = N_1 fan_1^n. \tag{5.4}$$

5.4.2 Retrieval Costs

IBSSF First, the retrieval cost for Case I is shown:

$$RC\{\mathcal{I}_{\text{BSSF}}, c\} = SC_{\text{bsf}}(PI) \times M_{\{c\}} + LC_{\text{OID}\{c\}}(PI) \quad (5.5)$$

$$+ FT\left(A_{\{c\}}(N_n) + Fd_{\{c\}}(N_n - A_{\{c\}}(N_n))\right). \quad (5.6)$$

Note that PI, the number of path instances, is used instead of N_n because the BSSF file stores PI entries. But note that $PI = N_n$ holds under the configuration of Case I.

Next, the retrieval cost for Case II is considered. The main part of the retrieval cost of $\mathcal{I}_{\text{BSSF}}$ is given as

$$SC_{\text{bsf}}(PI) \times M_{\{c\}} + LC_{\text{OID}\{c\}}(PI). \quad (5.7)$$

In the retrieval of $\mathcal{I}_{\text{BSSF}}$, OIDs of C_1 objects are obtained rather than of C_n objects. The number of OIDs of C_1 objects after duplicate elimination is estimated by Yao's formula:

$$npa\left(A_{\{c\}}(N_n) + Fd_{\{c\}}(N_n - A_{\{c\}}(N_n)),\ N_n,\ \frac{N_n}{fan_1^n}\right). \quad (5.8)$$

Then, forward traversals are performed for these C_1 objects. Therefore, the total retrieval cost is

$$RC\{\mathcal{I}_{\text{BSSF}}, c\} = [eq.(5.5)] + PFT([eq.(5.8)],\ A_{\{c\}}(fan_1^n)), \quad (5.9)$$

where PFT is the forward traversal cost and given in the Appendix C.3 (Eq. (C.4)).

INIX The retrieval costs of $\mathcal{I}_{\text{NIX}}$ depend on the query. The retrieval cost for $T \supseteq Q$ is common to Case I and II:

$$RC\{\mathcal{I}_{\text{NIX}},\, T \supseteq Q\} = RC_{\text{NIX}}\left(V,\, \frac{D_t N_n}{V},\, D_q\right) + P_o A_{\{T \supseteq Q\}}(N_n). \tag{5.10}$$

The retrieval costs for $T \subseteq Q$ are different. For Case I,

$$RC\{\mathcal{I}_{\text{NIX}},\, T \subseteq Q\} = RC_{\text{NIX}}\left(V,\, \frac{D_t N_n}{V},\, D_q\right) + FT\left(N_n\left(1 - \frac{\binom{V-D_q}{D_t}}{\binom{V}{D_t}}\right)\right). \tag{5.11}$$

For Case II, the number of OIDs of C_1 objects obtained by the retrieval of NIX is given by

$$npa\left(N_n\left(1 - \frac{\binom{V-D_q}{D_t}}{\binom{V}{D_t}}\right),\, N_n,\, \frac{N_n}{fan_1^n}\right). \tag{5.12}$$

Therefore, the retrieval cost is

$$\begin{aligned} RC\{\mathcal{I}_{\text{NIX}},\, T \subseteq Q\} \;=\; & RC_{\text{NIX}}\left(V,\, \frac{D_t N_n}{V},\, D_q\right) \\ & + PFT([eq.(5.12)],\, A_{\{T \subseteq Q\}}(fan_1^n)). \end{aligned} \tag{5.13}$$

IBSSF-NIX The retrieval cost is common to two cases.

$$RC\{\mathcal{I}_{\text{BSSF-NIX}},\, c\} = RC_{\text{BSSF}\{c\}}(N_n) + RC_{\text{NIX}}(N_n,\, 1,\, A_{\{c\}}(N_n)) + P_o A_{\{c\}}(N_n) \quad (5.14)$$

INIX-NIX The retrieval costs of $\mathcal{I}_{\text{NIX-NIX}}$ also depend on the query but are common to Case I and II:

$$\begin{aligned} RC\{\mathcal{I}_{\text{NIX-NIX}},\, T \supseteq Q\} &= RC_{\text{NIX}}\left(V,\, \frac{D_t N_n}{V},\, D_q\right) + RC_{\text{NIX}}(N_n,\, 1,\, A_{\{T\supseteq Q\}}(N_n)) \\ &\quad + P_o A_{\{T\supseteq Q\}}(N_n) \qquad (5.15) \\ RC\{\mathcal{I}_{\text{NIX-NIX}},\, T \subseteq Q\} &= RC_{\text{NIX}}\left(V,\, \frac{D_t N_n}{V},\, D_q\right) + P_o N_n \left(1 - \frac{\binom{V-D_q}{D_t}}{\binom{V}{D_t}}\right) \\ &\quad + RC_{\text{NIX}}\left(N_n,\, 1,\, A_{\{T\subseteq Q\}}(N_n)\right) + P_o A_{\{T\subseteq Q\}}(N_n). \qquad (5.16) \end{aligned}$$

5.4.3 Storage Costs

Storage costs for the four set access facilities are shown below:

$$\begin{aligned} SC\{\mathcal{I}_{\text{BSSF}}\} &= SC_{\text{BSSF}}(PI) \qquad (5.17) \\ SC\{\mathcal{I}_{\text{NIX}}\} &= SC_{\text{NIX}}\left(V,\, \frac{D_t N_n}{V}\right) \qquad (5.18) \\ SC\{\mathcal{I}_{\text{BSSF-NIX}}\} &= SC_{\text{BSSF}}(N_n) + SC_{\text{NIX}}(N_n,\, 1) \qquad (5.19) \end{aligned}$$

$$SC\{\mathcal{I}_{\text{NIX-NIX}}\} = SC_{\text{NIX}}\left(V, \frac{D_t N_n}{V}\right) + SC_{\text{NIX}}(N_n, 1). \quad (5.20)$$

5.4.4 Update Costs

Except for the deletion costs of $\mathcal{I}_{\text{NIX}}$, the update costs are common to Case I and II.

The costs to insert a path instance $P = C_1.A_1.A_2.\cdots.A_n$ are given by

$$IC\{\mathcal{I}_{\text{BSSF}}\} = IC_{\text{BSSF}} \quad (5.21)$$

$$IC\{\mathcal{I}_{\text{NIX}}\} = IC_{\text{NIX}}\left(V, \frac{D_t N_n}{V}, D_t\right) \quad (5.22)$$

$$IC\{\mathcal{I}_{\text{BSSF-NIX}}\} = IC_{\text{BSSF}} + IC_{\text{NIX}}(N_n, 1, 1) \quad (5.23)$$

$$IC\{\mathcal{I}_{\text{NIX-NIX}}\} = IC_{\text{NIX}}\left(V, \frac{D_t N_n}{V}, D_t\right) + IC_{\text{NIX}}(N_n, 1, 1). \quad (5.24)$$

For $\mathcal{I}_{\text{BSSF}}$, the cost for deleting an object O_n is given in the Appendix D (Eq. (D.1)).

$$DC\{\mathcal{I}_{\text{BSSF}}\} \approx SC_{\text{bsf}}(PI) \times nbs + npa(2, N_{\text{oid}} SC_{\text{OID}}(PI), SC_{\text{OID}}(PI)) + 2P_o n + 2, \quad (5.25)$$

where nbs is the weight of a query signature that satisfies $Fd_{\{T \supseteq Q\}}(PI - A_{\{T \supseteq Q\}}(PI)) \approx 1$. The second term is the lookup cost of the OID file. The third and fourth terms represent the forward traversal cost and the rewrite cost, respectively.

The deletion cost for $\mathcal{I}_{\text{NIX}}$ for Case I is estimated as

$$DC\{\mathcal{I}_{\text{NIX}}\} = RC_{\text{NIX}}\left(V, \frac{D_t N_n}{V}, D_t\right) + FT\left(N_n\left(1 - \frac{\binom{V-D_t}{D_t}}{\binom{V}{D_t}}\right)\right) + 2P_o D_t. \quad (5.26)$$

The third term in Eq. (5.25) represents the read and rewrite costs for the NIX entries.

The deletion cost for Case II is given as

$$DC\{\mathcal{I}_{\text{NIX}}\} = RC_{\text{NIX}}\left(V, \frac{D_t N_n}{V}, D_t\right) + [eq.(5.28)] + 2P_o D_t\left(1 - \frac{\binom{V-1}{D_t-1}}{\binom{V}{D_t}}\right)^{fan_1^n - 1}, \quad (5.27)$$

where Eq. (5.28) is the forward traversal cost and given by

$$FFT\left(npa\left(\left(1 - \frac{\binom{V-D_t}{D_t}}{\binom{V}{D_t}}\right) \times (N_n - 1) + 1,\ N_n,\ \frac{N_n}{fan_1^n}\right)\right) \quad (5.28)$$

(see Appendix E). In this case, all referenced C_n objects must be traversed. Therefore the full forward traversal cost FFT (Eq. (C.1) in Appendix C.1) is used. $\left(1 - \frac{\binom{V-1}{D_t-1}}{\binom{V}{D_t}}\right)^{fan_1^n - 1}$ represents an element of the set attribute value of O_n is not contained in other target sets, namely, the entry corresponding to the element can be deleted when the condition is satisfied.

For $\mathcal{I}_{\text{BSSF-NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$, the deletion costs are derived as

$$DC\{\mathcal{I}_{\text{BSSF-NIX}}\} = DC_{\text{BSSF}} + DC_{\text{NIX}}(N_n,\ 1,\ 1) \tag{5.29}$$

$$DC\{\mathcal{I}_{\text{NIX-NIX}}\} = DC_{\text{NIX}}\left(V,\ \frac{D_t N_n}{V},\ D_t\right) + DC_{\text{NIX}}(N_n,\ 1,\ 1). \tag{5.30}$$

5.5 Cost Analysis for Case I

First, the parameter settings are described. N_i, the number of objects in the class C_i, is set to $N_1 = N_2 = \cdots = N_n = 30,000$. As the cardinality of target sets D_t, two cases $D_t = 10$ and $D_t = 100$ are used, and for the length of the path, three cases $n = 2,\ 3,\ 4$ are compared. For $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$, it is necessary to set the BSSF parameters. Here the following policy is employed:

1. The storage costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are at most equal to that of $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. This policy restricts the signature size F. When $D_t = 10$, $F = 500$ (bits) is used, and when $D_t = 100$, $F = 5000$ (bits) is used.

2. The parameter m is set to $m = 2$ based on the analysis for non-nested objects in Chapter 4.

5.5.1 Retrieval Costs

The representative retrieval costs for $T \supseteq Q$ are shown in Figure 5.2 ($D_t = 10$) and Figure 5.3 ($D_t = 100$). Two figures have a similar tendency. For $D_q = 1$, $\mathcal{I}_{\text{BSSF}}$ is the worst cost. This is because $\mathcal{I}_{\text{BSSF}}$ requires forward traversals to process the query. When $D_q = 1$, actual drops and false drops exist so that the overhead of the forward traverse cost affects the overall cost. However, if $D_q \geq 2$, drops are almost negligible, so the retrieval costs increase almost linearly. The change of the path length n does not affect the retrieval costs very well because the numbers of drops are small except for $D_q = 1$.

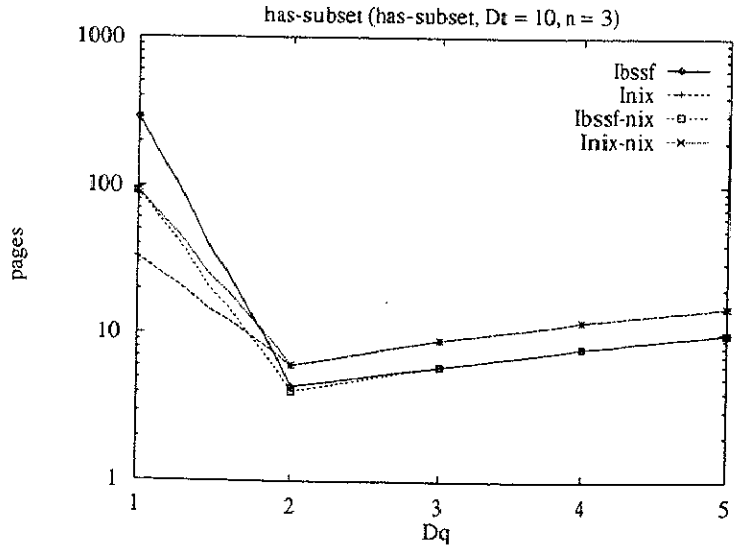

Figure 5.2: Retrieval Cost ($T \supseteq Q$, $D_t = 10$, $n = 3$)

The representative retrieval costs for $T \subseteq Q$ are shown in Figure 5.4 ($D_t = 10$) and Figure 5.5 ($D_t = 100$). $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ have almost the same costs and so do

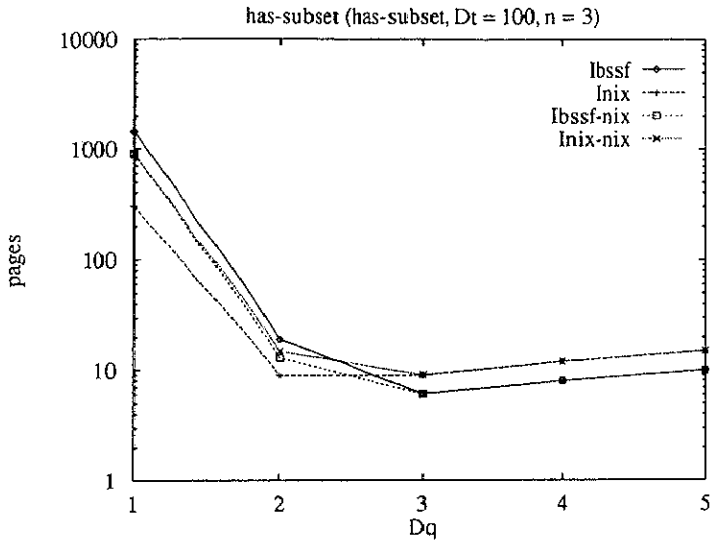

Figure 5.3: Retrieval Cost ($T \supseteq Q$, $D_t = 100$, $n = 3$)

$\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. When $D_t = 10$ and D_q is very small, the retrieval costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are larger than that of $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. But $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are generally better in other cases. Furthermore, the retrieval costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ can be improved by using the *smart retrieval strategy* described in Chapter 4. Therefore, for $T \subseteq Q$, $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are considered to be superior to $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. As it is not shown in Figure 5.4 and Figure 5.5, the cost of $\mathcal{I}_{\text{BSSF}}$ drastically increases for very large D_q values. Therefore, $\mathcal{I}_{\text{BSSF-NIX}}$ is the most efficient and stable one for $T \subseteq Q$.

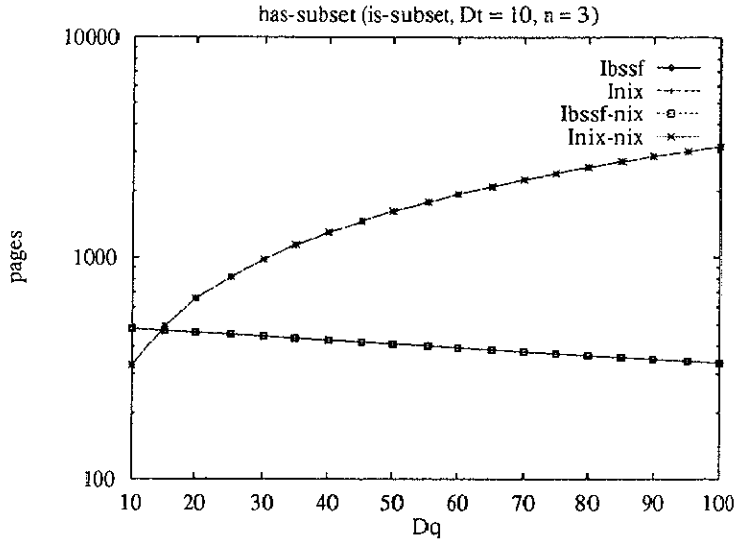

Figure 5.4: Retrieval Cost ($T \supseteq Q$, $D_t = 10$, $n = 3$)

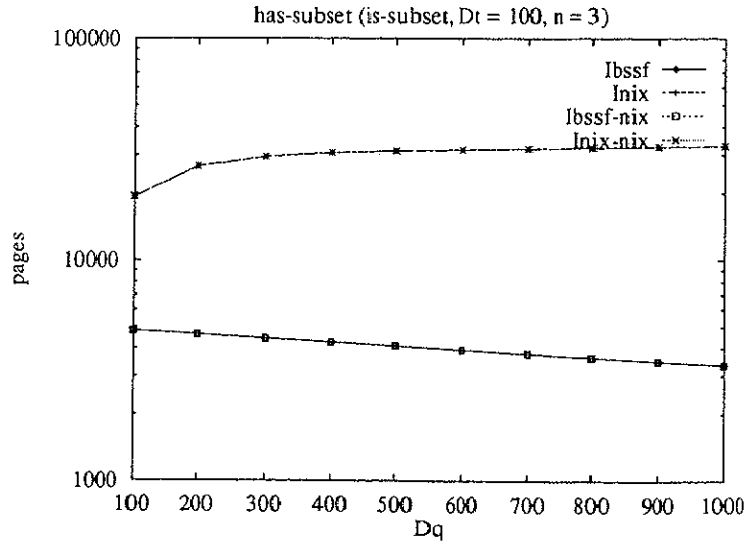

Figure 5.5: Retrieval Cost ($T \supseteq Q$, $D_t = 100$, $n = 3$)

5.5.2 Storage, Insertion, and Deletion Costs

Storage, insertion, and update costs are shown in Table 5.2. When $D_t = 10$, the storage costs are almost in the same degree. When $D_t = 100$, the storage costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are half of those of $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$.

Table 5.2: Storage, Insertion, and Deletion Costs

	SC		*IC*		*DC*	
	D_t=10	D_t=100	D_t=10	D_t=100	D_t=10	D_t=100
$\mathcal{I}_{\text{BSSF}}$	559	5059	41	394	12 †	12 †
$\mathcal{I}_{\text{NIX}}$	629	10048	24	242	930 ‡	57560 ‡
$\mathcal{I}_{\text{BSSF-NIX}}$	693	5193	44	397	73	426
$\mathcal{I}_{\text{NIX-NIX}}$	763	10193	27	245	27	245

† $n = 3$

‡ $n = 3$

The insertion costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are twice as much as $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. The deletion costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{NIX}}$ depend on n. Therefore, for $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{NIX}}$, representative costs are shown. The deletion cost of $\mathcal{I}_{\text{BSSF}}$ is very small. On the other hand, the cost of $\mathcal{I}_{\text{NIX}}$ is prohibitively larger than those of other access facilities. The reason is that $\mathcal{I}_{\text{NIX}}$ requires many forward traversals in deletion processing.

5.6 Cost Analysis for Case II

For Case II, the parameter settings are equal to Case I: $N_n = 30,000$, $D_t = 10$ or 100, $n = 2, 3, 4$, and for BSSF, $F = 500$ ($D_t = 10$), $F = 5000$ ($D_t = 100$), and $m = 2$. The only difference is the fanout parameters. The fanout parameters are set to

$$fan_i^{i+1} = fan \qquad (1 \leq i \leq n-1).$$

The constant fan value is set to 1, 5, or 10.

5.6.1 Retrieval Costs

The representative retrieval costs for $T \supseteq Q$ are shown in Figure 5.6 ($D_t = 10$) and Figure 5.7 ($D_t = 100$). In this case, forward traversals are only performed by $\mathcal{I}_{\text{BSSF}}$. The other three set access facilities do not depend on the fanout parameter fan or the path length n. The two figures indicate a similar tendency. Except for small D_q values (1 or 2), the retrieval costs are not different and increase monotonically. For small D_q values, $\mathcal{I}_{\text{BSSF}}$ configurations (especially $fan = 10$) give the worst costs. This is because $\mathcal{I}_{\text{BSSF}}$ needs forward traversals to process a query. In particular, when $fanout$ is large, many C_n objects correspond to one C_1 object so that the forward traversal cost increases. When $D_q = 1$, there are a considerable number of actual

drops and false drops. Therefore, the overhead of forward traversal cost determines the overall cost. However, when $D_q \geq 2$ or 3, drops are almost negligible, and the retrieval costs increase linearly. For other n-values, similar results are obtained.

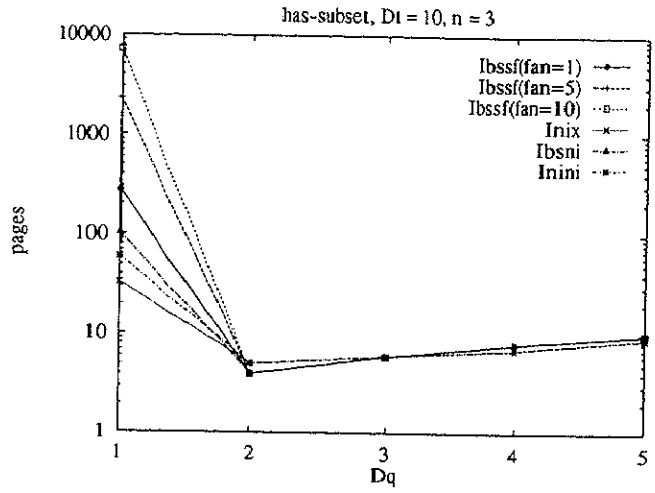

Figure 5.6: Retrieval Cost ($T \supseteq Q$, $D_t = 10$, $n = 3$)

The representative retrieval costs for $T \subseteq Q$ are shown in Figure 5.8 ($D_t = 10$) and Figure 5.9 ($D_t = 100$). In this query, $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{NIX}}$ need forward traversals. However, it seems that the retrieval cost of $\mathcal{I}_{\text{BSSF}}$ does not suffer from the penalty of forward traversals and its cost is almost the same as that of $\mathcal{I}_{\text{BSSF-NIX}}$. The reason is that the number of false drops of BSSF is very small for these D_q values so that few forward traversals occur. For this case, the smart retrieval strategy can also be applied to $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$.

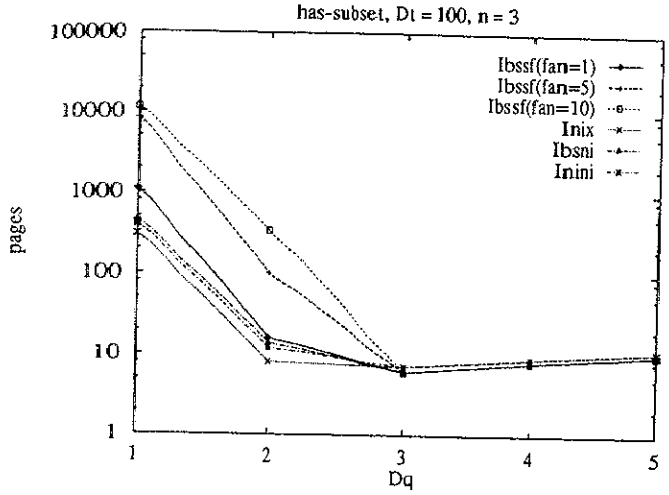

Figure 5.7: Retrieval Cost ($T \supseteq Q$, $D_t = 100$, $n = 3$)

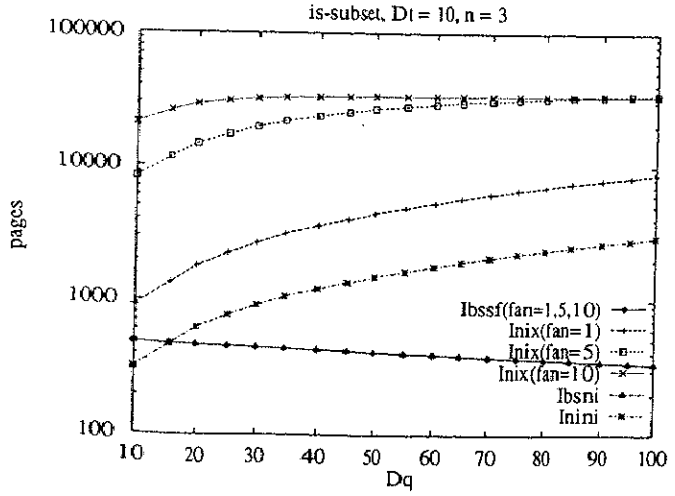

Figure 5.8: Retrieval Cost ($T \subseteq Q$, $D_t = 10$, $n = 3$)

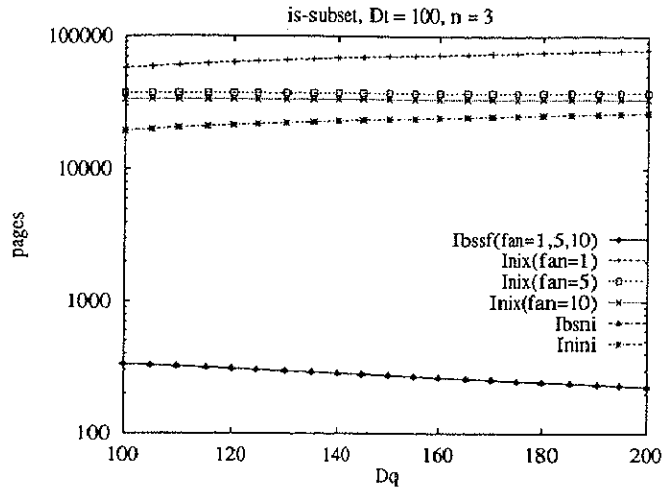

Figure 5.9: Retrieval Cost ($T \subseteq Q$, $D_t = 100$, $n = 3$)

5.6.2 Storage, Insertion, and Deletion Costs

Storage, insertion, and update costs are shown in Table 5.3. When $D_t = 10$, the storage costs are almost the same. When $D_t = 100$, the storage costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are almost half of those of $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. For insertion and deletion costs, they are not different from those of Case I except for the deletion cost of $\mathcal{I}_{\text{NIX}}$. Remind that the deletion cost formulas of $\mathcal{I}_{\text{NIX}}$ differs for two cases. The deletion cost of $\mathcal{I}_{\text{NIX}}$ depends not only on n but fan. Therefore the parameter fan greatly influences the performance.

Table 5.3: Storage, Insertion, and Deletion Costs

	SC		*IC*		*DC*	
	D_t=10	D_t=100	D_t=10	D_t=100	D_t=10	D_t=100
$\mathcal{I}_{\text{BSSF}}$	559	5059	41	394	12 †	12 †
$\mathcal{I}_{\text{NIX}}$	629	10044	24	242	934 ‡	57600 ‡
					21200 §	33500 §
$\mathcal{I}_{\text{BSSF-NIX}}$	662	5162	44	397	73	426
$\mathcal{I}_{\text{NIX-NIX}}$	732	10147	27	245	27	245

† $n = 3$

‡ $n = 3$, $fan = 1$

§ $n = 3$, $fan = 10$

5.6.3 Discussion

In this chapter, four set retrieval facilities $\mathcal{I}_{\text{BSSF}}$, $\mathcal{I}_{\text{NIX}}$, $\mathcal{I}_{\text{BSSF-NIX}}$, and $\mathcal{I}_{\text{NIX-NIX}}$ for nested objects are proposed, and their performance is compared. For the analysis, two configurations of nested objects (Case I and II) are considered. The costs of each set retrieval facility show similar tendency for two configurations. Therefore, it would be concluded that the result of the analysis holds for other situations.

For retrieval cost for $T \supseteq Q$, the analysis shows that four access facilities have similar performance except for $D_q = 1$. When $D_q = 1$, $\mathcal{I}_{\text{BSSF}}$ is the worst and $\mathcal{I}_{\text{NIX}}$ is the best. However, for the retrieval cost for $T \subseteq Q$, $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ show relatively stable performance and are better than $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$ for reasonable

range of D_q values. For very large D_q values, $\mathcal{I}_{\text{BSSF-NIX}}$ is superior to $\mathcal{I}_{\text{BSSF}}$. $\mathcal{I}_{\text{NIX}}$ suffers performance degradation for $T \subseteq Q$ of Case II as fanout increases.

The storage costs of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are equal to or smaller than those of $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. For insertion cost, $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are twice as much larger than $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$. For deletion cost, $\mathcal{I}_{\text{NIX}}$ is extremely high because of many forward traversals, and $\mathcal{I}_{\text{BSSF-NIX}}$ is slightly higher than those of $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{NIX-NIX}}$.

From the analysis, the following strategy selecting facilities for set retrieval:

1. If only one set retrieval facility is to be selected to support $T \supseteq Q$ and $T \subseteq Q$ queries, $\mathcal{I}_{\text{BSSF}}$ or $\mathcal{I}_{\text{BSSF}}$ is best suited within the retrieval facilities. $\mathcal{I}_{\text{BSSF}}$ is smaller than $\mathcal{I}_{\text{BSSF-NIX}}$ in terms of deletion cost, but $\mathcal{I}_{\text{BSSF-NIX}}$ is stable for large D_q values for $T \subseteq Q$. The selection is depend on the requirements.

2. If only one facility is to be selected and $T \ni q$ query frequently occurs, $\mathcal{I}_{\text{NIX-NIX}}$ may be better for the set retrieval facility. However, $\mathcal{I}_{\text{NIX-NIX}}$ cannot support $T \subseteq Q$ in an efficient manner.

3. If two or more facilities can be used, use $\mathcal{I}_{\text{NIX}}$ or $\mathcal{I}_{\text{NIX-NIX}}$ as a set retrieval facility for mainly $T \ni q$, and use $\mathcal{I}_{\text{BSSF}}$ or $\mathcal{I}_{\text{BSSF-NIX}}$ as for other queries.

Chapter 6

Discussion and Conclusion

In this dissertation, set-based signature files – indexing methods for set-valued objects – are proposed and their performance is examined with detailed cost models in some different situations.

In Chapter 3, false drop probability formulas estimating the number of false drops for four queries $T \supseteq Q$ (has-subset), $T \subseteq Q$ (is-subset), $T \sqcap Q$ (has-intersection), and $T \sqcap Q$ (is-equal) are derived.

In Chapter 4, four set retrieval facilities, three of them are set-based signature files (SSF, BSSF, BSSFcmpr) and the other is the nested index (NIX), are proposed for set retrieval of non-nested objects. Their retrieval cost for $T \supseteq Q$ and $T \subseteq Q$, storage cost, and update cost are are compared. Based on the analysis, novel query evaluation strategies to improve the retrieval costs of BSSF and NIX, called *smart object retrieval strategies*, are developed, and further the retrieval costs under the strategies are compared. The result indicates BSSF for small-scale databases and BSSFcmpr for medium-scale databases are promising set retrieval facilities.

In Chapter 5, four set retrieval facilities $\mathcal{I}_{\text{BSSF}}$, $\mathcal{I}_{\text{NIX}}$, $\mathcal{I}_{\text{BSSF-NIX}}$, and $\mathcal{I}_{\text{NIX-NIX}}$ are proposed combinating some existing index methods and applied to set retrieval of multilevel nested objects. The target queries for comparing their retrieval costs are root-level object retrieval with a leaf-level set comparison predicate. As the set comparison condition, $T \supseteq Q$ and $T \subseteq Q$ are considered. For each set access facility and query type, the retrieval algorithm is described in detail. Detailed algorithms for retrieval, insertion, and deletion are described, and cost formulas are derived. The analysis showed that $\mathcal{I}_{\text{BSSF}}$ and $\mathcal{I}_{\text{BSSF-NIX}}$ are efficient than $\mathcal{I}_{\text{NIX}}$ and $\mathcal{I}_{\text{NIX-NIX}}$ for those queries.

Based on the performance analysis in Chapter 4 and Chapter 5, it is concluded that set-based signature file is a promising set retrieval facility. But there still remain many issues. In the following, such issues are discussed.

Performance Analysis for Other Queries In Chapter 4 and Chapter 5, queries $T \supseteq Q$ and $T \subseteq Q$ are evaluated and analyzed. However, there are other queries $T \sqcap Q$ and $T \equiv Q$. For $T \equiv Q$, actual drops and false drops are considered to be very small and might be negligible. Therefore, the query cost will be small so that supporting $T \equiv Q$ query is not so difficult in terms of the performance. On the other hand, $T \sqcap Q$ query has many actual drops and false drops. In such a case, the final retrieval cost of the qualified objects might be very large, and dominate the retrieval cost. Therefore, there might be no differences between set-signature file organizations. But, the signature file may be inferior to NIX because of the existence of false drops. The analysis for these queries is important and is a very interesting research. As other kinds of queries, query conditions expressed by the combination of set predicates and logical operator ($\neg$, $\wedge$, and $\vee$) also be considered. Support for these queries should be also important.

Large-scale Databases In Chapter 5, set retrieval facilities are estimated for small-scale ($N = 32,000$) and medium-scale databases ($N = 320,000$). For medium-scale databases, the introduction of compression was effective. However, for large-scale databases, even compression scheme may be not efficient. Because the nested index, the rival of set-based signatures, is rather stable to some large N-value. To overcome this inferiority, the file structure of set based signature files should be devised to be suited to large-scale databases. For signature file in text retrieval area, various organization schemes are proposed [CCOL91, CZ93, Dep86, GTZ92, KP93, LL89, LL90a, LF92, Lin93, SDR83, ZRT91, ZCT93]. The application of these

organization schemes to set retrieval can also be considered. In fact, some methods are proposed by the author [IK94] (based on Hamming Filter) and Watanabe and Kitagawa [WIK94, WK95] (based on multilevel organization and partitioned organization).

Smart Retrieval In this study, the smart retrieval strategy that optimize the total retrieval cost at the sacrifice of many false drop occurrences. To the author's knowledge, such a scheme is not described in literature before the autor's proposal [IKO93]. In other study [PF94], this scheme is also called in *partial fetch* policy (usual retrieval is called *total fetch* policy). The remaining problem of the smart retrieval strategy is how to decide bit-slice files to be retrieved in a real situation.

Appendix

A. Properties of Two False Drop Probabilities

A.1 $T \supseteq Q$

For two false drop probabilities $Fd_{\{T \supseteq Q\},\mathrm{F1}}$ and $Fd_{\{T \subseteq Q\},\mathrm{F1}}$, their properties are examined here. $Fd_{\{T \supseteq Q\},\mathrm{F1}}$ and $Fd_{\{T \subseteq Q\},\mathrm{F1}}$ are abbreviated as $Fd_{\{T \supseteq Q\}}$ and $Fd_{\{T \subseteq Q\}}$.

For the false drop probability $Fd_{\{T \supseteq Q\}}$, the following properties hold:

Property 1 *$Fd_{\{T \supseteq Q\}}$ (Eq. (3.16)) also takes the minimum value when $m = m_{\mathrm{opt}}$.*

Since $Fd_{\{T \supseteq Q\}} = (Fd_{\{T \ni q\}})^{D_q}$, the partial derivative of $Fd_{\{T \supseteq Q\}}$ with respect to m is given as

$$\begin{aligned}\frac{\partial Fd_{\{T \supseteq Q\}}}{\partial m} &= \frac{\partial Fd_{\{T \supseteq Q\}}}{\partial Fd_{\{T \ni q\}}} \cdot \frac{\partial Fd_{\{T \ni q\}}}{\partial m} \\ &= D_q (Fd_{\{T \ni q\}})^{D_q - 1} \cdot \frac{\partial Fd_{\{T \ni q\}}}{\partial m}.\end{aligned}$$

As $0 < Fd_{\{T \ni q\}} < 1$, $Fd_{\{T \supseteq Q\}}$ corresponds to $Fd_{\{T \ni q\}}$ regarding increase and decrease.

■

When $m = m_{\text{opt}}$, the false drop probability Eq. (3.16) is simplified as

$$Fd_{\{T \supseteq Q\}} \simeq \left(\frac{1}{2}\right)^{\frac{D_q}{D_t} F \ln 2}. \tag{A.1}$$

Property 2 *$Fd_{\{T \supseteq Q\}}$ (Eq. (3.16)) is higher as D_t is larger.*

It is needed to show

$$Fd_{\{T \supseteq Q\}_{D_t}} \simeq \left(1 - e^{-\frac{mD_t}{F}}\right)^{mD_q} < Fd_{\{T \supseteq Q\}_{D_t+1}} \simeq \left(1 - e^{-\frac{m(D_t+1)}{F}}\right)^{mD_q}.$$

Since $0 < 1 - e^{-\frac{mD_t}{F}} < 1 - e^{-\frac{m(D_t+1)}{F}} < 1$ and $mD_q > 1$,

$$0 < \left(1 - e^{-\frac{mD_t}{F}}\right)^{mD_q} < \left(1 - e^{-\frac{m(D_t+1)}{F}}\right)^{mD_q} < 1.$$

Therefore, $Fd_{\{T \supseteq Q\}_{D_t}} < Fd_{\{T \supseteq Q\}_{D_t+1}}$.

■

Property 3 *$Fd_{\{T \supseteq Q\}}$ (Eq. (3.16)) is lower as D_q is larger.*

Since $mD_q < m(D_q + 1)$,

$$Fd_{\{T \supseteq Q\}_{D_q}} \simeq \left(1 - e^{-\frac{mD_t}{F}}\right)^{mD_q} > Fd_{\{T \supseteq Q\}_{D_q+1}} \simeq \left(1 - e^{-\frac{mD_t}{F}}\right)^{m(D_q+1)}.$$

■

Property 4 *$Fd_{\{T \supseteq Q\}}$ (Eq. (3.16)) is lower as F is larger.*

Since $0 < 1 - e^{-\frac{mD_t}{F+1}} < 1 - e^{-\frac{mD_t}{F}} < 1$,

$$Fd_{\{T \supseteq Q\}_{F+1}} \simeq \left(1 - e^{-\frac{mD_t}{F+1}}\right)^{mD_q} < Fd_{\{T \supseteq Q\}_F} \simeq \left(1 - e^{-\frac{mD_t}{F}}\right)^{mD_q}.$$

■

A.2 $T \subseteq Q$

For the false drop probability $Fd_{\{T \supseteq Q\}}$, the following properties hold. They are shown by the similar reasoning for $T \supseteq Q$.

Property 5 *$Fd_{\{T \subseteq Q\}}$ (Eq. (3.17)) takes the minimum value when $m = \frac{F \ln 2}{D_q}$.* ■

Property 6 *$Fd_{\{T \subseteq Q\}}$ (Eq. (3.17)) is lower as D_t is larger.* ■

Property 7 *$Fd_{\{T \subseteq Q\}}$ (Eq. (3.17)) is higher as D_q is larger.* ■

Property 8 *$Fd_{\{T \subseteq Q\}}$ (Eq. (3.17)) is lower as F is larger.* ■

B. Derivation of D_q^{opt}

To derive a rough estimation formula of D_q^{opt}, it is assumed that the number of actual drops is negligible and $Fd_{\{T \subseteq Q\}}$ is very small. In addition to this assumption, a simplified formula for $LC_{OID\{c\}}(N)$ [IKO93] is used:

$$LC_{OID\{c\}}(N) = SC_{OID}(N) \times \min(Fd_{\{c\}}(N_{oid} - \alpha_{\{c\}}(N)) + \alpha_{\{c\}}(N), 1), \quad \text{(B.1)}$$

where $\alpha_{\{c\}}(N)$ represents the expected number of actual drops per page of the OID file and given by

$$\alpha_{\{c\}}(N) = \frac{A_{\{c\}}(N)}{SC_{OID}(N)}. \quad \text{(B.2)}$$

Since $A_{\{T \subseteq Q\}}(N) \approx 0$ and $\alpha_{\{T \subseteq Q\}}(N) \approx 0$,

$$\begin{aligned} LC_{OID\{T \subseteq Q\}}(N) &\approx SC_{OID}(N) \times \min(Fd_{\{T \subseteq Q\}} \times N_{oid}, 1) \\ &= SC_{OID}(N) Fd_{\{T \subseteq Q\}} N_{oid} \\ &\approx Fd_{\{T \subseteq Q\}} N \qquad (SC_{OID}(N) = \lceil N/N_{oid} \rceil). \end{aligned}$$

Therefore, the total retrieval cost is

$$\begin{aligned} RC_{\mathrm{BSSF}\{T\subseteq Q\}}(N) &= SC_{\mathrm{bsf}}(N)M_{\{T\subseteq Q\}} + LC_{\mathrm{OID}\{T\subseteq Q\}}(N) \\ &\quad + P_{\mathrm{s}}A_{\{T\subseteq Q\}}(N) + P_{\mathrm{u}}Fd_{\{T\subseteq Q\}}(N - A_{\{T\subseteq Q\}}(N)) \qquad \text{(Eq. (4.7))} \\ &\approx SC_{\mathrm{bsf}}(N)M_{\{T\subseteq Q\}} + Fd_{\{T\subseteq Q\}}N + P_{\mathrm{u}}Fd_{\{T\subseteq Q\}}N \\ &\approx \left\lceil \frac{N}{Pb} \right\rceil (F - m_{\mathrm{q}}) + Fd_{\{T\subseteq Q\}}(N + P_{\mathrm{u}}N) \\ &= \left\lceil \frac{N}{Pb} \right\rceil Fe^{-\frac{m}{F}D_{\mathrm{q}}} + (1 - e^{-\frac{m}{F}D_{\mathrm{q}}})^{mD_{\mathrm{t}}}N(1 + P_{\mathrm{u}}). \end{aligned}$$

The partial derivative of this formula with respect to D_{q} is taken and the D_{q}-value that minimizes RC is obtained. The D_{q}-value is the estimated value of $D_{\mathrm{q}}^{\mathrm{opt}}$:

$$\begin{aligned} D_{\mathrm{q}}^{\mathrm{opt}} &\approx -\frac{F}{m}\ln\left[1 - \left(\frac{\left\lceil \frac{N}{Pb} \right\rceil F}{(1 + P_{\mathrm{u}})NmD_{\mathrm{t}}}\right)^{\frac{1}{mD_{\mathrm{t}}-1}}\right] \\ &\approx -\frac{F}{m}\ln\left[1 - \left(\frac{F}{(1 + P_{\mathrm{u}})mD_{\mathrm{t}}Pb}\right)^{\frac{1}{mD_{\mathrm{t}}}}\right], \end{aligned} \qquad \text{(B.3)}$$

where $mD_{\mathrm{t}} \gg 1$ and $\left\lceil \frac{N}{Pb} \right\rceil \approx \frac{N}{Pb}$ is used.

Next, the minimal value of $RC_{\mathrm{BSSF}\{T\subseteq Q\}}(N)$ at the point $D_{\mathrm{q}} = D_{\mathrm{q}}^{\mathrm{opt}}$ is derived.

$$\begin{aligned} RC_{\mathrm{BSSF}\{T\subseteq Q\}}(N)_{[D_{\mathrm{q}}=D_{\mathrm{q}}^{\mathrm{opt}}]} &= \left\lceil \frac{N}{Pb} \right\rceil Fe^{-\frac{m}{F}D_{\mathrm{q}}^{\mathrm{opt}}} + (1 - e^{-\frac{m}{F}D_{\mathrm{q}}^{\mathrm{opt}}})^{mD_{\mathrm{t}}}N(1 + P_{\mathrm{u}}) \\ &\approx \left\lceil \frac{N}{Pb} \right\rceil F\left[1 - \left(\frac{F}{(1 + P_{\mathrm{u}})mD_{\mathrm{t}}Pb}\right)^{\frac{1}{mD_{\mathrm{t}}}}\right] + \frac{FN}{mD_{\mathrm{t}}Pb} \\ &\approx \frac{FN}{Pb}\left\{\left[1 - \left(\frac{F}{(1 + P_{\mathrm{u}})mD_{\mathrm{t}}Pb}\right)^{\frac{1}{mD_{\mathrm{t}}}}\right] + \frac{1}{mD_{\mathrm{t}}}\right\}. \end{aligned} \qquad \text{(B.4)}$$

Therefore, the retrieval cost for $D_q = D_q^{opt}$ is approximately proportional to N.

C. Forward Traversal Costs

C.1 Derivation of *FFT*

Suppose that x C_1 objects are given. Here, the expected number of page accesses is derived for forward traversals from the C_1 objects to the descendant C_n objects. The use of the *nested-loop forward traversal* method [Ber93] is assumed and the assumptions made in Section 5.4.

There exist two cases for forward traversals:

1. The traversal cannot be finished until all reachable C_n objects are obtained (*full forward traversal*).
2. The traversal can be finished at the time a C_n object satisfying the condition is found (*partial forward traversal*).

The full forward traversal cost FFT is derived as

$$FFT(x) = P_o x \sum_{i=1}^{n} fan_1^i. \tag{C.1}$$

C.2 Derivation of $e(p, q)$

Next, let us consider the partial forward traversal cost PFT. As a preparation, first a formula $e(p, q)$, representing the answer for the following question, is derived:

> Suppose that there exist p lottery tickets and q of them are winning tickets. When we draw these lots until we meet an winning ticket, how many draws are required?

The probability that we meet an winning ticket at the i-th draw is [1]

$$\begin{aligned} p(p, q, i) &= \frac{q}{p} && (i = 1) \\ p(p, q, i) &= \left(1 - \frac{q}{p}\right) \times \left(1 - \frac{q}{p-1}\right) \times \cdots \times \left(1 - \frac{q}{p-i+2}\right) \times \frac{q}{p-i+1} \\ &= \frac{q}{p-i+1} \times \prod_{j=0}^{i-2} \left(1 - \frac{q}{p-j}\right) && (2 \leq i \leq p - q + 1). \end{aligned}$$

[1] Note that we will meet an winning ticket at most $p - q + 1$-th draw.

Therefore,

$$\begin{aligned} e(p,\ q) &= 1 \times \frac{q}{p} + 2 \times p(p,\ q,\ 2) + 3 \times p(p,\ q,\ 3) + \cdots \\ &\quad + (p - q + 1) \times p(p,\ q,\ p - q + 1) \\ &= \frac{q}{p} + \sum_{i=2}^{p-q+1} (i \times p(p,\ q,\ i)). \end{aligned} \tag{C.2}$$

C.3 Derivation of *PFT*

Let us consider a partial forward traversal from a C_1 object O_1, and assume that O_1 has fan_1^n descendant C_n objects and y of them satisfy the given query condition. If we access the fan_1^n C_n objects until we first meet one of the y objects, the expected number of accesses is $e(fan_1^n,\ y)$. In the case, as we are considering a partial forward traversal, we can finish the traversal immediately.

Next, we derive PFT formulas for two cases of y-values. First, we consider the case of $y \geq 1$. To derive PFT, we must estimate the number of accessed objects for each class C_i $(1 \leq i \leq n-1)$. Since $e_n(y) = e(fan_1^n,\ y)$ C_n objects are retrieved, and the fanout from C_{n-1} to C_n is fan_{n-1}^n,

$$e_{n-1}(y) = \left\lceil \frac{e_n(y)}{fan_{n-1}^n} \right\rceil$$

C_{n-1} objects are accessed. Similarly, the number of accessed C_i objects is

$$e_i(y) = \left\lceil \frac{e_{i+1}(y)}{fan_i^{i+1}} \right\rceil \quad (1 \leq i \leq n-1). \tag{C.3}$$

Thus, the partial forward cost for a C_n object is given as $P_o \sum_{i=1}^{n} e_i(y)$. Therefore, PFT, the partial forward cost for x C_1 objects, is [2]

$$PFT(x, y) = P_o x \sum_{i=1}^{n} e_i(y).$$

Next, suppose that $y < 1$. Consider partial forward traversals from x C_1 objects. In this case, actually xy C_1 objects only have descendant C_n objects satisfying the query condition. Remaining $x(1-y)$ C_1 objects do not have such C_n objects. Thus, in the traversal processing, partial forward traversals with the cost $PFT(1, 1)$ are performed for the xy C_1 objects, and for $x(1-y)$ C_1 objects, full forward traversals are performed.. Therefore, the forward traversal cost is

$$PFT(x, y) = xyPFT(1, 1) + x(1-y)FFT.$$

This is equivalent to

$$PFT(x, y) = PFT(xy, 1) + FFT(x(1-y)).$$

[2]We used the assumption that two objects do not share their references.

In summary,

$$PFT(x,\, y) = \begin{cases} P_o x \sum_{i=1}^{n} e_i(y) & \text{if } y \geq 1 \\ PFT(xy,\, 1) + FFT(x(1-y)) & \text{if } y < 1. \end{cases} \tag{C.4}$$

D. Derivation of $DC\{\mathcal{I}_{\text{BSSF}}\}$

In the deletion process of $\mathcal{I}_{\text{BSSF}}$, in first, all C_1 objects that reference O_n are found (step 1 in Section 3.4). As we assume here that any objects do not share their references, only one C_1 object references O_n. However, the resulting OIDs of the BSSF retrieval are not necessarily one due to the existence of false drops.

In this case, the pattern match condition for BSSF is "find all target signatures that perfectly match the signature S generated from the A_n-value of O_n," but it is not practical to take complete matches. For example, the following processing scheme is considered:

1. Retrieve some (not necessarily all) bit-slices corresponding "1"'s in the signature S.

2. take the bitwise-AND of such bit-slices.

3. If the number of "1"'s in the resulting bit-slice, namely, the number of drops, is sufficiently small, then finish the process.

In detail, we assume the following strategy. First, bit-slice files are retrieved and AND-ed until the following condition is satisfied:

$$\text{No. of false drops} = Fd_{\{T \supseteq Q\}}(PI - A_{\{T \supseteq Q\}}(PI)) = 1.$$

Next, the OIDs are retrieved. Let *nbs* be the number of bit-slices retrieved in this scheme. As the number of actual drop is *one*, the cost to obtain the OID of C_1 object using BSSF is approximately

$$SC_{\text{bsf}}(PI) \times nbs + npa(2,\ N_{\text{oid}} SC_{\text{OID}}(PI),\ SC_{\text{OID}}(PI)).$$

Thus, the forward traversal is performed based on the two C_1 objects. The cost for the forward traversal is about $2P_o n$ pages. Since we expect two pages are required to read and write a BSSF entry,

$$DC\{\mathcal{I}_{\text{BSSF}}\} \approx SC_{\text{bsf}}(PI) \times nbs + npa(2,\ N_{\text{oid}} SC_{\text{OID}}(PI),\ SC_{\text{OID}}(PI)) + 2P_o n + 2. \tag{D.1}$$

E. Derivation of $DC\{\mathcal{I}_{\text{NIX}}\}$

To derive the deletion cost of $\mathcal{I}_{\text{NIX}}$, let us suppose that the target sets have equal cardinality D_t and that no two objects do not share their references, and let O_n be the OID of the C_n object to be deleted, and let O_1 be the C_1 object that has O_n as a descendant [3].

In the deletion process of $\mathcal{I}_{\text{NIX}}$, NIX is first retrieved with D_t keys. The cost is expressed as

$$RC_{\text{NIX}}\left(V, \frac{D_t N_n}{V}, D_t\right).$$

As the result of the first step, D_t NIX entries $\langle v_i, \{O_1, \ldots\}\rangle$ $(1 \leq i \leq D_t)$ are obtained. Next, the union of OID sets in these D_t entries are taken. The resulting set contains OIDs of all C_1 objects referencing C_n objects that contains at least one element of the A_n value of O_n $(\{v_1, \ldots, v_{D_t}\})$ in their A_n values. The number of such C_n objects is given by

$$\left(1 - \frac{\binom{V-D_t}{D_t}}{\binom{V}{D_t}}\right) \times (N_n - 1) + 1, \tag{E.1}$$

where the second term represents O_n itself. Thus, the number of C_1 objects refer-

[3] Since no two objects do not share their references, such C_1 object is uniquely determined for a O_n.

encing the C_n objects is estimated by Yao's formula.

$$npa\left([Eq.\ (E.1)],\ N_n,\ \frac{N_n}{fan_1^n}\right). \tag{E.2}$$

In the second step, forward traversals are performed. In this case, we must check all descendants so that full forward traversals are used. The cost is given as

$$FFT([Eq.\ (E.2)]). \tag{E.3}$$

As the result of the forward traversals, $fan_1^n \times ([Eq.\ (E.2)])$ C_n objects are obtained. However, in these C_n objects, only fan_1^n objects are supposed to be referenced from O_1 [4]. In the deletion process of $\mathcal{I}_{\text{NIX}}$, we must check these fan_1^n objects.

We can delete the OID O_1 from the NIX entry $\langle v_i, \{O_1, \ldots, \}\rangle$ when the remaining $fan_1^n - 1$ objects do not contain v_i in their A_n values ($1 \leq i \leq D_t$). The probability that we can delete O_1 from the entry is

$$\left(1 - \frac{\binom{V-1}{D_t-1}}{\binom{V}{D_t}}\right)^{fan_1^n-1}.$$

Thus, the number of entries to be deleted is

$$D_t\left(1 - \frac{\binom{V-1}{D_t-1}}{\binom{V}{D_t}}\right)^{fan_1^n-1}. \tag{E.4}$$

[4]Note that O_n is contained in the fan_1^n C_n objects.

Therefore, the deletion cost of $\mathcal{I}_{\text{NIX}}$ is

$$DC\{\mathcal{I}_{\text{NIX}}\} = npa\left(V, \frac{D_t N_n}{V}, D_t\right) + [Eq.\ (E.3)] + 2P_o[Eq.\ (E.4)]. \qquad \text{(E.5)}$$

References

[AB84] S. Abiteboul and N. Bidoit: "Non First Normal Form Relations to Represent Hierarchically Organized Data", in *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pp. 191–200, Waterloo, Ontario, Canada, Apr. 1984.

[AB86] S. Abiteboul and N. Bidoit: "Non First Normal Form Relations: An Algebra Allowing Data Restructuring", *J. Comput. Syst. Sci.*, 33:361–393, 1986.

[AFS89] S. Abiteboul, P. C. Fisher, and H.-J. Schek: *Nested Relations and Complex Objects in Databases*, Vol. 361 of *Lecture Notes in Computer Science*, Springer-Verlag, June 1989.

[BCH87] P. B. Berra, S. M. Chung, and N. I. Hachem: "Computer Architecture for a Surrogate File to a Very Large Data/Knowledge Base", *IEEE Computer*, pp. 25–32, Mar. 1987.

[Ber90] E. Bertino: "Optimization of Queries using Nested Indices", in F. Bancilhon, C. Thanos, and D. Tsichritzis eds., *Advances in Database Technology - EDBT '90*, Vol. 416, pp. 44–59, Springer-Verlag, 1990.

[Ber91] E. Bertino: "An Indexing Technique for Object-Oriented Databases", in *Proc. IEEE Conf. on Data Eng.*, pp. 160–170, Kobe, Japan, Apr. 1991.

[Ber93] E. Bertino: "A Survey of Indexing Techniques for Object-Oriented Database Management Systems", in J. C. Freytag, D. Maier, and G. Vossen eds., *Query Processing for Advanced Database Systems*, chapter 13, pp. 383–418, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1993.

[Ber94] E. Bertino: "Index Configuration in Object-Oriented Databases", *VLDB Journal*, 3(3):355–399, July 1994.

[Bid87] N. Bidoit: "The Verso Algebra and How to Answer Queries without Joins", *J. Comput. Syst. Sci.*, 35(3):321–364, Dec. 1987.

[BK89] E. Bertino and W. Kim: "Indexing Techniques for Queries on Nested Objects", *IEEE Trans. on Knowledge and Data Engineering*, 1(2):196-214, June 1989.

[BM93] E. Bertino and L. Martino: *Object-Oriented Database Systems*, Addison-Wesley, 1993.

[BRG88] E. Bertino, F. Rabitti, and S. Gibbs: "Query Processing in a Multimedia Document System", *ACM Trans. Office Inf. Syst.*, 6(1):1-41, Jan. 1988.

[Cat91] R. G. G. Cattell: *Object Data Management - Object-Oriented and Extended Relational Database Systems*, Addison-Wesley, 1991.

[CBBC94] S. Choenni, E. Bertino, H. M. Blanken, and T. Chang: "On the Selection of Optimal Index Configuration in OO Databases", in *Proc. IEEE Conf. on Data Eng.*, pp. 526–537, Houston, Texas, Feb. 1994.

[CCOL91] J. W. Chang, H. J. Cho, S. H. Oh, and Y. J. Lee: "Hybrid access method: an extended two-level signature file approach", in *Int'l. Conf. on Multimedia Information Systems*, pp. 51–62, 1991.

[CF84] S. Christodoulakis and C. Faloutsos: "Design Considerations for a Message File Server", *IEEE Trans. Softw. Eng.*, SE-10(2):201–210, Mar. 1984.

[CHT86] S. Christodoulakis, F. Ho, and M. Theodoridou: "The Multimedia Object Presentation Manager of MINOS: A Symmetric Approach", in *Proc. ACM SIGMOD Conf.*, pp. 295–310, Washington, D.C., May 1986.

[CJ86] R. M. Colomb and J. Jayasooriah: "A Clause Indexing System for PROLOG Based on Superimposed Coding", *The Australian Compututer Journal*, 18(1):18–25, Feb. 1986.

[CJ94] C.-C. Chang and J.-H. Jiang: "A Fast Spatial Match Retrieval Using a Superimposed Coding Technique", in *Intl. Symp. on Advanced Database Technologies and Their Integration (ADTI'94)*, pp. 71–78, Nara, Japan, Oct. 1994.

[CL89] J. W. Chang and Y. J. Lee: "Multikey Access Scheme Based on Term Discrimination and Signature Clustering", in *Intl. Symp. on Database Systems for Advanced Applications*, pp. 211–218, Soul, Korea, Apr. 1989.

[CL92] J. W. Chang and Y. J. Lee: "Multikey Access Scheme Based on Term Discrimination and Signature Clustering", in W. Kim, Y. Kambayashi, and I. S. Paik eds., *Database Systems for Next-Generation Applications - Principles and Practice*, Vol. 1 of *Advanced Database and Development Series*, World Scientific Publishing, Singapore, 1992.

[Col89] L. S. Colby: "A Recursive Algebra and Query Optimization for Nested Relations", in *Proc. ACM SIGMOD Conf.*, pp. 273–283, Portland, OR, May-June 1989.

[Col90] L. S. Colby: "A Recursive Algebra for Nested Relations", *Information Systems*, 15(5):567–582, 1990.

[CS89] W. W. Chang and H. J. Schek: "A Signature Access Method for the Starburst Database System", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 145–153, Amsterdam, The Netherlands, Aug. 1989.

[CYKL93] J. W. Chang, J. W. Yoo, M. H. Kim, and Y. J. Lee: "A Signature-based Hybrid Access Scheme for Text Databases", in *Intl. Symp. on Next Generation Database Systems and Their Applications (NDA '93)*, pp. 138–144, Fukuoka, Japan, Sept. 1993.

[CYL92] J. W. Chang, J. S. Yoo, and Y. J. Lee: "Performance Comparison of Signature-based Multikey Access Methods", *Microprocessing and Microprogramming*, 35:345–352, 1992.

[CZ93] P. Ciaccia and P. Zezula: "Estimating Accesses in Partitioned Signature File Organizations", *ACM Trans. Database Syst.*, 11(2):133–142, Apr. 1993.

[Dad88] P. Dadam: "Advanced Information Management (AIM): Research in Extended Nested Relations", *IEEE Data Eng.*, 11(3):4-14, Dec. 1988.

[Dep86] U. Deppisch: "S-tree: A dynamic balanced signature index for office retrieval", in *Proc. of the 1986 ACM Conf. "Research and Development in Information Retrieval"*, Pisa, Italy, Sept. 1986.

[DG88] A. Deshpande and D. V. Gucht: "An Implementation for Nested Relational Databases", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 76–87, Los Angeles, Aug.-Sept. 1988.

[DGMS89] D. H.-C. Du, S. Ghanta, K. J. Maly, and S. M. Sharrock: "An Efficient File Structure for Document Retrieval in the Automated Office Environment", *IEEE Trans. on Knowledge and Data Engineering*, 1(2):258–273, June 1989.

[Dit90] K. R. Dittrich: "Object-Oriented Database Systems: The Next Miles of the Marathon", *Information Systems*, 1990.

[Dit91] K. R. Dittrich: "Object-Oriented Database Systems: The Notion and the Issues", in *On Object-Oriented Database Systems*, Topics in Information Systems, chapter 1, pp. 3–10, Springer-Verlag, 1991.

[DKA+86] P. Dadam, K. Küspert, F. Andersen, H. Blanken, R. Erbe, J. Güenauer, V. Lum, P. Pistor, and G. Walch: "A DBMS Prototype to Support Extended NF^2 Relations: An Integrated View on Flat Tables and Hierarchies", in *Proc. ACM SIGMOD Conf.*, pp. 356–367, Washington, DC, May 1986.

[DL89] P. Dadam and V. Linnemann: "Advanced Information Management (AIM): Advanced database technology for integrated applications", *IBM Syst. J.*, 28(4):661–681, 1989.

[Fal85a] C. Faloutsos: "Access Methods for Text", *ACM Comput. Surv.*, 17(1):49–74, Mar. 1985.

[Fal85b] C. Faloutsos: "Signature Files: Design and Performance Comparison of Some Signature Extraction Methods", in *Proc. ACM SIGMOD Conf.*, pp. 63–82, Austin, Texas, May 1985.

[Fal88] C. Faloutsos: "Signature Files: An Integrated Access Method for Text and Attributes, Suitable for Optical Disk Storage", *BIT*, 28:736–754, 1988.

[Fal90] C. Faloutsos: "Signature-Based Text Retrieval Methods: A Survey", *IEEE Database Engineering*, 13(1):25–32, Mar. 1990.

[FC84] C. Faloutsos and S. Christodoulakis: "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation", *ACM Trans. Office Inf. Syst.*, 2(4):267–288, Oct. 1984.

[FC87] C. Faloutsos and S. Christodoulakis: "Description and Performance Analysis of Signature File Methods for Office Filing", *ACM Trans. Office Inf. Syst.*, 5(3):237–257, July 1987.

[FC88] C. Faloutsos and R. Chan: "Fast Text Access Methods for Optical and Large Magnetic Disks: Design and Performance Comparison", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 280–293, Los Angeles, Aug.-Sept. 1988.

[FSTG85] P. C. Fischer, L. V. Saxton, S. J. Thomas, and D. V. Gucht: "Interactions between Dependencies and Nested Relational Structures", *J. Comput. Syst. Sci.*, 31:343–354, 1985.

[FT83] P. C. Fischer and S. J. Thomas: "Operators for Non-First-Normal-Form Relations", in *Proc. IEEE COMPSAC*, pp. 464–475, Chicaco, IL, 1983.

[GF88] D. V. Gucht and P. C. Fischer: "Multilevel Nested Relational Structures ", *J. Comput. Syst. Sci.*, 36:77–105, 1988.

[GG88] M. Gyssens and D. V. Gucht: "The Powerset Algebra as a Result of Adding Programming Constructs to the Nested Relational Algebra", in *Proc. ACM SIGMOD Conf.*, pp. 225–232, Chicago, IL, June 1988.

[GTZ92] F. Grandi, P. Tiberio, and P. Zezula: "Frame-Sliced Partitioned Parallel Signature Files", in *Proc. of 15th ACM SIGIR Conf.*, pp. 286–297, Copenhagen, Denmark, June 1992.

[GV75] R. G. Gallager and D. C. V. Voorhis: "Optimal Source Codes for Geometrically Distributed Integer Alphabets", *IEEE Trans. on Information Theory*, pp. 228–230, Mar. 1975.

[HHR93] E. N. Hanson, T. M. Harvey, and M. A. Roth: "Experiences in Database System Implementation Using a Persistent Programming Language", *Software - Practice and Exprerience*, 23(12):1361–1377, Dec. 1993.

[HSR91] T. M. Harvey, C. W. Schnepf, and M. A. Roth: "The Design of the Triton Nested Relational Database System", *ACM SIGMOD Record*, 20(3):62–72, Sept. 1991.

[Hug91] J. G. Hughes: *Object-Oriented Databases*, Prentice-Hall, 1991.

[IK94] Y. Ishikawa and H. Kitagawa: "Set-valued Object Retrieval based on Hamming Filter", in *IPSJ 49th Annual Conf.*, Sept. 1994, 6W–9, (in Japanese).

[IKO93] Y. Ishikawa, H. Kitagawa, and N. Ohbo: "Evaluation of Signature Files as Set Access Facilities in OODBs", in *Proc. ACM SIGMOD Conf.*, pp. 247–256, Washington, D.C., May 1993.

[JS82] G. Jaeschke and H.-J. Schek: "Remarks on the Algebra of Non First Normal Form Relations", in *Proc. ACM SIGACT-SIGMOD Symp. on*

Principles of Database Systems, pp. 124–138, Los Angeles, CA, Mar. 1982.

[KFIO93] H. Kitagawa, Y. Fukushima, Y. Ishikawa, and N. Ohbo: "Estimation of False Drops in Set-valued Object Retrieval with Signature Files", in D. B. Lomet ed., *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, Vol. 730 of *Lecture Notes in Computer Science*, pp. 146–163, Springer-Verlag, Oct. 1993.

[Kho93] S. Khoshafian: *Object-Oriented Databases*, John Wiley & Sons, 1993.

[Kim90a] W. Kim: *Introduction to Object-Oriented Databases*, MIT Press, 1990.

[Kim90b] W. Kim: "Object-Oriented Databases: Definition and Research Directions", *IEEE Trans. on Knowledge and Data Engineering*, 2(3):327–341, Sept. 1990.

[KK89] H. Kitagawa and T. L. Kunii: *The Unnormalized Relational Data Model: For Office Form Processor Design*, Computer Science Workbench Series, Springer-Verlag, Tokyo, Japan, 1989.

[KKD90] W. Kim, K.-C. Kim, and A. Dale: "Indexing Techniques for Object-Oriented Databases", in W. Kim and F. H. Lochovsky eds., *Object-Oriented Concepts, Databases, and Applications*, chapter 15, pp. 371–394, ACM Press, New York, 1990.

[KKO91] H. Kitagawa, T. L. Kunii, and N. Ohbo: "Classification of Nested Tables under Deepely Nested Algebra", in *Proc. 24th Hawaii Intl. Conf. on System Sciences*, pp. 165–173, Hawaii, Jan. 1991.

[KL89] W. Kim and F. H. Lochovsky eds.: *Object-Oriented Concepts, Databases, and Applications*, Frontier Series, ACM Press, 1989.

[KM94a] A. Kemper and G. Moerkotte: *Object-Oriented Database Management – Applications in Engineering and Computer Science*, Prentice-Hall, 1994.

[KM94b] C. Kilger and G. Moerkotte: "Indexing Multiple Sets", in *Proc. Int'l. Conf. on Very Large Data Bases*, Santiago, Chile, Sept. 1994.

[KP92] Y.-M. Kwon and Y.-J. Park: "A New Indexing Technique for Nested Queries on Composite Objects", *IEICE Trans. Inf. & Syst.*, E75-D(6):861–872, Nov. 1992.

[KP93] Y.-M. Kwon and Y.-J. Park: "Generalized Partitioning Scheme of Signature File for Information Retrieval", *IEICE Trans. Inf. & Syst.*, E76-D(2):189–198, Feb. 1993.

[LF92] Z. Lin and C. Faloutsos: "Frame-Sliced Signature Files", *IEEE Trans. on Knowledge and Data Engineering*, 4(3):281–289, June 1992.

[Lin87] V. Linnemann: "Non First Normal Form Relations and Recursive Queries: An SQL-Based Approach", in *Proc. IEEE Conf. on Data Eng.*, pp. 591–598, Los Angeles, CA, Feb. 1987.

[Lin91] Z. Lin: "CAT: An Execution Model for Concurrent Full Text Search", in *Proc. of 1st Intl. Conf. on Parallel and Distributed Information Systems*, pp. 151–158, Florida, Dec. 1991.

[Lin93] Z. Lin: "Concurrent Frame Signature File", *Distributed and Parallel Databases*, 1(3):231–249, July 1993.

[LL89] D. L. Lee and C.-W. Leng: "Partitioned Signature Files: Design Issues and Performance Evaluation", *ACM Trans. Office Inf. Syst.*, 7(2):158–180, Apr. 1989.

[LL90a] D. L. Lee and C.-W. Leng: "A Partitioned Signature File Structure for Multiattribute and Text Retrieval", in *Proc. IEEE Conf. on Data Eng.*, pp. 389–397, Los Angeles, CA, Feb. 1990.

[LL90b] M. Levene and G. Loizou: "The Nested Relation Type Model: An Application of Domain Theory to Databases", *The Computer Journal*, 33(1):19–30, 1990.

[LL92a] D. L. Lee and F. H. Lochovsky: "HYTREM – A Hybrid Text-Retrieval Machine for Large Databases", *IEEE Trans. on Computers*, 39(1):111–123, Jan. 1992.

[LL92b] W. Lee and D. L. Lee: "Signature File Methods for Indexing Object-Oriented Database Systems", in *Intl. Computer Science Conf. (ICSC)*, pp. 616–622, Hong Kong, 1992.

[LYC92] S.-Y. Lee, M.-C. Yang, and J.-W. Chen: "Signature File as a Spatial Filter for Iconic Image Database", *Journal of Visual Languages and Computing*, 3:373–397, 1992.

[MA] E. Murphree and D. Aktug: "Derivation of probability distribution of the weight of the query signature", (Preprint. 1st author's address: Department of Mathematics and Statistics, Miami University, Oxford, OH 45056, USA).

[Mak82] A. Makinouchi: "A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model", in *Proc. 3rd VLDB Conf.*, pp. 447–453, 1982.

[Mak91] A. Makinouchi: "Architectures of the Object-Oriented Database Management Systems", *Journal of IPSJ*, 32(5):514–522, May 1991, (in Japanese).

[Mas91] Y. Masunaga: "Object-Oriented Database System: The Next Generation Database System", *Journal of IPSJ*, 32(5):490–499, May 1991, (in Japanese).

[MS86] D. Maier and J. Stein: "Indexing in an object-oriented DBMS", in *Proc. Int. Workhop Object-Oriented Database Syst.*, pp. 171–182, Asilomar, CA, sep 1986.

[Obj] Object Design, Inc., One New England Executive Park, Burlington, MA: *ObjectStore Reference Manual Release 2.0 for UNIX Systems.*

[ONT] ONTOS, Inc., Three Burlington Woods, Burlington, MA: *ONTOS DB 3.0 Reference Manual.*

[Ozs88] Z. M. Ozsoyoglu: "Special Issue on Nested Relations", *IEEE Database Engineering*, 11(3), 1988.

[PA86] P. Pistor and F. Andersen: "Designing a Generalized NF^2 Model with an SQL-Type Language Interface", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 278–285, Kyoto, Japan, Aug. 1986.

[PBC80] J. L. Pfaltz, W. J. Berman, and E. M. Cagley: "Partial-Match Retrieval Using Indexed Descriptor Files", *Commun. ACM*, 23(9):522–528, Sept. 1980.

[PD89] P. Pistor and P. Dadam: "The Advanced Information Management Prototype", in S. Abiteboul, P. C. Fischer, and H.-J. Schek eds., *Nested Relations and Complex Objects*, Vol. 416 of *Lecture Notes in Computer Science*, pp. 3–26, Springer-Verlag, 1989.

[PF94] G. Panagopoulos and C. Faloutsos: "Bit-Sliced Signature Files for Very Large Text Databases on a Parallel Machine Architecture", in *Proc. of 4th Intl. Conf. on EDBT*, pp. 378–392, Cambridge, UK, Mar. 1994.

[PSSD87] H.-B. Paul, H.-J. Schek, M. H. Scholl, and U. Deppisch: "Architecture and Implementation of the Darmstadt Database Kernel System", in *Proc. ACM SIGMOD Conf.*, pp. 196–207, San Francisco, CA, May 1987.

[PT85] P. Pistor and R. Traunmüller: "A Data Base Language for Sets, Lists, and Tables", Technical Report TR 85.10.004, IBM Heidelberg Research Center, Oct. 1985.

[PY94] W.-W. Pan and W.-P. Yang: "Indexing Retrievals Based on Signature Files in OODBs", Department of Computer and Information Science, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., 1994.

[RKS88] M. A. Roth, H. F. Korth, and A. Silberschatz: "Extended Algebra and Calculus for Nested Relational Databases", *ACM Trans. Database Syst.*, 13(4):389–417, Dec. 1988.

[RKS89] M. A. Roth, H. F. Korth, and A. Silberschatz: "Null Values in Nested Relational Databases", *Acta Inf.*, 26:615-642, 1989.

[Rob79] C. S. Roberts: "Partial-Match Retrieval via the Method of Superimposed Codes", *Proceedings of the IEEE*, 67(12):1624–1642, Dec. 1979.

[RS86] K. Ramamohanarao and J. A. Shepherd: "A Superimposed Codeword Indexing Scheme for Very Large Prolog Databases", in *Proc. of the Third Intl. Conf. on Logic Programming*, pp. 569–576, London, 1986.

[RS91] F. Rabitti and P. Savino: "Image Query Processing Based on Multi-level Signatures", in *ACM SIGIR '91*, pp. 305-314, Chicago, Illinois, Oct. 1991.

[SAB+89] M. Scholl, S. Abiteboul, F. Bancilhon, N. Bidoit, S. Gamerman, D. Plateau, P. Richard, and A. Verroust: "VERSO: A Database Machine Based On Nested Relations", in S. Abiteboul, P. C. Fischer, and H.-J. Schek eds., *Nested Relations and Complex Objects in Databases*, Vol. 361 of *Lecture Notes in Computer Science*, pp. 27–49, Springer-Verlag, 1989.

[Sai95] J. Saitoh: "Internal Schema Design of an ASN.1 Database System", Master's thesis, Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Nara, Japan, Mar. 1995, (in Japanese).

[Sch86] M. H. Scholl: "Theoretical Foundation of Algebraic Optimization Utilizing Unnormalized Relations", Technical Report DVSI-1986-T3, Technische Hochschule Darmstadt, Darmstadt, West Germany, Mar. 1986.

[SD85] R. Sacks-Davis: "Performance of a Multikey Access Method Based on Descriptors and Superimposed Coding Techniques", *Inform. Systems*, 10(4):391–403, 1985.

[SDR83] R. Sacks-Davis and K. Ramamohanarao: "A Two Level Superimposed Coding Scheme for Partial Match Retrieval", *Inform. Systems*, 8(4):273–280, 1983.

[SHH+95] J. Saitoh, T. Hara, K. Harumoto, M. Tsukamoto, and S. Nishio: "Design of the Data Storage for an ASN.1 Database System", *IPSJ Research Report*, 95(12):17–24, Jan. 1995, 95-DBS-101, (in Japanese).

[SK86] C. Stanfill and B. Kahle: "Parallel Free-Text Search on the Connection Machine System", *Commun. ACM*, 29(12):1229–1239, Dec. 1986.

[SM91] J. Stein and D. Maier: "Associative Access Support in GemStone", in K. R. Dittrich, U. Dayal, and A. P. Buchmann eds., *On Object-Oriented Database Systems*, Topics in Information Systems, chapter 20, pp. 323–339, Springer-Verlag, 1991.

[SPS87] M. H. Scholl, H.-B. Paul, and H.-J. Schek: "Supporting Flat Relations by a Nested Relational Kernel", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 137–146, Brighton, England, Aug. 1987.

[SPSW90] H.-J. Schek, H.-B. Paul, M. H. Scholl, and G. Weikum: "The DASDBS Project: Objectives, Experiences, and Future Prospects", *IEEE Trans. on Knowledge and Data Engineering*, 2(1):25–43, Mar. 1990.

[SS86] H.-J. Schek and M. H. Scholl: "The Relational Model with Relation-Valued Attributes", *Inform. Systems*, 11(2):137–147, 1986.

[SS89] H.-J. Schek and M. H. Scholl: "The Two Roles of Nested Relations in the DASDBS Project", in S. Abiteboul, P. C. Fischer, and H.-J. Schek eds., *Nested Relations and Complex Objects in Databases*, Vol. 361 of *Lecture Notes in Computer Science*, pp. 50–68, Springer-Verlag, 1989.

[Sta90] C. Stanfill: "Information Retrieval Using Parallel Signature Files", *IEEE Database Engineering*, 13(1):34–40, Mar. 1990.

[Sti60] S. Stiassny: "Mathematical Analysis of Various Superimposed Coding Methods", *American Documentation*, 11(2):155–169, 1960.

[Tan91] K. Tanaka: "Fundamental Concepts of Object-Oriented Databases", *Journal of IPSJ*, 32(5):500–513, May 1991, (in Japanese).

[TC83] D. Tsichritzis and S. Christodoulakis: "Message Files", *ACM Trans. Office Inf. Syst.*, 1(1):88–98, Jan. 1983.

[Teu94] J. Teuhola: "An Efficient Relational Implementation of Recursive Relationships Using Path Signatures", in *Proc. IEEE Conf. on Data Eng.*, Houston, Texas, Feb. 1994.

[TF86] S. J. Thomas and P. C. Fischer: "Nested Relational Structures", in P. C. Kanellakis ed., *Advances in Computing Research III, The Theory of Databases*, Vol. 3, pp. 269–307, JAI Press Inc., 1986.

[TR92] N. Tavakoli and A. Ray: "A New Signature Approach for Retrieval of Documents from Free-Text Databases", *Information Process & Management*, 28(2):153–163, 1992.

[TRN86] J. A. Thom, K. Ramamohanarao, and L. Naish: "A Superjoin Algorithm for Deductive Databases", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 189–196, Kyoto, Japan, Aug. 1986.

[Uni] UniSQL, Inc., 9390 Research II, Suite 200, Austin, Texas: *UniSQL/X User's Manual*, release 1.2 edition.

[VER86] J. VERSO: "VERSO: A Data Base Machine Based on Non 1NF Relations", Technical Report 523, INRIA, Apr. 1986.

[WIK94] N. Watanabe, Y. Ishikawa, and H. Kitagawa: "Evaluation of Two-level Signature Files as Set-valued Object Retrieval Facilities", in *IPSJ 49th Annual Conf.*, Sept. 1994, 6W–1, (in Japanese).

[WK95] N. Watanabe and H. Kitagawa: "Evaluation of Partitioned Signature Files for Set-valued Object Retrieval", in *IPSJ 50th Annual Conf.*, Mar. 1995, 3F–5, (in Japanese).

[WLO+85] H. K. T. Wong, H. Liu, F. Olken, D. Rotem, and L. Wong: "Bit Transposed Files", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 448–457, Stockholm, Aug. 1985.

[WW91] K.-F. Wong and M. H. Williams: "A Superimposed codeword Indexing Scheme for Handling Sets in Prolog Databases", in *Proc. Int'l. Symp. on Database Sys. for Advanced Applications*, pp. 468–476, Tokyo, Japan, Apr. 1991.

[Yao77] S. B. Yao: "Approximating Block Accesses in Database Organizations", *Commun. ACM*, 20(4):260–261, Apr. 1977.

[YCLK93] J. S. Yoo, J. W. Chang, Y. J. Lee, and M. H. Kim: "Performance Evaluation of Signature-Based Access Mechanisms for Efficient Information Retrieval", *IEICE Trans. Inf. & Syst.*, E76-D(2):179–188, Feb. 1993.

[YLK94] H.-S. Yong, S. Lee, and H.-J. Kim: "Applying Signatures for Forward Traversal Query Processing in Object-Oriented Databases", in *Tenth Intl. Conf. on Data Engineering*, pp. 518–525, Houston, Texas, Feb. 1994.

[Yos91] M. Yoshikawa: "Query Languages and Query Processing in Object-Oriented Databases", *Journal of IPSJ*, 32(5):523–531, May 1991, (in Japanese).

[ZCT93] P. Zezula, P. Ciaccia, and P. Tiberio: "Hamming Filter: A Dynamic Signature File Organization for Parallel Stores", in *Proc. Int'l. Conf. on Very Large Data Bases*, pp. 314–327, Dublin, Ireland, Aug. 1993.

[ZM90] S. B. Zdonik and D. Maier eds.: *Readings in Object-Oriented Database Systems*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[ZRT91] P. Zezula, F. Rabitti, and P. Tiberio: "Dynamic Partitioning of Signature Files", *ACM Trans. Inf. Syst.*, 9(4):336–369, Oct. 1991.