B

B+-Tree

Donghui Zhang[1], Kenneth Paul Baclawski[1], Vassilis J. Tsotras[2]
[1]Northeastern University, Boston, MA, USA
[2]University of California-Riverside, Riverside, CA, USA

Synonyms

B-tree

Definition

The B+-tree is a disk-based, paginated, dynamically updateable, balanced, and tree-like index structure. It supports the exact match query as well as insertion/ deletion operations in $O(\log_p n)$ I/Os, where n is the number of records in the tree and p is the page capacity in number of records. It also supports the range searches in $O(\log_p n + t/p)$ I/Os, where t is the number of records in the query result.

Historical Background

The binary search tree is a well-known data structure. When the data volume is so large that the tree does not fit in main memory, a disk-based search tree is necessary. The most commonly used disk-based search trees are the B-tree and its variations. Originally invented by Bayer and McCreight [2], the B-tree may be regarded as an extension of the balanced binary tree, since a B-tree is always balanced (i.e., all leaf nodes are on the same level). Since each disk access retrieves or updates an entire block of information between memory and disk rather than a few bytes, a node of the B-tree is expanded to hold more than two child pointers, up to the block capacity. To guarantee worst-case performance, the B-tree requires that every node (except the root) has to be at least half full. Because of this requirement, an exact match query, insertion or deletion operation must access at most $O(\log_p n)$ nodes, where p is the page capacity in number of child pointers, and n is the number of objects. The most popular variation of the B-tree is the B+-tree [3,4]. In a B+-tree, objects are stored only at the leaf level, and the leaf nodes are organized into a double linked list. As such, the B+-tree can be seen as an extension of the Indexed Sequential Access Method (ISAM), a static (and thus possibly unbalanced if updates take place) disk-based search tree proposed by IBM in the mid 1960's.

Foundations

Structure

The B+-tree is a tree structure where every node corresponds to a disk block and which satisfies the following properties:

- The tree is balanced, i.e., every leaf node has the same depth.
- An internal node stores a list of keys and a list of pointers. The number of pointers is one more than the number of keys. Every node corresponds to a key range. The key range of an internal node with k keys is partitioned into k+1 sub-ranges, one for each child node. For instance, suppose that the root node has exactly two keys, 100 and 200. The key range of the root node is divided into three sub-ranges $(-\infty, 100)$, $(100, 200)$ and $(200, +\infty)$. Note that a key in an internal node does not need to occur as the key of any leaf record. Such a key serves only as a means of defining a sub-range.
- A leaf node stores a list of records, each having a key and some value.
- Every node except the root node is at least half full. For example suppose that an internal node can hold up to p child pointers (and p-1 keys, of course) and a leaf node can hold up to r records. The half full requirement says any internal node (except the root) must contain at least $\lceil p/2 \rceil$ child pointers and any leaf node (except the root) must contain at least $\lceil r/2 \rceil$ records.
- If the root node is an internal node, it must have at least two child pointers.

- All the leaf nodes are organized, in increasing key order, into a double linked list.

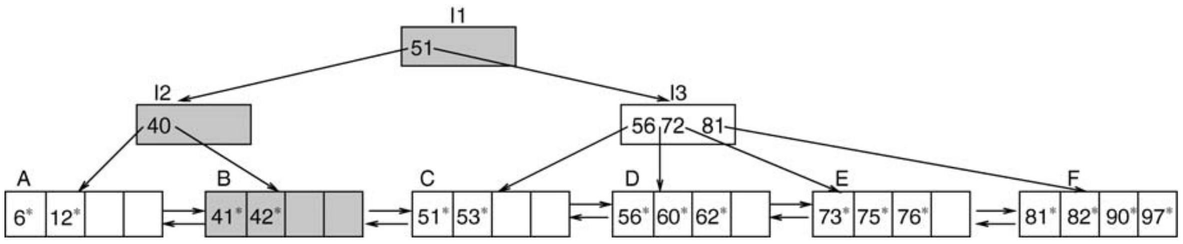
An example B+-tree is given in Fig. 1. It is assumed that every node has between two and four entries. In a leaf node, an entry is simply a record. In an internal node, an entry is a pair of (key, child pointer), where the key for the first entry is NULL. To differentiate a leaf entry (which corresponds to an actual record) from a key in an index entry, each leaf entry is followed by a "*".

Query Processing

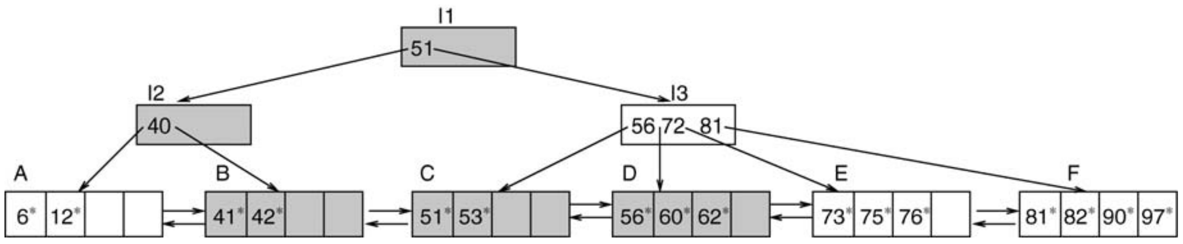
The B+-tree efficiently supports not only *exact-match queries*, which find the record with a given key, but also *range queries*, which find the records whose keys are in a given range. To perform an exact-match query, the B+-tree follows a single path from the root to a leaf. In the root node, there is a single child pointer whose key range contains the specified key. If one follows the child pointer to the corresponding child node, inside the child node there is also a single child pointer whose key range contains the desired key. Eventually, one reaches a leaf node. The desired record, if it exists, must be located in this node. As an example, Fig. 1 shows the search path if one searches for the record with key = 41. Besides exact-match queries, the B+-tree also supports range queries. That is, one can efficiently find all records whose keys belong to a range *R*. In order to do so, all the leaf nodes of a B+-tree are linked together. To search for all records whose keys are in the range *R* = [*low*, *high*], one performs an exact match query for key = *low*. This leads to a leaf node. One examines all records in this leaf node, and then follows the sibling link to the next leaf node, and so on. The algorithm stops when a record with key > *high* is encountered. An example is shown in Fig. 2.

Insertion

To insert a new record, the B+-tree first performs an exact-match query to locate the leaf node where the record should be stored, then the record is stored in the leaf node if there is enough space available. If there is not enough space, the leaf node is split. A new node is allocated, and half of the records, the ones with the larger keys in the overflowing node, are moved to the new node. A new index entry (the smallest key in the new node and a pointer to the new node) is inserted into the parent node. This may, in turn, cause



B+-Tree. Figure 1. Illustration of the B+-tree and exact-match query processing. To search for a record with key = 41, nodes I_1, I_2 and *B* are examined.



B+-Tree. Figure 2. Illustration of the range query algorithm in the B+-tree. To search for all records with keys in the range [41,60], the first step is to find the leaf node containing 41* (I_1, I_2 and *B* are examined). The second step is to follow the right-sibling pointers between leaf nodes and examine nodes *C* and *D*. The algorithm stops at *D* because a record with key >60 is found.

the parent node to overflow, and so on. In the worst case, all nodes along the insertion path are split. If the root node is split into two, a new root node is allocated and therefore the height of the tree increases by one.

As an example, Fig. 3 shows an intermediate result of inserting record 92* into Fig. 1. In particular, the example illustrates that splitting a leaf node results in a *"copy up"* operation. The result is intermediate because the parent node I_3 will also be split.

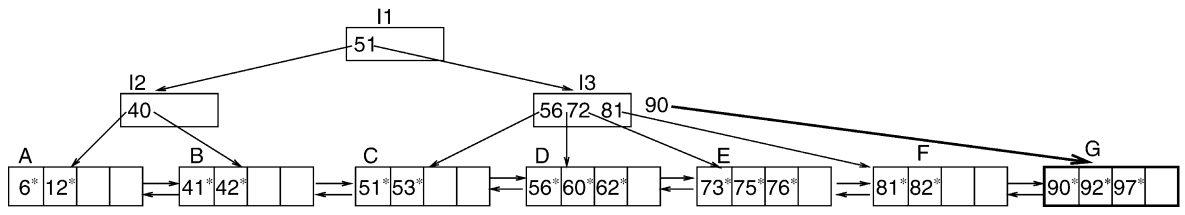
The complete result after inserting 92* is shown in Fig. 4. Here the overflowing internal node I_3 is split. In particular, the example illustrates that splitting an internal node can result in a *"push up"* operation.

Deletion

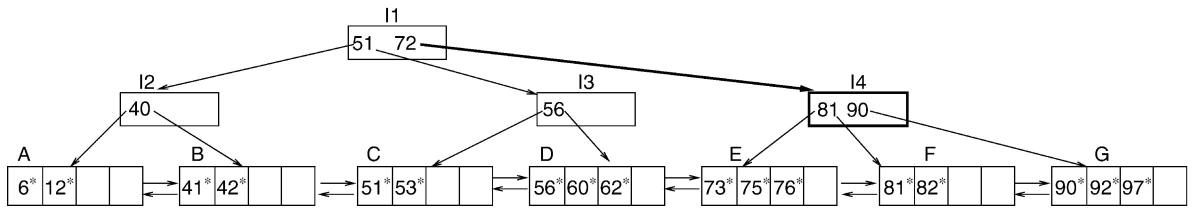
To delete a record from the B+-tree, one first uses the exact-match query algorithm to locate the leaf node that contains the record, and then the record is removed from the leaf node. If the node is at least half full, the algorithm finishes. Otherwise, the algorithm tries to re-distribute records between an immediate sibling node and the underflowing node. If redistribution is not possible, the underflowing node is merged with an immediate sibling node. Note that this merge is always possible.

As an example, Fig. 5 shows the intermediate result of deleting record 41* from the B+-tree shown in Fig. 4. Note that when merging two leaf nodes, a key in the parent node is *discarded*, which is the reverse operation of *copy up*. The result is intermediate because node I_2 is also underflowing.

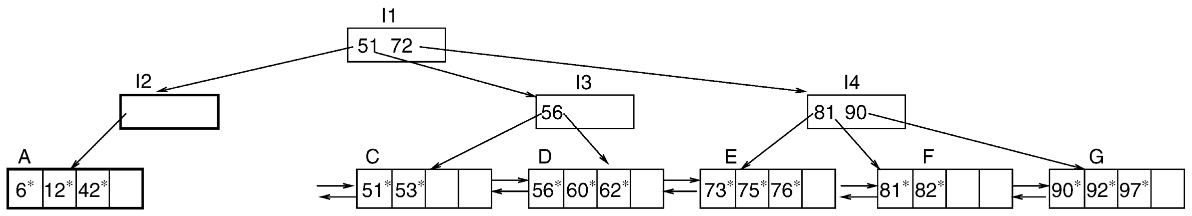
Figure 6 illustrates the final result of deleting 41*. The underflow of node I_2 is handled by merging I_2 with I_3. This merge causes key 51 from the parent node to be *dragged down* to I_2. It is the reverse operation of *push up*.



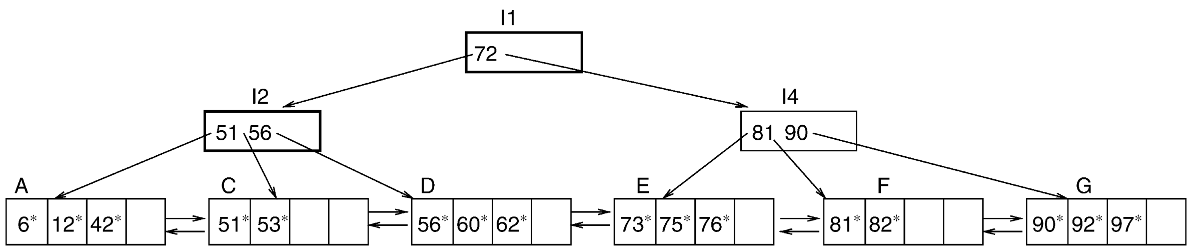
B+-Tree. Figure 3. Intermediate result of inserting record 92* into Fig. 1. The leaf node *F* is split. The smallest key in the new node *G*, which is 90, is *copied up* to the parent node.



B+-Tree. Figure 4. Continued from Fig. 3. Final result of inserting record 92*. The overflowing internal node I_3 is split. The middle key, 72, is *pushed up* to the parent node.



B+-Tree. Figure 5. Intermediate result of deleting 41* from Fig. 4. Node *B* is merged with node *A*. Key 40 (as well as the pointer to node *B*) is *discarded* from the parent node.



B+-Tree. Figure 6. Continued from Fig. 5. Final result of deleting 41*. Node I_2 is merged with node I_3. Key 51 from the parent node is *dragged down.*

Comparison with Some Other Index Structures

Compared with the B-tree, the B+-tree stores all records at the leaf level, and organizes the leaf nodes into a double linked list. This enables efficient range queries. Compared with ISAM, the B+-tree is a fully dynamic structure that balances itself nicely as records are inserted or deleted, without the need for overflow pages. Compared with external hashing schemes such as Linear Hashing and Extendible Hashing, the B+-tree can guarantee logarithmic query cost in the worst case (while hashing schemes have linear worst-case cost, although this is very unlikely), and can efficiently support range queries (whereas hashing schemes do not support range queries).

Key Applications

The B+-tree index has been implemented in most, if not all, relational database management systems such as Oracle, Microsoft SQL Server, IBM DB2, Informix, Sybase, and MySQL. Further, it is implemented in many filesystems including the NTFS filesystem (for Microsoft Windows), ReiserFS filesystem (for Unix and Linux), XFS filesystem (for IRIX and Linux), and the JFS2 filesystem (for AIX, OS/2 and Linux).

Future Directions

The impact of the B+-tree index is very significant. Many disk-based index structures, such as the R-tree [4] and k-d-B-tree [5] or their variants, are extensions of the B+-tree. Concurrency in B+-trees was studied in [6]. The Universal B-tree, which extends the B+-tree to index multi-dimensional objects, was studied in [1].

Cross-references

▶ Extendible Hashing
▶ Index Sequential Access Method (ISAM)
▶ k-d-B-Tree
▶ Linear Hashing
▶ Rtree
▶ Tree-based Indexing

Recommended Reading

1. Bayer R. The universal B-tree for multidimensional indexing: general concepts. In Proc. Int. Conf. on Worldwide Computing and Its Applications (WWCA), 1997, pp. 198–209.
2. Bayer R. and McCreight E.M. Organization and maintenance of large ordered indices. Acta Inf., 1, 1972.
3. Comer D. The ubiquitous B-tree. ACM Comput. Surv., 11(2), 1979.
4. Knuth D. The Art of Computer Programming, Vol. 3: Sorting and Searching. Addison Wesley, MA, USA, 1973.
5. Robinson J. The K-D-B tree: a search structure for large multidimensional dynamic indexes. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1981, pp. 10–18.
6. Srinivasan V. and Carey M.J. Performance of B+ tree concurrency algorithms. VLDB J., 2(4):361–406, 1993.
7. Theodoridis Y. The R-tree-portal. http://www.rtreeportal.org, 2003.

Backup

▶ Logging and Recovery

Backup and Restore

Kenichi Wada
Hitachi Limited, Tokyo, Japan

Synonyms

Backup copy

Definition

Backup is the action of collecting data stored on non-volatile storage media to aid recovery in case the original data are lost or becomes inaccessible due to corruption or damage caused by, mainly, a failure in storage component hardware (such as a hard disk drive or controller), a disastrous event, an operator's mistake, or intentional alteration or erasure of the data. In addition, the data collected by this action are called a backup. Restore is the action of copying a backup to on-line storage for use by applications in a recovery. The term "back up and restore" indicates both actions.

Key Points

The purpose of backup includes recovering data from storage media failure. For this reason, it is common to make backup to different storage media. Removable storage media is used because it is relatively less expensive than online storage media like hard disk drive, and it is easy to bring it offsite in case of site failure. Hard disk drives are becoming popular as backup storage media, because of its decreasing bit cost and quick restore capability.

There are many ways to take backup. A full backup is the way to make a full copy of data. It requires equal capacity of original data, unless data compression is not used. If full backup is taken daily from Monday to Friday and they are kept for the following week, five times capacity is required for backup.

The following two kinds of backup are used to reduce the size of backup data:

Incremental backup: A backup that copies all data modified since the last full backup. Modified data can be copied many times until next full backup is taken. To restore data when incremental backups are in use, the latest full backup and the latest incremental backup are required.

Differential backup: A backup that copies all data modified since the last backup. Modified data are not copied more than once. To restore data when differential backups are used, the latest full backup and all differential backups or all backups are required.

Cross-references

▶ Disaster Recovery
▶ Replication

Backup Copy

▶ Backup and Restore

Backup Mechanisms

▶ Replication for High Availability

Backward Recovery

▶ Crash Recovery

Bag Semantics

Todd J. Green
University of Pennsylvania, Philadelphia, PA, USA

Synonyms

Duplicate Semantics; Multiset Semantics

Definition

In the ordinary relational model, relations are sets of tuples, which by definition do not contain "duplicate" entries. However, RDBMSs typically implement a variation of this model where relations are *bags* (or *multisets*) of tuples, with duplicates allowed. Formally, a bag is a mapping of tuples to natural number *multiplicities*; a set can be viewed as a special case of a bag where all tuple multiplicities are 0 or 1. The operations of the relational algebra are extended to operate on bags by defining their action on tuple multiplicities. RDBMSs based on bags rather than sets are said to implement *bag semantics* (rather than *set semantics*). Duplicates may occur at multiple levels: in source relations, in materialized views, or in query answers. A variation of bag semantics called *bag-set semantics* is obtained by requiring source relations to be sets, while allowing views and query answers to contain duplicates. Bag-set semantics represents a practical compromise between bag semantics and set semantics.

Historical Background

In the ordinary relational model, relations are sets of tuples, which by definition do not contain "duplicate" entries. However, RDBMS implementations, beginning with System R and INGRES, deviated from the "pure" relational model by allowing duplicate tuples in query answers, thus making bags (multisets), rather than sets, the primary collection type in query processing. The primary motivation for allowing duplicate tuples was the need to avoid expensive duplicate elimination in query processing. Instead, duplicate elimination would only be performed if explicitly requested by the user, e.g., via the SQL `DISTINCT` keyword. In their use of bag relations, RDBMS implementations were ahead of theory, and the precise semantics of relational algebra on bags was not studied until the 1980s [6,11,12]. Klug [12] pointed out the failure of certain algebraic identities under bag semantics, and also gave a precise semantics of aggregate functions that did not depend on bag semantics. Dayal et al. initiated the study of query optimization under bag semantics in [6]. Bag containment of unions of conjunctive queries (UCQs) was shown to be undecidable by Ioannidis and Ramakrishan [9], while bag equivalence of conjunctive queries (CQs) was shown to be decidable, in fact the same as isomorphism, by Chaudhuri and Vardi [3]. The $\Pi_2{}^p$-hardness of checking bag containment of CQs was also established in [3], but the decidability of the problem remains open (see Future Directions). The same paper introduced the terminology "bag-set semantics" to describe the semantics obtained by requiring source relations to be sets while allowing views and query answers to contain duplicates, and established the decidability of bag-set equivalence of CQs. For conjunctive queries with inequalities ($\mathrm{CQ}^{\neq}$), bag containment and bag-set containment were both shown to be undecidable by Jayram et al. [10]. Bag equivalence of UCQs was shown to be decidable (and like CQs, the same as isomorphism) by Cohen et al. [5]. Cohen [4] studied query equivalence for a generalization of bag-semantics called *combined semantics*, which captures user-specified elimination of duplicates at intermediate stages of query processing. A bag semantics for Datalog was proposed by Mumick et al. [14], and Mumick and Shmueli [15] studied computational problems related to infinite multiplicities (which may occur in query answers because of recursion). Going beyond the relational model, several papers have studied bag semantics in the context of query languages for nested relations; see [2,8] and references therein.

Foundations

Bag relational algebra. For simplicity, the definitions here assume the unnamed perspective [1] of the relational model. Fix a countable domain $\mathbb{D}$ of database values. A *bag relation* of arity k is a mapping R: $\mathbb{D}^k \to \mathbb{N}$ associating *tuples* with their *multiplicities.* Conceptually, a tuple not present in the bag relation is mapped to zero. It is typically required that bag relations have *finite support*, i.e., R is zero on all but finitely many tuples. A *duplicate tuple* is a tuple whose multiplicity is greater than 1. From the perspective of bag semantics, an ordinary set relation of arity k is a mapping $R : \mathbb{D}^k \to \{0,1\}$ (again of finite support), in other words, a bag relation with no duplicate tuples. The relational algebra can be extended to bag relations by defining the action of the relational operators on tuple multiplicities:

Selection. If R is a bag relation of arity k and the selection predicate P maps each k-tuple to either 0 or 1 then $\sigma_P R$ is the bag relation of arity k defined by

$$(\sigma_P R)(t) \overset{\text{def}}{=} R(t) \cdot P(t).$$

Projection. If R is a bag relation of arity k and $V = (v_1,\ldots,v_n)$ is a list of indices, $1 \le v_i \le k$, then π_V R is a bag relation of arity n, defined by

$$(\pi_V R)(t) \overset{\text{def}}{=} \sum_{t' \text{s.t.} t' = \pi_V(t)} R(t').$$

Cross product. If R_1 is a bag relation of arity k_1 and R_2 is a bag relation of arity k_2, then $R_1 \times R_2$ is a bag relation of arity $k_1 + k_2$, defined by

$$(R_1 \times R_2)(t) \overset{\text{def}}{=} R_1(t_1) \cdot R_2(t_2),$$

where t is a $(k_1 + k_2)$-tuple obtained by concatenating t_1 and t_2.

Union. If R_1 and R_2 are bag relations of arity k, then R_1 $\cup$ R_2 is a bag relation of arity k, defined by

$$(R_1 \cup R_2)(t) \overset{\text{def}}{=} R_1(t) + R_2(t).$$

Intersection. If R_1 and R_2 are bag relations of arity k, then $R_1 \cap R_2$ is a bag relation of arity k, defined by

$$(R_1 \cap R_2)(t) \overset{\text{def}}{=} \min(R_1(t), R_2(t)).$$

Difference. If R_1 and R_2 are bag relations of arity k, then $R_1 - R_2$ is a bag relation of arity k, defined by

$$(R_1 - R_2)(t) \overset{\text{def}}{=} \max(R_1(t) - R_2(t), 0),$$

in other words, the calculation uses *proper subtraction* of natural numbers.

Example 1.

$$R = \begin{array}{|cc|c|} \hline a & b & 2 \\ c & b & 3 \\ c & d & 1 \\ \hline \end{array}\,, \quad S = \begin{array}{|cc|c|} \hline b & c & 5 \\ b & d & 1 \\ d & d & 2 \\ \hline \end{array}\,,$$

$$R \circ S = \begin{array}{|cc|c|} \hline a & c & 2 \cdot 5 = 10 \\ a & d & 2 \cdot 1 = 2 \\ c & c & 1 \cdot 5 = 5 \\ c & d & 3 \cdot 1 + 1 \cdot 2 = 5 \\ \hline \end{array}$$

where $R \circ S$ *denotes relational composition, i.e.,* $R \circ S \overset{\text{def}}{=} \pi_{1,4}\, \sigma_{2=3}(R \times S)$.

Example 2.

$$R = \begin{array}{|cc|c|} \hline a & b & 1 \\ c & b & 1 \\ c & d & 1 \\ \hline \end{array}\,, \quad S = \begin{array}{|cc|c|} \hline b & c & 1 \\ b & d & 1 \\ d & d & 1 \\ \hline \end{array}\,,$$

$$\pi_{1,3}\ (R \times S) = \begin{array}{|cc|c|} \hline a & b & 1 \cdot 1 + 1 \cdot 1 = 2 \\ a & d & 1 \cdot 1 = 1 \\ c & b & 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 4 \\ c & d & 1 \cdot 1 + 1 \cdot 1 = 2 \\ \hline \end{array}$$

Note that the use of projection results in duplicate tuples in the query output, even though the source tables R and S are duplicate-free. Duplicates may also be introduced by the union operator.

Bags and SQL. Practical query languages such as SQL are based on bag semantics rather than set semantics, since eliminating duplicates typically requires an expensive sort. However, SQL provides a way to emulate set semantics by the use of an explicit *duplicate elimination* operator, which is specified using the `DISTINCT` keyword. This operator converts a bag relation into the corresponding set relation. For example, to execute the SQL query

```
SELECT DISTINCT R.A, S.C
FROM R, S
WHERE R.B = S.B
```

the RDBMS will compute the bag join of R and S, then eliminate duplicates to produce a set relation. The result is the same as the join of R and S under set semantics.

SQL also provides alternate versions of the union, intersection, and difference operators which differ in their handling of duplicates. In particular, `UNION ALL`, `INTERSECT ALL`, and `EXCEPT ALL` correspond to the bag operations as defined above, i.e., they retain duplicates and produce bag relations as output. Meanwhile, `UNION`, `INTERSECT`, and `EXCEPT` perform duplicate elimination and produce set relations as output. For `UNION`, duplicate elimination is performed after the bag union, while for `INTERSECT` and `EXCEPT`, the duplicate elimination is performed beforehand on the operands.

Query reformulation with bag semantics. The context of bag semantics poses unique challenges in query reformulation and optimization, as classical optimization techniques such as query minimization do not necessarily transfer to bag semantics. For example, under set semantics, the query $R \bowtie R$ can be minimized by removing the redundant self-join to produce the equivalent query R. However, under bag semantics, the resulting query is not equivalent to the original, as the "redundant" self-join may increase the multiplicity of some output tuples. The rest of this section summarizes some of the theoretical results which are known regarding containment and equivalence of queries under bag semantics. In studying query reformulation, it is convenient to assume queries are expressed in a Datalog-style syntax. Thus, the algebraic query $R \bowtie R$ corresponds to the CQ

$$Q(x, y)\colon -R(x, y), R(x, y),$$

while the query R corresponds to the CQ

$$Q'(x, y)\colon -R(x, y).$$

As the example illustrates, repetitions of an atom in the body of a CQ are significant. As with set semantics, the bag semantics of CQs is defined in terms of *valuations* which assign the variables in the CQ values from the domain $\mathbb{D}$. The multiplicity of an output tuple is computed by summing over all valuations which produce that output tuple, and for each valuation, taking the product of the source multiplicities of the tuples corresponding to the body. Thus, for a CQ

$$Q(\bar{x})\colon -A_1(\bar{z}_1), \ldots, A_n(\bar{z}_n),$$

the multiplicity of an output tuple t is given by

$$Q(t) \overset{\text{def}}{=} \sum_{v} \prod_{i=1}^{n} A_i(v(\bar{z}_i)), \qquad (1)$$

where the sum is over valuations $v : \text{vars}(Q) \to \mathbb{D}$ such that $v(\bar{x}) = t$. Note that this definition agrees with the bag relational algebra definition for SPJ queries (using selection, projection, and cross product).

Example 3. *The relational algebra query from Example 1 corresponds to the CQ*

$$Q(x, z) : -R(x, y), S(y, z).$$

Applying Q to the bag relations R and S from Example 1 yields $Q(c,d) = 3 \cdot 1{+}1 \cdot 2{=}5$, *where the* $3 \cdot 1$ *is produced by the valuation v which sends* $x \mapsto c, y \mapsto b$, *and* $z \mapsto d$, *and the* $1 \cdot 2$ *is produced by the valuation* v' *which sends* $x \mapsto c, y \mapsto d$, *and* $z \mapsto d$.

For a UCQ $\bar{Q} = (Q_1,\ldots,Q_n)$ the multiplicity of an output tuple is computed by summing over all the CQs in $\bar{Q}$:

$$\bar{Q}(t) \overset{\text{def}}{=} \sum_{i}^{n} Q_i(t). \tag{2}$$

This definition agrees with the bag relational algebra definition for SPJU queries (using selection, projection, cross product, and union).

Example 4. *Applying the UCQ* Q' *defined by the CQs*

$$Q'(x, z) : -R(x, y), S(y, z)$$

$$Q'(z, y) : -R(x, y), S(x, z)$$

to the set relations R and S from Example 2, the multiplicity of (c, d) *in the output is* $1 \cdot 1 + 1 \cdot 1 = 2$, *with each of the CQs contributing a term to the sum.*

A query P is *bag-contained* in a query Q, denoted $P \sqsubseteq_b Q$, if for all bag instances I, for every output tuple t, $P(I)(t) \leq Q(I)(t)$. When $P \sqsubseteq_b Q$ and $Q \sqsubseteq_b P$, P and Q are said to be *bag-equivalent*, denoted $P \equiv_b Q$. The analogous notions of set-containment and set-equivalence are denoted $P \sqsubseteq_s Q$ and $P \equiv_s Q$, respectively. With set semantics, it is well-known that a CQ Q can be optimized by deleting atoms in the body of Q, producing a minimal *core* of Q which is set-equivalent to Q. However, under bag semantics, this optimization no longer works, as deleting an atom in the body of a CQ always produces an inequivalent query, in fact:

Theorem 1 ([3]). *For CQs P,Q,* $P \equiv_b Q$ *iff* $P \cong Q$.

Here, $P \cong Q$ denotes that P and Q are *isomorphic*. Two CQs P and Q are isomorphic if there is a (bijective) renaming of variables $\theta : \text{vars}(P) \to \text{vars}(Q)$ which transforms P into Q. Checking isomorphism of directed graphs (i.e., Boolean CQs over a single binary predicate), is known to be in *NP*, but is not known or believed to be either in *P* or *NP*-complete. This holds also for general relational structures (and arbitrary CQs). Thus, although there are fewer opportunities for optimization of CQs under bag semantics, checking bag-equivalence is presumably computationally easier than checking set-equivalence (which is known to be *NP*-complete). It turns out that Theorem 1 can be generalized to UCQs:

Theorem 2 ([5]). *For UCQs* $\bar{P}$, $\bar{Q}$, $\bar{P} \equiv_b \bar{Q}$ *iff* $\bar{P} \cong \bar{Q}$.

Bag-equivalence of unions of conjunctive queries with inequalities (UCQ$^{\neq}$s) is also known to be decidable:

Theorem 3 ([5]). For *UCQ*$^{\neq}$ *s P, Q, checking* $P \equiv_b Q$ *is in PSPACE.*

The same result was obtained earlier for CQ$^{\neq}$s by Nutt et al. [16]. The exact complexity in each case (CQ$^{\neq}$s or UCQ$^{\neq}$s) seems to be open. (Theorem 1 implies a graph isomorphism-hard lower bound).

In contrast to equivalence, which seems to only become easier to check for bag semantics than for set semantics, containment can become much harder. For CQs, the following lower bound on the complexity of bag containment is known:

Theorem 4 ([3]). For *CQs P, Q, checking* $P \sqsubseteq_b Q$ *is* $\Pi_2{}^P$*-hard.*

However, this is only a lower bound, and as of 2008, it is not known whether the problem is even decidable. For UCQs, the question has been resolved negatively:

Theorem 5 ([9]). For *UCQs* $\bar{P}, \bar{Q}$, *checking* $\bar{P} \sqsubseteq_b \bar{Q}$ *is undecidable.*

The reduction used to prove this result highlights a close connection between checking bag containment of UCQs and Hilbert's Tenth Problem, which concerns checking for the existence of solutions to Diophantine equations. Another (non-trivial) reduction from the same problem was used to establish a similar result for CQ$^{\neq}$s:

Theorem 6 ([10]). *For CQ*$^{\neq}$ *s P, Q, checking* $P \sqsubseteq_b Q$ *is undecidable.*

Query reformulation with bag-set semantics. Recall that with bag-set semantics, all source tuples are assumed to have cardinality 1 (or 0, if not present). For query optimization, the main ramification compared to bag semantics is that redundant (repeated) atoms in the body of a CQ are immaterial under bag-set semantics and can be simply deleted from the body, producing an *irredundant* CQ. For example, the CQs

$$Q_1(x,z) \colon - R(x,y), S(y,z)$$

$$Q_2(x,z) \colon - R(x,y), R(x,y), S(y,z)$$

are bag-set equivalent, denoted $Q_1 \equiv_{bs} Q_2$, and Q_1 is irredundant. The following result states that eliminating repeated atoms is essentially the only optimization possible for CQs under bag-set semantics:

Theorem 7 ([3]). *For irredundant CQs P, Q, $P \equiv_{bs} Q$ iff $P \equiv_b Q$ (hence $P \equiv_{bs} Q$ iff $P \cong Q$).*

This result was essentially a rediscovery of a well-known result in graph theory due to Lovász [13], who showed that for finite relational structures F, G, if $|\mathrm{Hom}(F,H)| = |\mathrm{Hom}(G,H)|$ for all finite relational structures H, where $\mathrm{Hom}(A,B)$ is the set of homomorphisms $h : A \to B$, then $F \cong G$. In database terminology, this says that bag-set equivalence of irredundant Boolean CQs is the same as isomorphism.

Theorem 7 was extended to UCQs in [5], and the PSPACE upper bound of Theorem 3 was also shown to hold for bag-set equivalence of UCQ$^{\neq}$s in [5].

In general, results on bag-set semantics correspond to results on bag semantics via the following *transfer lemma*:

Lemma 1 ([7]). *There exists a mapping $\varphi : CQ \to CQ$ (which extends to UCQs by applying it component-wise on CQs), a mapping f from bag instances to set instances, and a mapping g from set instances to bag instances, such that for any UCQ $\bar{Q}$, bag instance I, and set instance J:*

1. $\bar{Q}(I) = \varphi(\bar{Q})(f(I))$
2. $\varphi(\bar{Q})(J) = \bar{P}(g(J))$

In particular this lemma shows that bag-containment of CQs (UCQs) is polynomial time reducible to bag-set containment, as shown for CQs in [3]. Thus Theorem 5 also implies that bag-set containment of UCQs is undecidable. Also, Lemma 1 generalizes to queries with inequality predicates, hence Theorem [10] also implies also the undecidability of bag-set containment of CQ$^{\neq}$s, as shown in [10]. Finally, note that for UCQs $\bar{P}, \bar{Q}$ the transformation ϕ can be defined such that $\bar{P} \cong \bar{Q}$ iff $\varphi(\bar{P}) \cong \varphi(\bar{Q})$. Theorem 2 thus follows from a similar result for bag-set equivalence, also shown in [5]:

Theorem 8 ([5]). *For unions of irredundant CQs $\bar{P}, \bar{Q}$, we have $\bar{P} \equiv_{bs} \bar{Q}$ iff $\bar{P} \cong \bar{Q}$.*

Bag semantics of datalog queries. As with the relational algebra, the semantics of Datalog queries can be extended operate on bag instances. This is done by defining how *derivation trees* for an output tuple contribute to the multiplicity of the tuple. The *fringe* of a derivation tree τ of an output tuple t, denoted $\mathrm{fringe}(\tau)$, is the bag of leaves (source tuples) in the derivation tree. The count of an output tuple is computed by summing over all derivation trees, and taking the product of the multiplicities of the source tuples in the fringe. If a source tuple appears several times in the fringe, it is counted that many times in the product. Formally, for a Datalog program Q and source bag instance I, the multiplicity of a tuple t in the output is defined by

$$Q(t) \overset{\mathrm{def}}{=} \sum_{\tau} \prod_{A(t') \in \mathrm{fringe}(\tau)} A(t') \tag{3}$$

where the sum is over all derivation trees for t. Note that when Q is a CQ or UCQ, the above definition agrees with definitions (1.1) and (1.2). However, a basic difference is that for recursive queries, there may be *infinitely many* derivation trees for an output tuple. In this case, its count is defined to be ∞.

Example 5. *For the transitive closure query TC and bag relation R defined by*

$$\begin{array}{lcl} TC(x,y) & :- & R(x,y) \\ TC(x,z) & :- & TC(x,y), R(y,z) \end{array} \qquad R = \begin{array}{ccc} a & b & 1 \\ a & c & 2 \\ c & b & 1 \\ c & d & 1 \\ d & d & 3 \end{array}$$

evaluating the query yields $TC(a,\ b) = 1 + 2 \cdot 1 = 3$ but $TC(c,\ d) = \infty$.

Computational problems related to infinite counts are therefore of central interest for Datalog programs with bag semantics. The most basic problem is to check whether or not a count for a given output tuple is ∞. Clearly, computing the set of all derivation trees, and then checking whether or not the set is finite, is not feasible. However, it turns out that the problem is decidable in polynomial time (data complexity), even for Datalog extended with safe stratified negation [15]. The related problem of checking statically whether answer counts are finite for all possible source bag instances is also decidable, even for Datalog extended with and negation on unary edb's [15]. However, checking this property is undecidable for Datalog extended with safe stratified negation [15].

Key Applications

The use of bag semantics in practical RDBMS implementations has led to a reexamination of fundamental issues in query optimization, namely, query containment and query equivalence. The switch from classical set semantics to bag semantics turns out to have a radical impact on these issues. Bag semantics also poses challenges for processing of recursive Datalog programs, where infinite multiplicities may arise in the output.

Future Directions

The most salient open problem involving bag semantics concerns containment of conjunctive queries. This was shown to be Π_2^p-hard in [3], but the decidability of the problem remains open. In contrast, for set semantics, the problem is known to be *NP*-complete.

Cross-references

▶ Bag Semantics
▶ Data Models
▶ Duplicate Elimination
▶ Multiset Semantics

Recommended Reading

1. Abiteboul S., Hull R., and Vianu V. Foundations of Databases. Addison-Wesley, Reading, MA, 1995.
2. Buneman P., Naqvi S., Tannen V., and Wong L. Principles of programming with complex objects and collection types. Theor. Comput. Sci., 149(1):3–48, 1995.
3. Chaudhuri S. and Vardi M.Y. Optimization of *real* conjunctive queries. In Proc. 12th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1993.
4. Cohen S. Equivalence of queries combining set and bag-set semantics. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006, pp. 70–79.
5. Cohen S., Nutt W., and Serebrenik A. Rewriting aggregate queries using views. In Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1999.
6. Dayal U., Goodman N., and Katz R.H. An extended relational algebra with control over duplicate elimination. In Proc. 1st ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1982, pp. 117–123.
7. Green T.J. Containment of conjunctive queries on annotated relations. In Proc. 12th Int. Conf. on Database Theory, 2009.
8. Grumbach S., Libkin L., Milo T., and Wong L. Query languages for bags: Expressive power and complexity. SIGACT News, 1996, p. 27.
9. Ioannidis Y.E. and Ramakrishnan R. Containment of Conjunctive Queries: Beyond Relations as Sets. ACM TODS, 20 (3):288–324, 1995.
10. Jayram T.S., Kolaitis P.G., and Vee E. The containment problem for *real* conjunctive queries with inequalities. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006.
11. Klausner A. and Goodman N. Multirelations – semantics and languages. In VLDB, 1985.
12. Klug A.C. Equivalence of relational algebra and relational calculus query languages having aggregate functions. J. ACM, 29(3):699–717, 1982.
13. Lovász L. Operations with structures. Acta Math. Hungarica, 18(3–4):321–328, 1967.
14. Mumick I.S., Pirahesh H., and Ramakrishnan R. The magic of duplicates and aggregates. In Proc. 16th Int. Conf. on Very Large Data Bases, 1990, pp. 264–277.
15. Mumick I.S. and Shmueli O. Finiteness properties of database queries. In Proc. 4th Australian Database Conf. Brisbane, Australia, February 1993.
16. Nutt W., Sagiv Y., and Shurin S. Deciding equivalences among aggregate queries. In Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1998, pp. 214–223.

Bagging

Wei Fan[1], Kun Zhang[2]
[1]IBM T.J. Watson Research, Hawthorne, NY, USA
[2]Xavier University of Louisiana, New Orleans, LA, USA

Synonyms

Bootstrap aggregating

Definition

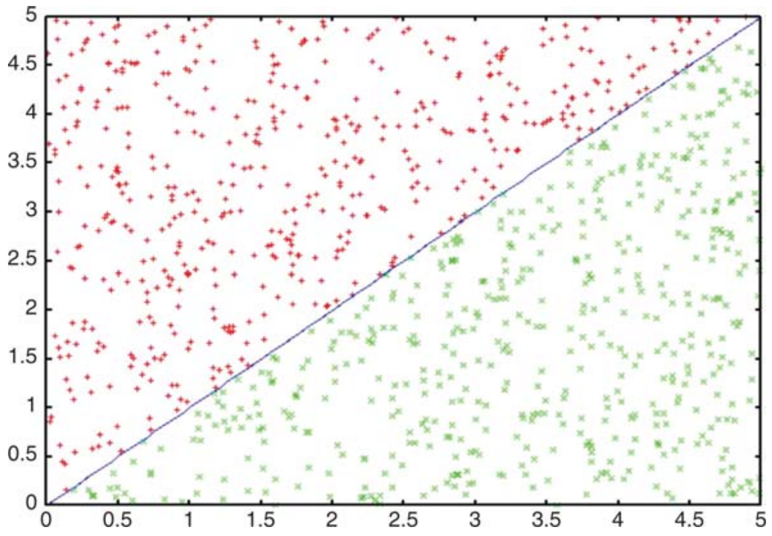
Bagging (**B**ootstrap **Agg**regating) uses "majority voting" to combine the output of different inductive models, constructed from bootstrap samples of the same training set. A bootstrap has the same size as the training data, and is uniformly sampled from the original training set with replacement. That is, after an example is selected from the training set, it is still kept in the training set for subsequent sampling and the same example could be selected multiple times into the same bootstrap sample. When the training set is sufficiently large, on average, a bootstrap sample has 63.2% unique examples from the original training set, and the rest are duplicates. In order to make full use of bagging, typically, one need to generate at least 50 bootstrap samples and construct 50 classifiers using these samples. During prediction, the class label receiving the most votes or most predictions from the base level 50 classifiers will be the final prediction. Normally, Bagging is more accurate than a single classifier trained from the original training set. Statistically, this is due to multiple classifiers' power to reduce the effect of overfitting on the training data and statistical variance. Bagging works the best for non-stable inductive models, such as decision trees, and its advantage is limited for stable methods such as logistic regression, SVM, naive Bayes, etc.

Historical Background

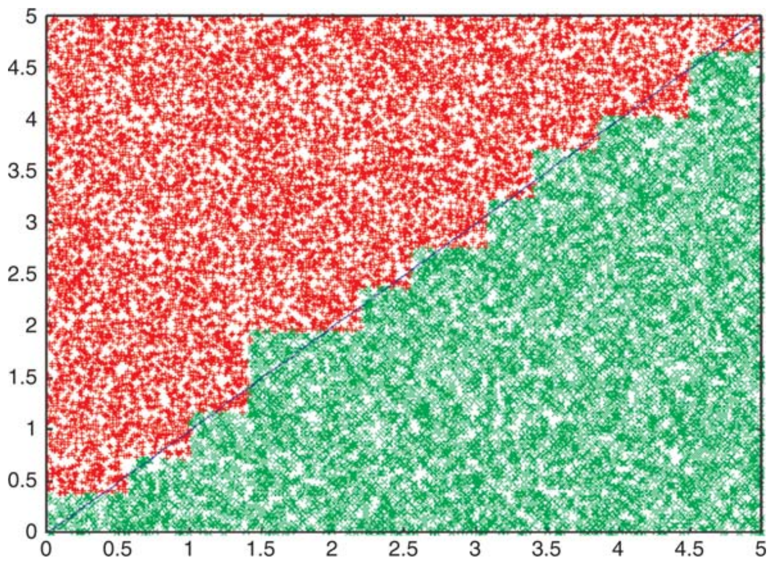
Bagging was originally proposed by Leo Breiman in 1996 [3]. The motivation comes from the fact that unstable inductive learners such as decision trees tend to generate very different models with just a slight change in the training data, for example, the inclusion or exclusion of one example. This is due to the "greedy-based" search heuristics of these methods to find the best hypothesis to fit the labeled training data. Statistically, the effect is that the variance in bias and variance decomposition of error is large. Bias is the systematic error of the chosen learning technique, and variance is due to variations in the training data, i.e., different training data produce different models. Bagging was proposed to remedy these problems. In order to overcome variations of a single training set, multiple bootstrap samples are used to replace the single original training set. Since each bootstrap sample is randomly selected from the original training data, it still "preserves" the main concept to be modeled. However, each bootstrap sample is different, thus offsetting the "individuality" of any single training set that contributes to variance. In the same time, any possible over-sample or "non-sample" of particular labeled examples from the training data (recall that 63.2% are unique examples and the rest are duplicates) that could reduce learning accuracy is resolved by majority voting or choosing the predicted labels with the most number of votes from base line models. The simple observation is that the probability for most models to make the same mistakes is much less than any single model itself.

Foundations

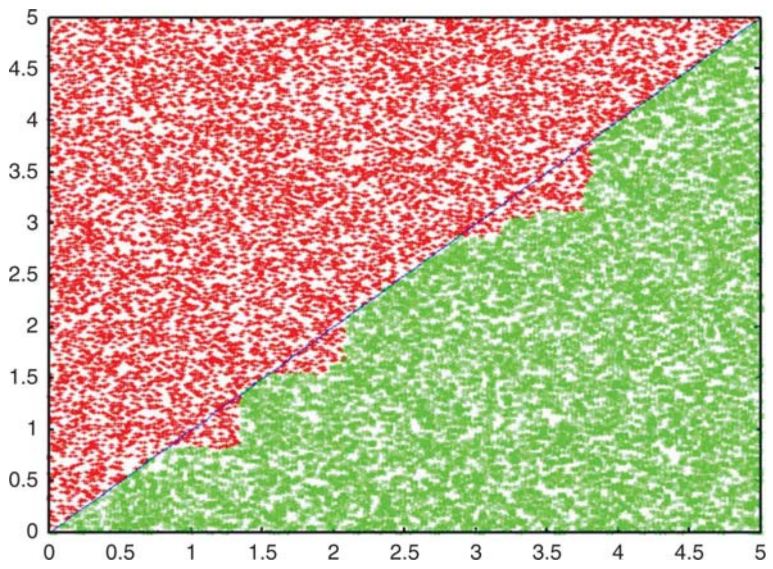
The key idea of Bagging can be illustrated by a simple synthetic example. In Fig. 1, the true decision boundary that separates two classes of examples is a simple linear function, $y = x$, where examples above the line belongs to one class and examples below the line belongs to another class. The labeled training examples are randomly sampled from the universe of examples and is obviously not exhaustive. When a decision tree algorithm is applied to the sampled training data, it will construct a tree model particular to that sample set. Figure 2 shows the decision boundary of a single tree constructed by C4.5 algorithm [12]. As observed, the "stair-step" shape of the decision boundary is due to "bias" or systematic error of decision tree algorithm, where the splits are always perpendicular to the axes. Once an algorithm is chosen, its bias is hard to avoid. At the same time, the exact position and size of each step is due to variations in individually sampled training data, and it contributes to "variance" term in the prediction error. Next, this entry studies how "variance" can be effectively reduced by applying Bagging. Figure 3 demonstrated the decision boundary of Bagging generated from 50 bagged C4.5 trees. It is clear



Bagging. Figure 1. Training data for a simple linear function.



Bagging. Figure 2. Decision boundary constructed by a single tree.



Bagging. Figure 3. Decision boundary constructed by bagging.

that the decision boundary is much closer to the perfect "$y = x$" line, and only three steps are left.

Formally, the reduction in variance can be approximated by the following equation:

$$\epsilon = \beta + \frac{\sigma}{\sqrt{n}}, \tag{1}$$

where the error ε is decomposed into bias β and variance σ. The reduction in variance is scaled by $\sqrt{n}$ where n is the number of bagging models. Additionally, the idea of Bagging can be applied to both classification problem or discrete variable prediction (such as the synthetic example given above) and regression problems or continuous variable prediction. In regression problems, the estimated value of different models are averaged as the final prediction.

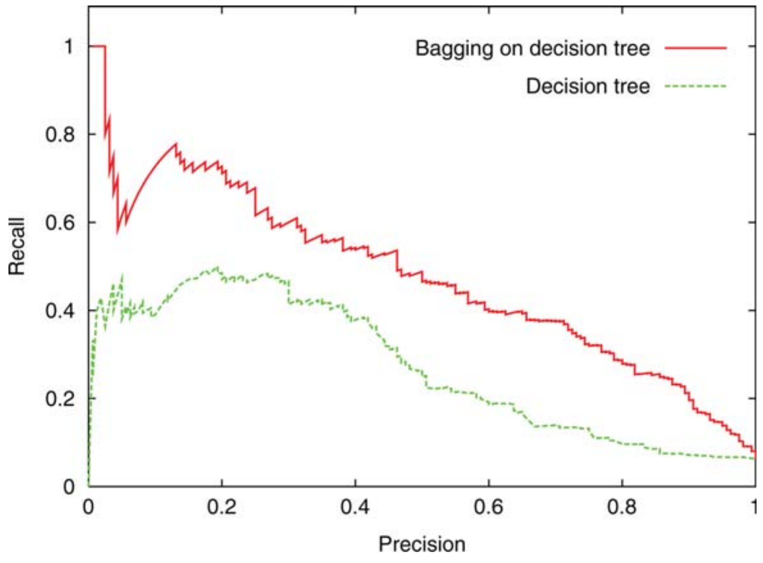
For further reading and understanding of Bagging and its various applications, refer to the 15 papers in the "Recommended Reading" section. For theories to explain how and why Bagging works, the best source is the original paper by Leo Breiman [3], that explains the statistical fundamentals that Bagging is built upon. Additionally, [5] explains the appropriate understanding of overfitting and how overfitting plays a role in classifers' generalization power. In essence, Bagging helps to correct the overfitting problem of unstable learners, such as decision trees. For various information on how to build an accurate decision tree, the base model where Bagging normally combines, the following list of papers contain solid information [2,4,9,11,12,13]. For state-of-the-art and future works of Bagging, one of the most novel ideas is "randomization" and different treatment on this subject can be found in various papers [1,6,7,14]. During the time when Bagging was proposed, an alternative method called "Boosting" was proposed to resolve the instability of inductive learners and some papers that theoretically describe its motivation and practice can be found in [8,10]. For a detailed and systematic comparison of Bagging and many other state-of-the art algorithms on a difficult application problem "skewed and stochastic ozone day forecasting", the works in [15] contain useful information.

Key Applications

Bagging is most suitable for applications where both the training set is not too big, i.e., the number of training examples can fit into main memory of the machine, and the chosen inductive learner is unstable, such as decision trees. This is mainly because generating multiple bootstrap sample involves multiple scans of the training data and can be a bottleneck if most operations rely on disk. At the same time, if the number of examples does not fit into main memory of the machine, learning can incur swapping cost and take a long time to complete. As discussed previously, bagging's success is mainly limited to unstable learners that normally have large variance.

As a good example, "ozone day prediction" [15] illustrates how well Bagging can perform in practice. What makes this problem interesting is that there are 72 continuous features in the collected dataset, only about 3% of 2,500+ collected days are positive. It is important to understand that for a problem with 72 dimensions, 2,500 training examples are trivial in size. Even if these 72 features are binary in values, the total problem size is an astronomical number 2^{72}, and obviously, 2,500 samples from this space is "really nothing".

Applying non-stable inductive learning method, such as decision tree, on this problem is unlikely to obtain satisfactory result. The simple reason is that these methods tend to "overfit" or build unnecessarily detailed models to fit well on the given 2,500 examples, but do not generalize well on unseen testing data in the problem space of 2^{72}. As illustrated in a precision-recall plot in Fig. 4, Bagging consistently obtains higher "precision" than its comparable single tree methods. Recall is the percentage of "true ozone days" that are correctly predicted as ozone days, and precision is the



Bagging. Figure 4. Bagging versus single decision tree.

percentage of "predicted ozone days" that are actually ozone days. By reading the consistently higher precision numbers by Bagging on the same recall number, it clearly demonstrates that Bagging is more accurate than single decision trees.

Future Directions

Most recently, a completely counter-intuitive method called "Random Decision Tree" [7,6,14] has been proposed to reduce both bias and variance in inductive learning. Each tree in a random tree ensemble is used as a structure to summarize the training data, and importantly, the structure of the tree is semi-randomly constructed, and data are "filled-in" the tree structure to summarize statistics of the training data. During prediction, the estimated probability from each tree is averaged as the final conditional probability $P(\mathbf{y}|\mathbf{x})$ and the class label with the highest probability is chosen as the predicted label. Since it does not incur any overhead of producing bootstraps or using information gain to perform feature selection as Bagging does, random decision tree can be highly efficient. One application on difficult skewed and high dimensional ozone day forecast can be found in [15]. Similar to Bagging, Random Decision Tree is applicable to both classification and regression problems.

Cross-references

► Boosting
► Decision Tree

Recommended Reading

1. Amit Y. and Geman D. Shape quantization and recognition with randomized trees. Neural Comput., 9(7):1545–1588, 1997.
2. Bradford J.P., Kunz C., Kohavi R., Brunk C., and Brodley C.E. Pruning decision trees with misclassification costs. In Proc. Eur. Conf. Mach. Learn., 131–136, 1998.
3. Breiman L. Bagging predictors. Mach. Learn., 24(2):123–140, 1996.
4. Buntine W. Learning classification trees. In Artificial Intelligence frontiers in statistics, D.J. Hand (ed.). Chapman & Hall, London, 1993, pp. 182–201.
5. Domingos P. Occam's two razors: The sharp and the blunt. In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, 1998.
6. Fan W., Greengrass E., McCloskey J., Yu P.S., and Drummey K. Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches. In Proc. IEEE Int. Conf. on Data Mining, 2005, pp. 154–161.
7. Fan W., Wang H., Yu P.S., and Ma S. Is random model better? on its accuracy and efficiency. In Proc. 19th Int. Conf. on Data Engineering, 2003.
8. Freund Y. and Schapire R. A decision-theoretic generalization of on-line learning and an application to boosting. Comput. Syst. Sci., 55(1):119–139, 1997.
9. Gehrke J., Ganti V., Ramakrishnan R., and Loh W.-Y. BOAT-optimistic decision tree construction. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999.
10. Kearns M. and Mansour Y. On the boosting ability of top-down decision tree learning algorithms. In Proc. Annual ACM Symp. on the Theory of Computing, 1996, pp. 459–468.
11. Mehta M., Rissanen J., and Agrawal R. MDL-based decision tree pruning. In Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining, 1995, pp. 216–221.
12. Quinlan R. C4.5: Programs for Machine Learning. Morgan Kaufmann, Los Altos, CA, 1993.
13. Shawe-Taylor J. and Cristianini N. Data-dependent structural risk minimisation for perceptron decision trees. In Advances in Neural Information Processing Systems 10, M. Jordan, M. Kearns, and S. Solla (eds.). MIT Press, Cambridge, MA, 1998, pp. 336–342.
14. Zhang K., Xu Z., Peng J., and Buckles B.P. Learning through changes: An empirical study of dynamic behaviors of probability estimation trees. In Proc. IEEE Int. Conf. on Data Mining, 2005, pp. 817–820.
15. Zhang K. and Fan W. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. Know. Inf. Syst., 14(3), 2008.

Base-line Clock

► Time-Line Clock

Bayes Classifier

► Bayesian Classification

Bayesian Classification

Wynne Hsu
National University of Singapore, Singapore, Singapore

Synonyms

Bayes classifier

Definition

In classification, the objective is to build a classifier that takes an unlabeled example and assigns it to a class. Bayesian classification does this by modeling the probabilistic relationships between the attribute set and the class variable. Based on the modeled relationships, it estimates the class membership probability of the unseen example.

Historical Background

The foundation of Bayesian classification goes back to Reverend Bayes himself [2]. The origin of Bayesian belief nets can be traced back to [15]. In 1965, Good [4] combined the independence assumption with the Bayes formula to define the Naïve Bayes Classifier. Duda and Hart [14] introduced the basic notion of Bayesian classification and the naïve Bayes representation of joint distribution. The modern treatment and development of Bayesian belief networks is attributed to Pearl [8]. Heckerman [13] later reformulated the Bayes results and defined the probabilistic similarity networks that demonstrated the practicality of Bayesian classification in complex diagnostic problems.

Foundations

Bayesian classification is based on Bayes Theorem. It provides the basis for probabilistic learning that accommodates prior knowledge and takes into account the observed data.

Let X be a data sample whose class label is unknown. Suppose H is a hypothesis that X belongs to class Y. The goal is to estimate the probability that hypothesis H is true given the observed data sample X, that is, P(Y|X).

Consider the example of a dataset with the following attributes: Home Owner, Marital Status, and Annual Income as shown in Fig. 1. Credit Risks are Low for those who have never defaulted on their payments and credit risks are High for those who have previously defaulted on their payments.

Assume that a new data arrives with the following attribute set: X = (Home Owner = Yes, Marital Status = Married, Annual Income = High). To determine the credit risk of this record, it is noted that the Bayes classifier combines the predictions of all alternative hypotheses to determine the most probable classification of a new instance. In the example, this involves computing P(High|X) and P(Low|X) and to determine whether P(High|X) > P(Low|X)?

However, estimating these probabilities is difficult, since it requires a very large training set that covers every possible combination of the class label and attribute values. Instead, Bayes theorem is applied and it resulted in the following equations:

$$\mathrm{P(High|X) = P(X|High)P(High)/P(X)} \text{ and}$$

$$\mathrm{P(Low|X) = P(X|Low)P(Low)/P(X)}$$

P(High), P(Low), and P(X) can be estimated from the given dataset and prior knowledge. To estimate the class-conditional probabilities P(X|High), P(X|Low), there are two implementations: the Naïve Bayesian classifier and the Bayesian Belief Networks.

In the Naïve Bayesian classifier [13], the attributes are assumed to be conditionally independent given the class label y. In other words, for an n-attribute set $X = (X_1, X_2,\ldots,X_n)$, the class-conditional probability can be estimated as follows:

$$P(Y|X) = \alpha P(Y) \prod_i P(X_i|Y)$$

Customer ID	Home owner	Marital status	Annual income	Credit risks
1	Yes	Married	High	Low
2	No	Single	Medium	High
3	No	Married	Medium	High
4	Yes	Divorced	Low	High
5	No	Married	High	Low
6	No	Divorced	High	Low
7	Yes	Single	Low	Low
8	No	Single	High	Low
9	No	Married	Medium	High
10	Yes	Single	Medium	Low

Bayesian Classification. Figure 1. Dataset example.

In the example,

$$\begin{aligned}(X|\text{Low}) &= \text{P(Home Owner} \\ &= \text{Yes|Credit Risk} = \text{Low})\times \\ &\text{P(Marital Status} = \text{Married|Credit Risk} \\ &= \text{Low})\times \\ &\text{P(Annual Income} = \text{High|Credit Risk} \\ &= \text{Low} = 3/4 \times 2/4 \times 4/4 = 3/8 \\ \text{P}(X|\text{High}) &= \text{P(Home Owner} \\ &= \text{Yes|Credit Risk} = \text{High})\times \\ &\text{P(Marital Status} \\ &= \text{High})\times \\ &\text{P(Annual Income} = \text{High|Credit Risk} \\ &= \text{High}) = 0\end{aligned}$$

Putting them together, P(High|X) = P(X|High)P (High)/P(X) = 0

$$\text{P}(\text{Low}|X) = \text{P}(X|\text{Low})\text{P}(\text{Low})/\text{P}(X) > 0$$

Since P(Low|X) > P(High|X), X is classified as having Credit Risk = Low.

In other words,

$$Classifiy(X) = \arg\max_{y} P(Y=y)\prod_{i} P(X_i|Y=y)$$

In general, Naïve Bayes classifiers are robust to isolated noise points and irrelevant attributes. However, the presence of correlated attributes can degrade the performance of naïve Bayes classifiers as they violate the conditional independence assumption. Fortunately, Domingos and Pazzani [3] showed that even when the independence assumption is violated in some situations, the naïve Bayesian classifier can still be optimal. This has led to a wide spread use of naïve Bayesian classifiers in many applications. Jaeger [9] also further clarifies and distinguishes the concepts that can be recognized by naïve Bayes classifiers and the theoretical limits on learning the concepts from data.

There are many extensions to the naïve Bayes classifier that impose limited dependencies among the feature/attribute nodes, such as tree-augmented naïve Bayes [6], and forest-augmented naïve Bayes [11].

Bayesian belief network [10] overcomes the rigidity imposed by this assumption by allowing the dependence relationships among a set of attributes to be modeled as a directed acyclic graph. Associated with each node in the directed acyclic graph is a probability table. Note that a node in the Bayesian network is conditionally independent of its non-descendants, if its parents are known.

Refer to the running example. Suppose the probabilistic relationships among Home Owner, Marital Status, Annual Income, and Credit Risks are shown in Fig. 2. Associated with each node is the corresponding conditional probability table relating the node to its parent node(s).

In the Bayesian belief network, the probabilities are estimated as follows:

$$P(Risk|X) = \alpha\sum_{I} P(Risk|Income)\sum_{O} P(Owner) \sum_{S} P(Status)P(Income|Owner, Status)$$

In the above example,

$$\begin{aligned}\text{P}(\text{Low}|X) =&\text{P}(\text{Low}|\text{Income} = \text{High}) \\ &*\text{P}(\text{Income} = \text{High}|\text{Owner} \\ &= \text{Yes}, \text{Status} = \text{Married}) = 1 * 1 = 1\end{aligned}$$

Alternatively, by recognizing that given Annual Income, Credit Risks is conditionally independent of Home Owner and Marital status, then

$$\text{P}(\text{Low}|X) = \text{P}(\text{Low}|\text{Income} = \text{High}) = 1$$

This example illustrates that the classification problem in Bayesian networks is a special case of belief updating for any node (target class) Y in the network, given evidence X.

While Bayesian belief network provides an approach to capture dependencies among variables using a graphical model, constructing the network is time consuming and costly. Substantial research has been and is still continuing to address the inference as well as the automated construction of Bayesian networks by learning from data. Much progress has been made recently. So applying Bayesian networks for classification is no longer as time consuming and costly as before, and the approach is gaining headway into the mainstream applications.

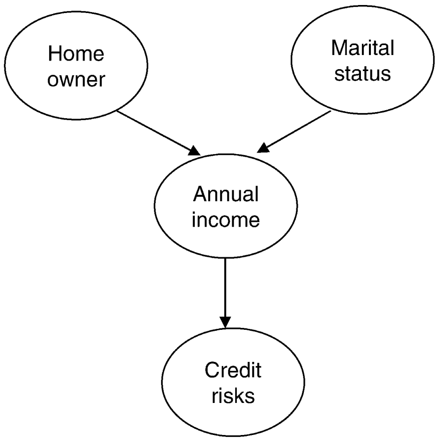
Key Applications

Bayesian classification techniques have been applied in many applications. Here, a few more common applications of Bayesian classifiers are mentioned.

Text Document Classification
Text classification refers to the grouping of texts into several clusters so as to improve the efficiency and effectiveness of text retrieval. Typically, the text documents are pre-processed and the key words chosen. Based on the selected keywords of the documents, probabilistic classifiers are built. Dumais et al. [5] show naïve Bayes classifier yields surprisingly good classifications for text documents.

Image Pattern Recognition
In image pattern recognition, a set of elementary or low level image features are selected which describe some characteristics of the objects. Data extracted based on this feature set are used to train Bayesian classifiers for subsequent object recognition. Aggarwal et al. [1] did a comparative study of three paradigms for object recognition – Bayesian Statistics, Neural Networks and Expert Systems.

Medical Diagnostic and Decision Support Systems
Large amounts of medical data are available for analysis. Knowledge derived from analyzing these data can be used to assist the physician in subsequent diagnosis. In this area, naïve Bayesian classifiers performed exceptionally well. Kononenko et al. [12] showed that



Owner = Yes
0.4

(Table associated with node owner)

Status	
Married	0.4
Single	0.4
Divorced	0.2

(Table associated with node status)

	Income = High	Income = Medium	Income = Low
HO = Yes, MS = Married	1	0	0
HO = Yes, MS = Divorced	0	0	1
HO = Yes, MS = Single	0	0.5	0.5
HO = No, MS = Married	0.33	0.67	0
HO = No, MS = Divorced	1	0	0
HO = No, MS = Single	0.5	0.5	0

(Table associated with node income)

	Risk = High	Risk = Low
Income = High	0	1
Income = Medium	0.75	0.25
Income = Low	0.5	0.5

(Table associated with node credit risks)

Bayesian Classification. Figure 2. Bayesian belief network of the credit risks dataset.

the naïve Bayesian classifier outperformed other classification algorithms on five out of the eight medical diagnostic problems.

Email Spam Filtering

With the growing problem of junk email, it is desirable to have an automatic email spam filters to eliminate unwanted messages from a user's mail stream. Bayesian classifiers that take into consideration domain-specific features for classifying emails are now accurate enough for real world usage.

Data Sets

http://archive.ics.uci.edu/beta/datasets.html
http://spamassassin.apache.org/publiccorpus/

URL to Code

More recent lists of Bayesian Networks software can be found at:

Kevin Murphy's website:
http://www.cs.ubc.ca/~murphyk/Bayes/bnsoft.html
Google directory:
http://directory.google.com/Top/Computers/Artificial_Intelligence/Belief_Networks/Software/
Specialized naïve Bayes classification software:
jBNC – a java toolkit for variants of naïve Bayesian classifiers, with WEKA interface

Cross-references

▶ Belief Networks
▶ Probabilistic Model
▶ Supervised Classification
▶ UnSupervised Classification

Recommended Reading

1. Aggarwal J.K., Ghosh J., Nair D., and Taha I. A comparative study of three paradigms for object recognition – Bayesian statistics, neural networks, and expert systems. Advances In Image Understanding: A Festschrift for Azriel Rosenfeld. IEEE Computer Society Press, Washington, DC, 1996, pp. 241–262.
2. Bayes T. An essay towards solving a problem in the doctrine of chances. Philos. Trans. R. Soc., 53:370–418, 1763.
3. Domingos P. and Pazzani M. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In Proc. 13th Int. Conf. on Machine Learning, 1996, pp. 105–112.
4. Duda R.O. and Hart P.E. Pattern Classification and Scene Analysis. Wiley, New York, 1973.
5. Dumais S., Platt J., Heckerman D., and Sahami M. Inductive learning algorithms and representations for text categorization. In Proc. Int. Conf. on Information and Knowledge Management, 1998.
6. Friedman N., Geiger D., and Goldszmidt M. Bayesian network classifiers. Mach. Learn., 29:131–163, 1997.
7. Good I.J. The Estimation of Probabilities: An Essay on Modern Bayesian Methods. MIT Press, Cambridge, MA, 1965.
8. Heckerman D. Probabilistic Similarity Networks. ACM Doctoral Dissertation Award Series. MIT Press, Cambridge, MA, 1991.
9. Jaeger M. Probabilistic classifiers and the concepts they recognize. In Proc. 20th Int. Conf. on Machine Learning, 2003, pp. 266–273.
10. Jensen F.V. An introduction to Bayesian networks. Springer, New York, 1996.
11. Keogh E. and Pazzani M. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In Proc. 7th Int. Workshop on Artificial Intelligence and Statistics, 1999.
12. Kononenko I., Bratko I., and Kukar M. Application of Machine Learning to Medical Diagnosis. Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications. Wiley, New York, 1998.
13. Langley P., Iba W., and Thompson K. An analysis of Bayesian classifiers. In Proc. 10th National Conf. on Artificial Intelligence, 1992, pp. 223–228.
14. Pearl J. Probabilistic Reasoning in Intelligenet Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA, 1988.
15. Wright S. Correlation and causation. J. Agric. Res., 20(7): 557–585, 1921.

BCNF

▶ Boyce-Codd Normal Form

Belief Time

▶ Transaction Time

Benchmark

▶ Application Benchmark

Biased Distribution

▶ Data Skew

Bibliography

▶ Citation

Bi-clustering

▶ Subspace Clustering Techniques

Bioinformatics

▶ Implications of Genomics for Clinical Informatics

Biological Data Retrieval, Integration, and Transformation

▶ Query Languages for the Life Sciences

Biological Metadata Management

Zoé Lacroix[1], Cartik R. Kothari[2], Peter Mork[3], Mark Wilkinson[2], Sarah Cohen-Boulakia[4]
[1]Arizona State University, Tempe, AZ, USA
[2]University of British Columbia, Vancouver, BC, Canada
[3]The MITRE Corporation, McLean, VA, USA
[4]University Paris-Sud, Orsay Cedex, France

Definition

Metadata characterize biological resources by core information including a name, a description of its input and its output (parameters or format), its address, and various additional properties. Resources are organized with respect to metadata that characterize their content (for data sources), their semantics (in terms of ontological classes and relationships), their characteristics (syntactical properties), their performance (with metrics and benchmarks), their quality (curation, reliability, trust), etc.

Historical Background

Digital resources for the Life Sciences include a variety of data sources and applications whose number increases dramatically every year [4]. Although this rich and valuable offering provides scientists with multiple options to implement and execute their scientific protocols (i.e., pipelines, dataflows, workflows), selecting the resources suitable for implementing each scientific step remains a difficult task. Scientific protocols are typically implemented using the resources a scientist is most familiar with, instead of the resources that may best meet the protocol's needs. The number of resources a scientist uses regularly, knowing their structure, the quality of data and annotations they offer, the capabilities made available by the provider to access, analyze, and display the data are ridiculously small compared to the thousands of resources made available on the Web [3]. Metadata are not only critical in selecting suitable resources for implement scientific protocols, but they are essential to the proper composition and integration of resources in a platform such as a workflow system or wrapped into a database mediation system. They also play a decisive role in the analysis of data, in particular to track data provenance. Finally, they contribute to data curation and the management of Life Sciences resources in a global and linked digital biological maze.

Foundations

Metadata management relies on the description of resources including the resource name, identification, and all additional information that may be relevant to locating, evaluating, and using the resource. A *resource identifier* is a sequence of characters that uniquely identifies a resource and is globally shared and understood over a network. A resource is analogous to a node on the Web. The ubiquitous Uniform Resource Locator (URL) is an example of a resource identifier, which uses the location, the local directory path, and the local file name of the resource to locate it on the Web. Unique Resource Identifiers (URIs) include URLs that not only identify the resource but describe its primary access mechanism or network location, and Uniform Resource Names (URN) that identify a resource by name in a particular namespace. The unique identification of resources is an unresolved problem in the life sciences community. Different protein, gene, and molecular interaction databases often assign separate identifiers to the same resource, a phenomenon known as *coreference*. Leveraging the information from all these databases becomes problematic, leading to duplicate records and inconsistency. To alleviate this problem, many Life Sciences databases cross reference their identifiers.

Metadata are data that describe a resource. Metadata include a wide range of information from attribution metadata, such as those attributes defined in the Dublin Core, to detailed policy metadata indicating who can access the resource under what conditions. *Semantic metadata* include the description of a resource with respect to the domain knowledge (e.g., a data source provides information about proteins, a tool computes the translation of a RNA sequence into an AA sequence). *Syntactic metadata* provide the description of the resource interface. *Summary metadata* describe the actual contents of the resource. These metadata include free text summaries and statistical summaries of the instances (values) contained in the database. Summary metadata can be classified along several axes: (i) textual versus quantitative, (ii) structured versus unstructured, and (iii) manually generated versus automatically generated. By far the most common type of summary metadata are textual. For example, NAR [4] maintains a listing of hundreds of biomedical resources. For each resource, they provide a brief description of the contents of that resource. Textual metadata allow an application developer or end-user to search for resources using keywords or phrases. The success of existing approaches seems to show that it is a familiar and intuitive operation, which works well when searching for reasonably well-defined concepts. Textual metadata are unstructured (i.e., free text) and manually curated. Alternatively, summary metadata can take the form of keywords drawn from a *controlled vocabulary*, such as Medical Subject Headings (MeSH) terms. A controlled vocabulary makes it easier to search for resources, assuming the vocabulary is sufficiently expressive and used consistently to annotate the resources. In most cases, textual metadata are generated manually, although there is some research in automatically extracting keywords from a resource for its annotation.

Quantitative metadata describe resources in terms of numeric datatypes. In the simplest case, these metadata specify the range of values that can be found in the resource. For example, all of the subjects in a pediatric database would be younger than 18. More detailed summaries are also possible. In the case of quantitative metadata, unstructured metadata make little sense. The end user needs to know what a given number represents, including relevant units. Quantitative metadata can still be generated manually or automatically. The former is required if the resource does not contain the necessary raw data. For example, if a pediatric database does not contain the ages of its subjects, the relevant age range must be specified manually. However, when the resource does contain the necessary raw data, quantitative metadata can be generated automatically. Moreover, the amount of detail in the metadata can vary depending on the needs of the resource owners and community members searching for resources. A current research challenge is determining the appropriate granularity for quantitative metadata and using these metadata to estimate the extent to which a given resource matches the end user's search criteria. *Statistical metadata* and benchmarks provide an additional layer exploited when the scientist wishes to predict the outcome of an execution. These metadata are particularly useful when several resources are combined to evaluate alternative evaluation strategies and select the most efficient one with respect to the protocols' aim [5]. The domain and range of resources, as well as resource overlaps contribute to the statistical description of resources. Similarly, information related to the quality of the resource (e.g., curation) may be exploited to optimize the quality of the execution.

Structural metadata describe the resource interface and the intention of the resource provider. These metadata can take on many forms including database schemata, Unified Modeling Language (UML) diagrams or Web service descriptions. What structural metadata provide are a description of how the resource provider intends to organize and deliver data. Controlled vocabularies capture domain knowledge and clarify resource descriptions (e.g., identical concepts). Controlled vocabularies are naturally extended by logical or conceptual representations such as expressed in a domain ontology. More generally, a metadata registry containing structural metadata allows an application developer to search for resources that are intended to contain particular types of data. Moreover, once a developer discovers a useful resource, he has a good idea of how to interact with that resource, both in terms of formulating queries and processing results. Structural metadata only indicate what sorts of queries can be posed, not whether those queries will return meaningful results. For example, the structural metadata for a card catalog might indicate that it includes, for each entry, a list of authors. Thus, one could reasonably query the card catalog for all books authored by John Grisham. However, if the card catalog supports a medical (non-fiction) library, it is unlikely that this query will return any record.

Bioinformatics resources may be represented with formats and standards developed by various communities driven by disparate motivations including business, library, Web, etc. The Resource Description Framework (RDF) and the RDF Schema (RDFS) were the earliest adopted standards for representing metadata about Web resources. The Dublin Core Metadata Elements Set (DCMES) is a standard set of metadata elements that can be used to describe a generic resource to facilitate its discovery and use. RDF specifically provides a very simple "triples" syntax or Subject-Predicate-Object syntax to capture resource metadata. Universal Description, Discovery and Integration (UDDI) is the XML-based format to register businesses on the Web proposed by OASIS. Dublin Core is a standard (NISO Standard Z39.85-2007) for cross-domain information resource description. The Web Ontology Language OWL, based on earlier languages OIL and DAML+OIL, is a W3C recommendation that extends XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics with descriptions of classes, along with their related properties and instances. The Web Service Description Language (WSDL) and its extension WSDL-S are used respectively as resource description and semantic annotation. Resource description and registration developed for the life science include BioMoby, PISE/Mobyle, caBIG, and SOAPlab.

The Life Sciences community has a number of metadata annotation standards. The Darwin Core (DwC) is a metadata standard from the National Biological Information Infrastructure (NBII) for annotating the objects contained within natural history specimen collections and species observation databases. These annotations are used to retrieve records of natural history specimens and observation records from local libraries, integrating them with other collections across the United States and making them available on the Web. The Access to Biological Collections Data Schema (ABCD Schema) is a complementary, hierarchical metadata standard for the annotation of biological specimens. Mappings exist from the terms of Darwin Core to the ABCD Schema that illustrate the overlap between the two standards and the few differences. Organizations such as NBII, the Integrated Taxonomic Information System and the Global Biodiversity Information Facility leverage metadata standards such as ABCD Schema and DwC to discover and utilize information pertaining to species and natural history specimens that is distributed around the world.

The Minimum Information About a Microarray Experiment (MIAME) [1] from the Microarray Gene Expression Data (MGED) Society, is used to describe sufficient information about a microarray experiment to reproduce it unambiguously. An increasing number of data providers are embracing the MIAME standard and several journals including NAR, Cell, and Nature require MIAME compliant data as a condition for publishing microarray based papers. The Minimum Information About a Proteomics Experiment (MIAPE) [7] is a minimum information reporting requirement for proteomics experiments. It is analogous to the MIAME standard for transcriptomics data. MIAPE is distributed across several modules, each of which is useful to describe a different proteomics experiment. Example modules are the MIAPE Gel Electrophoresis module, the MIAPE Mass Spectrometry module and the MIAPE Column Chromatography module. The Minimum Information required for reporting a Molecular Interaction Experiment (MIMIx) [6] has been developed as a framework to capture metadata about molecular interaction experiments. MIAME, MIAPE and MIMIx are being developed under the auspices of the Proteomics Standards Initiative at the Human Proteomics Organization.

Metadata standards vary in complexity from the simple format of the Dublin Core to the complex requirements of MIAME. A number of organizations are currently involved in the development of metadata standards in various fields of study. The Federal Geographic Data Committee (FGDC) is involved in the creation of several metadata standards such as the Spatial Data Transfer Standard for the exchange of spatial data, the National Vegetation Classification Standard to support a national vegetation classification system, the Biological Data Profile for the documentation of biological data, and the Utilities Data Content Standard to standardize geospatial information for utility systems. In addition the FGDC publishes the Content Standard for Digital Geospatial Metadata (CSDGM) for the annotation of geospatial data in the form of maps, atlases and satellite images. The Ocean Biogeographic Information System as an initiative to make marine biogeographic data available to a worldwide audience. OBIS has developed a taxonomic hierarchy of metadata elements, called the OBIS taxonomy for annotation of marine species observations. GeoConnections is a Canadian initiative to make geospatial data in the form of maps and satellite images easily available to a worldwide audience. GeoConnections uses the CSDGM published by the FGDC, to annotate its geospatial data.

Key Applications

Resource discovery systems exploit metadata in order to map the user requirements to the resource characteristics. For example, caCORE is an n-tier data management and integration infrastructure that combines several interconnected software and services. The Cancer Bioinformatics Infrastructure Objects (caBIO) model is at the heart of caCORE. The caBIO model contains definitions of concepts and inter-concept relationships that are common to biomedical research. These concept definitions in caBIO are the basis upon which data from distributed repositories are integrated. These repositories include gene and homolog databases (e.g., UniGene and Homologene), pathway databases (e.g., BioCarta), vocabulary and terminology repositories (e.g., the National Cancer Institute (NCI) Thesaurus, NCI Metathesaurus, and the Gene Ontology). The PathPort framework developed at The Virginia Bioinformatics Institute, presents a Web based interface that makes it possible for end users to invoke local and distributed biological Web services that are described in WSDL, in a location and platform independent manner.

Metadata offer metrics that may be used to predict the outcome of the execution of a scientific protocol on selected resources [5]. Metadata provide resource characterization that allows their comparison with similar resources, thus addressing the need of combining multiple complementary resources to implement completely a single task (e.g., data coverage). Path-based systems such as BioNavigation exploit statistical metadata to predict the resources that are likely to return the most entries, the best characterized (most attributes) entries, etc. [3].

Although some degree of transparency is often needed in queries, scientists also expect to be aware of the provenance of the answers. In order to analyze the results obtained from the execution of their scientific protocols, they often need to understand the process that produced the dataset. In particular, they need to know which resources have been used, and how entries have been linked to generate the answer to their question. Data traceability is related to the degree of information pertaining to the resources used to implement the process as well as the integration used to combine them. Because of this, reasoning on data provenance may exploit scientific resource metadata repositories. For example, BioGuideSRS allows the user to visualize the correspondence between the graph of entities and the graph of sources-entities [2]. By selecting an entity, the user visualizes the sources which provide information about this entity; similarly, by selecting a relationship, the user visualizes the links between sources which achieve this relationship. Second, the data obtained as a result yielded by BioGuideSRS to the user is systematically associated with the path which has been used to obtain it. In this way, the user knows the exact sequence of sources and links used. This approach was demonstrated with the ZOOM*UserViews system.

Future Directions

Metadata management remains a critical issue for the Life Sciences. First, the community has not agreed on common metadata to publish together with a resource so that it is properly identified, located, and used by scientists. Multiple discussions related to the representation of scientific objects generate the design of a large number of ontologies as published by the Open Biomedical Ontology (OBO) group. Although this effort contributes significantly to the better understanding of scientific information, it produces ontologies that may overlap and that are difficult to integrate. This semantic gap is aggravated by the diversity of models and formats used by biological data providers. Moreover, the community shows reluctance to adopting recommendations from the W3C Semantic Web for the specification of resources. The lack of a common publishing process for resources affects their impact significantly. In particular, it challenges the development of resource repositories to support resource discovery. Consequently, it affects the ability for scientists to select resources suitable to implement the scientific tasks involved in their protocols. The development of adequate technology, still in its infancy, is rather limited by the lack of a *franca lingua* for metadata. Future developments include the identification of metrics that adequalety capture the characteristics of resources, the design of benchmarks to evaluate and compare similar resources, automated data curation approaches that exploit and update resource metadata, automated classification of resources, data provenance analysis, etc.

Data Sets

NAR http://nar.oxfordjournals.org/

BMC Source Code for Biology and Medicine http://www.scfbm.org/home

Bioinformatics Links Directory http://bioinformatics.ca/links_directory/

Open Biomedical Ontologies (OBO) http://obofoundry.org/

Next Generation Biology Workbench (Swami) http://www.ngbw.org

URL to Code

Medical Subjects Headings (MeSH) http://www.nlm.nih.gov/mesh

UDDI http://www.uddi.org/

OWL Web Ontology Language http://www.w3.org/TR/owl-features/

DAML+OIL http://www.w3.org/TR/daml+oil-reference

BioMOBY http://biomoby.org/

PISE http://www.pasteur.fr/recherche/unites/sis/Pise/

Mobyle http://www.pasteur.fr/recherche/unites/sis/Pise/mobyle.html

caBIG http://cabig.nci.nih.gov/

SOAPlab http://www.ebi.ac.uk/Tools/webservices/soaplab/overview

National Biological Information Infrastructure (NBII) http://www.nbii.gov/

Microarray Gene Expression Data (MGED) Society http://www.mged.org

Federal Geographic Data Committee (FGDC) http://www.fgdc.gov

Ocean Biogeographic Information System (OBIS) http://www.iobis.org

GeoConnections http://www.geoconnections.org

caCORE http://ncicb.nci.nih.gov/infrastructure/cacoresdk

ZOOM*UserViews system http://db.cis.upenn.edu/research/provwf.html

BioNavigation http://bioinformatics.eas.asu.edu/

BioGuide http://bioguide-project.net/

Cross-references

▶ Benchmark
▶ Biological Resource Discovery
▶ Dublin Core
▶ Graph Management in the Life Sciences
▶ HTTP
▶ Metadata
▶ Ontology
▶ RDF
▶ UML
▶ URI
▶ URL
▶ URN
▶ Web Services
▶ Web Services and the Semantic Web for Life Science Data
▶ XML

Recommended Reading

1. Brazama A., Hingamp P., Quackenbush J., Sherlock G., Spellman P., Stoeckert C., Aach J., Ansorge W., Ball C.A., Causton H.C., Gaasterland T., Holstege F.C.P., Kim I.F., Markowitz V., Matese J.C., Parkinson H., Robinson A., Sarkans U., Schulze-Kremer S., Stewart J., Taylor R., Vilo J., and Vingron M. Minimum information about a microarray experiment (MIAME) – toward standards for microarray data. Nat. Genet., 29:365–371, 2001.
2. Cohen-Boulakia S., Biton O., Davidson S., Froidevaux C. BioGuideSRS: querying multiple sources with a user-centric perspective. Bioinformatics, 23(10):1301–1303, 2007.
3. Cohen-Boulakia S., Davidson S., Froidevaux C., Lacroix Z, and Vidal M.E. Path-based systems to guide scientists in the maze of biological resources. J. Bioinf. Comput. Biol., 4(5):1069–1095, 2006.
4. Galperin M.Y. The molecular biology database collection: 2007 update. Nucl. Acids Res., 35:D3–D4, 2007.
5. Lacroix Z., Raschid L., and Eckman B. Techniques for optimization of queries on integrated biological resources. J. Bioinf. Comput. Biol., 2(2):375–411, 2004.
6. Orchard S., Salwinski L., Kerrien S., Montecchi-Palazzi L., Oesterheld M., Stmpflen V., Ceol A., Chatr-aryamontri A., Armstrong J., Woollard P., Salama J.J., Moore S., Wojcik J., Bader G.D., Vidal M., Cusick M.E., Gerstein M., Gavin A.C., Superti-Furga G., Greenblatt J., Bader J., Uetz P., Tyers M., Legrain P., Fields S., Mulder N., Gilson M., Niepmann M., Burgoon L., De Las Rivas J., Prieto C., Perreau V.M., Hogue C., Mewes H.W., Apweiler R., Xenarios I., Eisenberg D., Cesareni G., and Hermjakob H. The minimum information required for reporting a molecular interaction experiment (MIMIx). Nat. Biotechnol., 25:894–898, 2007.
7. Taylor C.F., Paton N.W., Lilley K.S., Binz P.A., Julian R.K., Jones A.R., Zhu W., Apweiler R., Aebersold R., Deutsch E.W., Dunn M.J., Heck A.J.R., Leitner A., Macht M., Mann M., Martens L., Neubert T.A., Patterson1 S.D., Ping P., Seymour S.L., Souda P., Tsugita A., Vandekerckhove J., Vondriska T.M., Whitelegge J.P., Wilkins M.R., Xenarios I., Yates J.R., and Hermjakob H. The minimum information about a proteomics experiment (MIAPE). Nat. Biotechnol., 25:887–893, 2007.

Biological Networks

Amarnath Gupta
University of California-San Diego, La Jolla, CA, USA

Synonyms

Biological pathways; Molecular interaction graphs; Signal transduction networks; Transcriptional networks; Protein-protein interaction networks

Definition

A biological network is a graph-structured representation of binarized interactions among biological objects. Typically, the nodes in such a graph represent biological molecules, and the edges are labeled to represent different forms of interactions between molecules.

Example: A transcriptional network is a directed graph where a node represents either a protein (a transcription factor) or a region of the chromosome such that the edges can be constructed from the protein node to the chromosomal region. The edge in the graph represents that the protein can initiate the transcription (production of messenger RNA) process.

Key Points

A biological network is typically a node and edge attributed graph, where the edges can have different semantics depending on the kind of network. In some networks, the edges may be weighted, denoting, for instance, the probability of the interaction taking place. In some networks, like the protein-protein interaction graph, the edges are undirected. In some cases, like signal transduction networks, the edges represent the flow of time. Querying, integrating, and simulating are typical operations performed on biological networks.

Cross-references

▶ Graph Data Management in Scientific Applications

Recommended Reading

1. Baitaluk M., Qian X., Godbole S., Raval A., Ray A., and Gupta A. PathSys: integrating molecular interaction graphs for systems biology. BMC Bioinformatics, 7:55, 2006.
2. Eckman B.A. and Brown P.G. Graph data management for molecular and cell biology. IBM J. Res. Dev., 50(6):545–560, 2006.
3. Leser U. A query language for biological networks. Bioinformatics, 21(Suppl 2):ii33–ii39, 2005.

Biological Pathways

▶ Biological Networks

Biological Query Languages

▶ Query Languages for the Life Sciences

Biological Resource Discovery

Zoé Lacroix[1], Cartik R. Kothari[2], Peter Mork[3], Rami Rifaieh[4], Mark Wilkinson[2], Juliana Freire[5], Sarah Cohen-Boulakia[6]
[1]Arizona State University, Tempe, AZ, USA
[2]University of British Columbia, Vancouver, BC, Canada
[3]The MITRE Corporation, McLean, VA, USA
[4]University of California-San Diego, San Diego, CA, USA
[5]University of Utah, Salt Lake City, UT, USA
[6]University Paris-Sud, Orsay, France

Definition

Resources for the Life Sciences include various expedients including (access to) data stored in flat files or databases (e.g., a query form or a textual search engine), links between resources (index or hyperlink), or services such as applications or tools. *Resource discovery* is the process of identifying and locating existing resources that have a particular property. Machine-based resource discovery relies on crawling, clustering, and classifying resources discovered on the Web automatically. Resource discovery systems allow the expression of queries to identify and locate resources that implement scientific tasks and have properties of interest.

Historical Background

Resource selection relies on the identification of the resources suitable to achieve each task and the ability to compose the selected resources into a meaningful and efficient executable protocol. Metadata constitute the core information requisite to evaluate the suitability of Life Sciences resources to achieve a scientific task. Metadata critical to resource discovery include (i) resource publication, identification, and location, and

(ii) semantic and (iii) syntactic descriptions. First scientists need to be aware of existing resources. If academic publications such as Nucleic Acids Research (NAR) [7] or BMC Source Code for Biology and Medicine have provided valuable media where bioinformaticians may publish their resources, they require significant manpower to identify and evaluate the potential of each resource and compile and record their location and description for future use. Core resource description in a unified format accessible to scientists and machines alike and resource repositories contribute greatly to ease the problem of resource identification and location.

Resource discovery is the interface between resource metadata on one side and resource integration to implement complex scientific protocols (or workflows, queries, pipelines) on the other. Indeed the motivation for discovering a resource is drawn from the need to implement a scientific task. Most approaches to support resource discovery only locate one resource at a time, regardless of their future composition to implement complex workflows. In contrast, path-based guiding systems such as BioNavigation and BioGuide provide the ability to express resource discovery queries to identify resources that can be composed to express scientific protocols expressed as connected scientific tasks [4].

Foundations

Resource discovery relates to the activity of identifying a resource suitable to implement a particular task. Resource discovery relies on various metadata that specify the characteristics of resources thus allowing the mapping of the requirements to the resource specifications. The type of metadata chosen to represent resources will constrain resource discovery. For example, textual metadata drawn from a controlled, hierarchical vocabulary, support resource discovery by automatically expanding search terms to include more-specific terms. However, textual metadata are not sufficient when searching on specific criteria; for example, consider a researcher interested in finding datasets of "*MRI images for subjects between the ages of 18 and 24*". Syntactic (formats) and semantic (concepts) metadata describe how a resource is organized and provide some insight into what type of information might be found in the resource while summary metadata specify the content of the resource. Although structural metadata are normally generated to help application developers understand how to interact with the resource, they can also be collected to support resource discovery. For example, in the caBIG framework, structural metadata are represented as common data elements. Each common data element references a common terminology (the NCI thesaurus in this case) and may also contain free text documentation describing that data element both providing a semantic representation of the resource. The metadata registry also maps common data elements to resources that provide instances of that element. An application developer searches the metadata registry by providing a collection of keywords; the registry returns a list of data elements that contain those keywords.

Key Applications

BioMoby is an open source, extensible framework that enables the representation, discovery, retrieval, and integration of biological data from distributed data repositories and analysis services. By registering their analysis and data access services with BioMoby, service providers agree to use and provide service specifications in a shared semantic space. The BioMoby Central registry now hosts more than a thousand services in the United States, Canada, and several other countries across the world. BioMoby uses a datatype hierarchy to facilitate the automated discovery of Web services capable of handling specific input datatypes. As a minimal Web based interface, the Gbrowse Moby service browser can be used by biologists to discover and invoke biological Web services from the Moby registry and seamlessly chain these services to compose multi-step analytical workflows. The process is data centric, relying on input and output datatype specifications of the services. The Seahawk client interface can infer the datatype of the input data files directly and immediately presents the biologist with a list of Web services that can process the input file. Users of the Seahawk interface are relieved of the necessity to familiarize themselves with datatype hierarchies, and instead are free to concentrate on the analytical aspects of their work. The BioMoby service encyclopedia provides a query interface to the repository of services. MOBY-S Web Service Browser retrieves bioinformatics resources with respect to a data type. Additional interfaces to BioMoby services include registry browsers that provide access to the complete list of registered BioMoby services organized in a HTML page.

These interfaces are convenient when searching for services with respect to a specific data format (input), but they are not suitable when searching for services with respect to their scientific meaning rather than their format. Another critical limitation of the approaches occurs when no single service achieves the task. In order to allow the discovery of the services that can be used to express scientific protocols, combinations of services must be retrieved. Scientific data integration systems such as workflow systems [6] enable the composition and execution of bioinformatics services. Combining a workflow approach with a service representation that guarantees compatibility of data formats offers a great value to the scientist who has selected the services to use and wishes to combine them in an executable workflow. A resource is selected because it uses or produces the expected format (e.g., `FASTA`) rather than because it implements the expected scientific aim or because it is efficient. This is illustrated by the BioMoby plugin in Taverna. When a BioMoby service, e.g., '`DragonDB_TBlastN`', is included in a workflow its output format, e.g., `NCBI_BLAST_TEXT`, can be searched (brief search) against available formats to determine if it is an input to any other service registered in BioMoby [9]. The characterization of a resource provided by existing formats such as Web services does not include the level of metadata necessary to evaluate the suitability of resources beyond the description of its input and output. More advanced resource formats such as OWL-S, WSDL-S, SAWSDL, and BioMoby aim at providing a semantic layer to capture better what the resource does in addition to its input and output data formats. The semantic part of the resource registration allows the classification of resources into a hierarchy of classes thus enhancing resource discovery. For example, the semantic search method of the BioMoby plugin in Taverna traverses the object ontology and recursively extracts the parent nodes of that particular output object [9]. However existing approaches do not offer an interface that allows the discovery of services with respect to their scientific meaning expressed in an ontology. To overcome this difficulty, path-based guiding systems such as SemanticMap and BioGuide can be combined with integration platforms to allow the discovery of resources suitable to implement scientific workflows. For example, BioGuide extends SRS [5] to allow a unique interface to discover resources and express queries over integrated data sources [3].

Resource discovery can exploit further resource metadata to predict the outcome of a workflow execution and select the resource more likely to produce the expected output. For example, BioNavigation [4] exploits various statistical metadata combined with semantic data to rank resources with respect to the users' criteria. Resources are ranked with respect to their cardinality, the characterization of their entries (number of attributes), etc. A user interested in *retrieving as many genes involved in a particular disease* selects a path in a domain ontology together with the corresponding ranking criteria. BioNavigation returns a ranking of all implementations of the conceptual path [10].

BioSpider is a system that integrates biological and chemical online databases. Given a biological or chemical identifier, BioSpider produces a report containing physico-chemical, biochemical and genetic information about the identifier. Ngu et al. [11] proposed an approach to classify search interfaces by probing these interfaces and trying to match the control flow of the interface against a standard control flow. InfoSpiders is a multi-agent focused crawler specialized for biomedical information whose goal is to fetch information about diseases when given information about genes. The Adaptive Crawler for Hidden-Web Entry Points (ACHE) is a focused crawler specialized for locating searchable Web forms that serve as entry points to online databases and Web services. Context-Aware Form Clustering (CAFC) is a clustering approach that models Web forms as a set of hyperlinked objects and considers visible information in the form context – both within and in the neighborhood of forms – as the basis for similarity comparison. A repository of scientific resources was automatically compiled using the approach [1].

Future Directions

Biological resource discovery remains a critical issue for the Life Sciences. The development of a system to support resource discovery is directly constrained by the information pertaining to scientific resources made available to the users as well as the formats designed to represent these rich metadata. For these reasons research on resource discovery for Life Sciences still is in its infancy. Scientists dramatically need assistance at each level of the process from the identification of the resources that best would meet the experimental requirements to the actual composition of the resources in an executable workflow. Future developments include the design of systems that combine various orthogonal aims for resource selection such as semantics (what the resource does), statistics (prediction of the result), syntax (schema mapping for resource compsition), performance (efficiency), quality, etc.

Data Sets

NAR http://nar.oxfordjournals.org/

BMC Source Code for Biology and Medicine http://www.scfbm.org/home

Bioinformatics Links Directory http://bioinformatics.ca/links_directory/

Semantic Map for Structural Bioinformatics http://bioserv.rpbs.jussieu.fr/SBMap/

Automatically compiled list of biological resources http://formsearch.cs.utah.edu

Open Biomedical Ontologies (OBO) http://obofoundry.org/

Next Generation Biology Workbench (Swami) http://www.ngbw.org

URL to Code

caBIG http://cabig.nci.nih.gov/

BioMOBY http://biomoby.org/

SOAPlab http://www.ebi.ac.uk/Tools/webservices/soaplab/overview

SemanticMap http://bioinformatics.eas.asu.edu/

myGRID http://www.mygrid.org.uk/

Taverna http://taverna.sourceforge.net/

MOBY-S http://mobycentral.icapture.ubc.ca/

BioNavigation http://bioinformatics.eas.asu.edu/

BioGuide http://bioguide-project.net/

Seahawk http://biomoby.open-bio.org/

Remora http://lipm-bioinfo.toulouse.inra.fr/remora/cgi/remora.cgi

Kepler http://www.kepler-project.org/

BioSpider http://biospider.ca

InfoSpiders http://www.informatics.indiana.edu/fil/IS/

Cross-references

▶ Benchmark
▶ Biological Metadata Management
▶ Dublin Core
▶ Graph Management in the Life Sciences
▶ HTTP
▶ Metadata
▶ Ontology
▶ RDF
▶ UML
▶ URI
▶ URL
▶ URN
▶ Web Services
▶ Web Services and the Semantic Web for Life Science Data
▶ XML

Recommended Reading

1. Barbosa L., Tandon S., and Freire J. Automatically Constructing a Directory of Molecular Biology Databases. In Data Integration in the Life Sciences, LNCS, Vol. 4544, 2007, pp. 6–16.
2. Clark T., Martin S., and Liefeld T. Graphically Distributed Object Identification for Biological Knowledge Bases. Brief. Bioinformat., 5(1):59–70, 2004.
3. Cohen-Boulakia S., Biton O., Davidson S., and Froidevaux C. BioGuideSRS: q uerying multiple sources with a user-centric perspective. Bioinformatics, 23(10):1301–1303.
4. Cohen-Boulakia S., Davidson S., Froidevaux C., Lacroix Z., and Vidal M.E. Path-based systems to guide scientists in the maze of biological resources. J. Bioinformat. Comput. Biol., 4(5):1069–1095, 2006.
5. Etsold T., Harris H., and Beaulah S. Bioinformatics: Managing Scientific Data. Chapter 5 – SRS: An Integration Platform for Databanks and Analysis Tools in Bioinformatics, pp. 109–146. Z. Lacroix and T. Critchlow (eds.). Morgan Kaufmann, Los Altos, CA, 2003.
6. Fox G.C. and Gannon D. (eds). Concurrency and Computation: Practice and Experience, Special Issue: Workflow in Grid Systems, 2006.
7. Galperin M.Y. The Molecular Biology Database Collection: 2007 update. Nucl. Acids Res., 35:D3–D4, 2007.
8. Good B.M. and Wilkinson M.D. The Life Sciences Semantic Web is Full of Creeps! Brief. Bioinformat., 7(3):275–286, 2006.
9. Kawa, E.A., Senger M., and Wilkinson M.D. BioMoby extensions to the Taverna workflow management and enactment software BMC Bioinformat., 7:523, 2006.
10. Lacroix Z., Raschid L., and Eckman B. Techniques for optimization of queries on integrated biological resources. J. Bioinformat. Computat. Biol., 2(2):375–411, 2004.
11. Ngu A.H.H., Rocco D., Critchlow T., and Buttler D. Automatic discovery and inferencing of complex bioinformatics web interfaces. World Wide Web, 8(4):463–493, 2005.
12. Wolstencroft K., Alper P., Hull D., Wroe C., Lord P., Stevens R., and Goble C. The myGrid Ontology: Bioinformatics Service Discovery. Int. J. Bioinformat. Res. Appl., 3(3):303–325, 2007.

Biological Sequences

Amarnath Gupta
University of California-San Digeo, La Jolla, CA, USA

Synonyms

DNA sequences; Protein sequence

Definition

A biological sequence is a sequence with a small fixed alphabet, and represents a naturally occurring or experimental generated fragment of genetic or protein material or any intermediate product (like the messenger RNA).

Example: A DNA fragment has the 4 character alphabet 'A', 'C', 'T', 'G'. Chromosomes are long strings over this alphabet.

Key Points

Biological sequences can be long. A full chromosome may have millions of characters. Therefore development of proper storing and indexing strategies is very important for fast retrieval. Suffix tree based indexes have been used successfully for long biological sequences. Further, approximate string matching techniques with potential deletions and insertions are important for biological sequences. BLAST is a well known algorithm used for approximate matching and ranking of biological sequences.

Cross-references

► Index Structures for Biological Sequences
► Query Languages and Evaluation Techniques for Biological Sequence Data
► Query languages for the life sciences

Recommended Reading

1. Brown A.L. Constructing genome scale suffix trees. In Proc. 2nd Asia-Pacific Bioinformatics Conference, 2004.
2. Hunt E., Atkinson M.P., and Irving R.W. A database index to large biological sequences. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 139–148.
3. Phoophakdee B. and Zaki M.J. TRELLIS +: an effective approach for indexing genome-scale sequences using suffix trees. In Proceedings of the Pacific Symposium on Biocomputing (online proceedings), 2008, pp. 90–101.
4. Tian Y., Tata S., Hankins R.A., and Patel J.M. Practical methods for constructing suffix trees. VLDB J., 14(3): 281–299, 2005.

Biomedical Data Annotation

► Biomedical Data/Content Acquisition, Curation

Biomedical Data/Content Acquisition, Curation

Nigam Shah
Stanford University, Stanford, CA, USA

Synonyms

Biomedical data annotation

Definition

The largest source of biomedical knowledge is the published literature, where results of experimental studies are reported in natural language. Published literature is hard to query, integrate computationally or to reason over. The task of reading published papers (or other forms of experimental results such as pharmacogenomics datasets) and distilling them down into structured knowledge that can be stored in databases as well as knowledgebases is called curation. The statements comprising the structured knowledge are called annotations. The level of structure in annotation statements can vary from loose declarations of "associations" between concepts (such as associating a paper with the concept "colon cancer") to statements that declare a precisely defined relationship between concepts with explicit semantics. There is an inherent tradeoff between the level of detail of the structured annotations and the time and effort required to create them. Curation to create highly structured and computable annotations requires PhD level individuals to curate the literature. In the molecular biology research community, this task is performed primarily by curators employed by genome databases such as the saccharomyces genome database [7]. In the biomedical research community this task is performed by curators employed by community portals such as AlzForum for Alzheimer's research [9] and PharmGKB for pharmacogenomics [31]. In the medical community such curation is still an ignored task, with some groups, such as RCTBank [26], pioneering the effort to curate clinical trial reports.

Historical Background

In the biomedical domain, curation began with the formation of cDNA, EST and gene sequence databases such as GenBank. Initially, curation was restricted to the task of assigning a functional annotation (usually in free text) to a sequence being submitted to GenBank. Scientists performing the experiments and submitting the data performed the task on their own. With the rise in the amount of sequence data and subsequently data on the function, structure and cellular locations of gene products along with the formation of communities of researchers around specific model organisms, the task of curation gradually became centralized in the role of a curator at model organism databases. Interaction amongst the curators and leading scientists led to the creation of projects such as the gene ontology project [1] in 1998, which led to a systematic basis for creating annotations about the molecular function, biological process and cellular locations of gene products. In subsequent years, user groups formed around other kinds of data, such as microarray gene expression data, resulting in the creation of information models for structuring the metadata pertaining to high throughput experiments. Individual research groups, such as Ecocyc, have already maintained a high level of curation effort, particularly for Information about biological pathways; although it was the success of the gene ontology project that resulted in the widespread appreciation for the need of curated content. With the continued rise in the amount and diversity of biomedical data, the need for curation continues to increase; both in terms of the number of man-hours required and in the level detail desired in the resulting annotations.

Foundations

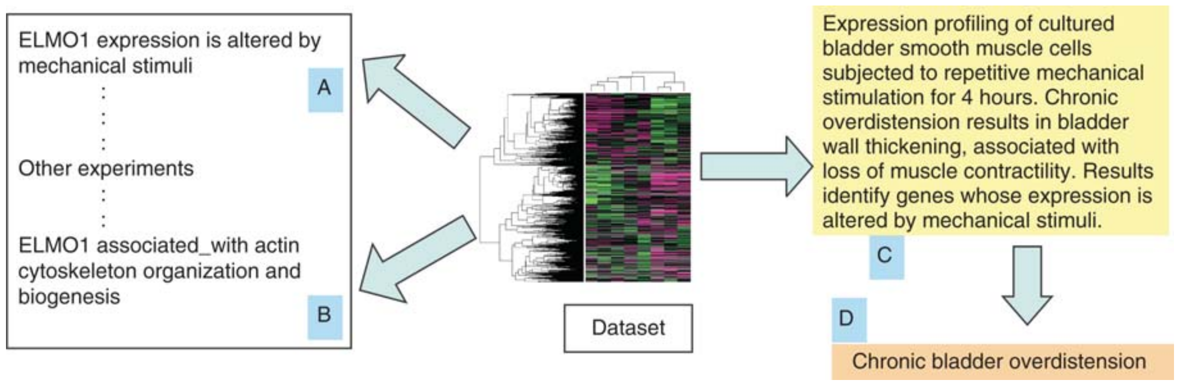
In the course of their work, biomedical investigators must integrate a growing amount of diverse information. It is not possible for scientists to bring together this large amount of information without the aid of computers. Researchers have turned to ontologies – which allow representation of experimental results in a structured form – to facilitate interoperability among databases by indexing them with standard terms as well as to create knowledge bases that store large amounts of knowledge in a structured manner. Ontologies provide researchers with both the structure into which experimental results, facts and findings have to be put into as well as the words (or terms) to be used in populating the structure with instances [8,14]. If the ontologies are well-designed, then the resulting knowledge bases can be used to retrieve relevant facts, to organize and interpret disparate knowledge, to infer non-obvious relationships, and to evaluate hypotheses posited by scientists.

Assertion annotations – assertions or statements about the relationships among biological entities and the processes in which they participate – are a crucial link between abstractions of experimental results and the theory (or theories) that explain the underlying results. The national center for biomedical ontology develops methods and tools that enable the easy creation of such assertion annotations [11]. However, even with tool support, the creation of assertion annotations is manual, hard and expensive. In addition to this annotation as assertion viewpoint, another predominant use of annotations is to provide metadata for datasets stored in databases. In this case, these metadata-annotations are not assertions about any biological entity but instead provide additional information about the experiment or dataset examining the biological entity. Such metadata-annotations provide information about experimental conditions, the disease that the dataset pertains to, the perturbation applied during the experiment, and so on. Metadata-annotations do not state a biological fact like an assertion based on interpretation of experimental results. These two kinds of annotations: (i) assertion annotations and (ii) metadata annotations, are highlighted in Fig. 1.

Curation is the process by which annotations (either assertions or metadata) are created. (The word "annotation" is also used by some as a verb to describe this process of curation to create annotations; this has lead to wide-spread confusion in the community about the meaning of annotation.) Until now curation has been largely a manual process requiring highly qualified individuals to read and interpret the text in published papers to create the annotations. It is important to note that even though curation is carried out by skilled personnel, different curators have different opinions on what "knowledge" is being reported in the paper. Increasingly, automated methods are being employed to assist in the curation task because of the fact that manual curation is unlikely to scale and keep pace with the growth of biomedical data and literature [3]. Curation is typically carried out using a tool that allows the curator to select relevant ontology terms and associate them with the entity being annotated. The same tool writes out the resulting annotations in a custom format.

Technical Issues

The Different Types of Expressivity of Ontologies/Vocabularies Used to Create the Annotations As discussed, annotations can range from simple terms that are "associated" with a particular resource to structured


ELMO1 expression is altered by mechanical stimuli
A
Other experiments
ELMO1 associated_with actin cytoskeleton organization and biogenesis
B
Dataset
Expression profiling of cultured bladder smooth muscle cells subjected to repetitive mechanical stimulation for 4 hours. Chronic overdistension results in bladder wall thickening, associated with loss of muscle contractility. Results identify genes whose expression is altered by mechanical stimuli.
C
D
Chronic bladder overdistension

Biomedical Data/Content Acquisition, Curation. Figure1. Shows the relationship of the assertion-annotations and metadata-annotations with datasets. The dataset is analyzed by a researcher to make a fine grained statement (a), which states the particular observation made in the dataset and reported in a publication. Several such statements get published in scientific papers. A curator, after reading a multitude of these papers creates an *assertion-annotation* (such as a GO annotation) as a summary statement (b) based on the fine grained statements. The dataset is also described by the researcher in terms of the disease studied, the cell lines used, the experimental conditions that existed etc. This description (c), comprises the *metadata-annotation* of the dataset, and is usually in natural language; although at times it is done using a CV. (d) shows the "tag" from a controlled vocabulary or ontology that can be assigned to this dataset upon processing the text description computationally.

assertions that use explicit logical relationships. Depending on the required use – that of creating assertion-annotations or creating metadata-annotations – the ontologies used in the annotation process need to have adequate expressivity in terms of the different relationships the user can use during the curation process. A detailed discussion on the kinds of relationships that are available in biomedical ontologies is can be found in [27]. The most widely used artifacts for annotation are controlled vocabularies (CVs). A CV provides a list of terms whose meanings are specifically defined. Terms from a CV are usually used for indexing records in a database. The Gene Ontology (GO) is the most widely used CV in databases serving biomedical researchers [1]. The GO provides terms that are "associated" with particular gene products for describing their molecular function (MF), biological process (BP) and cellular component (CC). Arguably, CVs provide the most return-on-effort in terms of facilitating database search and interoperability.

Storage Schemes and Data Models to Store These Annotations in Underlying Databases Most annotations when created initially are stored as flat text files. However in order for the annotations to be useful to researchers, they need to be stored in database systems that support efficient storage and querying. Naturally the database schema and the data model to which the annotations conform to becomes an important issue. Until recently the trend was to create a relational schema corresponding to the annotation model used for a particular curation workflow and each group created its own annotation model as well as schema. This lead to various "silo" databases that need to be mapped to one another. The need for such mapping lead to the creation of groups, such as BioPAX, which proposed "exchange formats" to map silo databases to one another [4]. Recently, semantic web technology is receiving a lot of attention in the biomedical community because of the promise of "automatic" interoperability if different groups use consistent identifier (URIs) as well as the Resource Description Framework (RDF) format to describe entities and resources in their annotations [22].

Techniques for Indexing the Curated Annotation for Retrieval Assuming the issue of creating annotations (manually or computationally) is adequately addressed, special attention needs to be paid to the appropriate indexing of annotations. For example, once a publication is annotated by associating it with the term *melanoma*, in order to ensure appropriate retrieval when someone searches for *skin neoplasms* it is essential to index the same paper with terms such as *skin neoplasm* (because *melanoma* is a kind of *skin neoplasm*). This can be accomplished by pre-computing all such inferred annotations or by real time query expansion using the hierarchy among the terms *melanoma* and *skin neoplasm*.

Workflow Aspects of the Curation Process As noted before, different curators can have different opinions on what "knowledge" is being reported in the paper. The level of this agreement is quantified by calculating inter-curator agreement using a variety of methods [6]. Using detailed curation guidelines, many projects achieve inter-curator agreement in the range of 85–90% and some as high as 94% [6]. Currently, curation is typically carried out using a tool, such as Phenote, (www.phenote.org.) that allows the curator to select relevant ontology terms and associate them with the entity being annotated. The same tool writes out the resulting annotations in a custom format. Usually the workflow for curation differs by organization and the kinds of source (such as published papers or medical records or clinical trials) being curated. Currently, there are no off-shelf workflow systems that provide a generic curation workflow. Increasingly curation is becoming web based and the tools used for curation are tied to a database which stores the annotations (See the Alzforum and SWAN projects for an example). There are also efforts to make curation collaborative, and several wiki-based projects such as wikipathways (www.wikipathways.org) are underway in the field.

Key Applications

The discovery process in biomedical research is cyclical; Scientists examine existing data to formulate models that explain the data, design experiments to test the hypotheses and develop new hypotheses that incorporate the data generated during experimentation. Currently, in order to advance this cycle, the experimentalist must perform several tasks: (i) gather information of many different types about the biological entities that participate in a BP, (ii) formulate hypotheses (or models) about the relationships among these entities, (iii) examine the different data to evaluate the degree to which his/her hypothesis is supported, and (iv) refine the hypotheses to achieve the best possible match with the data. In today's data-rich environment, this is a very difficult, time-consuming, and tedious task.

If existing data, information and knowledge are curated to create knowledge bases that store large amounts of knowledge in a structured manner [14,15] the resulting knowledge bases can be used to retrieve relevant facts, to organize and interpret disparate knowledge and to computationally evaluate hypotheses and model posited by scientists [2,18,19]. For example, EcoCyc is a comprehensive source of structured knowledge on metabolic pathways in E. Coli and can be used to reason about E. Coli metabolism. Reactome is a source of structured knowledge on BPs related to signal transduction, gene regulation and metabolism in eukaryotic organisms [13]. The creation of such knowledgebases requires that the task of curation be carried out with great detail and that the tradeoffs between the complexity of the annotation structure required and the curation overhead entailed by that be balanced. Understanding the curation cost is a significant factor in determining the feasibility of proposed knowledge-driven applications [12].

There are several public as well as private groups that curate biomedical literature and other data to create highly structured knowledge bases. A majority of these knowledge bases are centered on biological pathways and Ecocyc and Reactome are the leading examples. In Pharmacogenomics, PharmGKB is a resource that provides curated knowledge on the interactions between genotype and pharmacological effects of drugs [31]. The Semantic Web Applications in Neuroscience (SWAN) project is a resource providing curated knowledge on Alzheimer's research with a focus on capturing the evolving scientific discourse as the research progresses [16]. In the commercial sector, companies such as Ingenuity offer subscription access to curated literature content as well as curation-for-fee services.

Such curated and structured content is primarily used to interpret the results of high-throughput datasets in the light of prior knowledge. At the simplest level, coloring nodes of a pathway according the increase or decrease in their expression level in a particular assay is a widely used approach. Another widely used approach is that of counting the annotations, such as the association with a particular BP, assigned to a set of biological entities, such as genes deemed significant for a particular cancer, and analyzing for a statistically significant difference in the distribution of the annotation counts as compared to a reference such as the set of all the genes assayed.

The other key use of curated and structured content is to support computer aided reasoning; with the goal of inferring possible explanations for biological phenomena [25], for evaluating alternative explanations for biological phenomena[19], for automated question answering [29] and automatically constructing as well as extending existing structured descriptions of BPs such as pathways [23].

Future Directions

As the amount and diversity of data, information and knowledge rise in the biomedical domain, there is a recognized need to be able to compute with the existing knowledge [5]. As the use of ontology rises in the biomedical domain [30], the appreciation for the need of curated content is also rapidly increasing; along with the realization that manual curation is unlikely to keep pace with the needs of the community [3].

These trends have led several groups, such as the BioAI group at Arizona State University and the SWAN group, to propose the use of distributed and collaborative curation in an attempt to leverage the "wisdom of the masses" [10,17]. Collaborative curation holds tremendous promise for the field if the community can arrive at an agreed upon platform and formalism using which researchers can contribute structured content. The other clear future direction is the use of text-mining in the curation pipeline as a "force multiplier" to increase the productivity of existing curation efforts. The computational pharmacology group at University of Colorado is conducting exciting research in this direction. Both community-based collaborative curation as well as the use of text-mining to increase the efficiency of curation tools will be activities to follow closely for those interested in biomedical data acquisition and curation.

Cross-references

- ▶ Annotation
- ▶ Biological Metadata Management
- ▶ Curation

Recommended Reading

1. Ashburner M., et al. Gene ontology: tool for the unification of biology. Nat. Genet., 25(1):25–29, 2000.
2. Baral C., et al. A knowledge based approach for representing and reasoning about signaling networks. Bioinformatics, 20 (1):15–22, 2004.
3. Baumgartner Jr, et al. Manual curation is not sufficient for annotation of genomic databases. Bioinformatics, 23 (13):41–48, 2007.
4. BioPax-Consortium. BioPAX: Biological Pathways Exchange. Available from: http://www.biopax.org/ 2006.
5. Bodenreider O. and Stevens R. Bio-ontologies: current trends and future directions. Brief. Bioinform., 7(3):256–274, 2006.
6. Camon E.B., et al. An evaluation of GO annotation retrieval for BioCreAtIvE and GOA. BMC Bioinform., 6(Suppl 1):S17, 2005.
7. Cherry J.M., et al. SGD: saccharomyceas genome database. Nucleic Acids Res., 26(1):73–79, 1998.
8. Ciccaresse P., Wu E., and Clark T. An overview of the SWAN 1.0 ontology of scientific discourse. In Proc. 16th Int. World Wide Web Conference, 2007.
9. Clark T. and Kinoshita J. Alzforum and SWAN: the present and future of scientific web communities. Brief. Bioinform., 8(3):163–171, 2007.
10. Gao Y., et al. SWAN: a distributed knowledge infrastructure for Alzheimer disease research. J. Web Semantics, 4(3):222–228, 2006.
11. Gibson M. Phenote. Berkeley Bioinformatics and Ontology Project (BBOP), National Center for Biomedical Ontology, Lawrence Berkeley National Laboratory, 2007.
12. Hunter L. and Cohen K.B. Biomedical language processing: what's beyond PubMed? Mol. Cell., 21(5):589–594, 2006.
13. Joshi-Tope G., et al. Reactome: a knowledge base of biological pathways. Nucleic Acids Res., 33(Database Issue): D428–432, 2005.
14. Karp P.D. An ontology for biological function based on molecular interactions. Bioinformatics, 16(3):269–285, 2000.
15. Karp P.D. Pathway databases: a case study in computational symbolic theories. Science, 293(5537):2040–2044, 2001.
16. Katz A.E., et al. Molecular staging of genitourinary malignancies. Urology, 47(6):948–958, 1996.
17. Leslie M. Netwatch. Science, 312:1721, 2006.
18. Massar J.P., et al. BioLingua: a programmable knowledge environment for biologists. Bioinformatics, 21(2):199–207, 2004.
19. Racunas S.A., et al. HyBrow: a prototype system for computer-aided hypothesis evaluation. Bioinformatics, 20 (Suppl 1):257–264, 2004.
20. Reactome Curator Guide. http://wiki.reactome.org/index.php/Reactome_Curator_Guide
21. Rise of the Bio-Librarian – the field of biocuration expands as the data grow. http://www.the-scientist.com/article/display/23316/.
22. Ruttenberg A., et al. Advancing translational research with the Semantic Web. BMC Bioinform., 8(Suppl 3):S2, 2007.
23. Rzhetsky A., et al. GeneWays: a system for extracting, analyzing, visualizing, and integrating molecular pathway data. J. Biomed. Inform., 37(1):43–53, 2004.
24. Second International Biocuration Meeting, San Jose, CA, October 25–28, 2007. http://biocurator.org/Mtg2007/index.html.
25. Shrager J., et al. Deductive biocomputing. PLoS ONE, 2(4): e339, 2007.
26. Sim I., Olasov B., and Carini S. The Trial Bank system: capturing randomized trials for evidence-based medicine. AMIA Annu. Symp. Proc., 2003:1076, 2003.

27. Smith B., et al. Relations in biomedical ontologies. Genome Biol., 6(5):R46, 2005.
28. Spasic I., Ananiadou S., McNaught J., and Kumar A. Text mining and ontologies in biomedicine: making sense of raw text. Brief. Bioinform. 6(3):239–251, 2005.
29. Tari L., et al. BioQA. http://cbioc.eas.asu.edu/bioQA/v2/index.html, 2007.
30. The National Center for Biomedical Ontology. Available at: www.biontology.org, 2006.
31. Thorn C.F., Klein T.E., and Altman R.B. PharmGKB: the pharmacogenetics and pharmacogenomics knowledge base. Meth. Mol. Biol., 311:179–91, 2005.

Biomedical Image Data Types and Processing

Sameer Antani
National Institutes of Health, Bethesda, MD, USA

Synonyms

Data Types: Image, Video, Pixel, Voxel, Frame; Conceptual data types: Pixel, Point, Edge, Volume, Region of interest, Shape, Color, Texture, Feature; Format: Joint photographic experts group (JPEG), Digital imaging and communications in medicine (DICOM), JPEG2000, Imaging Technique: X-Ray, Magnetic resonance imaging (MRI), Computerized tomography (CT), Ultrasound, Positron emission tomography (PET), Nuclear magnetic resonance (NMR), Microscopy, Single photon emission computerized tomography (SPECT), Fluoroscopy; Image Processing: Compression, Wavelet compression, Functional mapping, Image reconstruction, 2D image processing, Texture analysis, Edge detection, 3D image processing, Surface detection, Image content analysis; Storage and Retrieval: Image databases, Content-based image retrieval (CBIR), Visual similarity, Feature indexing, Multimedia information retrieval

Definition

The entry term describes biomedical image types (X-Ray, CT, MR, PET) stored in a particular format (DICOM, JPEG) that can be processed for visual enhancement (windowing, leveling) or extraction of features for further processing as needed in specific applications (generate 3D volumes from 2D slices, Content-Based Image Retrieval (CBIR)).

Historical Background

Both image processing and databases have been studied for over four decades. Biomedical processing and storage systems have received significant attention within the last two decades. Imaging and image processing has gained significant importance in clinical medicine, biomedical research and education and correspondingly, biomedical image databases have also found increasing use in recent years. Images are still largely stored as flat files on file servers and made accessible via links stored in revelant database records. Significant progress has been made in image types, formats, and content being computed and stored in these databases. This information can help in processing and further use of these data. Image and image feature indexing is a topic of significant research interest with some specialized types are already in practical use.

Foundations

Imaging has taken on a very important role in clinical medicine, biomedical research, and education. Biomedical visual data are acquired using a variety of techniques: single frame images; 3D volumes composed of single frame images; and made and as time-synchronized multiple frames as video data. In addition, these data are acquired at varying scales ranging from gross anatomy to the cellular level. Each image data type has specific acquisition methods, set of image processing methods for feature extraction that aid in analysis for targeted purposes, compression and storage methods, and particular data handling methods [1]. The image database primarily serves as a file storage mechanism with various processes for analysis and retrieval traditionally included in utility applications. Image databases are typically found in practical use as "multimedia databases" or "multimedia information systems" in the form of Radiological Information Systems (RIS), Hospital Information Systems (HIS), and Picture Archiving and Communication Systems (PACS). Such systems link textual data to image data through file links stored in database records. Image databases imply use of image feature indexing strategies such as metric index trees, multidimensional data trees, spatial databases, and R-trees, for specialized use such as Content-Based Image Retrieval (CBIR) [8].

Image data types are challenging to define in standard terms such as integers, characters, strings, etc. An image is composed of pixels or in case of 3D images may be considered to composed of a set of elements of conceptual data type called voxels. Wikipedia (http://www.wikipedia.org) defines voxel as a portmanteau of words volumetric and pixel representing a unit element on a 3D image. Each such element (pixel or voxel) can be considered a complex data type as its content may be expressed using *n*-bits where *n* may be 8, 12, 16, 24, or 32. Typically 8-, 12-, and 16-bit images are gray scale images. A color pixel is typically 24-bits in depth comprising of three 8-bit channels for the additive color primaries (RED, GREEN, BLUE), though it is possible to have color images with other bit depths. This information is not natively stored in the image but needs to be exposed to the application through image metadata that may be stored in particular formats, such as a DICOM (http://dicom.nema.org/) on JPEG image header [6].

The images can be generated using a variety of techniques. Radiographic or X-Ray images, Computerized Tomography (CT), Magnetic Resonance images (MRI), Positron Emission Tomography (PET) images are examples of various imaging techniques. Techniques such as CT and MRI image the desired anatomical region in closely spaced sections. These sectional images can then be processed to create views along desired axes (axial, coronal, sagittal, oblique) as well as generate 3D volumetric data rendering.

Image processing is a term that includes functions and methods that focus on enhancement of images for improved human visualization or computer analysis, such as windowing, leveling, object edge detection, among others [3,9]. It also is synonymous with application of methods whereby features such as edges, textures, and surfaces, among others, can be computed for making measurements, computer-aided diagnosis, visual enhancement, identifying anatomical structures, determining unique image content signature, etc. For instance, using the above example of generating 3D volumetric data from 2D image slices, for a data set of MRI slices of the brain, it would be necessary to segment the edges from each 2D MRI image slice and register them with corresponding edges from the same anatomy in other slices. The next step would be to convert these edges into surfaces formed across these slices in order to generate 3D volumetric data.

With the increasing use of images in medical care and research, it becomes necessary make this data connect with other image and non-image data. The resulting database systems have evolved as PACS, RIS, and HIS and are commonly found in modern hospitals and medical centers. These database systems are capable of storing and retrieving text data, for example, a patient record containing test results and other medical history, along with image data. The systems may exist on a single computer, a local network of computers, or distributed over a wide area network. Variants of these systems developed for medical research studies can also correlate between different study participant information and keep track of longitudinal information.

In database processing it is often necessary to define the image to be of a particular type. This can assist in data and type verification as well as communication of semantics to other applications that may be using the data. Some database systems require storage of images in their native form as undefined BLOB data types while others, including most PACS, prefer to maintain references to image data files that are stored in traditional directory (folder) file structures. The choice between these approaches is largely determined by storage and computational efficiency and dependent on particular applications and solutions. In either scenario it is efficient to store the image metadata as database records. While this information may be available in image header files, it requires the additional step of accessing and opening each image file for any database operation involving images.

Image processing steps often result in features extracted from images. These features could be regions of pixels, measurements of color, texture, edges forming a shape, surfaces, etc. Each of these features could be standardized to be a data type or could use standard data types, e.g., a 3-channel color histogram could be represented using 3D arrays that could represent 3D histograms. Each such conceptual type may be used as a predefined data type. Other operations could include transforming the image or extracted characteristics from the spatial domain several into other domains through Fourier analysis or Wavelet transforms. Selecting these or other image processing methods is heavily dependent on the nature of the images and several methods are covered in [1,3,9]. Further, it may be necessary to compress the images in order to minimize data storage requirements or improve transmission efficiency over networks. These decisions must be made carefully in light of possible data loss found in typical implementations of common image compression methods such as JPEG or JPEG2000 [6].

In summary, biomedical image processing is critical to analysis and use of biomedical images for clinical medicine, research and education. These images may also have other associated images as well as text data. All this information is stored in biomedical databases that use a combination of image types, header information, image units, and content through the extracted features as data types. Images may be indexed through multidimensional indexing trees or be linked to flat files stored in a folder or accessed via a file server.

Key Applications

Multimedia Medical Information Systems: Medical Information Systems, like the PACS, hold medical data about patients, medical research study participants, etc. This medical data is typically heterogeneous comprising of electronic medical records containing the medical history, clinical notes, lab reports, and any acquired images. The text data can be fielded or exist as a block of free text. The image data can be from various sources and in a variety of formats. As such, a PACS can be considered a special type of a Multimedia Medical Information System. Another example of a Multimedia Medical Information System is the Multimedia Database Tool (MDT) being developed at the National Library of Medicine (NLM), part of the National Institutes of Health (NIH). The MDT is a Web-based system [5] with a MySQL back-end database that enables retrieval of text and image data in response to specific queries. For example, from a database of a cervical cancer study being conducted by the National Cancer Institute (NCI) one could query for "all women with cervical intraepithelial neoplasia 3 whose cytologic results are atypical squamous cells of undetermined significance."

The MDT was developed to access, evaluate, and collect information from thousands of uterine cervix images or cervigrams for cancer studies. It is one of two systems developed to work with these images: the boundary marking tool (BMT) for marking areas of particular importance in the images and the MDT for Internet dissemination of the images and to relate them with text data and information collected with the BMT. The MDT has system architecture capable of deploying color images and related information on the Web with minimal reprogramming. It has the flexibility to accept new datasets with the required customization performed at the level of a database administrator, rather than a programmer. It also has the capability of querying a database of text and images over the Web, of showing the query results consisting of multiple images and text data, and of exporting these results for statistical analysis. Additionally, the MDT is designed for data collection from remote users; given adequate password protection and anonymized data to assure patient privacy, experts worldwide will be able to access the data resource. Through the Internet, they will evaluate images and test data, perform analysis of the information, and record their evaluations in a central database. Additionally, because of its architecture, the MDT system can support a broad class of text and image databases. Therefore, the MDT is designed to grow if additional groups wish to merge their data on cervical cancer or to use it to manage their own multimedia collections.

Content-Based Image Retrieval (CBIR): While the MDT allows querying of text and images using text keywords and structured SQL-like queries, an alternative complementary form of image-based querying has gained significant research interest in recent years. This approach, called Content-Based Image Retrieval (CBIR), uses distinguishing features extracted from images to serve as indices. These features are then used to find images similar to an image query. For example, in the Spine Pathology Image Retrieval System (SPIRS) [4], developed at NLM, the boundary edges of individual vertebra are used to query the X-ray images of the spine captured as a part of the Second National Health and Nutrition Examination Survey (NHANES II). This shape feature was chosen because the pathology of interest is expressed along the vertebral boundary seen in the spine imaged on the sagittal plane. Perturbations along the boundary are indicative of pathology. In contrast, the set of cervigrams from the NCI cancer study is an example of an image class where color, texture, and location information are much more important than edge information. In this latter case, the pathology of interest is acetowhitened regions [10] on the cervical wall.

Finding similar images tends to be a very subjective matter and is heavily dependent on the extracted features from the images. Additionally, it is also dependent on the level of detail extracted. For example, one measure of image similarity is overall appearance of the image, which in case of X-Ray images, can be characterized by a histogram of pixel intensity levels in the image. While this *global* measure may be sufficient for overall similarity, it is insufficient in expressing *local* pathology that can only be captured by feature extraction within the region of interest. An intelligently implemented hierarchical strategy works better in a heterogeneous collection of images.

Given the subjective nature of image content and human perception, it is challenging to evaluate systems through system characteristics or reported performance measures alone. These results are sensitive to the kinds of image data that the system is operating on, extracted features, query capability, and several other gaps that need to be overcome for developing an "ideal" system. A framework for these gaps is discussed in [2]. Other medical systems are reviewed in [7]. The Cross Language Evaluation Forum (CLEF) benchmarking competition has evaluates image classification and image retrieval in a biomedical setting on an annual basis (CLEF-Campaign, http://www.clef-campaign.org) and permits use of text data commonly found with medical images to improve usability. Due to the transient nature of academic systems, and lack of detail available in commercial systems, such a venue provides valuable metrics for comparing various systems and assessing the state-of-the-art.

Cross-references

▶ Annotation-Based Image Retrieval
▶ 2D Shape Retrieval
▶ Feature-Based 3D Object Retrieval
▶ Feature Extraction for Content-Based Image Retrieval
▶ Image
▶ Image Content
▶ Image Database
▶ Image Management for Biological Data
▶ Image Metadata
▶ Image Representation
▶ Image Retrieval
▶ Image Retrieval and Relevance Feedback
▶ Image Salient Points and Features
▶ Image Segmentation
▶ Indexing and Similarity Search
▶ Indexing Metric Spaces
▶ Lossless Data Compression
▶ Low Level Image Content Analysis (Color, Texture Shape)
▶ Multimedia Data
▶ Multimedia Databases
▶ Multimedia Data Indexing
▶ Multimedia Data Storage
▶ Radiology Information Systems
▶ Relevance Feedback for Content-Based Information Retrieval
▶ Spatial Data Types
▶ Visual Content Analysis

Recommended Reading

1. Beutel J., Kundel H.L., and Van Metter R.L. (eds.). Handbook of Medical Imaging. Vols. 1, 2, and 3. SPIE Press, Bellingham, WA.
2. Deserno T.M., Antani S., Long R. Ontology of Gaps in Content-Based Image Retrieval. J Digital Imaging, February 2008.
3. Gonzales R.C. and Woods R.E. (eds.). Digital Image Processing (2nd edn.). Prentice Hall, Upper Saddle River, NJ.
4. Hsu W., Antani S., Long LR. SPIRS: a framework for content-based image retrieval from large biomedical databases. Medinfo, 12 (Pt 1):188–92, 2007.
5. Jeronimo J., Long L.R., Neve L., Bopf M., Antani S., Schiffman M. Digital tools for collecting data from cervigrams for research and training in colposcopy. J. Lower Genital Tract Dis., 10(1):16–25, 2006
6. Joint Photographic Experts Group (JPEG) http://www.jpeg.org/. American Medical Information Association (AMIA 2007), Chicago, November 2007, pp. 826–830.
7. Müller H., Michoux N., Bandon D., Geissbuhler A. A Review of Content-Based Image Retrieval Systems in Medical Applications – Clinical Benefits and Future directions. Int J Med Inform., 73(1):1–23, 2004.
8. Samet H. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufman. San Francisco, CA, 2006.
9. Sonka M., Hlavac V., and Boyle R. (eds.). Image Processing, Analysis, and Machine Vision (2nd edn.). PWS Publishing, Washington, DC.
10. Xue Z., Antani S.K., Long L.R., Jeronimo J., Thoma G.R. Investigating CBIR techniques for cervicographic images. In Proc. 2007 Annual Symposium of the American Medical Information Association, 2007, pp. 826–830.

Biomedical Informatics

▶ Implications of Genomics for Clinical Informatics
▶ Taxonomy: Biomedical Health Informatics

Biomedical Literature

▶ Biomedical Scientific Textual Data Types and Processing

Biomedical Scientific Textual Data Types and Processing

Li Zhou[1], Hua Xu[2]
[1]Partners HealthCare System Inc., Boston, MA, USA
Harvard Medical School, Boston, MA, USA
[2]Columbia University, New York, NY, USA

Synonyms

Scientific knowledge bases; Biomedical literature; MEDLINE/PubMed; Curation; Annotation; Information retrieval; Information retrieval models/metrics/operations; Indexing; Semi-structured text retrieval; Text extraction; Text mining; Web search and crawling

Definition

Vast amounts of biomedical scientific information and knowledge are recorded in text [1,7]. Various scientific textual data in the biomedical domain may generally be disseminated through the following resources [7,11]: biomedical literature (e.g., original reports and summaries of research in journals, books, reports, and guidelines), biological databases (e.g., annotations in gene/protein databases), patient records (e.g., clinical narrative reports), and web content.

A variety of techniques have been applied to identify, extract, manage, integrate and exploit knowledge from biomedical text. Some researchers [11] divide biomedical scientific textual data processing into three major activities as shown in Figure 1: information retrieval (IR), information extraction (IE), and text mining (TM).

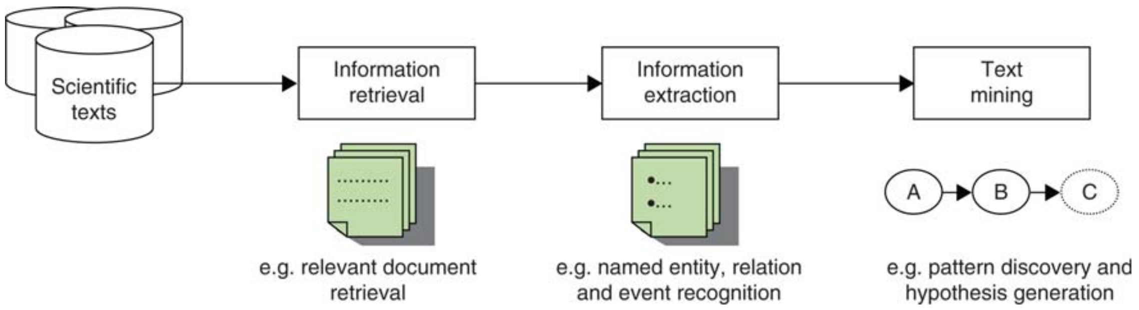
Information retrieval [2,11] is the science of indexing and searching for information particularly in text or other unstructured forms. The aim of IR is to identify relevant documents in response to a particular query, which forms the basis of any knowledge discovery process.

Information extraction [11] aims to identify and extract categorized or semantically well-defined data (entities, relations or events) from text documents in a certain domain, as well as to create structured knowledge bases that can be accessed by other informatics applications. Typical subtasks of IE are named entity, relation and event recognition (e.g., recognition of protein names and interactions between proteins), coreference (e.g., identifying whether a chain of noun phrases refer to the same object), and terminology extraction (e.g., finding the relevant terms for a given corpus).

Text mining (TM) [2,6] is the process of discovering and extracting interesting and non-trivial patterns and knowledge from unstructured text data. The primary goal of TM is to retrieve knowledge that is hidden in text, and to present the distilled knowledge to users in a concise form. Typical subtasks of TM may include pattern discovery, hypothesis generation, correlation discovery, etc. However, some researchers give a broader definition of TM which overlaps with IR and IE on certain tasks such as text classification, text clustering and named entity recognition.

Historical Background

Before the invention of computers, results of biomedical research have been published as journal or conference prints for a long time. Bibliographic databases that typically contained references to literature on library shelves was the first application of using computers to improve library service. MEDLINE (Medical Literature Analysis and Retrieval System Online) is the U.S. National Library of Medicine's (NLM) premier bibliographic database that contains over 16 million references to journal articles in life sciences with a concentration on biomedicine. As an online interactive searchable bibliographic database, MEDLINE was introduced in 1971 by NLM, to replace its previous version called MEDLARS (Medical Literature Analysis and Retrieval System). In 1997, PubMed was developed by the National Center for Biotechnology Information (NCBI) at the NLM, to provide free and efficient access to MEDLINE through the World Wide Web. Currently, MEDLINE/PubMed is probably the best-known biomedical literature reference database. The use of high-throughput experimental technologies has dramatically increased the pace of biomedical knowledge discovery. In 2006, over 623,000 references to published articles were added to the MEDLINE database. A large amount of effort has been spent on improving the performance of IR on the MEDLINE database. A distinctive feature of MEDLINE is that the records are indexed with NLM's Medical Subject Headings (MeSH).


Scientific texts
Information retrieval
Information extraction
Text mining
A
B
C
e.g. relevant document retrieval
e.g. named entity, relation and event recognition
e.g. pattern discovery and hypothesis generation

Biomedical Scientific Textual Data Types and Processing. Figure 1. Major stages of processing biomedical scientific textual data and relevant subtasks [11].

During the past decade, more and more full-text biomedical publications have become available on the Internet, though most of them have restricted access. In 1999, a bold new initiative called PubMed Central (PMC) was designed at the U.S. National Institutes of Health (NIH) to provide a central repository for literature in the life sciences with open access. To date, there are more than 300 journals that have joined PMC and they provide free access to their publications. BioMed Central, a commercial publisher, also provides free access to papers published in their journals.

Various text processing methods, such as natural language processing (NLP) and machine learning (ML) technologies, have been extensively studied in the domain of computer and information science. However, they have not been widely applied to biomedical text before the 1990s, largely due to the lack of available biomedical text. Starting at the mid 1990s, text processing technologies have been gradually applied to biomedical text on different tasks, such as information retrieval, biomedical entity recognition, text clustering and classification, and knowledge discovery.

Foundations

As mentioned above, biomedical scientific textual data processing applies methods and technologies from multiple disciplines, including linguistics, computer science, statistics, and so on. In general, the major stages of processing scientific textual data to exploit rich knowledge include retrieval of relevant documents, extraction of named entities and relations, and discovery of new knowledge. However, some processes may not follow the exact steps. This entry adopts a classification by Natarajan et al. [11] on major constituent technologies for knowledge discovery in text (see Fig. 1). Scientific fundamentals for each stage will be discussed in the following sections.

Information Retrieval

Conventional IR methods are often based on keyword queries, using Boolean logic, vector space models or probabilistic models [1,7]. One of the simplest forms of IR is to search keywords in documents that are indexed by a set of keywords. Search algorithms are used to identify the relevant documents based on the number of index keywords that match query keywords. One disadvantage of this approach is that the documents are determined either relevant or irrelevant. There is no further ranking. Vector space model is an algebraic model for representing text documents. When applying the model to IR, both the query and documents are represented as vectors, whose dimensions correspond to different terms in the query or documents. There are different methods to compute the weight of terms in the vectors and the tf-idf weighting is one of the best known schemes. Relevancy rankings of documents to a query can be determined by calculating the document similarities between the query and documents, via measurements such as cosine similarity of two vectors. Probabilistic models treat the process of document retrieval as a probabilistic inference and similarities are computed as probabilities that a document is relevant for a given query. An advantage of probabilistic model is that documents are ranked in decreasing order of their probability of being relevant to the query.

Information Extraction

Approaches to named entity recognition generally fall into three categories: lexicon based, rule based and statistically-based [1]. For example, part-of-speech tagging, inductive rule learning, decision trees, Bayesian model, support vector machines, as well as combined methods have been applied to this problem. For discovering relationships among entities, variant techniques have been used. Shallow parsing is often used to focus on specific parts of the text to analyze predefined words such as verbs and nouns. Some systems combine natural language processing and co-occurrence techniques, while others apply machine learning techniques.

Text Mining

Text classification and clustering are the most widely used techniques in biomedical text mining. While text classification is a form of learning from pre-classified examples, text clustering is referred to as unsupervised learning. Bayesian models were widely used in the early days. In recent years, more advanced machine learning methods, such as k-nearest neighbors, artificial neural networks, support vector machines, expectation maximization, and fuzzy clustering have been used. Logical inference models [13] have been applied to hypothesis generation which attempts to uncover relationships that are not present in the text but instead are inferred by the other existing relationships. There are a variety of techniques for knowledge discovery from biomedical text using graphs and knowledge models.

Key Applications

Information retrieval technologies have been used extensively to help users to find relevant articles that they are interested in. MEDLINE/PubMed has used various methods to improve the performance of searches. It provides keyword-based Boolean search to allow users to search by keywords, as well as document-based search, which implements the vector space model and could find documents for similar topics. With the availability of full-text articles online, more IR applications have tried to search full-text articles for detailed information. For example, the focus of the 2006 genomics track of the Text Retrieval Conference (TREC) [8] was to retrieve answers for biological questions from full text articles. The European Bioinformatics Institute (EBI) at the European Molecular Biology Laboratory (EMBL) has developed a biomedical information retrieval system called "CiteXplore," which combines literature search with text mining tools for biology. It also links biomedical literature sources to existing bioinformatics databases, such as SwissProt.

Although most of biomedical text processing tools are still in the research stage, some of them have shown potential uses. Many information extraction systems have been used to build knowledge bases from biomedical literature. Different approaches have been reported to extract relations among biomedical entities of interest (e.g., gene/protein). GENIES (GENomic Information Extraction System) [4] is an NLP-based system that extracts molecular pathways from literature. It semantically parses sentences into a structured form for relation extraction. PASTA (Protein Active Site Template Acquisition) [5] is a system that uses manually created templates to extract relationships between amino acid residues and their functions within a protein. The PreBIND [3] system uses Support Vector Machine (SVM) technology to locate protein-protein interaction data in the literature, thus to facilitate the curation process for protein databases. IR and IE systems are also combined to build more sophisticated systems to help specific tasks in biology, such as biological database curation tools that can help curators find related articles and identify critical findings from biological articles [14]. The iHOP [9] system extracts protein-relationship from the literature. It also includes advanced search modes for discovery and visualization of protein-protein-interaction network [9].

Another potential application of text mining tools is to discover new knowledge from literature, for example, helping biomedical researchers to generate new research hypotheses. ARROWSMITH and BITOLA [9,14] are two online tools that provide the function of literature-based knowledge discovery. ARROWSMITH detects indirect associations between concepts that are not directly linked in the literature. BITOLA is designed for disease candidate gene discovery by mining the bibliographic database MEDLINE.

Cross-references

- ▶ Data Mining
- ▶ Data, Text, and Web Mining in Healthcare
- ▶ Information Retrieval
- ▶ Text Mining of Biological Resources
- ▶ Text Mining

Recommended Reading

1. Chen H., Friedman C., Hersh W., and Fuller S.S. (eds.) Medical Informatics: Knowledge Management and Data Mining in Biomedicine. Springer, Secaucus, NJ, 2005.

2. Cohen A.M. and Hersh W.R. A survey of current work in biomedical text mining. Brief Bioinform., 6(1):57–71, 2005.
3. Donaldson I., Martin J., deBruijn B., Wolting C., Lay V., Tuekam B., Zhang S., Baskin B., Bader G., Michalickova K., et al. PreBIND and Textomy – mining the biomedical literature for protein-protein interactions using a support vector machine. BMC Bioinformatics, 4:11, 2003.
4. Friedman C., Kra P., Yu H., Krauthammer M., and Rzhetsky A. GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. Bioinformatics, 17 (Suppl 1):S74–S82, 2001.
5. Gaizauskas R., Demetriou G., Artymiuk P.J., and Willett P. Protein structures and information extraction from biological texts: the PASTA system. Bioinformatics, 19(1):135–143, 2003.
6. Hearst M. Untangling text data mining. In Proc. 27th Annual Meeting of the Assoc. for Computational Linguistics, 1999.
7. Hersh W. Information Retrieval: A Health and Biomedical Perspective. Springer, NY, 2003.
8. Hersh W., Cohen A., Roberts P., and Rekapalli H.K. TREC 2006 genomics track overview. In Proc. TREC 2006. Available at: http://trec.nist.gov/pubs/trec15/papers/GEO06. OVERVIEW. pdf
9. Hoffmann R. and Valencia A. A gene network for navigating the literature. Nat. Genet., 36(7):664, July 2004.
10. Hristovski D. and Peterlin B. Literature-based disease candidate gene discovery. In Proc. Medinfo. American Medical Informatics Association, Bethesda, 2004, p. 1649.
11. Natarajan J., Berrar D., Hack C.J., and Dubizky W. Knowledge discovery in biology and biotechnology texts: a review of techniques, evaluation strategies, and applications. Crit. Rev. Biotechnol., (25):31–52, 2005.
12. Smalheiser N. and Swanson D. Using ARROWSMITH: a computer-assisted approach to formulating and assessing scientific hypotheses. Comput. Methods Programs Biomed., 57:149–153, 1998.
13. Swanson D.R. Complementary structure in disjoint science literatures. In Proc. 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1990, pp. 280–289.
14. Yeh A.S., Hirschman L., and Morgan A.A. Evaluation of text data mining for database curation: lessons learned from the KDD Challenge Cup. Bioinformatics, 19 (Suppl 1):i331–i339, 2003.

Biostatistics and Data Analysis

MEHMET M. DALKILIÇ
Indiana University, Bloomington, IN, USA

Definition

Biostatistics is the application of probability and statistical techniques to the biological sciences. Probability has played a significant role in areas like genetics where combinatorics validate conjectures about the relationships of genes and the environment. Recently, combinatorics has become one of the main approaches to solving problems *e.g.*, motif discovery in bioinformatics. In the nineteenth century, well-known biologists like Herman von Helmholtz advocated that biological phenomenon could be understood using techniques in the physical sciences (remnants of this view still are present today). That approach, together with the "vitalism" movement [3] impeded the use of statistics. By the beginning of this century, however, statistics has become *de rigueur* in virtually all biological publications.

Historical Background

Although statistics as a mathematical area can be traced further, biostatistics is often associated with the work of Francis Galton (1822–1911). His major contribution was demonstrating that statistical methods could be beneficial in biology. Carrying on this tradition were Karl Pearson (1857–1936) and R.A. Fisher (1890–1962). While a number of standard applications and techniques have not changed for more than half-a-century, the availability of computing has made some (heretofore infeasible) techniques available. The classic text in this area is by Sokal and Rohlf, "Biometry" [5] now in its third edition. It should be pointed out that the demarcation often cited between "frequentists" [6] and "Bayesians" [1] is fairly well evident in biostatistics. In the former, physical, repeatable, identical, and random experiments can be associated to a mathematical limit *e.g.*, $P(heads) = \lim_{flips\to\infty} \frac{\#heads}{flips} = 1/2$ is the probability of getting heads when flipping a coin. Bayesians, on the other hand presume probability to be a degree of belief (or subjective probability). This can be written as *posterior* $\propto$ *prior* $\times$ *likelihood*, in symbols $P(H|E) \propto P(H)\, P(E|H)$. The distinctive feature is that Bayesian statisticians will associate values with the *posterior*, *hypothesis*, and *prior* (likelihood), whereas non-Bayesian statisticians will only consider the hypotheses in constrained settings. Classical statistics is often the primary choice. However, Bayesian techniques are becoming increasingly popular in systems biology–biology that integrates and evaluates disparate data usually in the form of very large graphs. In these graphs, nodes are typically genes and edges evidence of relationship.

Foundations

As explained eloquently in [5], there is a deeper philosophical debate centered on whether biological phenomena can ultimately be modeled using deterministic approaches. That debate aside, the use of statistics is continuing to grow as the amounts of biological data grow. Indeed, recent technologies (high-throughput) are producing several orders of magnitude more data in comparison to traditional approaches. Statistical tools, from this perspective, becomes a necessity. Furthermore, a growing number of biologists now believe that there is benefit in examining data normally not studied within one's own specialized domain–this is called a systems biology. While the foundations for systems biology are still under development it is clear that it will rely heavily on Bayesian reasoning. Typically the disparate experimental, textual, *etc.* data are structured as a directed, acyclic graph where nodes are random variables and edges are effects. The basic attempt is to discover sets of independent variables and form a better understanding of the joint probability. For example, one might take the some 14,000 *Drosophila m.* genes and presume a joint distribution $P(X_1, X_2, \ldots, X_{14000})$ where X_i represents the probability that gene i has some expression level. Bayesians build large graphs relying on conditional probabilities and so-called "separations" that expose which random variables are independent of one another. They then examine the behavior of the graph under particular conditions. The interested reader is guided to [4]. Topics studied in biostatistics are too numerous to list (for example multivariate regression, analysis of covariance, linear discriminant analysis, principal component analysis, and so forth; therefore, a sample that reflects the kind of tools that are used and most prevalent techniques will be given. Most of the biostatistics used is parameteric; the models result from human expertise. This is opposed to nonparametric models (or data-driven) that rely on the data itself. With the availability of cheap, fast computation, however, the use of nonparametric models has exploded. Given a set of data $\mathbf{x} = x_1, x_2, \ldots, x_n$, the probability $P(\mathbf{x} | \theta_1, \theta_2, \ldots, \theta_m)$ is parametric if m is not dependent on n; otherwise it is non-parametric. Linear Regression is a parametric model that presumes a linear relationship between two random variables (rv) both having Gaussian distributed noise. It is presumed the variables are real valued. If rv Y is a function of rv X, then the phrase, "A regression of Y on X," is used. One is determining the optimal coefficients β_0, β_1 given data $D = \{\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle\}$ on function $Y = \beta_1 X + \beta_0$. Regression, though simple, provides a first step in understanding the relationship between to rvs. Regression can be used to predict, adjust, explain, *etc.* and is common in biostatistics. Regression produces an optimal linear relationship between rvs Y and X. Typically, one minimizes the squares of the residuals–the difference between the hypothetical point and observed point. The use is so ubiquitous that virtually every mathematical or statistical package has this available. As pointed out many times in the area, because two rvs are *not* linearly related, does not mean they are *not* functionally related.

Correlation, related to Linear Regression, is as often used. It is so similar in so many ways to Linear Regression that the literature will often have the use of one, when in fact, it is the other that is warranted or even makes sense. Succinctly, Regression examines how one rv depends on another; Correlation examines how two rvs behave together–in concert or more formally *covary*. Correlation is related to moments and is often called product-moments. One of the most popular is the Pearson product-moment $\rho = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$ where σ_i is the variance of the joint (numerator) and individual rvs (denominator). The value $\rho \in [-1, 1]$ where as $|\rho| \approx 1$, the variables appear to have an association. As ρ approaches 0, this indicates they do not. The positive and negative score reflects the direction of the association. Another popular correlation examines how pairs of pairs of values behave called ranks. Kendall's τ examines two pairs of pairs $\langle x_i, y_y \rangle, \langle x_j, y_j \rangle$ observing if *both* values of one pair are greater or smaller than the other pair; if so, then the pair of pairs is called concordant. If this is not satisfied, then the pair of pairs is discordant. The formula for Kendall's $\tau = \frac{N}{n(n-1)}$ where N is the count of ranks and n the sample size. It should be pointed out this is a nonparametric statistic and value $\tau \in (-1, 1)$.

A good deal of analysis is done with simple hypothesis testing of sample statistics. Because an entire population can be seldom checked, to establish a degree of certainty about a property we use sampling techniques (statistics). Sampling is done randomly to hopefully reflect the underlying distribution, especially if it is not known. If the sample is consistent with the conjecture about the existence of a property (called a hypothesis), then the hypothesis is said to be *accepted*; if not, it is *rejected*. There exists two disparate hypotheses that are simultaneously proposed when doing this analysis. The Null hypothesis, typically denoted H_0, is that the property was observed likely through chance. The Alternative hypothesis, typically denoted by H_a, is that the property is *not* due to chance. While recapitulating this process in its entirety is not appropriate here, we summarize the following sequence of standard steps that are typically followed for such analysis: **(0)** State the two hypotheses H_0 and H_a. It must be the case that only one can be true; **(1)** Establish the protocol for acquiring and using sample data. The decision usually depends on a chosen test statistic (a real-valued function of the sample); **(3)** Calculate the test statistic's value; **(4)** Check whether the statistic's value is likely to have occurred by chance or not typically using tables [2].

There are two types of errors that can result in this procedure. A Type I error will reject a null hypothesis when it is actually true. The significance level is the probability of committing this error and is often denoted by α. A Type II error is accepting a null hypothesis when it is actually false. The power of the test is the probability of not committing a Type II error. A *P*-value is, in a sense, an extreme case in examining H_0. It will reflect the strength of the evidence. *P*-values are very useful, since one can apply any number of significance levels reflecting confidence in the data.

ANOVA (analysis of variance), developed by Fisher almost a century ago remains a popular tool that examines differences in populations means $\mu = \Sigma_{i=1}^{N} \frac{X_i}{N}$, for values $X_1, X_2, \ldots, X_N$ one of the measures of central tendency of a population (*e.g.*, median, mode). Population variance, $\sigma^2 = \Sigma_{i=1}^{N} \frac{(X_i - \mu)^2}{N}$, measures the spread of values. There are two models of ANOVA, Model I and Model II, which form H_0 and H_a based on mean and variance, respectively. The interested reader is guided to [5] for an in-depth presentation.

Key Applications

The initial application of biostatistics was to study evolution and natural selection. It now plays an essential role in sequence and genome analysis, protein structure prediction, proteomics, phylogenetics, *etc.* A current challenge for data analysis is to consider extensions to relational data, to include probabilistic data and uncertainty, as they their roles in biological inquiry. Database researchers, one of the challenges is to move in thinking from a Boolean model (relational) to one that involves probability and uncertainty, conflicting data, non-replicable data, and data that has orders of magnitude more attributes than tuples. Data in biology is often un-normalized and may lack primary key identifiers normalization.

Future Directions

Biologists and computer scientists use different paradigms when considering data and analysis. Biologists are reductionists and focus on tightly constrained problems, whereas computer scientists pursue generic technological solutions that can be applied to multiple problems. Researchers in data management and biostatistics must keep these differences in mind as they move towards developing useful yet generic tools to serve biologists into the future. The future of biostatistics will depend directly on the computational resources available. Typically, two different approaches to problems are taken: combinatorial (or enumerative) or statistical. The former is usually very fast, but lacks the ability to discover nuanced relationships. Statistical approaches are computationally expensive, *e.g.*, Expectation-Maximization, but with increasingly powerful computers and clusters, this computational impediment is slowly eroding.

Cross-reference

- ▶ Annotation
- ▶ Biomedical Data/Content Acquisition, Curation
- ▶ Clustering
- ▶ Curation
- ▶ Data Quality Assessment
- ▶ F-measure
- ▶ Metadata-based Query Processing for Statistical Data
- ▶ Principal Component Analysis
- ▶ Probabilistic Databases
- ▶ Spectral Clustering
- ▶ Taxonomy: Biomedical Health Informatics
- ▶ Term Weighting
- ▶ Text Analytics
- ▶ Text Compression
- ▶ Two-Poisson Model
- ▶ Uncertainty in Events

Recommended Reading

1. Lee P.M. Bayesian Statistics, 2nd Ed. Arnold, 2003.
2. Lindley D.V. and Scott W.F. New Cambridge Statistics Tables 2nd Ed. Cambridge University Press, 1995.

3. Myers C.S. Vitalism: A Brief Historical and Critical Review. Mind, 9(35):319–331, 1900.
4. Neapolitan R.E. Learning Bayesian Networks. Prentice Hall, 2003.
5. Sokal R. and Rohlf F.J. *Biometry.* W.H. Freeman and Company, NY, 3rd edition, 1995.
6. von Mises R. Probability, Statistics, and Truth. 1939. Translated by J. Neyman and D. Scholl and E. Rabinowitsch.

BIR Model

▶ Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model

Bit Vector Join

▶ Semijoin

Bi-temporal Access Methods

▶ Bi-Temporal Indexing

Bitemporal Algebras

▶ Temporal Algebras

Bitemporal Data Model

▶ Temporal Data Models

Bi-Temporal Indexing

Mirella M. Moro[1], Vassilis J. Tsotras[2]
[1]The Federal University of Rio Grande dosul, Porto Alegre, Brazil
[2]University of California-Riverside, Riverside, CA, USA

Synonyms

Bi-temporal access methods

Definition

A bi-temporal index is a data structure that supports both temporal time dimensions, namely, transaction-time (the time when a fact is stored in the database) and valid-time (the time when a fact becomes valid in reality). The characteristics of the time dimensions supported imply various properties that the bi-temporal index should have to be efficient. As traditional indices, the performance of a temporal index is described by three costs: (i) storage cost (i.e., the number of pages the index occupies on the disk), (ii) update cost (the number of pages accessed to perform an update on the index; for example when adding, deleting or updating a record), and (iii) query cost (the number of pages accessed for the index to answer a query).

Historical Background

Most of the early work on temporal indexing has concentrated on providing solutions for transaction-time databases. A basic property of transaction-time is that it always *increases*. Each newly recorded piece of data are time-stamped with a new, larger, transaction time. The immediate implication of this property is that previous transaction times *cannot* be changed. Hence, a transaction-time database can “rollback” to, or answer queries for, any of its previous states.

On the other hand, a valid-time database maintains the entire temporal behavior of an enterprise as best known now. It stores the current knowledge about the enterprise’s past, current or even future behavior. If errors are discovered about this temporal behavior, they are corrected by modifying the database. In general, if the knowledge about the enterprise is updated, the new knowledge modifies the existing one. When a correction or an update is applied, previous values are not retained. It is thus not possible to view the database as it was before the correction/update.

By supporting both valid and transaction time, a bi-temporal database combines the features of the other temporal database types. While it keeps its past states, it also supports changes anywhere in the valid time domain. Hence, the overlapping and persistent methodologies proposed for transaction-time indexing can be applied [5,6,9]. The difference with transaction-time indexing is that the underlying access method should be able to dynamically manage intervals (like an R-tree, a quad-tree etc.). For a worst-case comparison of temporal access methods, the reader is referred to [7].

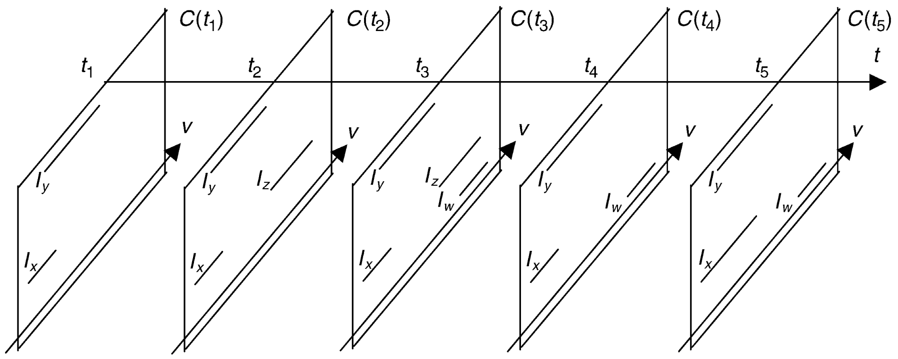
Foundations

When considering temporal indexing, it is important to realize that the valid and transaction time dimensions are *orthogonal* [3]. While in various scenarios it may be assumed that data about a fact is entered in the database at the same time as when it happens in the real world (i.e., valid and transaction time coincide), in practice, there are many applications where this assumption does not hold. For example, data records about the sales that occurred during a given day are recorded in the database at the end of the day (when batch processing of all data collected during the day is performed). Moreover, a recorded valid time may represent a later time instant than the transaction time when it was recorded. For example, a contract may be valid for an interval that is later than the (transaction) time when this information was entered in the database. The above properties are critical in the design of a bi-temporal access method since the support of both valid and transaction time affects directly the way records are created or updated. Note that the term "interval" is used here to mean a "convex subset of the time domain" (and not a "directed duration"). This concept has also been named a "period"; in this discussion however, only the term "interval" is used.

The reader is referred to the entry on Transaction-Time Indexing, in which a transaction time database was abstracted as an evolving collection of objects; updates arrive in increasing transaction-time order and are always applied on the latest state of this set. In other words, previous states cannot be changed. Thus a transaction-time database represents and stores the database activity; objects are associated with intervals based on this database activity. In contrast, in the chapter on Valid-Time Indexing, a valid-time database was abstracted as an evolving collection of interval-objects, where each interval represents the validity interval of an object. The allowable changes in this environment are the addition/deletion/modification of an interval-object. A difference with the transaction-time abstraction is that the collection's evolution (past states) is *not* kept. Note that when considering the valid time dimension, changes do not necessarily come in increasing time order; rather they can affect *any* interval in the collection. This implies that a valid-time database can correct errors in previously recorded data. However, only a single data state is kept, the one resulting after the correction is applied.

A bi-temporal database has the characteristics of both approaches. Its abstraction maintains the evolution (through the support of transaction-time) of a dynamic collection of (valid-time) interval-objects. Figure 1 offers a conceptual view of a bi-temporal database. Instead of maintaining a single collection of interval-objects (as a valid-time database does) a bi-temporal database maintains a sequence of such collections $C(t_i)$ indexed by transaction-time. Assume that each interval I represents the validity interval of a contract in a company. In this environment, the user can represent how the knowledge about company contracts evolved. In Fig. 1, the t-axis (v-axis) corresponds to transaction (valid) times. At transaction time t_1, the database starts with interval-objects I_x and I_y. At t_2, a new interval-object I_z is recorded, etc. At t_5 the valid-time interval of object I_x is modified to a new length.

When an interval-object I_j is inserted in the database at transaction-time t, a record is created with the object's surrogate (contract_no I_j), a valid-time interval (contract duration), and an initial transaction-time interval $[t, UC]$. When an object is inserted, it is not yet known if it (ever) will be updated. Therefore, the right endpoint of the transaction-time interval is filled with



Bi-Temporal Indexing. Figure 1. A bi-temporal database.

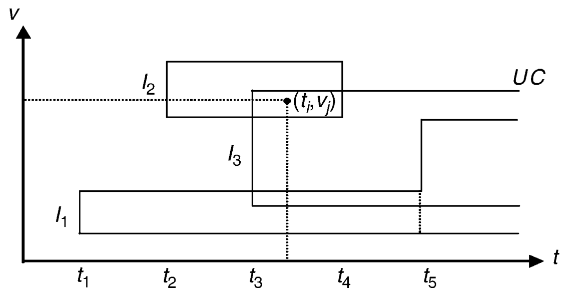
the variable *UC* (Until Changed), which will be changed to another transaction time if this object is later updated. For example, the record for interval-object I_z has transaction-time interval $[t_2, t_4]$, because it was inserted in the database at transaction-time t_2 and was "deleted" at t_4. Note that the collections $C(t_3)$ and $C(t_4)$ correspond to the collections C_a and C_b of Fig. 1 in the Valid-Time Indexing chapter, assuming that at transaction-time t_4 the erroneous contract I_z was deleted from the database.

Based on the above discussion, an index for a bi-temporal database should: (i) store past states, (ii) support addition/deletion/modification changes on the interval-objects of its current logical state, and (iii) efficiently access and query the interval-objects on any state.

Figure 1 summarizes the differences among the various database types. Each collection $C(t_i)$ can be thought of on its own, as a separate valid-time database. A valid-time database differs from a bi-temporal database since it keeps *only one* collection of interval-objects (the latest). A transaction-time database differs from a bi-temporal database in that it maintains the history of an evolving set of *plain*-objects instead of *interval*-objects. A transaction-time database differs from a conventional (non-temporal) database in that it also keeps its *past* states instead of only the latest state. Finally, the difference between a valid-time and a conventional database is that the former keeps *interval*-objects (and these intervals can be queried).

There are three approaches that can be used for indexing bi-temporal databases.

Approach 1: The first one is to have each bi-temporal object represented by a "bounding rectangle" created by the object's valid and transaction-time intervals, and to store it in a conventional multi-dimensional structure like the R-tree. While this approach has the advantage of using a single index to support both time dimensions, the characteristics of transaction-time create a serious overlapping problem [5]. A bi-temporal object with valid-time interval I that is inserted in the database at transaction time t, is represented by a rectangle with a transaction-time interval of the form $[t, UC]$. All bi-temporal objects that have not been deleted (in the transaction sense) will share the common transaction-time endpoint *UC* (which in a typical implementation, could be represented by the largest possible transaction time). Furthermore, intervals that remain unchanged will create



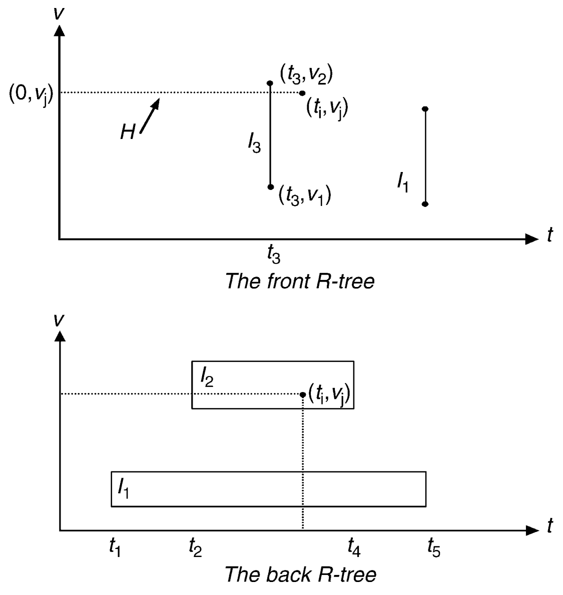
Bi-Temporal Indexing. Figure 2. The bounding-rectangle approach for bi-temporal objects.

long (in the transaction-time axis) rectangles, a reason for further overlapping. A simple bi-temporal query that asks for all valid time intervals that at transaction time t_i contained valid time v_j, corresponds to finding all rectangles that contain point (t_i, v_j).

Figure 2 illustrates the bounding-rectangle approach; only the valid and transaction axis are shown. At t_5, the valid-time interval I_1 is modified (enlarged). As a result, the initial rectangle for I_1 ends at t_5, and a new enlarged rectangle is inserted ranging from t_5 to *UC*.

Approach 2: To avoid overlapping, the use of two R-trees has also been proposed [5]. When a bi-temporal object with valid-time interval I is added in the database at transaction-time t, it is inserted at the *front* R-tree. This tree keeps bi-temporal objects whose right transaction endpoint is unknown. If a bi-temporal object is later deleted at some time $t' > t$, it is physically deleted from the front R-tree and inserted as a rectangle of height I and width from t to t' in the *back* R-tree. The back R-tree keeps bi-temporal objects with known transaction-time interval. At any given time, all bi-temporal objects stored in the front R-tree share the property that they are alive in the transaction-time sense. The temporal information of every such object is thus represented simply by a vertical (valid-time) interval that "cuts" the transaction axis at the transaction-time when this object was inserted in the database. Insertions in the front R-tree objects are in increasing transaction time while physical deletions can happen anywhere on the transaction axis.

In Fig. 3, the two R-trees methodology for bi-temporal data are divided according to whether their right transaction endpoint is known. The scenario of Fig. 2 is presented here (i.e., after time t_5 has elapsed). The query is then translated into an interval intersection and a point enclosure problem. A simple



Bi-Temporal Indexing. Figure 3. The two R-tree methodology for bi-temporal data.

bi-temporal query that asks for all valid time intervals which contained valid time v_j at transaction time t_i, is answered with two searches. The back R-tree is searched for all rectangles that contain point (t_i, v_j). The front R-tree is searched for all vertical intervals that intersect a horizontal interval *H* that starts from the beginning of transaction time and extends until point t_i at height v_j.

When an R-tree is used to index bi-temporal data, overlapping may also incur if the valid-time intervals extend to the ever-increasing *now*. One approach could be to use the largest possible valid-time timestamp to represent the variable *now*. In [2] the problem of addressing both the *now* and *UC* variables is addressed by using bounding rectangles/regions that increase as the time proceeds. A variation of the R-tree, the GR-tree is presented. The index leaf nodes capture the exact geometry of the bi-temporal regions of data. Bi-temporal regions can be static or growing, rectangles or stair-shapes. Two versions of the GR-tree are explored, one using minimum bounding rectangles in non-leaf nodes, and one using minimum bounding regions in non-leaf nodes. Details appear in [2].

Approach 3: Another approach to address bi-temporal problems is to use the notion of partial persistence [1,4]. This solution emanates from the abstraction of a bi-temporal database as a sequence of collections *C*(*t*) (in Fig. 1) and has two steps. First, a good index is chosen to represent each *C*(*t*). This index must support dynamic addition/deletion of (valid-time) interval-objects. Second, this index is made partially persistent. The collection of queries supported by the interval index structure implies which queries are answered by the bi-temporal structure. Using this approach, the Bi-temporal R-tree that takes an R-tree and makes it partially persistent was introduced in [5].

Similar to the transaction-time databases, one can use the “overlapping” approach [3] to create an index for bi-temporal databases. It is necessary to use an index that can handle the valid-time intervals and an overlapping approach to provide the transaction-time support. Multi-dimensional indexes can be used for supporting intervals. For example, an R-tree or a quad-tree. The Overlapping-R-tree was proposed in [6], where an R-tree maintains the valid time intervals at each transaction time instant. As intervals are added/deleted or updated, overlapping is used to share common paths in the relevant R-trees. Likewise, [9] proposes the use of quad-trees (which can also be used for spatiotemporal queries).

There are two advantages in “viewing” a bi-temporal query as a “partial persistence” or “overlapping” problem. First, the valid-time requirements are disassociated from the transaction-time ones. More specifically, the valid time support is provided from the properties of the R-tree while the transaction time support is achieved by making this structure “partially persistent” or “overlapping.” Conceptually, this methodology provides fast access to the *C*(*t*) of interest on which the valid-time query is then performed. Second, changes are always applied to the most current state of the structure and last until updated (if ever) at a later transaction time, thus avoiding the explicit representation of variable *UC*. Considering the two approaches, overlapping has the advantage of simpler implementation, while the partial-persistence approach avoids the possible logarithmic space overhead.

Key Applications

The importance of temporal indexing emanates from the many applications that maintain temporal data. The ever increasing nature of time imposes the need for many applications to store large amounts of temporal data. Accessing such data specialized indexing techniques is necessary. Temporal indexing has offered many such solutions that enable fast access.

Cross-references

▶ B+-Tree
▶ Rtree
▶ Temporal Database
▶ Transaction-Time Indexing
▶ Valid-Time Indexing

Recommended Reading

1. Becker B., Gschwind S., Ohler T., Seeger B., and Widmayer P. An asymptotically optimal multiversion B-tree. VLDB J., 5 (4):264–275, 1996.
2. Bliujute R., Jensen C.S., Saltenis S., and Slivinskas G. R-tree based indexing of now-relative bitemporal data. In Proc. 24th Int. Conf. on Very Large Data Bases, 1998, pp. 345–356.
3. Burton F.W., Huntbach M.M., and Kollias J.G. Multiple generation text files using overlapping tree structures. Comput. J., 28(4):414–416, 1985.
4. Driscoll J.R., Sarnak N., Sleator D.D., and Tarjan R.E. Making data structures persistent. J. Comput. Syst. Sci., 38(1):86–124, 1989.
5. Kumar A., Tsotras V.J., and Faloutsos C. Designing access methods for bitemporal databases. IEEE Trans. Knowl. Data Eng., 10(1):1–20, 1998.
6. Nascimento M.A. and Silva J.R.O. Towards historical R-trees. In Proc. 1998 ACM Symp. on Applied Computing, 1998, pp. 235–240.
7. Salzberg B. and Tsotras V.J. A comparison of access methods for time-evolving data. ACM Comput. Surv., 31(2):158–221, 1999.
8. Snodgrass R.T. and Ahn I. Temporal databases. IEEE Comput., 19(9):35–42, 1986.
9. Tzouramanis T., Vassilakopoulos M., and Manolopoulos Y. Overlapping linear quadtrees and spatio-temporal query processing. Comput. J., 43(4):325–343, 2000.

Bitemporal Interval

Christian S. Jensen[1], Richard T. Snodgrass[2]
[1]Aalborg University, Aalborg, Denmark
[2]University of Arizona, Tucson, AZ, USA

Definition

Facts in a bitemporal database may be timestamped by time values that are products of time intervals drawn from two orthogonal time domains that model valid time and transaction time, respectively. A *bitemporal interval* then is given by an interval from the valid-time domain and an interval from the transaction-time domain, and denotes a rectangle in the two-dimensional space spanned by valid and transaction time.

When associated with a fact, a bitemporal interval then identifies an interval (valid time) during which that fact held (or holds or will hold) true in reality, as well as identifies an interval (transaction time) when that belief (that the fact was true during the specified valid-time interval) was held, i.e., was part of the current database state.

Key Points

In this definition, a time interval denotes a convex subset of the time domain. Assuming a discrete time domain, a bitemporal interval can be represented with a non-empty set of bitemporal chronons or granules.

Cross-references

▶ Bitemporal Relation
▶ Chronon
▶ Temporal Database
▶ Temporal Granularity
▶ Time
▶ Time Domain
▶ Time Interval
▶ Transaction Time

Recommended Reading

1. Bettini C., Dyreson C.E., Evans W.S., Snodgrass R.T., and Wang X.S. A glossary of time granularity concepts. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399. Springer, Berlin Heidelberg New York, 1998, pp. 406–413.
2. Jensen C.S. and Dyreson C.E. (eds.). A consensus glossary of temporal database concepts–February 1998 version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399. Springer, Berlin Heidelberg New York, 1998, pp. 367–405.

Bitemporal Relation

Christian S. Jensen[1], Richard T. Snodgrass[2]
[1]Aalborg University, Aalborg, Denmark
[2]University of Arizona, Tucson, AZ, USA

Synonyms

Temporal relation; Fully temporal relation; Valid-time and transaction-time relation

Definition

A *bitemporal relation* captures exactly one valid time aspect and one transaction time aspect of the data it contains. This relation inherits its properties from valid-time relations and transaction-time relations. There are no restrictions as to how either of these temporal aspects may be incorporated into the tuples.

Key Points

In this definition, "bi" refers to the capture of exactly two temporal aspects. An alternative definition states that a bitemporal relation captures one or more valid times and one or more transaction times. In this definition, "bi" refers to the existence of exactly two types of times.

One may adopt the view that the data in a relation represents a collection of logical statements, i.e., statements that can be assigned a truth values. The valid times of these so-called facts are the times when these are true in the reality modeled by the relation. In cases where multiple realities are perceived, a single fact may have multiple, different valid times. This might occur in a relation capturing archaeological facts for which there no agreements among the archaeologists. In effect, different archaeologists perceive different realities.

Transaction times capture when database objects are current in a database. In case an object migrates from one database to another, the object may carry along its transaction times from the predecessor databases, termed *temporal generalization*. This then calls for relations that capture multiple transaction times.

The definition of bitemporal is used as the basis for applying bitemporal as a modifier to other concepts such as "query language." A query language is bitemporal if and only if it supports any bitemporal relation. Hence, most query languages involving both valid and transaction time may be characterized as bitemporal.

Relations are named as opposed to databases because a database may contain several types of relations. Most relations involving both valid and transaction time are bitemporal according to both definitions.

Concerning synonyms, the term "temporal relation" is commonly used. However, it is also used in a generic and less strict sense, simply meaning any relation with time-referenced data.

Next, the term "fully temporal relation" was originally proposed because a bitemporal relation is capable of modeling both the intrinsic and the extrinsic time aspects of facts, thus providing the "full story." However, this term is no longer used.

The term "valid-time and transaction-time relation" is precise and consistent with the other terms, but is also lengthy.

Cross-references

- ▶ Bitemporal Interval
- ▶ Temporal Database
- ▶ Temporal Generalization
- ▶ Transaction Time
- ▶ Valid Time

Recommended Reading

1. Jensen C.S. and Dyreson C.E. (eds.). A consensus glossary of temporal database concepts–February 1998 version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399. Springer, Berlin Heidelberg New York, 1998, pp. 367–405.

Bitmap Index

CHEE YONG CHAN
National University of Singapore, Singapore, Singapore

Definition

An index on an attribute provides an efficient way to access data records associated with a given range of values for the indexed attribute. Typically, an index stores a list of RIDs (called a RID-list) of all the records associated with each distinct value v of the indexed attribute. In a *bitmap index*, each RID-list is represented in the form of a bit vector (i.e., bitmap) where the size of each bitmap is equal to the cardinality of the indexed relation, and the ith bit in each bitmap corresponds to the ith record in the indexed relation. The simplest bitmap index design is the *Value-List index*, which is illustrated in Fig. 1b for an attribute A of a 12-record relation R in Fig. 1a. In this bitmap index, there is one bitmap E^v associated with each attribute value $v \in [0,9]$ such that the ith bit of E^v is set to 1 if and only if the ith record has a value v for the indexed attribute.

a

	A
1	3
2	2
3	1
4	2
5	8
6	2
7	9
8	0
9	7
10	5
11	6
12	4

b

E^9	E^8	E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0

c

R^8	R^7	R^6	R^5	R^4	R^3	R^2	R^1	R^0
1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0	0
1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0

d

I_1^4	I_1^3	I_1^2	I_1^1	I_1^0
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	1	1	1
1	0	0	0	0
0	0	1	1	1
0	0	0	0	0
0	0	0	0	1
1	1	0	0	0
1	1	1	1	0
1	1	1	0	0
1	1	1	1	1

Bitmap Index. Figure 1. Examples of bitmap indexes. (a) indexed attribute *A* (b) equality-encoded index (or value-list index) (c) range-encoded index (or base-10 bit-sliced index) (d) interval-encoded index.

Historical Background

The idea of using bitmap indexes to speed up selection predicate evaluation has been recognized since the early 1970s [4]. Some early implementations of bitmap processing techniques include PC DBMSs (e.g., FoxPro, Interbase), a scientific/statistical database application developed at Lawrence Berkeley Laboratory [11], and Model 204, which is a commercial DBMS for the IBM mainframe [7].

The main advantage of using a bitmap index is the CPU efficiency of bitmap operations (AND, OR, XOR, NOT). Furthermore, compared to RID-based indexes, bitmap indexes are more space-efficient for attributes with low cardinality and more I/O-efficient for evaluating selection predicates with low selectivities. For example, assuming each RID requires four bytes of storage and ignoring any compression, bitmap indexes are more space-efficient if the attribute cardinality is less than 32, and reading a bitmap is more I/O-efficient than reading a RID-list if the selectivity factor of the selection predicate is more than $\frac{1}{32}(\approx 3.2\%)$. Another advantage of bitmap indexes is that they are very amenable to parallelization due to the equal-sized bitmaps and the nature of the bitwise operations.

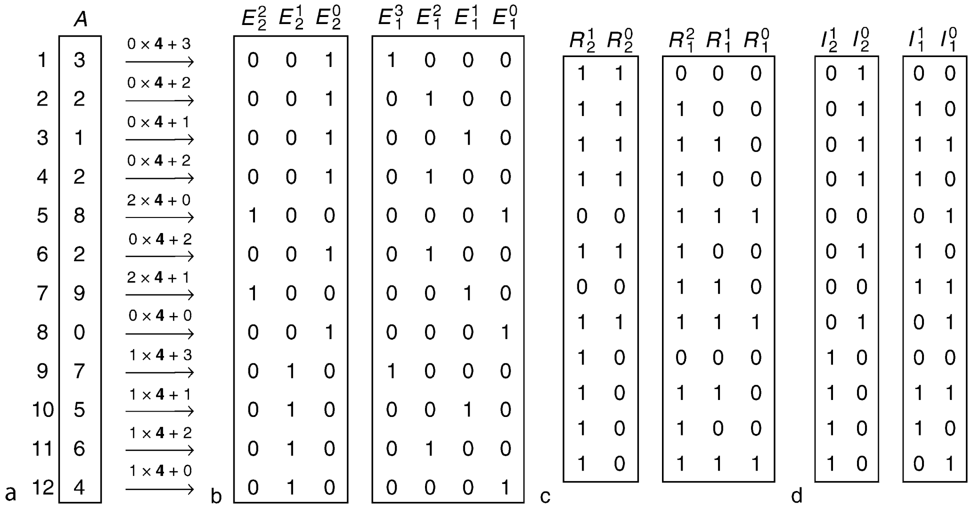
A variety of bitmap index designs have been proposed since the early days. Besides the simple Value-List index illustrated in Fig. 1b, another early bitmap index design is the *Bit-Sliced index (BSI)* which is implemented in Model 204 and Sybase IQ [7]. A BSI for an attribute with a cardinality of C consists of $k = \lceil log_2(C) \rceil$ bitmaps, with one bitmap associated with each bit in the binary representation of C. Compared to the Value-List index, the BSI is more space-efficient with an attribute value v being encoded by a string of k bits corresponding to its binary representation. The BSI design can be generalized to use a non-binary base b such that it consists of $k(b - 1)$ bitmaps $\{B_i^j : 1 \leq i \leq k, 0 \leq j < b\}$, where $k = \lceil log_b(C) \rceil$. Each attribute value v is expressed in base b as a sequence of k base-b digits $v_k v_{k-1} ... v_2 v_1$, and each bitmap B_i^j represents the set of records with $v_i \leq j$. Using a larger base number improves the index's performance for evaluating range predicates at the cost of an increased space cost. An example of a base-10 BSI is shown in Fig. 1c. Both Model 204 and Sybase IQ implemented base-10 Bit-Sliced indexes [7].

Several bitmap index designs have also been implemented in a scientific/statistical database application at Lawrence Berkeley Laboratory [11]. These bitmap indexes include binary encoded indexes (equivalent to binary BSI), unary encoded indexes (equivalent to non-binary BSI), K-of-N encoded indexes (generalizations of Value-List indexes where each attribute value is encoded by a N-bit string with exactly K bits set to 1), and superimposed encoded indexes based on superimposed encoding which is useful for indexing set-valued attributes.

Interest in bitmap indexes was revived in the mid 1990s due to the emergence of data warehousing applications which are characterized by read-mostly query workloads dominated by large, complex ad hoc queries [7]. All the major DBMS vendors (IBM, Microsoft, Oracle, and Sybase) also started to support bitmap indexes in their products around this time.

Foundations

Chan and Ioannidis propose a two-dimensional framework to characterize the design space of bitmap indexes [2]. The two orthogonal parameters identified for bitmap indexes (with an attribute cardinality of C) are (i) the arithmetic used to represent attribute values; i.e., how an attribute value is decomposed into digits according to some base (e.g., base-C arithmetic is used in a Value-List index); and (ii) the encoding scheme of each decomposed digit in bits (e.g., each attribute value in a Value-List index is encoded by turning on exactly one out of C bits). Consider an attribute value $v \in [0,C)$ and a sequence of n base numbers $\mathcal{B} = <b_n, b_{n-1},\ldots,b_1>$, where $b_n = \left\lceil \frac{C}{\prod_{i=1}^{n-1} b_i} \right\rceil$ and $b_i \geq 2$, $i \in [1,n]$. Using $\mathcal{B}$, v can be decomposed into a unique sequence of n digits $<v_n, v_{n-1},\ldots,v_1>$ as follows: $v_i = V_i \ mod \ b_i$, where $V_1 = v$ and $V_i = \left\lfloor \frac{V_{i-1}}{b_{i-1}} \right\rfloor$, for $1 < i \leq n$. Thus, $v = v_n\left(\prod_{j=1}^{n-1} b_j\right) + \ldots + v_i\left(\prod_{j=1}^{i-1} b_j\right) + \ldots + v_2 b_1 + v_1$. Note that each v_i is a base-b_i digit (i.e., $0 \leq v_i < b_i$). Each choice of n and base-sequence $\mathcal{B}$ gives a different representation of attribute values, and therefore a different index (known as a Base-$\mathcal{B}$ index). The index consists of n *components* (i.e., one component per digit) where each component individually is now a collection of bitmaps. Figure 2b shows a base-<3,4> Value-List index that consists of two components: the first component has four bitmaps $\{E_1^3, E_1^2, E_1^1, E_1^0\}$, and the second component has three bitmaps $\{E_2^2, E_2^1, E_2^0\}$. Note that the kth bit in each bitmap E_i^j is set to 1 if and only if $v_i = j$, where $<v_2,v_1>$ is the <3,4>-decomposition of the kth record's indexed attribute value. For the encoding scheme dimension, there are two basic encoding schemes: equality encoding and range encoding. Consider the ith component of an index with base b_i, and a value $v_i \in [0,b_i - 1]$. In the *equality encoding scheme*, v_i is encoded by b_i bits, where all the bits are set to 0 except for the bit corresponding to v_i, which is set to 1. Thus, an equality-encoded component (with base b_i) consists of b_i bitmaps $\{E_i^{b_i-1},\ldots,E_i^0\}$ such that the kth bit in each bitmap E_i^j is set to 1 if and only if $v_i = j$, where v_i is the ith digit of the decomposition of the kth record's indexed attribute value. In the *range encoding scheme*, v_i is encoded again by b_i bits, with the v_i rightmost bits set to 0 and the remaining bits (starting from the one corresponding to v_i and to the left) set to 1. The kth bit in each bitmap R_i^j is set to 1 if and only if $v_i \leq j$, where v_i is the ith digit of the decomposition of the kth record's indexed attribute value. Since the bitmap $R_i^{b_i-1}$ has all bits set to 1, it does not need to be stored, so a range-encoded component consists of $(b_i - 1)$ bitmaps $\{R_i^{b_i-2},\ldots,R_i^0\}$. Value-List and Bit-Sliced indexes therefore correspond to equality-encoded and range-encoded indexes, respectively. Figures 1c and 2c show the range-encoded indexes corresponding to the equality-encoded indexes in Figs. 1b and 2b. Details of



	a: A		b: E_2^2	E_2^1	E_2^0	E_1^3	E_1^2	E_1^1	E_1^0	c: R_2^1	R_2^0	R_1^2	R_1^1	R_1^0	d: I_2^1	I_2^0	I_1^1	I_1^0
1	3	$\xrightarrow{0\times\mathbf{4}+3}$	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0
2	2	$\xrightarrow{0\times\mathbf{4}+2}$	0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	0
3	1	$\xrightarrow{0\times\mathbf{4}+1}$	0	0	1	0	0	1	0	1	1	1	1	0	0	1	1	1
4	2	$\xrightarrow{0\times\mathbf{4}+2}$	0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	0
5	8	$\xrightarrow{2\times\mathbf{4}+0}$	1	0	0	0	0	0	1	0	0	1	1	1	0	0	0	1
6	2	$\xrightarrow{0\times\mathbf{4}+2}$	0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	0
7	9	$\xrightarrow{2\times\mathbf{4}+1}$	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1
8	0	$\xrightarrow{0\times\mathbf{4}+0}$	0	0	1	0	0	0	1	1	1	1	1	1	0	1	0	1
9	7	$\xrightarrow{1\times\mathbf{4}+3}$	0	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0
10	5	$\xrightarrow{1\times\mathbf{4}+1}$	0	1	0	0	0	1	0	1	0	1	1	0	1	0	1	1
11	6	$\xrightarrow{1\times\mathbf{4}+2}$	0	1	0	0	1	0	0	1	0	1	0	0	1	0	1	0
12	4	$\xrightarrow{1\times\mathbf{4}+0}$	0	1	0	0	0	0	1	1	0	1	1	1	1	0	0	1

Bitmap Index. Figure 2. Example of base- <3,4> indexes. (a) indexed attribute A (b) equality-encoded index (c) range-encoded index (d) interval-encoded index.

query processing algorithms and space-time tradeoffs of equality/range-encoded, multi-component bitmap indexes are given in [4].

A new encoding scheme, called *interval encoding*, was proposed in [5]. For an attribute with a cardinality of C, a value $v \in [0,C)$ is encoded using $\lceil \frac{C}{2} \rceil$ bits such that if $v < \lceil \frac{C}{2} \rceil$, then v is encoded by setting the $(v+1)$ rightmost bits to 1 and the remaining bits to 0; otherwise, v is encoded by setting the $(C-1-v)$ leftmost bits to 1 and remaining bits to 0. Thus, the interval encoding scheme consists of $\lceil \frac{C}{2} \rceil$ bitmaps $\{I^{\lceil \frac{C}{2} \rceil - 1},\ldots,I^0\}$, where each bitmap I^j is associated with a range of $(m+1)$ values $[j,j+m]$, $m = \lfloor \frac{C}{2} \rfloor - 1$, such that the kth bit in a bitmap I^j is set to 1 if and only if the kth record's indexed attribute value is in $[j,j+m]$. Figures 1d and 2d show the interval-encoded indexes corresponding to the equality-encoded indexes in Figs. 1b and 2b. Note that interval encoding has better space-time tradeoff than range encoding: it has the same worst-case evaluation cost of two bitmap scans as range encoding but its space requirement is about half that of range encoding.

Wu and Bachmann proposed a variant of binary BSI called *encoded bitmap index (EBI)* [12,13]. Instead of encoding each attribute value simply in terms of its binary representation, an EBI uses a lookup table to map each attribute value to a distinct bit string; this flexibility enables optimization of the value-to-bit-string mapping, by exploiting knowledge of the query workload, to reduce the number of bitmap scans for query evaluation. Thus, binary BSI is a special case of EBI. Another similar index design called *Encoded-Vector index (EVI)* is used in IBM DB2. Instead of storing the index as a collection of $\lceil log_b(C) \rceil$ bitmaps as in EBI, EVI is organized as a single vector of $\lceil log_b(C) \rceil$-bit strings, and the purpose of the lookup table optimization is to reduce the CPU cost of bit string comparisons when evaluating selection queries of the form "$A \in \{v_1,v_2,\ldots,v_n\}$." Another related index is the *Projection index* [7], which is implemented in Sybase IQ.

Complex, multi-table join queries (such as star-join queries) can also be evaluated very efficiently using *bitmapped join indexes* [6], which are indexes that combine the advantages of *join indexes* and bitmap representation. A *join index* for the join between two relations R and S is a precomputation of their join result defined by $\Pi_{R.rid,S.rid}(R \bowtie_p S)$, where p is the join predicate between R and S. Thus, a join index on $R \bowtie S$ can be thought of as a conventional index on the table R, where the attribute being indexed is the "virtual" attribute $S.rid$; i.e., each distinct $S.rid$ value v is associated with a list of all $R.rid$ values that are related to v via the join. A *bitmapped join index* [6] is simply a join index with the RID-lists represented using bitmaps. Bitmapped join indexes are implemented in Informix Red Brick Warehouse and Oracle. Bitmap indexes can also be applied to evaluate queries that involve aggregate functions (e.g., SUM, MIN/MAX, MEDIAN); evaluation algorithms for Value-List and Bit-Sliced indexes are discussed in a paper by O'Neil and Quass [7]. Efficient algorithms for performing arithmetic operations (addition and subtraction) on binary BSIs are proposed in [8].

As bitmap indexes become less efficient for larger attribute cardinality, a number of approaches have been developed to reduce their space requirement. Besides using multi-component bitmap indexes [2,11], another common space-reduction technique is to apply compression. In Model 204 [7], the bitmaps are compressed by using a hybrid representation; specifically, each individual bitmap is partitioned into a number of fixed-size segments, and segments that are dense are stored as verbatim bitmaps while sparse segments are converted into RID-lists. While generic compression techniques (e.g., LZ77) are effective in reducing both the disk storage and retrieval cost of bitmap indexes, the savings in I/O cost can be offset by the high CPU cost incurred for decompressing the compressed bitmaps before they can be operated with other bitmaps. A number of specialized compression techniques that enable bitmaps to be operated on without a complete decompression have been proposed: Byte-aligned Bitmap Code (BBC) (which is used by Oracle), and Word-Aligned Hybrid code (WAH) [14]. Some performance study of compressed bitmap indexes are reported in [1–3,14].

A different approach proposed to reduce the size of bitmaps is to use *range-based bitmaps (RBB)* [15], which have been applied to index large data sets in tertiary storage systems as well as large, multi-dimensional data sets in scientific applications [14]. Unlike Value-List indexes where there is one bitmap for each distinct attribute value, the RBB approach partitions the attribute domain into a number of disjoint ranges and constructs one bitmap for each range of values (this is also known as *binning*). Thus, RBB provides a form of lossy compression which requires additional post-processing to filter out false positives.

Koudas [5] has examined space-optimal RBBs for equality queries when both the attribute and query distributions are known; these results have been extended for range queries [9]. More recently, Sinha and Winslett have proposed *multi-resolution bitmap indexes* to avoid the cost of filtering out false positives for RBB [10]. For example, in a two-resolution bitmap index, it has a lower resolution index consisting of RBB (i.e., with each bitmap representing a range of attribute values) and a higher resolution index consisting of one bitmap for each distinct attribute value. By combining the efficiency of lower resolution indexes and the precision of higher resolution indexes, queries can be evaluated efficiently without false positive filtering.

Key Applications

Today, bitmap indexes are supported by all major database systems, and they are particularly suitable for data warehousing applications [7]. Bitmap indexes have also been used in scientific databases (e.g., [10,11,14]), indexing data on tertiary storage systems (e.g., [1]), and data mining applications.

Future Directions

The design space for bitmap indexes is characterized by four key parameters: levels of resolution (which affects the number of levels of bitmap indexes and the index granularity at each level), attribute value representation (which affects the number and size of the index components at each level) encoding scheme (which affects how the bitmaps in each component are encoded), and storage format (i.e., uncompressed, compressed, or a combination of compressed and uncompressed). While there are several performance studies that have examined various combinations of the above parameter space, a comprehensive investigation into the space-time tradeoffs of the entire design space is, however, still lacking and deserves to be further explored. The result of such a study can be applied to further enhance automated physical database tuning tools.

Cross-references

▶ Access Path
▶ Bitmap-based Index Structures
▶ Data Warehouse

Recommended Reading

1. Amer-Yahia S. and Johnson T. Optimizing queries on compressed bitmaps. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 329–338.
2. Chan C.Y. and Ioannidis Y.E. Bitmap index design and evaluation. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 355–366.
3. Chan C.Y. and Ioannidis Y.E. An efficient bitmap encoding scheme for selection queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 215–226.
4. Knuth D.E. Retrieval on secondary keys. In The Art of Computer Programming: Sorting and Searching, vol. 3. Chap. 6. Addison-Wesley, Reading, Mass., 1973, pp. 550–567.
5. Koudas N. Space efficient bitmap indexing. In Proc. Int. Conf. on Information and Knowledge Management, 2000, pp. 194–201.
6. O'Neil P. and Graefe G. Multi-table joins through bitmapped join indices. ACM SIGMOD Record. September 1995, pp. 8–11.
7. O'Neil P. and Quass D. Improved query performance with variant indexes. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 38–49.
8. Reinfret D., O'Neil P., and O'Neil E. Bit-sliced index arithmetic. In Proc. ACM SIGMOD Int. Conf. on Management of Data. 2001, pp. 47–57.
9. Rotem D., Stockinger K., and Wu K. Optimizing candidate check cost for bitmap indices. In Proc. Int. Conf. on Information and Knowledge Management, 2005, pp. 648–655.
10. Sinha R. and Winslett M. Multi-resolution bitmap indexes for scientific data. ACM Trans. Database Syst., 32(3):1–39, 2007.
11. Wong H.K.T., Liu H-F., Olken F., Rotem D., and Wong L. Bit transposed files. In Proc. 11th Int. Conf. on Very Large Data Bases, 1985, pp. 448–457.
12. Wu M.C. Query optimization for selections using nitmaps. In Proc. ACM SIGMOD Int. Conf. on Management of Data 1999, pp. 227–238, 1999.
13. Wu M.C. and Buchmann A.P. Encoded bitmap indexing for data warehouses. In Proc. 14th Int. Conf. on Data Engineering, 1998, pp. 220–230.
14. Wu K., Otoo E.J., and Shoshani A. Optimizing bitmap indices with efficient compression. ACM Trans. Database Syst., 31(1):1–38, 2006.
15. Wu K.L. and Yu P.S. Range-based bitmap indexing for high cardinality attributes with skew. Technical report, IBM Watson Research Center, May 1996.

Bitmap-based Index Structures

GUADALUPE CANAHUATE, HAKAN FERHATOSMANOGLU
The Ohio State University, Columbus, OH, USA

Synonyms

Bitmap Index; Projection Index

Definition

A bitmap-based index is a binary vector that represents an interesting property and indicates which objects in the dataset satisfy the given property. The vector has a 1 in position *i* if the *i*-th data object satisfies the property, and 0 otherwise. Queries are executed using fast bitwise logical operations supported by hardware over the binary vectors.t

Historical Background

Bitmap-based indexing was first implemented in Computer Corporation of America's Model 204 in the mid-1980s by Dr. Patrick O'Neil. The bitmap index from Model 204 was a hybrid between verbatim (uncompressed) bitmaps and RID lists. Originally, a bitmap was created for each value in the attribute domain. The entire bitmap index is smaller than the original data as long as the number of distinct values is less than the number of bits used to represent the attribute in the original data. For example, if an integer attribute has cardinality 10 and integers are stored using 32 bits, then the bitmap index for such an attribute, which only has 10 bit vectors, is 3.2 times smaller than the original data. For floating point attributes, the bitsliced index (BSI) [8] stores each bit of the binary representation of the attribute independently. Bitsliced indexes are never more than the size of the original data. However, in general, all bitslices need to be accessed to answer a query.

In order to reduce the number of bit vectors that needed to be read to answer a query, more complex encodings for bitmap-based indexes have been proposed, such as range encoded bitmaps [15,8], encoded bitmaps [16], and interval encoded bitmaps [4,5]. For attributes with high cardinality including floating point attributes, binning [7,13] was proposed to reduce the number of values in the domain, and therefore the number of bit vectors in the index. In addition, special compression techniques were developed to improve the performance of the bitmap indexes. The two most popular techniques are Byte-aligned Bitmap Code (BBC) [1], and Word Aligned Hybrid (WAH) [18] compression method. These compression techniques allow query execution over the compressed bitmaps. With compression, bitmap indexes have been proven to perform well with high cardinality attributes [17].

Foundations

Bitmap tables are a special type of binary matrices. Each binary row in the bitmap table represents one tuple in the database. The bitmap columns are produced by quantizing the attributes in the database into categories or bins. Each tuple in the database is then encoded based on which bin each attribute value falls into. Bitmap index encoding is based on the properties of physical row identifiers and there is a one-to-one correspondence between the data objects and the bits in the bitmap vector. Therefore, given a bit position, the location of its corresponding table row can be computed by simple arithmetic operations. For tables with a clustered-index or index-ordered tables, a mapping table is used to map bit positions to row locations.

Bitmap Encoding

For equality encoded bitmaps (also called simple encoding or projection index) [8], if a value falls into a bin, this bin is marked "1", otherwise "0". Since a value can only fall into a single bin, only a *single* "1" can exist for each row of each attribute. After binning, the whole database is converted into a large 0–1 bitmap, where rows correspond to tuples and columns correspond to bins. Figure 1 shows an example using a table with one attribute with cardinality 4. Columns

Tuple	Value	Equality				Eq w/binning		Range			Interval			Bitsliced		
		= 1	= 2	= 3	= 4	{1,2}	{3,4}	≤3	≤2	≤1	[1,2]	[2,3]	[3,4]	2^2	2^1	2^0
t_1	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	1
t_2	2	0	1	0	0	1	0	1	1	0	1	1	0	0	1	0
t_3	3	0	0	1	0	0	1	1	0	0	0	1	1	0	1	1
t_4	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	1
t_5	2	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0
t_6	4	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0
t_7	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	1
t_8	3	0	0	1	0	0	1	1	0	0	0	1	1	0	1	1

Bitmap-based Index Structures. Figure 1. Bitmap index examples for a table with one attribute with cardinality 4.

3–6 of Fig. 1 show the equality encoded bitmap for this table. The first tuple t_1 has value 1, therefore only the corresponding bit in the bitmap =1 is set. Columns 7 and 8, show the bitmaps for equality encoding with binning. In this example, the attribute was quantized into two bins.

For range encoded bitmaps [4], a bin is marked “1” if the value falls into it or a smaller bin, and “0” otherwise. Using this encoding, the last bin for each attribute is all 1s. Thus, this column is not explicitly stored. Columns 9–11 in Fig. 1 show the range encoding for the attribute. The first tuple t_1 has the smallest value 1, therefore all the bitmaps have the first bit set. For interval encoded bitmaps, every bitmap represents a range of $\lceil \frac{C}{2} \rceil$ values, where C is the cardinality of the attribute. This encoding allows to answer any range or point query on one attribute by reading at most two bit vectors [4,5]. Columns 12–14 in Fig. 1 show the interval encoded bitmap. The fifth tuple t_5 has value 2, therefore the fifth bit is set for the [1,2] and [2,3] interval bitmaps. Bit-sliced indexes (BSI) [8] can be considered as a special case of the encoded bitmaps [16]. With the bit-sliced index the bitmaps encode the binary representation of the attribute value. Columns 15–17 of Fig. 1 show the bit-sliced index of the table. The third tuple t_3 has value 3 which is represented by the binary number 011, therefore the corresponding bit in slices 2^1 and 2^0 are set. Encoded bitmaps encode the binary representation of the attribute bins. Therefore, only $\lceil \log_2 bins \rceil$ bitmaps are needed to represent all values.

With binning, several values are encoded in the same bitmap. However, the results from the queries are supersets of the actual results, therefore additional disk access may be needed to evaluate the candidates and retrieve the exact results. Several binning strategies, based on data distribution and query workloads, and several strategies for candidate evaluation have been proposed [10,11]. A binned bitmap index augmented with an auxiliary order-preserving bin-based Clustering (OrBiC) [19] can significantly reduce the I/O cost of candidate evaluation.

Query Execution

Bitmap indexes can provide efficient performance for point and range queries thanks to fast bitwise operations (AND, OR, NOT) which are supported by hardware. With equality encoded bitmaps a point query is executed by ANDing together the bit vectors corresponding to the values specified in the query conditions. For example, finding the data points that correspond to a query where Attribute 1 is equal to 3 and Attribute 2 is equal to 5 is only a matter of ANDing the two bitmaps together. Equality Encoded Bitmaps are optimal for point queries. Range queries are executed by first ORing together all the bit vectors specified by each range in the query conditions and then ANDing the answers together. If the query range for an attribute queried includes more than half of the cardinality then one executes the query by taking the complement of the ORed bitmaps that are not included in the query condition. The performance of queries with large-range query conditions can be improved by using multi-resolution bitmap indexes [12], i.e., bitmaps with different interval size: at the highest level one bit vector corresponds to an individual value and at lower levels each bit vector corresponds to a bin of values. Queries are executed using combined resolution bitmaps that minimize both the number of bitwise operations and the number of candidate points. With more complex encodings, the query evaluation strategy depends on the encoding and the range being queried.

Bitmap Compression

The goal of the bitmap index compression is twofold: to reduce the space requirement of the index and to maintain query execution performance by executing the query over the compressed index. The two most popular run-length compression techniques are the Byte-aligned Bitmap Code (BBC) [1] and the Word-Aligned Hybrid (WAH) compression method [18]. BBC stores the compressed data in bytes while WAH stores it in words. WAH is simpler because it only has two types of words, while BBC has four. Both techniques are based on the idea of run length encoding that represents consecutive bits of the same symbol (also called a fill or a gap) by their bit value and their length. The bit value of a fill is called the fill bit. BBC first divides the bit sequence into bytes and then group bytes into runs. A run consists of a fill word followed by a tail of literal bytes. A run always contains a number of whole bytes as it represents the fill length as number of bytes rather than number of bits. The byte alignment property limits a fill length to be an integer multiple of bytes. This ensures that during any bitwise logical operation a tail byte is never broken into individual bits. Similar to BBC, WAH is a hybrid between the run length encoding and the literal scheme. WAH stores compressed data in words rather than in bytes. The most significant bit of a word distinguishes between a literal word(0) and a fill word(1). Lower bits of a literal word contain the bit values from the bit sequence. The second most significant bit of a fill word is the fill bit and the lower (w-2) bits store the fill length. Imposing word alignment ensures that the logical operation functions only need to access words not bytes or bits. In general, bit operations over the compressed WAH bitmap file are faster than BBC while BBC gives better compression ratio.

Typically, complex encoded bitmaps do not compress well as the bit density of such bitmaps is relatively high. Recently, reordering has been proposed as a preprocessing step for improving the compression of bitmaps. The objective with reordering is to increase the performance of run length encoding. By reordering the data, the compression ratio of large boolean matrices can be improved [6]. However, optimal matrix reordering is NP-hard and the authors use traveling salesman heuristics to compute the new order. The idea of reordering is also applied to compression of bitmap indexes [9]. The authors show that the tuple reordering problem is NP-complete and propose a Gray code ordering heuristic.

Bitmap update is an expensive operation. The common practice is to drop the index, do a batch update, and recreate the index. For improved update performance, it is possible to add to each compressed bitmap a pad-word that encodes all possible rows using non-set bits [3]. For insertions, only the updated column needs to be accessed as opposed to all the bitmap columns. This technique is suitable for maintaining an online index when the update rate is not very high. An alternative to run-length compression proposed in the literature is the Approximate Bitmap Encoding [2], where hashing is used to encode the data objects in a bloom filter. This technique can offer improved performance for selection queries over small number of rows.

Key Applications

Bitmap indexing has been traditionally used in data warehouses to index large amount of data that are infrequently updated. More recently, bitmaps have found many other applications such as visualization and indexing of scientific data. Other works have applied bitmaps for term matching and similarity searches. In general, one could claim that bitmaps would outperform most approaches in the domains where full scan of the data are unavoidable and computations can be done using parallel bitwise operations.

Bitmap-based indexes are successfully implemented in commercial Database Management Systems. Oracle implements a version of the simple bitmap encoding compressed using BBC. Sybase IQ has two different types of bitwise indexes. IBM DB2 dynamically builds bitmaps from a single-column B-tree to join tables. Informix uses bitmapped indexes since version 8.21 released in 1998. PostgreSQL supports bitmap index scans since version 8.1 where dynamic bitmaps are created to combine the results of multiple index conditions using bitwise operations.

Cross-references

► Approximation and Data Reduction Techniques
► Indexing
► Query Processing and Optimization in Object Relational Databases

Recommended Reading

1. Antoshenkov, G. Byte-aligned bitmap compression. In Data Compression Conference, 1995. Oracle Corp.
2. Apaydin, T., Canahuate, G., Ferhatosmanoglu, H., and Tosun, A. Approximate encoding for direct access and query processing over compressed bitmaps. In Proc. 32nd Int. Conf. on Very Large Data Bases, 2006.
3. Canahuate, G., Gibas, M., Ferhatosmanoglu, H. Update conscious bitmap indices. In Proc. 19th Int. Conf. on Scientific and Statistical Database Management, 2007.
4. Chan, C.Y. and Ioannidis, Y.E. Bitmap index design and evaluation. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998.
5. Chan, C.Y. and Ioannidis, Y.E. An efficient bitmap encoding scheme for selection queries. ACM SIGMOD Rec., 1999.
6. Johnson, D., Krishnan, S., Chhugani, J., Kumar, S., and Venkatasubramanian, S. Compressing large boolean matrices using reordering techniques. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004.
7. Koudas, N. Space efficient bitmap indexing. In Proc. Int. Conf. on Information and Knowledge Management, 2000.
8. O'Neil, P. and Quass, D. Improved query performance with variant indexes. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997.
9. Pinar, A., Tao, T., and Ferhatosmanoglu, H. Compressing bitmap indices by data reorganization. In Proc. 21st Int. Conf. on Data Engineering, 2005.
10. Rotem, D., Stockinger, K., and Wu, K. Optimizing candidate check costs for bitmap indices. In Proc. Int. on Information and Knowledge Management, 2005.
11. Rotem, D., Stockinger, K., and Wu, K. Minimizing I/O costs of multi-dimensional queries with bitmap indices. In Proc. 18th Int. Conf. on Scientific and Statistical Database Management, 2006.

12. Sinha, R., Winslett, M. Multi-resolution bitmap indexes for scientific data. ACM Trans. Database Syst., 2007.
13. Stockinger, K. Design and implementation of bitmap indices for scientific data. In Proc. Int. Conf. on Database Eng. and Applications, 2001.
14. Stockinger, K., Wu, K., and Shoshani, A. Evaluation strategies for bitmap indices with binning. In Proc. 15th Int. Conf. Database and Expert Syst. Appl., 2004.
15. Wong, H.K., Liu, H., Olken, F., Rotem, D., and Wong, L. Bit transposed files. In Proc. 11th Int. Conf. on Very Large Data Bases, 1985.
16. Wu, M-C. and Buchmann, A. Encoded bitmap indexing for data warehouses. In Proc. 14th Int. Conf. on Data Engineering, 1998.
17. Wu, K., Otoo, E.J., and Shoshani, A. On the performance of bitmap indices for high cardinality attributes. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004.
18. Wu, K., Otoo, E.J., and Shoshani, A. Optimizing bitmap indexes with efficient compression. ACM Trans Database Syst., 2006.
19. Wu, K., Stockinger, K., Shoshani, A. Breaking the curse of cardinality on bitmap indexes. In Proc. 20th Int. Conf. on Scientific and Statistical Database Management, 2008.

Blind Signatures

Barbara Carminati
University of Insubria, Varese, Italy

Definition

In blind signature schemes, the sender and the signer of the message are two distinguished entities. Given a message m, blind signatures have the property that the signer digitally signs a blinded version of m, i.e., m', without the disclosure of any information about the original message. The obtained blind signature can be verified by using the public key of the signer and the original message m, instead of m'.

Key Points

In 1982, David Chaum introduced the concept of blind signatures for protecting user privacy during electronic payment transactions [1]. This scheme has been devised for scenarios where the sender and the signer of the message are two distinguished entities, with the aim of preventing the signer from observing the message he or she signs. More precisely, given a message m generated by A, a signer B is able to digitally sign a blinded version of m', i.e., $DS_B(m')$, without the disclosure of any information about the original message. The key property of blind signatures is that the obtained signature $DS_B(m')$ can be verified by using the public key of B and the original message m, instead of m'. This property makes blind signatures particularly suitable for applications where sender privacy is the main concern, like for instance, in digital cash protocols and electronic voting systems.

Several digital signature schemes can be used to obtain a blind signature, like for instance RSA and DSA schemes. In the following to explain the generation and verification of a blind signature, the RSA algorithm is considered. Let $PK_B = (n, e)$ and $SK_B = (n, d)$ be the public and private keys of signer B.

Blinding. Let k be a random integer chosen by the sender A, such that $0 \leq k \leq n - 1$ and $gcd(n, k) = 1$. Thus, given a message m, the sender generates the blinded version of it, i.e., m', by combing m with k. According to RSA, the blinded version of m is computed as $m' = (mk^e) \bmod n$.

Blind signature generation. The signer digitally signs the blinded version m' using the RSA digital signature algorithm; that is, $DS_B(m') = (m')^d \bmod n$.

Signature unblinding. The intended verifier can obtain the signature of m, i.e., $DS_B(m)$, as follows $DS_B(m) = k^{-1}DS_B(m')$. This signature can be validated by the RSA verification algorithm, according to the standard process.

Cross-references

▶ Digital Signatures
▶ Privacy Enhancing Technologies

Recommended Reading

1. Chaum D. Blind signatures for untraceable payments, advances in cryptology. In Proceedings of Crypto '82, 1983, pp. 199–203.

Bloom Filter Join

▶ Semijoin

Bloom Filters

Michael Mitzenmacher
Harvard University, Boston, MA, USA

Synonyms

Hash filter

Definition

A Bloom filter is a simple, space-efficient randomized data structure based on hashing that represents a set in a way that allows membership queries to determine whether an element is a member of the set. False positives are possible, but not false negatives. In many applications, the space savings afforded by Bloom filters outweigh the drawbacks of a small probability for a false positive. Various extensions of Bloom filters can be used to handle alternative settings, such as when elements can be inserted and deleted from the set, and more complex queries, such as when each element has an associated function value that should be returned.

Historical Background

Burton Bloom introduced what is now called a Bloom filter in his 1970 paper [2], where he described the technique as an extension of hash-coding methods for applications where error-free methods require too much space and were not strictly necessary. The specific application he considered involved hyphenation: a subset of words from a standard dictionary require specialized hyphenation, while the rest could be handled by a few simple and standard rules. Keeping a Bloom filter of the words requiring specialized hyphenation dramatically cut down disk accesses. Here, false positives caused words that could be handled by the simple rules to be treated as special cases. Bloom filters were also used in early UNIX spell-checkers [12,13], where a false positive would allow a misspelled word to be ignored, and were suggested as a way of succinctly storing a dictionary of insecure passwords, where a false positive would disallow a potentially secure password [16].

Bloom filters were applied in databases to reduce the amount of communication and computation for join operations, especially distributed join operations [1,4,11]. The term Bloomjoin is sometimes used to describe a semijoin operation that utilizes Bloom filters. The Bloom filter is used to represent join column values, so that matching values can be found by querying the Bloom filter. False positives cause false matches of tuples that must later be removed. However, the communication and computation gains from using the filter can yield significant advantages even with these false positives.

Foundations

A Bloom filter for representing a set $S = \{x_1,x_2,\ldots,x_n\}$ of n elements from a universe U consists of an array of m bits, initially all set to 0. The filter uses k independent hash functions $h_1,\ldots,h_k$ with range $\{1,\ldots,m\}$. For each element $x \in S$, the bits $h_i(x)$ are set to 1 for $1 \leq i \leq k$. (A location can be set to 1 multiple times.) To check if an item y is in S, one checks whether all $h_i(y)$ are set to 1. If not, then clearly y is not a member of S. If all $h_i(y)$ are set to 1, the data structure returns that y is in S. There is no possibility of a false negative, but it is possible that for $y \notin S$, all $h_i(y)$ are set to 1, in which case the data structure gives a false positive.

The fundamental issue in the analysis of the Bloom filter is the *false positive probability* for an element not in the set. In the analysis, generally it is assumed that the hash functions map each element in the universe to a random number independently and uniformly over the range. While this is clearly an optimistic assumption, it appears to be suitable for practical implementations. With this assumption, after all the elements of S are hashed into the Bloom filter, the probability that a specific bit is still 0 is

$$p' = (1 - 1/m)^{kn} \approx \mathrm{e}^{-kn/m}.$$

It is convenient to use the approximation $p = \mathrm{e}^{-kn/m}$ in place of p'. If ρ is the proportion of 0 bits after all the n elements are inserted in the table, then conditioned on ρ the probability f of a false positive is

$$f = (1 - \rho)^k \approx (1 - p')^k \approx (1 - p)^k = \left(1 - \mathrm{e}^{-kn/m}\right)^k.$$

These approximations follow since $\mathrm{E}[\rho] = p'$, and ρ can be shown to be highly concentrated around p' using standard techniques [8].

The optimal number of hash function can be found by finding where $(1 - \mathrm{e}^{-kn/m})^k$ is minimized as a function of k. Simple calculus reveals the optimum occurs when $k = \ln 2 \cdot (m/n)$, giving a false positive probability f of

$$f = \left(1 - \mathrm{e}^{-kn/m}\right)^k = (1/2)^k \approx (0.6185)^{m/n}.$$

In practice, k must be an integer, and the choice of k might depend on application-specific questions.

Key Applications

Essentially any application that requires membership checks against a list or set of objects, and for which space is at a premium, is a candidate for a Bloom filter. In some applications, Bloom filters may also save computational resources. The consequences of false positives, however, need to be carefully considered in all circumstances.

Bloom filters enjoyed a recent resurgence in the networking community after their use in a paper by Fan et al. on distributed Web caches [9]. Instead of having caches distribute lists of URLs (or lists of their 16-byte MD5 hashes) corresponding to the cache contents, the authors demonstrate a system that saves in communication costs by sharing Bloom filters of URLs. False positives may cause a server to request a page from another nearby server, even when that server does not hold the page. In this case, the page must subsequently be retrieved from the Web. If the false positive probability is sufficiently low, the penalty from these occasional failures is dominated by the improvement in overall network traffic. Countless further applications in databases, peer-to-peer networks, overlay networks, and router architectures have arisen, and the use of Bloom filters and their many variants and extensions has expanded dramatically [5].

In databases, applications include the previously mentioned Bloomjoin variation of the semijoin. In a similar manner, Bloom filters can be effectively used to estimate the size of semijoin operations, using the fact that Bloom filters can be used to estimate the size of a set (and the size of set unions and intersections) [14]. Another early database application utilizes differential files [10]. In this setting, all changes to a database that occur during a given time period are processed as a batch job, and a differential file tracks changes that occur until the batched update occurs. To read a record then requires determining if the record has been changed by some transaction in the differential file. If not, the record can be read directly from the database, which is generally much quicker and more efficient than performing the necessary processing on the differential file. Instead of keeping a list of all records that have changed since the last update, a Bloom filter of the records that have been changed can be kept. Here, a false positive would force the database to check both the differential file and the database on a read even when a record has not been changed. If false positives are sufficiently rare, this cost is minimal.

Future Directions

Recent research related to Bloom filters has followed three major directions: alternative constructions, improved analysis, and extensions to more challenging questions. All three of these directions are likely to continue to grow.

Research in alternative constructions has focused on building data structures with the same capabilities as a Bloom filter or related data structures with improved efficiency, sometimes particularly for a specialized domain, such as router hardware. Improved analysis has considered reducing for example the amount of randomness required for a Bloom filter, or its performance under specific classes of hash functions.

The largest area of research has come from extending the basic idea of the Bloom filter to more general and more difficult problems. Arguably, the success of Bloom filters relies on their flexibility, and numerous variations on the theme have arisen. For example, counting Bloom filters extend Bloom filters by allowing sets to change dynamically via insertions and deletions of elements [9,15]. Spectral Bloom filters extend Bloom filters to handle multi-sets [7]. Count-min sketches extend Bloom filters to approximately track counts associated to items in a data stream, such as byte-counts for network flows [8]. Bloomier filters extend Bloom filters to the situation where each element of S is associated with a function value from a discrete, finite set of values, and the data structure should return that value [6]. Approximate concurrent state machines further extend Bloom filter to the setting where both the set can change due to insertions and deletions and the function values associated with set elements can change dynamically [3]. The list of variations continues to grow steadily as more applications are found.

URL to Code

- en.wikipedia.org/wiki/Bloom_filter
- www.patrow.net/programming/hashfunctions/index.html
- search.cpan.org/~mceglows/Bloom-Filter-1.0/Filter.pm

Cross-references

► Join
► Semijoin

Recommended Reading

1. Babb E. Implementing a relational database by means of specialized hardware. ACM Trans. Database Syst., 4(1):1–29, 1979.
2. Bloom B. Space/time tradeoffs in hash coding with allowable errors. Commun. ACM, 13(7):422–426, 1970.
3. Bonomi F., Mitzenmacher M., Panigrahy R., Singh S., and Varghese G. Beyond Bloom filters: from approximate membership checks to approximate state machines. Comput. Commun. Rev., 36(4):315–326, 2006.
4. Bratbergsengen K. Hashing methods and relational algebra operations. In Proc. 10th Int. Conf. on Very Large Data Bases, 1984, pp. 323–333.
5. Broder A. and Mitzenmacher M. Network applications of Bloom filters: a survey. Internet Math., (4):485–509, 2005.
6. Chazelle B., Kilian J., Rubinfeld R., and Tal A. The Bloomier filter: an efficient data structure for static support lookup tables. In Proc. 15th Annual ACM-SIAM Symp. on Discrete Algorithms, 2004, pp. 30–39.
7. Cohen S. and Matias Y. Spectral Bloom filters. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 241–252.
8. Cormode G. and Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications. J. Algorithms, 55(1):58–75, 2003.
9. Fan L., Cao P., Almeida J., and Broder A.Z. Summary cache: a scalable wide-area Web cache sharing protocol. IEEE/ACM Trans. Network., 8(3):281–293, 2000.
10. Gremilion L.L. Designing a Bloom filter for differential file access. Commun. ACM, 25:600–604, 1982.
11. Mackett L.F. and Lohman G.M. R* optimizer validation and performance evaluation for distributed queries. In Proc. 27th Int. Conf. on Very Large Data Bases, 1986, pp. 149–159.
12. McIlroy M.D. Development of a spelling list. IEEE Trans. Commun., 30(1):91–99, January 1982.
13. Mullin J.K. and Margoliash D.J. A tale of three spelling checkers. Software Pract. Exp., 20(6):625–630, June 1990.
14. Mullin J.K. Estimating the size of a relational join. Inf. Syst., 18(3):189–196, 1993.
15. Mitzenmacher M. Compressed Bloom filters. IEEE/ACM Trans. Network., 10(5):604–612, October 2002.
16. Spafford E.H. Opus: preventing weak password choices. Comp. Sec., 11:273–278, 1992.

Bloom Join

▶ Semijoin

BM25

GIAMBATTISTA AMATI

Ugo Bordoni Foundation, Rome, Italy

Synonyms

OKAPI retrieval function; Probabilistic model

Definition

BM25 is a ranking function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. It is used by search engines to rank matching documents according to their relevance to a given search query and is often referred to as "Okapi BM25," since the Okapi information retrieval system was the first system implementing this function. The BM25 retrieval formula belongs to the BM family of retrieval models (BM stands for Best Match), that is the weight of a term **t** in a document **d** is:

$$\frac{\mathbf{tf}}{k+\mathbf{tf}}\ln\frac{(r_{\mathbf{t}}+0.5)\cdot(\mathbf{N}-R-\mathbf{n_t}+r_{\mathbf{t}}+0.5)}{(\mathbf{n_t}-r_{\mathbf{t}}+0.5)\cdot(R-r_{\mathbf{t}}+0.5)} \quad [BM\text{'s family}]$$

where

- R is the number of documents known to be relevant to a specific topic,
- $r_{\mathbf{t}}$ is the number of relevant documents containing the term,
- $\mathbf{N}$ is the number of documents of the collection,
- $\mathbf{n_t}$ is the document frequency of the term,
- $\mathbf{tf}$ is the frequency of the term in the document,
- k is a parameter.

The BM25 document-query matching function is:

$$\sum_{\mathbf{t}\in\mathbf{q}} w_{\mathbf{t}}\cdot \mathrm{In}\frac{(r_{\mathbf{t}}+0.5)\cdot(\mathbf{N}-R-\mathbf{n_t}+r_{\mathbf{t}}+0.5)}{(\mathbf{n_t}-r_{\mathbf{t}}+0.5)\cdot(R-r_{\mathbf{t}}+0.5)} \quad [BM25] \qquad (1)$$

where

- $\mathbf{q}$ is the query,
- $w_{\mathbf{t}}=(k_1+1)\frac{\mathbf{tf}}{k+\mathbf{tf}}\cdot(k_3+1)\frac{\mathbf{tf}_q}{(k_3+\mathbf{tf_q})}$
- $\mathbf{tf}_q$ is the frequency of the term within the topic from which $\mathbf{q}$ was derived
- l and $\bar{\mathbf{l}}$ are respectively the document length and average document length.

- k is $k_1\left((1-b)+b(\frac{l}{\bar{l}})\right)$.
- k_1, b and k_3 are parameters which depend on the nature of the queries and possibly on the database.
- k_1 and b are set by default to 1.2 and 0.75 respectively, k_3 to 1000.

By using these default parameters, the unexpanded BM25 ranking function, that is the BM25 applied in the absence of information about relevance ($R{=}r{=}0$), is:

$$\sum_{\mathbf{t}\in\mathbf{q}} \frac{2.2\cdot\mathbf{tf}}{0.3+0.9\frac{l}{\bar{l}}+\mathbf{tf}}\cdot\frac{1001\cdot\mathbf{tf}_q}{1000+\mathbf{tf_q}}\ln\frac{N-\mathbf{n_t}+0.5}{\mathbf{n_t}+0.5} \qquad (2)$$

Historical Background

The successful formula of the BM family, the BM25, was introduced by Robertson and Walker in 1994 [1,3] and it has its root in Harter's 2-Poisson model of litheness for indexing. Before BM25, a direct exploitation of itheness for document retrieval was explored by Robertson, Van Rijsbergen, Porter, Williams and Walker [3,4] who plugged the Harter 2-Poisson model into the standard probabilistic model of relevance o Robertson and Spark Jones [2]. The evolution of the 2-Poisson model as designed by Robertson, Van Rijsbergen and Porter has thus motivated the birth of the BM family of term-weighting forms. The BM25 formula is the matching function of the Okapi information retrieval system of City University in London.

Foundations

BM25 derives from both the 2-Poisson model and the probabilistic binary independence model of relevance, that is as a combination of the probabilistic model

$$\mathrm{Prob}(rel,\mathbf{d}|\mathbf{q}) \propto \prod_{\mathbf{t}\in\mathbf{q}\cap\mathbf{d}} \frac{\mathrm{Prob}(\mathbf{t}|rel)\cdot\mathrm{Prob}(\bar{\mathbf{t}}|\overline{rel})}{\mathrm{Prob}(\mathbf{t}|\overline{rel})\cdot\mathrm{Prob}(\bar{\mathbf{t}}|rel)} \qquad (3)$$

and the 2-Poisson model

$$\mathrm{Prob}(X=\mathbf{tf}) = p\cdot\frac{e^{-\lambda_{\mathbf{E_t}}}\lambda_{\mathbf{E_t}}{}^{\mathbf{tf}}}{\mathbf{tf}!}+(1-p)\cdot\frac{e^{-\lambda_{\overline{\mathbf{E_t}}}}\lambda_{\overline{\mathbf{E_t}}}{}^{\mathbf{tf}}}{\mathbf{tf}!} \qquad (4)$$

where p is the probability of a document to belong to the Elite set.

However, the 2-Poisson has three parameters: the mean term frequency of the term in the elite set $\lambda_{\overline{\mathbf{E_t}}}$ (a set of documents with a large number of occurrences of the term), the mean term frequency of the term in the rest of the collection, $\lambda_{\overline{\mathbf{E_t}}}$, and the mixing parameter p. Therefore, the 2-Poisson model needs reasonable approximations in order to make the probabilistic mixture a workable retrieval model.

The combination of the notion of eliteness with that of relevance generates the Robertson, Van Rijsbergen and Porter's query term-document matching function:

$$w=\ln\frac{(p_1\lambda_{\mathbf{E_t}}{}^{\mathbf{tf}}e^{-\lambda_{\mathbf{E_t}}}+(1-p_1)\lambda_{\overline{\mathbf{E_t}}}{}^{\mathbf{tf}}e^{-\lambda_{\overline{\mathbf{E_t}}}})}{(p_2\lambda_{\mathbf{E_t}}{}^{\mathbf{tf}}e^{-\lambda_{\mathbf{E_t}}}+(1-p_2)\lambda_{\overline{\mathbf{E_t}}}{}^{\mathbf{tf}}e^{-\lambda_{\overline{\mathbf{E_t}}}})}\;\frac{(p_2e^{-\lambda_{\mathbf{E_t}}}+(1-p_2)e^{-\lambda_{\overline{\mathbf{E_t}}}})}{(p_1e^{-\lambda_{\mathbf{E_t}}}+(1-p_1)e^{-\lambda_{\overline{\mathbf{E_t}}}})} \qquad (5)$$

where p_1 and p_2 are the conditional probabilities of a document to be or not in the elite set respectively, given the set of relevant documents. Since elite set (documents with high query-term frequency) and relevance (documents relevant to the query) are highly correlated then it can be assumed that $p_1{>}p_2$.

With little algebra, (5) is equivalent to

$$w=\ln\frac{\left(p_1+(1-p_1)\left(\frac{\lambda_{\overline{\mathbf{E_t}}}}{\lambda_{\mathbf{E_t}}}\right)^{\mathbf{tf}}e^{\lambda_{\mathbf{E_t}}-\lambda_{\overline{\mathbf{E_t}}}}\right)}{\left(p_2+(1-p_2)\left(\frac{\lambda_{\overline{\mathbf{E_t}}}}{\lambda_{\mathbf{E_t}}}\right)^{\mathbf{tf}}e^{\lambda_{\mathbf{E_t}}-\lambda_{\overline{\mathbf{E_t}}}}\right)}\;\frac{\left(p_2e^{-\lambda_{\mathbf{E_t}}+\lambda_{\overline{\mathbf{E_t}}}}+(1-p_2)\right)}{\left(p_1e^{-\lambda_{\mathbf{E_t}}+\lambda_{\overline{\mathbf{E_t}}}}+(1-p_1)\right)} \qquad (6)$$

Equation (6) can be rewritten as

$$w(\mathbf{tf})=\ln C(\mathbf{tf})+\ln C_0$$

where C_0 is the ratio of the two components of the cross-product ratio not containing the variable **tf**. The first derivative with respect to the variable **tf** is

$$\frac{dw}{d\mathbf{tf}}=\frac{(p_1-p_2)\cdot e^{\lambda_{\mathbf{E_t}}-\lambda_{\overline{\mathbf{E_t}}}}\cdot\left(\frac{\lambda_{\overline{\mathbf{E_t}}}}{\lambda_{\mathbf{E_t}}}\right)^{\mathbf{tf}}\cdot\ln\left(\frac{\lambda_{\mathbf{E_t}}}{\lambda_{\overline{\mathbf{E_t}}}}\right)}{C(\mathbf{tf})}$$

The derivative is always positive because $p_1{>}p_2$, $\lambda_{\overline{\mathbf{E_t}}}\ll\lambda_{\mathbf{E_t}}$ and thus $\ln\left(\frac{\lambda_{\mathbf{E_t}}}{\lambda_{\overline{\mathbf{E_t}}}}\right)>0$ in the 2-Poisson model. Therefore $w(\mathbf{tf})$ is a monotonically increasing function. The limiting form of (6) for $\mathbf{tf}\rightarrow\infty$ is

$$w = \ln \frac{p_1\left(p_2 e^{-\lambda_{E_t}+\lambda_{\overline{E_t}}} + (1-p_2)\right)}{p_2\left(p_1 e^{-\lambda_{E_t}+\lambda_{\overline{E_t}}} + (1-p_1)\right)}$$

Since $e^{-\lambda_{E_t}+\lambda_{\overline{E_t}}} \sim 0$, this limit is very close to

$$w \sim \ln \frac{p_1(1-p2)}{p_2(1-p1)} \qquad (7)$$

Because (6) is monotonic with respect to the within–document term-frequency **tf**, document ranking is obtained by decreasing order of the **tf** value. Hence because for the highest values of the term frequency **tf** (i.e., for **tf** $\to\infty$), the limiting form of (7) can be taken as the actual score of the topmost documents.

In 1994, Robertson and Walker defined as an approximation of w of (7) the product

$$w = \frac{\mathbf{tf}}{\mathbf{tf}+K} \cdot \ln \frac{\text{Prob}(\mathbf{t}|rel)\text{Prob}(\bar{\mathbf{t}}|\overline{rel})}{\text{Prob}(\mathbf{t}|\overline{rec})\text{Prob}(\bar{\mathbf{t}}|\overline{rel})} \qquad (8)$$

Indeed, both (6) and (8) have Formula 7 as their limit for large **tf**. Varying the parameter K and using equation

$$\frac{\text{Prob}(\mathbf{t}|rec)\text{Prob}(\bar{\mathbf{t}}|\overline{rec})}{\text{Prob}(\mathbf{t}|\overline{rec})\text{Prob}(\bar{\mathbf{t}}|rec)} = \frac{(r_t+0.5)\cdot(\mathbf{N}-R-\mathbf{n_t}+r_t+0.5)}{(\mathbf{n_t}-r_t+0.5)\cdot(R-r_t+0.5)} \qquad (9)$$

the BM weighting formulas are derived.

Cross-references

▶ Divergence from Randomness Models
▶ Information Retrieval
▶ Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model
▶ Relevance
▶ Term Weighting
▶ TF*IDF
▶ Two-Poisson Model

Recommended Reading

1. Robertson S.E., Walker S., Beaulieu M.M., Gatford M., and Payne A. Okapi at trec-4. In Harman D.K. (ed.). NIST Special Publication 500-236: In Proc. The 4th Text Retrieval Conference. 1996.
2. Robertson S.E. and Sparck-Jones K. Relevance weighting of search terms. J. Am. Soc. Inform. Sci., 27:129–146, 1976.
3. Robertson S.E. and Walker S. Some simple approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, June 1994, pp. 232–241.
4. Robertson S.E., Van Rijsbergen C.J., and Porter M. Probabilistic models of indexing and searching. In Robertson S.E., Van Rijsbergen C.J., and Williams P.W. (eds.). Information retrieval Research, Chap. 4, Butterworths, 1981, pp. 35–56.

Boolean Model

Massimo Melucci
University of Padua, Padua, Italy

Definition

In the Boolean Model for Information Retrieval, a document collection is a set of documents and an index term is the subset of documents indexed by the term itself. An index term can also be seen as a proposition which asserts whether the term is a property of a document, that is, if the term occurs in the document or, in other words, if the document is about the concept represented by the term.

The interpretation of a query is set-theoretical. In practice, a query is a Boolean expression where the set operators are the usual intersection, union and complement, and the operands are index terms. The document subsets which corresponds to the index terms of the query are combined through the set operators. The system returns the documents which belong to the subset expressed by the query.

Historical Background

The Boolean model for Information Retrieval was proposed as a paradigm for accessing large scale systems since the 1950s. The idea of composing queries as Boolean propositions was at that time considered advantageous for the end user and simple to implement, so that its use rapidly spread in the 1960s. The main reason for the rapid advent of the Boolean model was the presence of experienced end users who also were expert of the domain of the searched documents. These users were expected to be able to quite effectively express their own information needs by composing index terms and operators as propositions. More recently, some user studies suggested that even experienced users tended to employ the less Boolean operators as possible and to submit quite simple queries.

A possible reason for the potential success of the Boolean model was the controlled, coordinated and very often manual indexing of the collections; these terms were often quite homogeneous terms, namely, referred to a restricted domain, therefore, indexing carefully avoided ambiguous terms and correctly associated related terms. Since then, the Boolean model was adopted by various other information management systems, e.g., office information systems, library catalogue systems, and database systems.

Currently, the Boolean model is quite ubiquitous since the advanced search functions constitute a constant of almost every information management systems, Web search engines included. However, it is a matter of fact that the success of this model decreased over time because of the decreasing proportion of experienced end users, the increasing heterogeneity of the document collections and the lack of a support for ranking documents. The most glaring example is the World Wide Web, which fuels the search engine indexes with more and more heterogenous Web pages accessed by an increasing number of unexperienced end users. Various research works showed that the percentage of users who use advanced search functions is very limited, even when they are quite expert of the domain or in using the interface.

Foundations

A parallel can be established between database design and Boolean model. As far as database systems are concerned, the design process aims at structuring entities, relationships and attributes accessed through relational query languages. When Boolean Information Retrieval systems are considered, the document properties are index terms and the design process aims at selecting the index terms of the document through an indexing process. Once every document has been indexed, the document collection is described by an index whose terms are associated to the documents in which they occur. The basic information retained by the system is the membership of the document to the index term.

Suppose, for example, some authors are writing their own documents, say, d_1,d_2,d_3 using, say, three terms, namely, A,B,C. The terms *unambiguously* describe one concept each – the absence of ambiguity of the natural language is an important assumption so as to make this model effective. As a document may be about one or two concepts, a document may belong to more subsets. Suppose, for example, that the collection stores the documents d_1,d_2,d_3, which respectively contains the following index terms: $\{A,C\}$, $\{A,B\}$, $\{B\}$. According to the Boolean model, three subsets $A = \{d_1,d_2\}$, $B = \{d_2,d_3\}$, $C = \{d_1\}$ are defined after indexing the collection. (It is customary to label the subset with the index term.)

The example shows that a concept is addressed by one or more documents and a term is associated to a single document subset. These document subsets are the extensional expressions of the concept described by an index term. In other words, the enumeration of the document subset is the description of the concept labeled by the index term. Indeed, the Boolean model was thought as a means for describing concepts without recurring to ontological description, but to a simple, yet powerful logical language.

The index terms and the associated document subsets are decided at indexing time by a human indexer or an automated process driven by an indexing algorithm. If the set operators were not available, the end user could use one index term at a time for formulating his own query thus having a very few degrees of freedom for formulating his own information need. Thanks to the set-based view imposed by the Boolean model, the end user can utilize the Boolean algebra for constructing new subsets. These new subsets are what results from Boolean expressions based on the classical operators, namely, intersection, union and complement. Suppose, for example, A,B,C are the document subsets associated to three index terms; let $A = \{d_1,d_2\}$, $B = \{d_2,d_3\}$, $C = \{d_1\}$ and A `AND NOT` C be a new subset. This subset contains d_2 only thus expressing the fact that d_2 is about A `AND NOT` C.

What is important here is that the construction of document subsets through the Boolean algebra allows the end user for expressing new concepts which were not explicitly thought by the authors. Indeed, the author of d_2 did not think that d_2 was about A `AND NOT` C. With this respect, the Boolean model is a powerful language because would permit to represent the knowledge stored in a document collection by means of an algebra. A system based on the Boolean model can efficiently answer these queries by performing some simple algorithms which implement set operations and process sets.

Despite its strengths and simplicity, the Boolean model has some weaknesses. Understanding the main weaknesses of the Boolean model is crucial for making the application of this model to realistic contexts effective. The weaknesses of the Boolean model can be summarized as follows (see also Cooper, 1988):

- Set operator confusion
- Expressiveness gap
- Index term ambiguity
- Null output
- Output overload

Null output and output overload are due to the lack of support for ranking documents.

Set operator confusion occurs whenever intersection and union, namely, `AND` and `OR` are exchanged when composing a query. Because of the imprecision of the natural language, humans often use "or" for saying "and," and viceversa. This confusion causes a similar exchange when using the artificial Boolean language. Suppose, for example, the collection is $\{d_4,d_5\}$, d_4 is about *A* and d_5 is about *B*. Using the natural language, end user's information need might be "the documents about *A and B*" to mean the whole document collection. To obtain the whole collection as a result, the user should use *A* `OR` *B*. However, if the user translated the "and" of his own natural language request into an `AND` operator, the translation of the request into an artificial language would be *A* `AND` *B*, which returns the empty set. Things are more difficult as the query is more complex.

Expressiveness gap is the loss of expressiveness encountered whenever an information need has to be translated to an artificial language like the Boolean language from a more expressive natural language. Suppose, for example, the end user wants to retrieve documents about the Boolean language in Information Retrieval or Database Systems. The use of the Boolean query language requires that each word or group of words is corresponded to an index term, namely, to a document subsets, and that some set operators are applied to these sets; for example, a query might be `(boolean AND information retrieval) OR database systems`. The problem of the expressiveness gap is amplified by the multiplicity of queries which can be formulated as a representation of the same information need. As a consequence, a variety of document subsets not all being the same can be retrieved. Suppose, for example, the end user wants to retrieve documents about the Boolean language in Information Retrieval or Database Systems. Potential Boolean queries are, for example:

- `(boolean AND information retrieval) OR database systems`
- `boolean AND (information retrieval OR database systems)`

The expressiveness gap occurs because the request expressed in natural language does not indicate if and where the parentheses are located, whereas the location of the parentheses is crucial when using the Boolean algebra. Although both are valid expressions, they provide different results.

Index term ambiguity occurs whenever a term has two distinct meanings (polysemy) or two terms have the same meaning (synonymy). When a term is polysemous, documents about two distinct concepts co-occur in the same subset. As a consequence, irrelevant documents may be retrieved when the end user employs the term in his own query. Suppose, for example, `bank` $= \{d_6,d_7\}$, but d_6 is about bank as river bank and d_7 is about bank as bank branch. When the user expresses his own Boolean query using `bank` as term, both documents are retrieved. If, however, the user needs information about river banks, one irrelevant document, i.e., d_7, is retrieved. In this event, precision is lower than the precision measured if ambiguity does not occur. When two terms are synonyms, two distinct document subsets are defined, yet the documents are about the same concept. As a consequence, relevant documents may be missed when the end user employs only one of the two terms in his own query. Suppose, for example, `personalcomputer` $= \{d_8,d_9\}$ and `PC` $= \{d_{10}\}$, but `PC` is the acronym of `personal computer` and therefore d_{10} should be retrieved, but it is not. When the user expresses his own Boolean query using `PC` as term, only d_{10} is retrieved. If, however, the use needs information about personal computers, two relevant documents, i.e., d_8,d_9, are missed. In this event, recall decreases.

Null output occurs whenever the end user submits a very restrictive query to the system to an extent to make the returned document subset very small, if not even empty. Limit examples are terms being absent from the index, which are associated to the empty document set or two index terms whose intersection is empty because there are no common documents. The event of intersecting disjoint document subsets is very probable since the document subsets are usually small if compared with the document collection. Moreover, Boolean systems do not usually keep track of the semantic relationships between terms, and therefore connecting by AND two semantically related terms whose document subsets are disjoint would give the empty set as result. To reduce null output, some query terms can be related by OR or some AND should be removed by the user.

Output overload occurs whenever the result document subset is too large for being effectively browsed by the end user. This drawback often happens when too many ORs or too few ANDs are utilized for formulating the query. Even though a document subset is small if compared with the document collection, a subset of, say, 1,000 documents is very large for the end user who is required to inspect all of them. A strategy for avoiding output overload is reducing the number of OR or adding some AND.

Overall, the side effects of null output and output overload are hardly controlled by the end user who is requested to add and delete terms or Boolean operators so as to make the query an effective description of his own information need. This task, which is called Query Expansion, would overload the cognitive effort of the end user, if performed using a Boolean systems.

To overcome the problems of null output and output overload, the notion of level of coordination was introduced. The level of coordination is a measure of the degree to which each returned document matches the query. In this way, the level of coordination provides a score for ranking the documents. This ranking allows the user for deciding how many documents to inspect and the system for cutting the bottom ranked documents off the list.

The level of coordination is calculated as follows. A Boolean query is transcribed in conjunctive normal form, namely, as a list of propositions related by AND – each proposition is a disjunction of terms. The level of coordination of a document is the number of propositions satisfied by the document. Suppose, for example, A,B,C are the document subsets associated to three index terms; let $A = \{d_1,d_2\}$, $B = \{d_2,d_3\}$, $C = \{d_3\}$ and A AND (C OR B) be the query; the level of coordination of d_1 is 1 because only the first proposition is satisfied by d_1, the level of d_2 is 2 and that of d_3 is 1 – d_1,d_3 would have not been retrieved if the level of coordination were not been calculated because they do not satisfy the query.

A variation of the level of coordination was introduced for taking the variable size of document subsets into account – indeed, the document subsets are of arbitrary size and therefore a small subset may be treated as a large subset. This variation has been called weighted level of coordination: instead of assigning a constant weight to each proposition made true by a document, a different weighted is assigned depending on the proposition; for example, an IDF may be used. The weighed level of coordination is then the sum of the weights assigned to every proposition.

Key Applications

The research papers and articles on the Boolean model flourished until when other models, e.g., the probabilistic models and the vector-space model, appeared on the scene of the Information Retrieval theater and the Web search engines drastically changed the audience and the document collections. The knowledge of this model, however, is a basic element of an Information Retrieval system because the Boolean query language is provided by many if not all information management systems. Of course, this model is the starting point for facing the non-classical logic models for Information Retrieval.

Cross-references

▶ Indexing Units
▶ Logical Models of Information Retrieval
▶ Precision
▶ Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model
▶ Query Expansion for Information Retrieval
▶ Query Expansion Models
▶ Recall
▶ Relevance Feedback
▶ Vector-Space Model

Recommended Reading

1. Bar-Hillel Y. Language and information. Addison-Wesley, Reading, MA, USA, 1964.
2. Belkin N.J., Cool C., Croft W.B., and Callan J.P. The Effect of Multiple Query Representations on Information Retrieval System Performance. In Proc. 16th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1993, pp. 339–346.
3. Blair D. Language and representation in information retrieval. Elsevier, Amsterdam, 1990.
4. Cooper W. Getting beyond Boole. Inform. Process. Manage., 24:243–248, 1988.
5. Croft W., Turtle H., and Lewis D. The Use of phrases and structured queries in information retrieval. In Proc. 14th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1991, pp. 32–45.

6. Grefenstette G. (ed.), Cross-Language Information Retrieval, International Series on Information Retrieval, Kluwer Academic, Dordecht, 1998.
7. Hersh W. and Hickam D. An evaluation of interactive Boolean and natural language searching with an online medical textbook. J. Am. Soc. Inform. Sci., 46(7):478–489, 1995.
8. Hull D. A weighted Boolean model for Cross Language Text Retrieval. In Grefenstette [6], pp. 119–136.
9. Korfhage R. Information Storage and Retrieval. Wiley, New York, 1997.
10. Kowalski G. and Maybury M. Information retrieval systems: Theory and implementation. Kluwer, Dordecht, 2000.
11. Lancaster F. and Warner A. Information retrieval today. Information Resources, Arlington, VA, 1993.
12. Lee J. Properties of Extended Boolean Models in Information Retrieval. In Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1994, pp. 182–190.
13. Lee J., Kim W., Kim M., and Lee Y. On the evaluation of Boolean operators in the extended Boolean retrieval framework. In Proc. 16th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1993, pp. 291–297.
14. Radecki T. Generalized boolean methods of information Retrieval. Int. J. Man–Machine Studies, 18(5):407–439, 1983.
15. van Rijsbergen C. The Geometry of Information Retrieval. Cambridge University Press, UK, 2004.
16. Wong S., Ziarko W., Raghavan V., and Wong P. Extended boolean query processing in the generalized vector space model. Inform. Syst., 14(1):47–63, 1989.

Boosting

Zhi-Hua Zhou

Nanjing University, Nanjing, China

Definition

Boosting is a kind of ensemble method which produces a strong learner that is capable of making very accurate predictions by combining rough and moderately inaccurate learners (which are called as *base learners* or *weak learners*). In particular, Boosting sequentially trains a series of base learners by using a *base learning algorithm*, where the training examples wrongly predicted by a base learner will receive more attention from the successive base learner. After that, it generates a final strong learner through a weighted combination of these base learners.

Historical Background

In 1988, Kearns and Valiant posed an interesting question for the research of computational learning theory, i.e., whether a *weak* learning algorithm that performs just slightly better than random guess can be "boosted" into an arbitrarily accurate *strong* learning algorithm. In other words, whether two complexity classes, *weakly learnable* and *strongly learnable* problems, are equal. In 1989, Schapire [9] proved that the answer to the question is "yes", and the proof he gave is a construction, which is the first Boosting algorithm. One year later, Freund developed a more efficient algorithm. However, both algorithms suffered from some practical drawbacks. Later, in 1995, Freund and Schapire [4] developed the AdaBoost algorithm

```
Input: Data set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$;
       Base learning algorithm $\mathcal{L}$;
       Number of learning rounds $T$.
Process:
  $D_1(i) = 1/m$.        % Initialize the weight distribution
  for $t = 1, \cdots, T$:
      $h_t = \mathcal{L}(\mathcal{D}, D_t)$;        % Train a base learner $h_t$ from $\mathcal{D}$ using distribution $D_t$
      $\epsilon_t = \Pr_{i \sim D_i}[h_t(x_i \neq y_i)]$;        % Measure the error of $h_t$
      $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$; % Determine the weight of $h_t$
      $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases}$
                $= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$   % Update the distribution, where $Z_t$ is
                        % a normalization factor which enables $D_{t+1}$ be a distribution
  end.
Output: $H(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

Boosting. Figure 1. The AdaBoost algorithm.

which is effective and efficient in practice, and then a hot wave of research on Boosting arose.

Foundations

AdaBoost is the most influential Boosting algorithm. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the instance space and the set of class labels, respectively, and assume $\mathcal{Y} = \{-1, +1\}$. Given a training set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ $(i = 1, \ldots, m)$, and a base learning algorithm which can be decision tree, neural networks or any other learning algorithms, the AdaBoost algorithm works as follows.

First, it assigns equal weights to all the training examples (x_i, y_i) $(i \in \{1, \ldots, m\})$. Denote the distribution of the weights at the tth learning round as D_t. From the training set and D_t the algorithm generates a base learner $h_t : \mathcal{X} \to \mathcal{Y}$ by calling the base learning algorithm. Then, it uses the training examples to test h_t, and the weights of the incorrectly classified examples will be increased. Thus, an updated weight distribution D_{t+1} is obtained. From the training set and D_{t+1} AdaBoost generates another base learner by calling the base learning algorithm again. Such a process is repeated for T times, each of which is called a *round*, and the final learner is derived by weighted majority voting of the T base learners, where the weights of the learners are determined during the training process. In practice, the base learning algorithm may be a learning algorithm which can use weighted training examples directly; otherwise the weights can be exploited by sampling the training examples according to the weight distribution D_t. The pseudo-code of AdaBoost is shown in Fig. 1.

Freund and Schapire [4] proved that the training error of the final learner H is upper-bounded by

$$\epsilon_D = \Pr_{i \sim \mathcal{D}}[H(x_i) \neq y_i] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1 - \epsilon_t)},$$

which can be written as

$$\epsilon_D \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2 \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)},$$

where $\gamma_t = 1/2 - \epsilon_t$. Thus, if each base learner is slightly better than random so that $\gamma_t \geq \gamma$ for some $\gamma > 0$, the training error will drop exponentially fast in T since the upper bound is at most $e^{-2T\gamma^2}$.

Freund and Schapire [4] also gave a generalization error bound of H in terms of its training error $\epsilon_{\mathcal{D}}$, the size m of the training set, the VC-dimension d of the base learner space, and the number of rounds T, by

$$\epsilon \leq \epsilon_{\mathcal{D}} + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

with high probability, where $\tilde{O}(\cdot)$ is used to hide all logarithmic and constant factors [10] instead of using $O(\cdot)$ which hides only constant factors.

The above generalization error bound suggests that AdaBoost will overfit if it runs for many rounds since T is in the numerator. However, empirical observations show that AdaBoost often does *not* overfit even after a large number of rounds, and sometimes it is even able to reduce the generalization error after the training error has already reached zero. Thus, later, Schapire et al. [11] presented another generalization error bound,

$$\epsilon \leq \Pr_{i \sim \mathcal{D}}\left[\text{margin}_f(x_i, y_i) \leq \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

for any $\theta > 0$ with high probability, where the *margin* of f on (x_i, y_i) was defined as

$$\text{margin}_f(x_i, y_i) = \frac{y_i f(x_i)}{\sum_{t=1}^{T} |\alpha_t|} = \frac{y_i \sum_{i=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} |\alpha_t|},$$

whose value is in $[-1, +1]$ and is positive only if H classifies (x_i, y_i) correctly. In fact, the magnitude of margin can be explained as a measure of confidence in prediction. The larger the magnitude of margin, the higher confidence of prediction. Note that when $H(x_i) = y_i$, $\text{margin}_f(x_i, y_i)$ can still be increased as t increases. Thus, the above margin-based generalization bound gives an answer to the question of why AdaBoost is able to reduce the generalization error even after the training error reaches zero, that is, the confidence in prediction can be increased further. However, Breiman [2] showed that improving the margin does not necessarily lead to the improvement of generalization, which doubted the above margin-based explanation to AdaBoost.

In addition to the previously mentioned studies, the behavior of AdaBoost has been explained from the views of game theory [2,3], additive model [5], etc. Many variants or extensions of AdaBoost have been developed [6,10], which makes Boosting become a big family of ensemble methods.

In contrast to another famous ensemble method, Bagging (which reduces *variance* significantly but has little effect on *bias*), Boosting can significantly reduce bias in addition to reducing variance. So, on weak learners such as decision stumps, which are one-level decision trees, Boosting is usually more effective.

Key Applications

The first application of Boosting was on Optical Character Recognization by Drucker et al. Later, Boosting was applied to diverse tasks such as text categorization, speech recognization, image retrieval, medical diagnosis, etc. [6,10]. It is worth mentioning that AdaBoost has been combined with a cascade process for face detection [12], and the resulting face detector was 15 times faster than state-of-the-art face detectors at that time but with comparable accuracy, which was recognized as one of the major breakthroughs in computer vision (in particular, face detection) during the past decade.

Future Directions

The margin-based explanation to why Boosting often does not overfit was seriously challenged by Breiman's indication that larger margin does not necessarily mean better generalization [2]. Recently, Reyzin and Schapire [8] found that Breiman considered minimum margin instead of average or median margin. If the margin-based explanation can survive, it may be possible to establish a unified theoretical framework for the two powerful learning approaches, i.e., Boosting and support vector machine, since it is well-known that support vector machine works by maximizing the margin in a feature space.

It has been observed that Boosting performs poorly when abundant noise exists. Making Boosting more robust to noise is an important task. Moreover, Boosting suffers from some general deficiencies of ensemble methods, such as the lack of comprehensibility, i.e., the knowledge learned by Boosting is not understandable to the user. Trying to overcome those deficiencies is an important future direction.

Experimental Results

Empirical studies on Boosting have been reported in many papers, such as [1,7].

Data Sets

A large collection of datasets commonly used for experiments can be found at http://www.ics.uci.edu/∼mlearn/MLRepository.html.

URL to Code

The code of an extended AdaBoost algorithm, BoosTexter, which was designed for multi-class text categorization, can be found at http://www.cs.princeton.edu/ schapire/boostexter.html.

Cross-references

▶ Bagging
▶ Decision Tree
▶ Ensemble
▶ Neural Networks
▶ Support Vector Machine

Recommended Reading

1. Bauer E. and Kohavi R. An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants. Mach. Learn., 36(1–2):105–139, 1999.
2. Breiman L. Prediction games and arcing classifiers. Neural Comput., 11(7):1493–1517, 1999.
3. Freund Y. and Schapire R.E. Game theory, on-line prediction and Boosting. In Proc. Ninth Annual Conf. on Computational Learning Theory, 1996, pp. 325–332.
4. Freund Y. and Schapire R.E. A decision-theoretic generalization of on-line learning and an application to Boosting. J. Comput. Syst. Sci., 55(1):119–139, 1997. (A short version appeared in the Proceedings of EuroCOLT'95).
5. Friedman J., Hastie T., and Tibshirani R. Additive logistic regression: A statistical view of Boosting with discussions. Ann. Stat., 28(2):337–407, 2000.
6. Meir R. and Rätsch G. An introduction to Boosting and leveraging. In Advanced Lectures in Machine Learning, S. Mendelson and A.J. Smola (eds.). LNCS, Vol. 2600, Springer, Berlin, 2003, pp. 118–183.
7. Opitz D. and Maclin R. Popular ensemble methods: An empirical study. J. Artif. Intell. Res., 11:169–198, 1999.
8. Reyzin L. and Schapire R.E. How boosting the margin can also boost classifier complexity. In Proc. 23rd Int. Conf. on Machine Learning, 2006, pp. 753–760.
9. Schapire R.E. The strength of weak learnability. Mach. Learn., 5(2):197–227, 1990.
10. Schapire R.E. The Boosting approach to machine learning: An overview. In Nonlinear Estimation and Classification. D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick, and B. Yu (eds.). Springer, Berlin, 2003.
11. Schapire R.E., Freund Y., Bartlett P., and Lee W.S. Boosting the margin: A new explanation for the effectiveness of voting methods. Ann. Stat., 26(5):1651–1686, 1998.

12. Viola P. and Jones M. Rapid object detection using a boosted cascade of simple features. In Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition, 2001, pp. 511–518.

Bootstrap

Hwanjo Yu
University of Iowa, Iowa City, IA, USA

Synonyms

Bootstrap sampling; Bootstrap estimation

Definition

The bootstrap is a statistical method for estimating the performance (e.g., accuracy) of classification or regression methods. The bootstrap is based on the statistical procedure of sampling *with replacement*. Unlike other estimation methods such as cross-validation, the same object or tuple can be selected for the training set more than once in the boostrap. That is, each time a tuple is selected, it is equally likely to be selected again and re-added to the training set.

Historical Background

The bootstrap sampling was developed by Bradley Efron in 1979, and mainly used for estimating the statistical parameters such as mean, standard errors, etc. [2]. A meta-classification method using the bootstrap called *bootstrap aggregating* (or *bagging*) was proposed by Leo Breiman in 1994 to improve the classification by combining classifications of randomly generated training sets [1].

Foundations

This section discusses a commonly used bootstrap method, *.632 bootstrap*. Given a dataset of N tuples, the dataset is sampled N times, with replacement, resulting in a *bootstrap sample* or training set of N tuples. It is very likely that some of the original data tuples will occur more than once in the training set. The data tuples that were not sampled into the training set end up forming the test set. If this process is repeated multiple times, on average 63.2% of the original data tuples will end up in the training set and the remaining 36.8% will form the test set (hence, the name, .632 bootstrap).

The figure, 63.2%, comes from the fact that a tuple will not be chosen with probability of 36.8%. Each tuple has a probability of $1/N$ of being selected, so the probability of not being chosen is $(1 - 1/N)$. The selection is done N times, so the probability that a tuple will not be chosen during the whole time is $(1 - 1/N)^N$. If N is large, the probability approaches $e^{-1} = 0.368$. Thus, 36.8% of tuples will not be selected for training and thereby end up in the test set, and the remaining 63.2% will form the training set.

The above procedure can be repeated k times, where in each iteration, the current test set is used to obtain an accuracy estimate of the model obtained from the current bootstrap sample. The overall accuracy of the model is then estimated as

$$Acc(M) = \sum_{i=1}^{k} (0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set}), \quad (1)$$

where $Acc(M_i)_{train\_set}$ and $Acc(M_i)_{test\_set}$ are the accuracy of the model obtained with bootstrap sample i when it is applied to training set and test set respectively in sample i.

Key Applications

The bootstrap method is preferably used for estimating the performance when the size of dataset is relatively small.

Cross-references

▶ Cross-Validation
▶ Holdout Test
▶ Sampling
▶ Validation

Recommended Reading

1. Breiman L. Bagging predictors. Machine Learning, 1996.
2. Efron B. and Tibshirani R.J. An Introduction to the Bootstrap. CRC Press, Boca Raton, 1994.

Bootstrap Aggregating

▶ Bagging

Bootstrap Estimation

▶ Bootstrap

Bootstrap Sampling

▶ Bootstrap

Bottom-up Semantics

▶ Emergent Semantics

Boyce-Codd Normal Form

Marcelo Arenas
Pontifical Catholic University of Chile, Santiago, Chile

Synonyms

BCNF

Definition

Let $R(A_1,...,A_n)$ be a relation schema and Σ a set of functional dependencies over $R(A_1,...,A_n)$. Then (R, Σ) is said to be in Boyce-Codd Normal Form (BCNF) if for every nontrivial functional dependency $X \rightarrow A$ implied by Σ, it holds that X is a superkey for R.

Key Points

In order to avoid update anomalies in database schemas containing functional dependencies, BCNF was introduced by Codd (In [2], Codd pointed out that this normal form was developed by Raymond F. Boyce and himself.) in [2]. This normal form is defined in terms of the notion of superkey as shown above. For example, given a relation schema $R(A, B, C)$ and a set of functional dependencies $\Sigma = \{AB \rightarrow C, C \rightarrow B\}$, it does not hold that $(R(A, B, C), \Sigma)$ is in BCNF since C is not a superkey for R. On the other hand, $(S(A, B, C), \Gamma)$ is in BCNF if $\Gamma = \{A \rightarrow BC\}$, since A is a superkey for S in this case.

It should be noticed that relation schema $R(A, B, C)$ above is in 3NF if $\Sigma = \{AB \rightarrow C, C \rightarrow B\}$, although this schema is not in BCNF. In fact, BCNF is strictly stronger than 3NF; every schema in BCNF is in 3NF, but there exist schemas (as the one shown above) that are in 3NF but not in BCNF.

For every normal form two problems have to be addressed how to decide whether a schema is in that normal form, and how to transform a schema into an equivalent one in that normal form. As opposed to the case of 3NF, it can be tested efficiently whether a relation schema is in BCNF. A relation schema (R, Σ) is in BCNF if and only if for every nontrivial functional dependency $X \rightarrow Y \in \Sigma$, it holds that X is a superkey. Thus, it is possible to check efficiently whether (R, Σ) is in BCNF by using the linear-time algorithm for functional dependency implication developed by Beeri and Bernstein [1]. On the negative side, given a relation schema S, it is not always possible to find a database schema S' such that S' is in BCNF and S' is a lossless and dependency preserving decomposition of S. In fact, relation schema $R(A, B, C)$ and set of functional dependencies $\{AB \rightarrow C, C \rightarrow B\}$ does not admit a dependency preserving decomposition in BCNF.

Cross-references

▶ Fourth Normal Form (4NF)
▶ Normal Forms and Normalization
▶ Second Normal Form (2NF)
▶ Third Normal Form (3NF)

Recommended Reading

1. Beeri C. and Bernstein P. Computational Problems Related to the Design of Normal Form Relational Schemas. ACM Trans. Database Sys., 4(1):30–59, 1979.
2. Codd E.F. Recent Investigations in Relational Data Base Systems. In Proc. IFIP Congress. 1974, pp. 1017–1021.

BP-Completeness

Dirk Van Gucht
Indiana University, Bloomington, IN, USA

Synonyms

Instance-completeness; Relation-completeness

Definition

A relational query language Q is BP-complete if for each relational database D, the set of all relations defined by the queries of Q on D is equal to the set of all first-order definable relations over D. More formally, fix some infinite universe **U** of atomic data elements. A *relational database schema* S is a finite set of relation names, each with an associated arity. A *relational database* D with schema S assigns to each relation name of S a *finite* relation over **U** of its arity. The domain of D, $dom(D)$, is the set of all atomic data elements occurring in the tuples of its relations. Let FO^S be the set of first-order formulas over signature S and the equality predicate, and let $FO^S(D) = \{\varphi(D) | \varphi \in FO^S\}$. (For a formula $\varphi \in FO^S$ with free variables $(x_1,\ldots,x_m)$, $\varphi(D)$ denotes the m-ary relation over $dom(D)$ defined by φ, where the variables in φ are assumed to range over $dom(D)$.) Let Q^S denote those queries of Q defined over schema S, and let $Q^S(D) = \{q(D) | q \in Q^S\}$, i.e., the set of relations defined by queries of Q applied to D. Then, Q is *BP- complete* if for each relational database D over schema S,

$$Q^S(D) = FO^S(D).$$

In the words of Chandra and Harel, "BP-completeness can be seen to be a measure of the power of a language to express relations and *not* of its power to express functions having relations as outputs, i.e., queries." In fact, there exist BP-complete languages that do not express the same queries.

Key Points

Chandra and Harel introduced the concept of BP-completeness and attributed it to Bancilhon and Paredaens who were the first to study it. Bancilhon and Paredaens considered the following decision problem: given a relational database D and a relation R defined over $dom(D)$, does there exists a first-order formula φ such that $\varphi(D) = R$? (Paredaens considered this problem for the relational algebra, but by Codd's theorem on the equivalence of first-order logic and the relational algebra, these decision problems are the same.) They gave an algebraic, language-independent characterization of this problem by showing that such a first-order formula exists if and only if for each bijection $h : dom(D) \rightarrow dom(D)$, if $h(D) = D$ then $h(R) = R$. Equivalently, if h is an automorphism of D then it is also an automorphism of R. (Here, $h(D)$ and $h(R)$ are the natural extensions of h to D and R, respectively.) For a relational database D over schema S and a relation R, denote by $Aut(D)$ and $Aut(R)$ the sets of automorphisms of D and R, respectively, and let $R^S(D) = \{R \mid R$ is a relation over $dom(D)$ such that $Aut(D) \subseteq Aut(R)\}$. Then, an alternative characterization for the BP-completeness of Q is to require that for each relational database D over schema S,

$$Q^S(D) = R^S(D).$$

Van den Bussche showed that this characterization follows from Beth's Theorem about the explicit and implicit definability of first-order logic. The concept of BP-completeness has been generalized as well as specialized to query languages over other database models.

Cross-references

▶ Complete Query Languages
▶ Query Language
▶ Relational Calculus and Algebra

Recommended Reading

1. Bancilhon F. On the completeness of query languages for relational databases. In Proc. Mathematical Foundations of Computer Science, 1978.
2. Chandra A. and Harel D. Computable Queries for relational databases. J. Comput. Syst. Sci., 25:99–128, 1982.
3. Paredaens J. On the expressive power of the relational algebra. Inform. Process. Lett., 7(2):107–111, 1978.
4. Van den Bussche. J. Applications of Alfred Tarski's Ideas in Database Theory. In Computer Science Logic, Lect. Notes Comput. Sci., 2142:20–37, 2001.

BPEL

▶ Business Process Execution Language

BPEL4WS

▶ Business Process Execution Language

BPMN

▶ Business Process Modeling Notation

Bpref

Nick Craswell
Microsoft Research Cambridge, Cambridge, UK

Definition

Bpref is a preference-based information retrieval measure that considers whether relevant documents are ranked above irrelevant ones. It is designed to be robust to missing relevance judgments, such that it gives the same experimental outcome with incomplete judgments that Mean Average Precision would with complete judgments.

Key Points

In a test collection where all relevant documents have been identified, experiments using bpref and MAP should give the same outcome, for example both systems should agree that system A is better than system B. However, if the relevance judgments are incomplete, for example where only half the pool has been judged, MAP becomes unstable and may incorrectly show that system B is better than system A. The bpref measure was developed to maintain the correct ordering of systems (A better than B) even with incomplete judgments.

Given a ranked list of search results and a set of R known relevant documents and N known irrelevant documents, bpref first identifies the top-R irrelevant documents in the list. For these irrelevant documents n and relevant documents r the measure is:

$$bpref = \frac{1}{R}\sum_{r}\left(1 - \frac{|n \text{ ranked higher than } r|}{\min(R, N)}\right)$$

The bpref paper [1] found good agreement between full-judgment MAP and reduced-judgment bpref even if the judged set was reduced by 80%. Later work has introduced other measures with similar or better properties, notably Inferred Average Precision [2].

Cross-references

▶ Mean Average Precision
▶ Precision-Oriented Effectiveness Measures

Recommended Reading

1. Buckley C. and Voorhees E.M. Retrieval evaluation with incomplete information. In Proc. 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 25–32.
2. Yilmaz E. and Aslam J.A. Estimating average precision with incomplete and imperfect judgments. In Proc. Int. Conf. on Information and Knowledge Management, 2006, pp. 102–111.

Branch

▶ OR-Split

Bridging

▶ Mediation

Browsing

Kent Wittenburg
Mitsubishi Electric Research Laboratories, Inc., Cambridge, MA, USA

Synonyms

Perusal; Scanning

Definition

"Browsing" has two definitions in the context of visual interfaces for database systems:

1. The human activity of visual perception and interpretation of electronic content when there is no specific target object being sought.
2. The human activity of clicking or tapping on a sequence of elements in an information display that results in a sequence of screens of information state.

Definition (1) implies something about the mental intent of the user performing the act of information processing. The intent can be to learn the gist of "what is there." It may be to take in information in order to be entertained or informed. Or it may simply be a part of involuntary visual scanning within an environment. In the physical world, examples include flipping through the pages of a book, scanning quickly through a menu, or glancing down a store aisle.

Definition (2) is used in the context of contrasting the human-computer interaction paradigm of link-following in World Wide Web applications versus searching, which entails forming queries. "Browsing" in this context refers to a sequence of clicking or tapping behaviors on highlighted elements (hyperlinks) in order to go directly to a subsequent state of the information display. In contrast, "searching" entails specifying and then executing a query. The interaction paradigms of browsing and searching are independent of the intentional state of the user – the individual may or may not be looking for something specific in either case.

Key Points

A branch of research in visual interfaces for browsing in the first sense is Rapid Serial Visual Presentation (RSVP) [2, Sect. 4.2]. The human psychology of visual perception, specifically models of short-term visual memory, can inform the design of visual interfaces for browsing. Visual information can be presented in space or in time or in some combination thereof. For image presentation specifically, is it better to utilize screen real estate to present multiple images concurrently or to present them over time? If over time, should images move in some specified path or remain motionless? What are the best controls to give to the user in order to support the tasks of rapid perusal in order to get a gist of the content within that information space or to help in making a selection for further detailed investigation? These are some of the questions being asked by researchers and designers in the context of RSVP interfaces.

An important aspect of browsing as link-following has to do with information "scent." Just as animals' scent can be used as an indicator of their territory or former presence, the presentation of information links provides hints of what a user would find were he or she to follow the link. The theory of information foraging [1] provides insight into human behavior in information seeking that can help inform the design for interfaces and electronic documents. According to Pirolli, humans tend to make decisions about what links to follow and how long to continue along certain paths based on resource-limited assessments of the costs of such activities relative to a judgment of payoff.

Cross-references

- ▶ Browsing in Digital Libraries
- ▶ Discovery
- ▶ Human-Computer Interaction
- ▶ Information Foraging
- ▶ Information Navigation
- ▶ Information Retrieval
- ▶ Searching Digital Libraries
- ▶ Visual Interaction
- ▶ Visual Interfaces
- ▶ Web Information Retrieval Models

Recommended Reading

1. Pirolli P. Information Foraging Theory: Adaptive Interaction with Information. Oxford University Press, New York, NY, 2007.
2. Spence R. Information Visualization: Design for Interaction (2nd edn.). Pearson/Prentice Hall, Upper Saddle River, NJ, 2007.

Browsing in Digital Libraries

RAO SHEN[1], EDWARD A. FOX[2]
[1]Yahoo!, Sunnyvale, CA, USA
[2]Virginia Tech, Blacksburg, VA, USA

Synonyms

Exploring; Surfing; Looking over/through

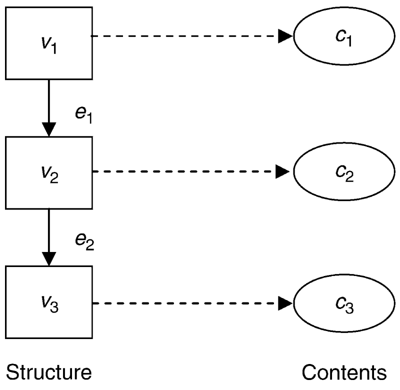
Definition

Informally, browsing is a process that involves looking through a collection of information. Thus, in a traditional library, one may wander about the stacks, glancing at titles of works, in regions where one expects to find interesting material. In the broadest sense, browsing is considered as one type of exploration, typically less directed or purposeful than searching, in that the goal or result is not always precisely known in advance. In the World Wide Web, this is very common, making use of "browsers" like Firefox or Internet Explorer. Sometimes the colorful term "surfing" is used when it seems appropriate to emphasize the excitement that some feel when exploring new Web content, or the thrill of getting closer to some desirable result. Yet, the WWW is but one example of a hypertext (or, if multiple media types are considered, hypermedia) environment. Accordingly, in a digital library, which many believe must include a hypertext [4], browsing involves moving from one information object to another according to some connections or links or structures.

Historical Background

A hypertext is a high level interactive navigational structure which allows non-sequential exploration of and access to information [3]. In the most general case, it consists of nodes connected by directed links in a graph structure. A directed link in the hypertext graph is called a hyperlink. Today's most famous hypertext system is the World Wide Web. The Web's structure contains hyperlinks connecting nodes that each can be associated with either a complete document (e.g., HTML page) or with a location in a document (e.g., a start tag in an HTML page).

Three models for Web browsing defined by Baeza-Yates and Ribeiro-Neto [1] are flat browsing, structure guided browsing, and utilizing the hypertext model. If a document collection is organized as a one dimensional list, exploring the document space is an example of flat browsing; if the organization is hierarchical instead of flat, then browsing is structure guided. The first two models have a hypertext graph that is made up of a disconnected bipartite graph and a tree, respectively, so they are special cases of the third model.



Browsing in Digital Libraries. Figure 1. A simple hypertext.

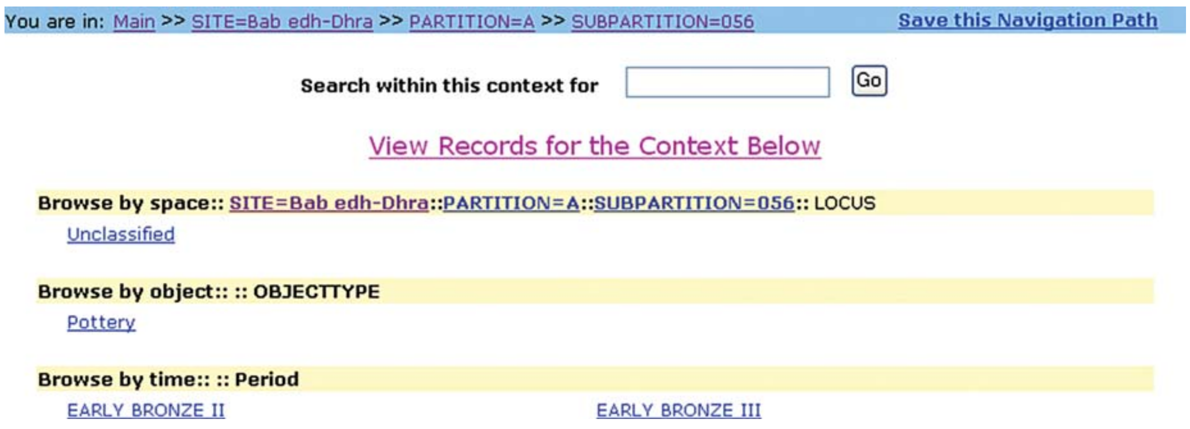
Browsing and searching are two paradigms for finding things on the Web. Some Web search engines provide Web directories used for browsing (and also for searching). Web directories are hierarchical taxonomies that classify human knowledge. Although a taxonomy can be considered as a tree, there can be cross references. One of the advantages of browsing a Web directory, in most cases, is that if a user finds what she is looking for in the taxonomy, then the answer that is found classified under that category almost certainly will be useful. In Web directories, a search can be reduced to a sub-tree of the taxonomy. However, searching may miss related pages that are not in that part of the taxonomy, if a user cannot formulate her information need sufficiently broadly.

Foundations

Browsing and searching are two methods of pursuing information on the Web; they also are two major paradigms for exploring digital libraries. The Web has no maintenance organization. Individuals add and delete pages at will. On the other hand, digital libraries require proper collection maintenance, and they will succeed only if their content is well organized [2].

The 5S (Streams, Structures, Scenarios, Spaces, and Societies) framework [4], which is used as a formal base upon which to describe digital libraries, defines structures to specify the way in which parts of a whole are managed or organized. In such digital libraries, structures can represent hypertexts.

For example, Fig. 1 illustrates a simple hypertext designed to provide access to objects in a digital library, in chronological order. It is made up of structural hyperlinks that follow chronological order (see solid



You are in: Main >> SITE=Bab edh-Dhra >> PARTITION=A >> SUBPARTITION=056 Save this Navigation Path

Search within this context for [ ] Go

View Records for the Context Below

Browse by space:: SITE=Bab edh-Dhra::PARTITION=A::SUBPARTITION=056:: LOCUS
Unclassified

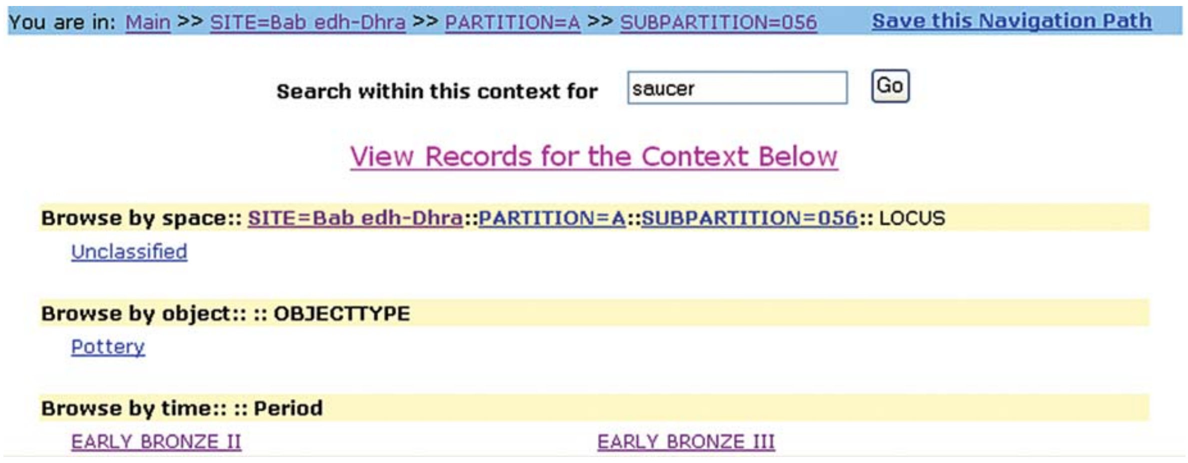
Browse by object:: :: OBJECTTYPE
Pottery

Browse by time:: :: Period
EARLY BRONZE II EARLY BRONZE III

Browsing in Digital Libraries. Figure 2. Multi-dimensional browsing.

arrows in Fig. 1) and external reference links (see dashed arrows in Fig. 1).

Thus, a formal and precise definition of browsing is illustrated in Fig. 1: Given a hypertext of a digital library, with vertex set V and edge set E, browsing is a set of sequences of traverse link events over the hypertext, such that event e of traversing from node v_k to v_t is associated with a function which retrieves the contents of node v_t. Hence, browsing a digital library is the


You are in: Main >> SITE=Bab edh-Dhra >> PARTITION=A >> SUBPARTITION=056 Save this Navigation Path

Search within this context for saucer Go

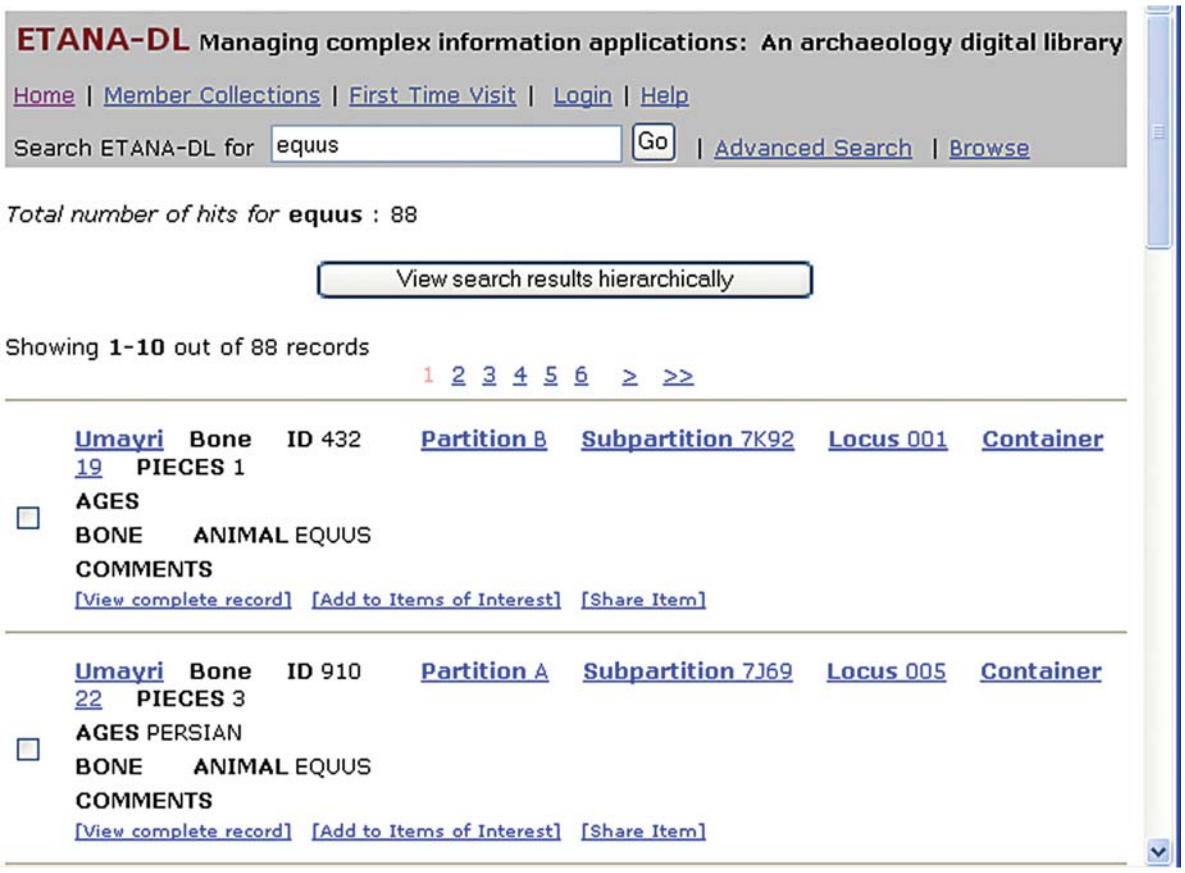
View Records for the Context Below

Browse by space:: SITE=Bab edh-Dhra::PARTITION=A::SUBPARTITION=056:: LOCUS
Unclassified

Browse by object:: :: OBJECTTYPE
Pottery

Browse by time:: :: Period
EARLY BRONZE II EARLY BRONZE III

Browsing in Digital Libraries. Figure 3. Search saucer records.


ETANA-DL Managing complex information applications: An archaeology digital library

Home | Member Collections | First Time Visit | Login | Help

Search ETANA-DL for equus Go | Advanced Search | Browse

Total number of hits for **equus** : 88

View search results hierarchically

Showing **1-10** out of 88 records

1 2 3 4 5 6 > >>

Umayri Bone ID 432 Partition B Subpartition 7K92 Locus 001 Container 19 PIECES 1
AGES
BONE ANIMAL EQUUS
COMMENTS
[View complete record] [Add to Items of Interest] [Share Item]

Umayri Bone ID 910 Partition A Subpartition 7J69 Locus 005 Container 22 PIECES 3
AGES PERSIAN
BONE ANIMAL EQUUS
COMMENTS
[View complete record] [Add to Items of Interest] [Share Item]

Browsing in Digital Libraries. Figure 4. Equus records are retrieved through basic searching.

procedure of navigating the hypertext, and it also can be understood as a traversal of a directed linked graph.

Returning to the previous example, one can observe that the left part of Fig. 1 is a directed linked graph (V, E), where each the right part is a set of contents $C = \{c_1, c_2, c_3\}$, and where c_i $(1 <= i <= 3)$ is the contents of a Web page associated with node v_i. An event of traversing along edge $e_1 = (v_1, v_2)$ is associated with a function: $E \rightarrow C$, which retrieves the contents of node v_2, i.e., c_2.

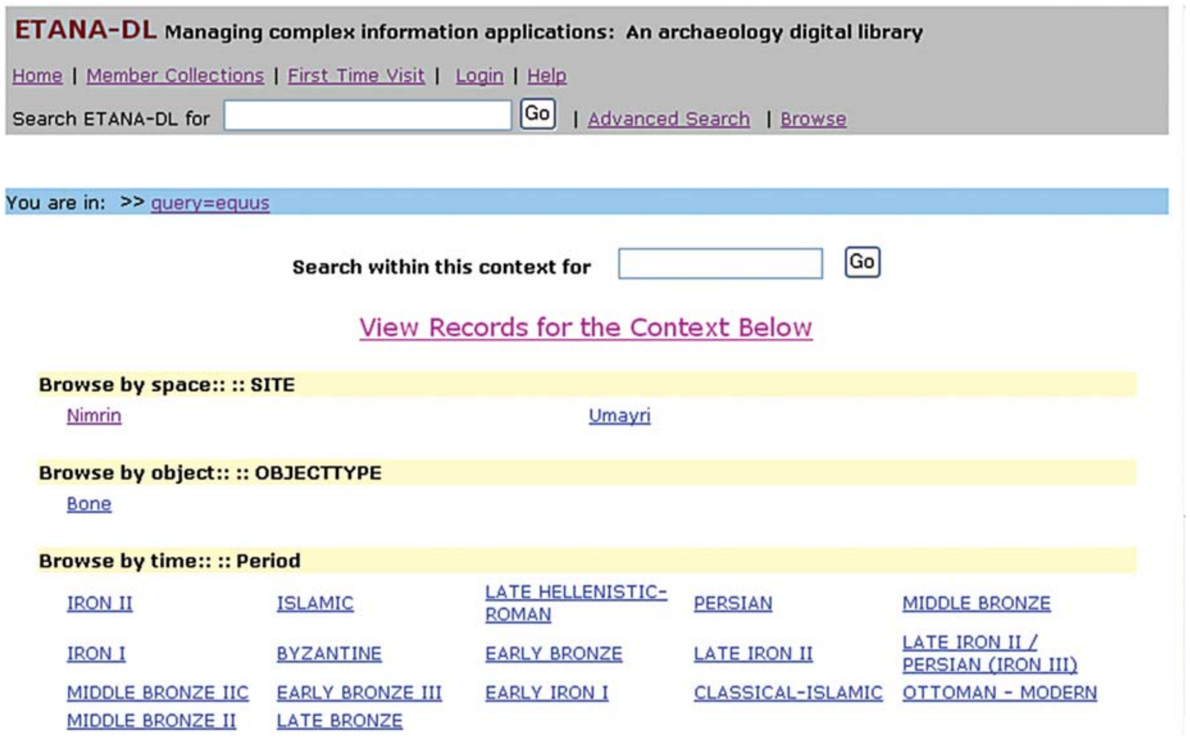
Note that (sorted) lists and hierarchical trees can be represented as hypertexts, so browsing can be over common organizations, such as alphabetical lists of authors, or indented lists of categories and subcategories; these are familiar from books with author indexes or tables of contents.

Key Applications

Modeling and analyzing browsing in digital libraries helps ease development and maintenance in those digital libraries. Some domain specific digital libraries have heterogeneous data that should be organized using several schemes. Thus, digital objects in an archaeological digital library may fit into various categories of archaeological data such as figurine images, bone records, locus sheets, and site plans. They can be organized according to different hierarchical structures (e.g., animal bone records are organized based on sites where they are excavated, temporal sequence, and animal names). These hierarchical structures contain one or more hierarchically arranged categories. In addition, they can be refined based on taxonomies existing in botany and zoology, or through classification and description of artifacts by archaeologists.

Thus, an archaeological digital library may provide multi-dimensional browsing (see Fig. 2) to allow users to move along any of the navigational dimensions, or a combination thereof. Navigational dimensions correspond to hierarchical structures used to browse digital objects, as mentioned above.

For example, an archaeologist might browse through three dimensions: space, object, and time. She can start from any of these dimensions and move along by clicking. The scenario shown in Fig. 2 tells that she is interested in the artifact records from the tomb numbered 056 in area A of the Bab edh-Dhra site. The clickstream representing her navigation path is


ETANA-DL Managing complex information applications: An archaeology digital library

Home | Member Collections | First Time Visit | Login | Help

Search ETANA-DL for [Go] | Advanced Search | Browse

You are in: >> query=equus

Search within this context for [Go]

View Records for the Context Below

Browse by space:: :: SITE

Nimrin Umayri

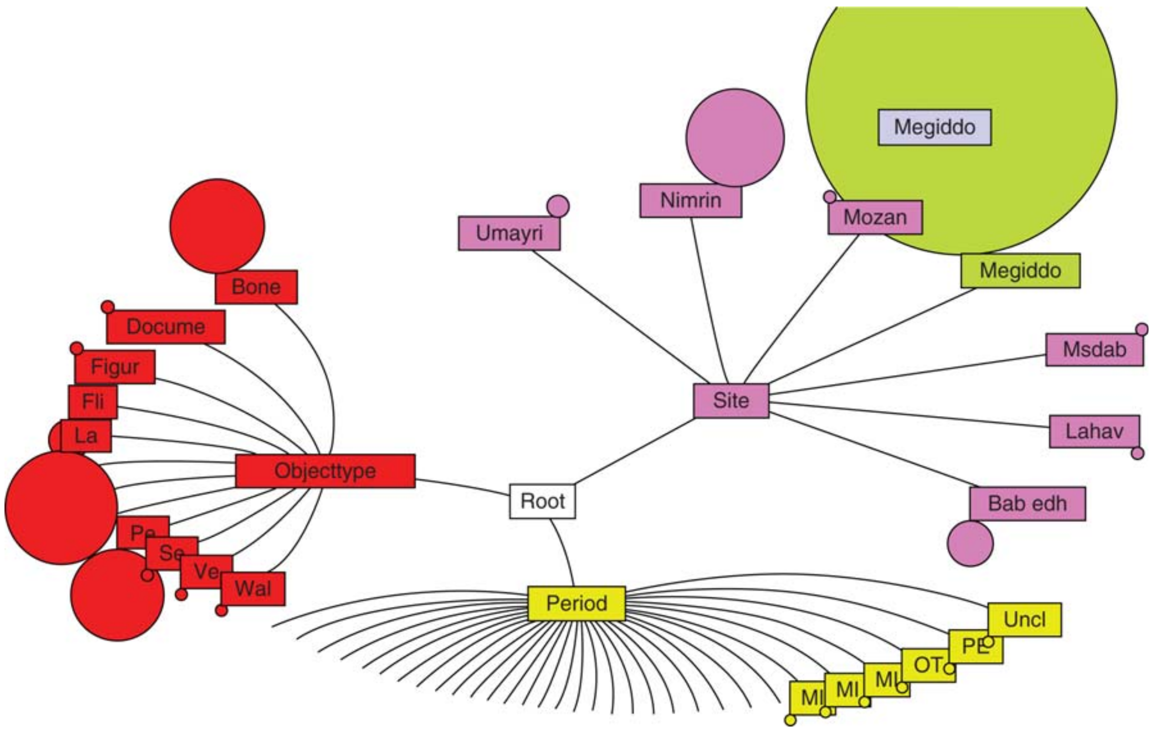
Browse by object:: :: OBJECTTYPE

Bone

Browse by time:: :: Period

IRON II	ISLAMIC	LATE HELLENISTIC-ROMAN	PERSIAN	MIDDLE BRONZE
IRON I	BYZANTINE	EARLY BRONZE	LATE IRON II	LATE IRON II / PERSIAN (IRON III)
MIDDLE BRONZE IIC	EARLY BRONZE III	EARLY IRON I	CLASSICAL-ISLAMIC	OTTOMAN - MODERN
MIDDLE BRONZE II	LATE BRONZE			

Browsing in Digital Libraries. Figure 5. Retrieved equus records are organized into 3 dimensions.



Browsing in Digital Libraries. Figure 6. Overview of an archaeological digital library.

denoted "Site=Bab edh-Dhra>>PARTITION=A>> SUBPARTITION=056." While this navigation path is within the first dimension, it also is associated with the other dimensions. The second dimension shows there is only one type of object, i.e., pottery, from that particular location. The third dimension presents the two time periods associated with those pottery records. Hence, the dynamic coverage and hierarchical structure of those dimensions yields a tool supporting learning and exploring. The user can navigate across dimensions. By clicking "EARLY BRONZE II" in the third dimension, she can view all of the interesting artifact records from the EARLY BRONZE II period.

Browsing may present a useful starting point for active exploration of an answer space. Subsequent browsing and searching, in any combination, also can be employed to refine or enhance users' initial, possibly under-specified, information needs.

Browsing context is associated with a user's navigation path. Browsing results within a certain browsing context typically are a set of records (web pages), e.g., there are 35 pottery records within the browsing context represented by the navigation path denoted "Site=Bab edh-Dhra>>PARTITION=A>>SUBPARTITION=056." For example, assume a user wants to find saucer records in the set of thirty five pottery records. She types "saucer" in the search box as shown in Fig. 3. She switches from browsing to searching, so searching then is a natural extension of browsing.

Browsing may be provided as a post-retrieval service to organize searching results hierarchically in digital libraries. For example, in Fig. 4 one can see that eighty eight equus records are retrieved through the basic searching service, in response to a query "equus". They are organized into three dimensions after the user clicks the button "View search results hierarchically" (see Fig. 5).

Browsing may be supported by visualization to provide a starting point for users. Graphic overviews of a digital library collection can display category labels hierarchically based on the facets. Categories can be visualized as a hyperbolic tree [5] as well as through a traditional node-link representation of a tree.

A hyperbolic tree in Fig. 6 shows hierarchical relationships among excavation data in an archaeological digital library based on spatial, temporal, and artifact-related taxonomies. A node name represents a category, and a bubble attached to a node represents a set of archaeological records. The size of a bubble attached to a node reflects the number of records belonging to that category. The hyperbolic tree supports "focus + context" navigation; it also provides an overview of records organized in the archaeological digital library. It shows that the records are from seven archaeological sites (the Megiddo site has the most) and are of twelve different types.

Future Directions

Browsing and searching are often provided by digital libraries as separate services. Developers commonly see these functions as having different underlying mechanisms, and they follow a functional, rather than a task-oriented, approach to interaction design. While exhibiting complementary advantages, neither paradigm alone is adequate for complex information needs. Searching is popular because of its ability to identify information quickly. On the other hand, browsing is useful when appropriate search keywords are unknown or unavailable to users. Browsing also is appropriate when a great deal of contextual information is obtained along the navigation path. Therefore, a synergy between searching and browsing is required to support users' information seeking goals. Browsing and searching can be converted and switched to each other under certain conditions [5]. This suggests some new possibilities for blurring the dividing line between browsing and searching. If these two services are not considered to have different underlying mechanisms, they will not be provided as separated functions in digital libraries, and may be better integrated.

Text mining and visualization techniques provide digital libraries additional powerful exploring services, with possible beneficial effects on browsing and searching. Digital library exploring services such as browsing, searching, clustering, and visualization can be generalized in the context of a formal digital library framework [5]. The theoretical approach may provide a systematic and functional method to design and implement exploring services for domain focused digital libraries.

Experimental Results

See Fig. 2 – Fig. 6 above, and the corresponding explanation.

Data Sets

See Fig. 2 – Fig. 6 above, and the ETANA Digital Library [5].

Cross-references

▶ Digital Libraries

Recommended Reading

1. Baeza-Yates R. and Ribeiro-Neto B. Modern Information Retrieval. Addison-Wesley, Reading, MA, 1999.
2. Fox E.A. and Urs S.R. Digital libraries, chap. 12. In Annual Review of Information Science and Technology, Vol. 36, B. Cronin (ed.); Medford, NJ, Information Today, Inc. 2002, pp. 503–589.
3. Fox E.A., Rous B., and Marchionini G. ACM's hypertext and hypermedia publishing projects. In Hypertext/Hypermedia Handbook, E. Berk, J. Devlin (eds.). McGraw-Hill, NY, 1991, pp. 465–467.
4. Goncalves M., Fox E.A., Watson L., and Kipp N. Streams, structures, spaces, scenarios, societies (5S): a formal model for digital libraries. ACM Trans. Inf. Syst., 22(2):270–312, 2004.
5. Shen R., Vemuri N., Fan W., Torres R., and Fox E.A. Exploring digital libraries: integrating browsing, searching, and visualization. In Proc. 6th ACM/IEEE-CS joint Conference on Digital Libraries, 2006, pp. 1–10.

B-Tree

▶ B+-Tree

B-Tree Concurrency Control

▶ B-Tree Locking

B-Tree Locking

GOETZ GRAEFE
Hewlett-Packard Laboratories, Palo Alto, CA, USA

Synonyms

B-tree concurrency control; Row-level locking; Key value locking; Key range locking; Lock coupling; Latching; Latch coupling; Crabbing

Definition

B-tree locking controls concurrent searches and updates in B-trees. It separates transactions in order to protect the B-tree contents and it separates threads in order to protect the B-tree data structure. Nowadays, the latter is usually called latching rather than locking.

Historical Background

Bayer and Schkolnick [1] presented multiple locking (latching) protocols for B^*-trees (all data records in the leaves, merely separator keys or "reference keys" in upper nodes) that combined high concurrency with deadlock avoidance. Their approach for insertion and deletion is based on deciding during a root-to-leaf traversal whether a node is "safe" from splitting (during an insertion) or merging (during a deletion), and on reserving appropriate locks (latches) for ancestors of unsafe nodes.

Lehman and Yao defined B^{link}-trees by relaxing the B-tree structure in favor of higher concurrency [8]. Srinivasan and Carey demonstrated their high performance using detailed simulations [13]. Jaluta et al. recently presented a detailed design for latching in B^{link}-trees, including a technique to avoid excessive link chains and thus poor search performance [7].

IBM's System R project explored multiple transaction management techniques, including transaction isolation levels and lock duration, predicate locking and key value locking, multi-granularity and hierarchical locking, etc. These techniques have been adapted and refined in many research and product efforts since then. Research into multi-level transactions [14] and into open nested transactions [3,12] enables crisp separation of locks and latches – the former protecting database contents against conflicts among transactions and the latter protecting in-memory data structures against conflicts among concurrent threads.

Mohan's ARIES/KVL design [10,11] explicitly separates locks and latches, i.e., logical database contents versus "structure maintenance" in a B-tree. A key value lock covers both a gap between two B-tree keys and the upper boundary key. In non-unique indexes, an intention lock on a key value permits operations on separate rows with the same value in the indexed column. In contrast, other designs include the row identifier in the unique lock identifier and thus do not need to distinguish between unique and non-unique indexes.

Lomet's design for key range locking [4] attempts to adapt hierarchical and multi-granularity locking to keys and half-open intervals but requires additional lock modes, e.g., a "range insert" mode, to achieve the desired concurrency. Graefe's design [9] applies traditional hierarchical locking to keys and gaps (open intervals) between keys, employs ghost (pseudo-deleted) records during insertion as well as during deletion, and permits more concurrency with fewer special cases. The same paper also outlines hierarchical locking exploiting B-trees' hierarchical structure or multi-field B-tree keys.

Foundations

The foundations of B-tree locking are the well-known transaction concepts, including multi-level transactions and open nested transactions, and pessimistic concurrency control, i.e., locking. Multiple locking concepts and techniques are employed, including two-phase locking, phantom protection, predicate locks, precision locks, key value locking, key range locking, multi-granularity locking, hierarchical locking, and intention locks.

Preliminaries

Most work on concurrency control and recovery in B-trees assumes what Bayer and Schkolnick call B^*-trees [1] and what Comer calls B^+-trees [2], i.e., all data records are in leaf nodes and keys in non-leaf or "interior" nodes act merely as separators enabling search and other operations but not carrying logical database contents. Following this tradition, this entry ignores the original design of B-trees with data records in interior nodes.

Also ignored are many other variations of B-trees here. This includes what Comer, following Knuth, calls B^*-trees, i.e., attempting to merge an overflowing node with a sibling rather than splitting it immediately. Among the ignored techniques are whether or not underflow is recognized and acted upon by load balancing and merging nodes, whether or not empty nodes are removed immediately or ever, whether or not leaf nodes form a singly or doubly linked list using physical pointers (page identifiers) or logical boundaries (fence keys equal to separators posted in the parent node during a split), whether suffix truncation is employed when posting a separator key, whether prefix truncation or any other compression is employed on each page, and the type of information associated with B-tree keys. Most of these issues have little or no bearing on locking in B-trees, with the exception of sibling pointers, as indicated below where appropriate.

Two Forms of B-Tree Locking

B-tree locking, or locking in B-tree indexes, means two things. First, it means concurrency control among concurrent database transactions querying or modifying database contents and its representation in B-tree indexes. Second, it means concurrency control among concurrent threads modifying the B-tree data structure in memory, including in particular images of disk-based B-tree nodes in the buffer pool.

These two aspects have not always been separated cleanly. Their difference becomes very apparent when a single database request is processed by multiple parallel threads. Specifically, two threads within the same transaction must "see" the same database contents, the same count of rows in a table, etc. This includes one thread "seeing" updates applied by the other thread. While one thread splits a B-tree node, however, the other thread should not observe intermediate and incomplete data structures. The difference also becomes apparent in the opposite case when a single operating system thread is multiplexed to serve all user transactions.

These two purposes are usually accomplished by two different mechanisms, locks and latches. Unfortunately, the literature on operating systems and programming environments usually uses the term locks for the mechanisms that in database systems are called latches, which can be confusing.

Locks separate transactions using read and write locks on pages, on B-tree keys, or even gaps (open intervals) between keys. The latter two methods are called key value locking and key range locking. Key range locking is a form of predicate locking that uses actual key values in the B-tree and the B-tree's sort order to define predicates. By default, locks participate in deadlock detection and are held until end-of-transaction. Locks also support sophisticated scheduling, e.g., using queues for pending lock requests and delaying new lock acquisitions for lock conversions, e.g., an existing shared lock to an exclusive lock. This level of sophistication makes lock acquisition and release fairly expensive, often thousands of CPU cycles, some of those due to cache faults in the lock manager's hash table.

Latches separate threads accessing B-tree pages, the buffer pool's management tables, and all other in-memory data structures shared among multiple threads. Since the lock manager's hash table is one of the data structures shared by many threads, latches are required while inspecting or modifying a database system's lock information. With respect to shared data structures, even threads of the same user transaction conflict if one thread requires a write latch. Latches are held only during a critical section, i.e., while a data structure is read or updated. Deadlocks are avoided by appropriate coding disciplines, e.g., requesting multiple latches in carefully designed sequences. Deadlock resolution requires a facility to roll back prior actions, whereas deadlock avoidance does not. Thus, deadlock avoidance is more appropriate for latches, which are designed for minimal overhead and maximal performance and scalability. Latch acquisition and release may require tens of instructions only, usually with no additional cache faults since a latch can be embedded in the data structure it protects.

Latch Coupling and B^{link}-Trees

Latches coordinate multiple concurrent threads accessing shared in-memory data structures, including images of on-disk storage structures while in the buffer pool. In the context of B-trees, latches solve several problems that are similar to each other but nonetheless lend themselves to different solutions.

First, a page image in the buffer pool must not be modified (written) by one thread while it is interpreted (read) by another thread. For this issue, database systems employ latches that differ from the simplest implementations of critical sections and mutual exclusion only by the distinction between read-only latches and read-write latches, i.e., shared or exclusive access. Latches are useful not only for pages in the buffer pool but also for the buffer pool's table of contents or the lock manager's hash table.

Second, while following a pointer (page identifier) from one page to another, e.g., from a parent node to a child node in a B-tree index, the pointer must not be invalidated by another thread, e.g., by deallocating a child page or balancing the load among neighboring pages. This issue requires retaining the latch on the parent node until the child node is latched. This technique is traditionally called "lock coupling" or better "latch coupling."

Third, "pointer chasing" applies not only to parent-child pointers but also to neighbor pointers, e.g., in a chain of leaf pages during a scan. This issue is similar to the previous, with two differences. On the positive side, asynchronous read-ahead may alleviate the frequency of buffer faults. On the negative side, deadlock avoidance among scans in opposite directions requires that latch acquisition code provides an immediate failure mode.

Fourth, during a B-tree insertion, a child node may overflow and require an insertion into its parent node, which may thereupon also overflow and require an insertion into the child's grandparent node. In the most extreme case, the B-tree's root node splits and a new root node is added. Going back from the leaf towards the B-tree root works well in single-threaded B-tree implementations, but in multi-threaded code it introduces the danger of deadlocks. This issue affects all updates, including insertion, deletion, and even record updates, the latter if length changes in variable-length records can lead to nodes splitting or merging. The most naïve approach, latching an entire B-tree with a single exclusive latch, is obviously not practical in multi-threaded servers.

One approach latches all nodes in exclusive mode during the root-to-leaf traversal. The obvious problem in this approach is the potential concurrency bottleneck, particularly at a B-tree's root. Another approach performs the root-to-leaf search using shared latches and attempts an upgrade to an exclusive latch when necessary. A third approach reserves nodes using "update" or "upgrade" latches. A refinement of these three approaches retains latches on nodes along its root-to-leaf search only until a lower, less-than-full node guarantees that split operations will not propagate up the tree beyond the lower node. Since most nodes are less than full, most insertion operations will latch no nodes in addition to the current one.

A fourth approach splits nodes proactively during a root-to-leaf traversal for an insertion. This method avoids both the bottleneck of the first approach and the failure point (upgrading a latch) of the second approach. Its disadvantage is that it wastes some space by splitting earlier than truly required. A fifth approach protects its initial root-to-leaf search with shared latches, aborts this search when a node requires splitting, restarts a new one, and upon reaching the node requiring a split, acquires an exclusive latch and performs the split.

An entirely different approach relaxes the data structure constraints of B-tress and divides a node split into two independent steps. Each node has a high fence key and a pointer to its right neighbor, thus the name B$^{\text{link}}$-trees. The right neighbor might not yet be referenced in the node's parent and a root-to-leaf search might need to proceed to the node's right neighbor. The first step of splitting a node creates the high fence key and a new right neighbor. The second, independent step posts the high fence key in the parent. The second step should happen as soon as possible yet it may be delayed beyond a system reboot or even a crash. The advantage of B$^{\text{link}}$-trees is that allocation of a new node and its initial introduction into the B-tree is a local step, affecting only one preexisting node. The disadvantages are that search may be a bit less efficient, a solution is needed to prevent long linked lists among neighbor nodes during periods of high insertion rates, and verification of a B-tree's structural consistency is more complex and perhaps less efficient.

Key Range Locking

Locks separate transactions reading and modifying database contents. For serializability, read locks are retained until end-of-transaction. Write locks are always retained until end-of-transaction in order to ensure the ability to roll back all changes if the transaction aborts. High concurrency requires a fine granularity of locking, e.g., locking individual keys in B-tree indexes. The terms key value locking and key range locking are often used interchangeably.

Key range locking is a special form of predicate locking. The predicates are defined by intervals in the sort order of the B-tree. Interval boundaries are the key values currently existing in the B-tree, which form half-open intervals including the gap between two neighboring keys and one of the end points.

In the simplest form of key range locking, a key and the gap to the neighbor are locked as a unit. An exclusive lock is required for any form of update of this unit, including modifying non-key fields of the record, deletion of the key, insertion of a new key into the gap, etc. Deletion of a key requires a lock on both the old key and its neighbor; the latter is required to ensure the ability to re-insert the key in case of transaction rollback.

High rates of insertion can create a hotspot at the "right edge" of a B-tree index on an attribute correlated with time. With next-key locking, one solution verifies the ability to acquire a lock on $+\infty$ but does not actually retain it. Such "instant locks" violate two-phase locking but work correctly if a single acquisition of the page latch protects both verification of the lock and creation of the new key on the page.

In those B-tree implementations in which a deletion does not actually erase the record and instead merely marks the record as invalid, "pseudo-deleted," or a "ghost" record, each ghost record's key participates in key range locking just like a valid record's key. Another technique to increase concurrency models a key, the appropriate neighboring open interval, and the combination of key and open interval as three separate items [9]. These items form a hierarchy amenable to multi-granularity locking. Moreover, since key, open interval, and their combination are all identified by the key value, additional lock modes can replace multiple invocations of the lock manager by a single one, thus eliminating the execution costs of this hierarchy.

Multi-granularity locking also applies keys and individual rows in a non-unique index, whether such rows are represented using multiple copies of the key, a list of row identifiers associated with a single copy of the key, or even a bitmap. Multi-granularity locking techniques exploiting a B-tree's tree structure or a B-tree's compound (multi-column) key have also been proposed. Finally, "increment" locks may be very beneficial for B-tree indexes on materialized summary views [5].

Both proposals need many details worked out, e.g., appropriate organization of the lock manager's hash table to ensure efficient search for conflicting locks and adaptation during structure changes in the B-tree (node splits, load balancing among neighboring nodes, etc.).

Key Applications

B-tree indexes have been called ubiquitous more than a quarter of a century ago [2], and they have become ever more ubiquitous since. Even for single-threaded applications, concurrent threads for maintenance and tuning require concurrency control in B-tree indexes, not to mention online utilities such as online backup. The applications of B-trees and B-tree locking are simply too numerous to enumerate them.

Future Directions

Perhaps the most urgently needed future direction is simplification – concurrency control and recovery functionality and code are too complex to design, implement, test, tune, explain, and maintain. Elimination of any special cases without a severe drop in performance or scalability would be welcome to all database development and test teams.

At the same time, B-trees are employed in new areas, e.g., Z-order UB-trees for spatial and temporal information, various indexes for unstructured data and XML documents, in-memory and on-disk indexes for data streams and as caches of reusable intermediate query results. It is unclear whether these application areas require new concepts or techniques in B-tree concurrency control.

Online operations – load and query, incremental online index creation, reorganization & optimization, consistency check, trickle load and zero latency in data warehousing including specialized B-tree structures.

Scalability – granularities of locking between page and index based on compound keys or on B-tree structure; shared scans and sort-based operations including "group by," merge join, and poor man's merge join (index nested loops join); delegate locking (e.g., locks on orders cover order details) including hierarchical delegate locking.

B-tree underpinnings for non-traditional database indexes, e.g., blobs, column stores, bitmap indexes, and master-detail clustering.

Confusion about transaction isolation levels in plans with multiple tables, indexes, materialized and indexed views, replicas, etc.

URL to Code

Gray and Reuter's book [6] shows various examples of sample code. In addition, the source of various open-source database systems is readily available.

Cross-references

- ▶ Concurrency Control and Recovery
- ▶ Database benchmarks – online transaction processing
- ▶ Locking Granularity and Lock Types
- ▶ pessimistic concurrency control
- ▶ phantoms
- ▶ precision locks
- ▶ predicate locks
- ▶ relational data warehousing
- ▶ System Recovery
- ▶ two-phase commit
- ▶ two-phase locking
- ▶ write-ahead logging

Recommended Reading

1. Bayer R. and Schkolnick M. Concurrency of operations on B-trees. Acta Inf., 9:1–21, 1977.
2. Comer D. The ubiquitous B-tree. ACM Comput. Surv., 11(2):121–137, 1979.
3. Eliot J. and Moss B. Open Nested Transactions: Semantics and Support. In Proc. Workshop on Memory Performance Issues 2006.

4. Graefe G. Hierarchical locking in B-tree indexes. BTW Conf., 2007, pp. 18–42.
5. Graefe G. and Zwilling M.J. Transaction support for indexed views. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004.
6. Gray J. and Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, San Francisco, CA, 1993.
7. Jaluta I., Sippu S., and Soisalon-Soininen E. Concurrency control and recovery for balanced B-link trees. VLDB J., 14(2):257–277, 2005.
8. Lehman P.L. and Yao S.B. Efficient locking for concurrent operations on B-trees. ACM Trans. Database Syst., 6(4):650–670, 1981.
9. Lomet D.B. Key range locking strategies for improved concurrency. In Proc. 19th Int. Conf. on Very Large Data Bases, 1993, pp. 655–664.
10. Mohan C. ARIES/KVL: A key-value locking method for concurrency control of multiaction transactions operating on B-tree indexes. In Proc. 16th Int. Conf. on Very Large Data Bases, 1990, pp. 392–405.
11. Mohan C., Haderle D.J., Lindsay B.G., Pirahesh H., and Schwarz P.M. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. Database Syst., 17(1):94–162, 1992.
12. Ni Y., Menon V., Adl-Tabatabai A-R., Hosking AL., Hudson RL., Moss JEB., Saha B., and Shpeisman T. Open nesting in software transactional memory. In Proc. 12th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming, 2007, pp. 68–78.
13. Srinivasan V. and Carey M.J. Performance of B-tree concurrency algorithms. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991, pp. 416–425.
14. Weikum G. Principles and realization strategies of multilevel transaction management. ACM Trans. Database Syst., 16(1):132–180, 1991.

Buffer Management

GIOVANNI MARIA SACCO
University of Torino, Torino, Italy

Definition

The database buffer is a main-memory area used to cache database pages. Database processes request pages from the buffer manager, whose responsibility is to minimize the number of secondary memory accesses by keeping needed pages in the buffer. Because typical database workloads are I/O-bound, the effectiveness of buffer management is critical for system performance.

Historical Background

Buffer management was initially introduced in the 1970s, following the results in virtual memory systems. One of the first systems to implement it was IBM System-R. The high cost of main-memory in the early days forced the use of very small buffers, and consequently moderate performance improvements.

Foundations

The buffer is a main-memory area subdivided into frames, and each frame can contain a page from a secondary storage database file. Database pages are requested from the buffer manager. If the requested page is in the buffer, it is immediately returned to the requesting process with no secondary memory access. Otherwise, a fault occurs and the page is read into a free frame. If no free frames are available, a "victim" page is selected and its frame is freed by clearing its content, after writing it to secondary storage if the page was modified. Usually any page can be selected as a victim, but some systems allow processes actively using a page to fix or pin it, in order to prevent the buffer manager from discarding it [5]. Asynchronous buffered write operations have an impact on the recovery subsystem and require specific protocols not discussed here.

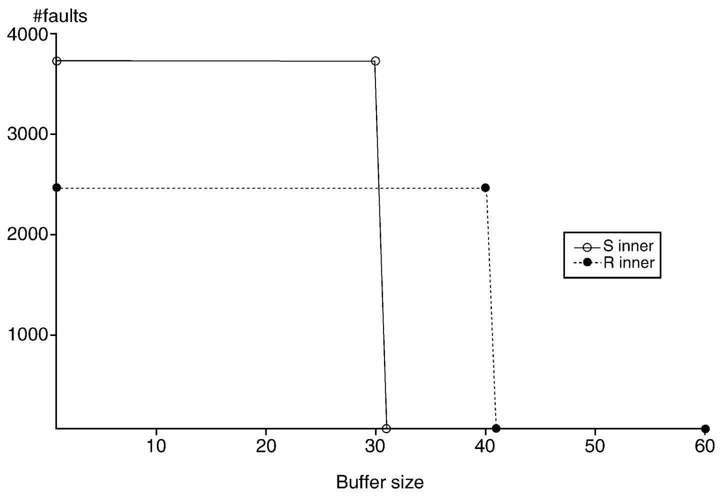
There are obvious similarities between buffer management and virtual memory (VM) systems [3]. In both cases, the caching system tries to keep needed pages in main-memory in order to minimize secondary memory accesses and hence speed up execution. As in VM systems, buffer management is characterized by two policies: the *admission policy*, which determines when pages are loaded into the buffer, and the *replacement policy*, which selects the page to be replaced when no empty frames are available. The admission policy normally used is demand paging (i.e., a missing page is read into the buffer when requested by a process), although prefetching (pages are read before processes request them) was studied (e.g., [1]). Since the interaction with the caching system is orders of magnitude less frequent in database systems than in VM systems, "intelligent" replacement policies such as LRU [3] (the Least Recently Used page is selected for replacement) can be implemented in software, with no performance degradation. Inverted page tables are used because their space requirement is proportional to the buffer size rather than to the entire database space as in normal page tables. Finally, database pages in the buffer can be shared among different processes, whereas the amount of sharing in VM systems is usually negligible.

The VM Approach

Besides minor architectural differences, the buffer manager can be used exactly as a virtual memory system for database pages, so that buffer management is transparent for the database system. In this VM approach, research is focused on effective replacement policies, which include GCLOCK and LRD (Least Reference Density) [5], LRU, and, more recently, efficient policies that account for page popularity, such as LRU-K [11].

LRU does not discriminate between frequently and infrequently referenced pages, and once a page is admitted in a buffer of B frames, it will stay there for at least B-1 references, even if not referenced again. Therefore, a potentially large portion of the buffer can be wasted by caching useless (i.e., infrequent) pages. *LRU-K* dynamically estimates the interreference distance for each page in the buffer by keeping the history of the last K references for each page in the buffer. A shorter interreference distance means a more frequently accessed page. Consequently, the page selected for replacement is the one with the largest interreference distance, i.e., the one with the maximum backward K-distance. The backward K-distance $b_t(p, K)$ at time t is defined as the distance backward to the Kth most recent reference the page p. When K = 1, only the last reference is considered, and LRU-1 is equivalent to LRU. As K grows, so do space and time overheads, because longer histories must be stored and kept ordered. At the same time, larger values of K improve the estimate of the interreference distance, but make the algorithm less responsive to dynamic changes in page popularity, so that, in practice, LRU-2 is normally used. The *2Q* (two queues) [7] algorithm provides an efficient, constant-time algorithm equivalent to LRU-2 replacement.

Although LRU-K improves LRU replacement through additional information about page access frequency, it is fundamentally different from the Least Frequently Used (LFU) replacement, because LFU does not adapt its estimate to evolving access patterns. The idea of exploiting both recency and frequency of access (also present in LRD replacement [5]) is extended by *LRFU* (least recently/frequently used) replacement [9], to model a parametric continuum of replacement policies ranging from LRU to LFU. The Combined Recency and Frequency (CRF) information is associated to each page in the buffer. CRF weights page references giving higher weights to more recent references, and combines frequency and recency information through a parameter λ $(0 \leq \lambda \leq 1)$. The page to be replaced is the one with minimum CRF. When $\lambda = 0$, LRFU becomes LFU; when $\lambda = 1$, it becomes LRU. The optimal λ depends on the actual workload, but a self-tuning strategy can be used. An efficient implementation of LRFU exists, and experiments show this replacement strategy to outperform LRU-K with small buffer sizes.



Buffer Management. Figure 1. Number of faults as a function of the available buffer space, under LRU replacement.

A known problem in the VM approach is that processes with fast sequential scans over large relations tend to fill the buffer with useless pages accessed in the scan and flush the active pages of other processes out of memory, thereby significantly increasing the overall fault rate [12]. In a sequential scan, the current page is (i) rapidly accessed several times in order to read all the tuples on it, (ii) once the last tuple in the page is read, the page will never be reaccessed again. Reaccess in (i) makes VM strategies to incorrectly estimate that the current page is likely to be reaccessed in the future, and therefore to keep it in the buffer at the expense of other pages. In order to avoid this problem, some strategies use parametric correctives. LRU-K, for instance, uses the Correlated Reference Period. During this period, a page freshly admitted to the buffer cannot be replaced because additional references are expected. At the same time, subsequent references during this period are not tracked because they are expected to be correlated and do not give a reliable indication of future behavior. The length of such period is at the same time critical and difficult to estimate.

The Predictive Approach

A completely different, predictive approach was proposed with the *hot set model* [12,13]. In database systems with non-procedural data manipulation languages (such as relational or object-relational database systems), it is not the programmer but the system query optimizer that determines the access plan for a query. Such access plan is based on a small number of access primitives, such as sequential scans, nested loop joins, etc., whose reference string is known or can be estimated before actual execution. As an example, consider the execution of a tuplewise nested loop join between relation R (3 tuples/page, 40 pages) and relation S (2 tuples/page, 30 pages). Assuming S inner and indicating by Xi the i-th page of relation X, and by $\{a\}^N$ N repetitions of string a, the reference string is exactly $\{R1, \{S1\}^2,\ldots,\{S30\}^2\}^3$, $\{R2, \{S1\}^2,\ldots,\{S30\}^2\}^3,\ldots,\{R40, \{S1\}^2,\ldots,\{S30\}^2\}^3$. Figure 1 plots the number of faults as a function of the available buffer space, under LRU replacement.

Figure 1 shows that, differently from what happens in VM systems where fault curves tend to be smooth, the fault curve here is discontinuous. If the buffer is not large enough to contain all the pages in the inner relation plus one frame for the current page in the outer one, no reusal occurs and the fault rate is the same as for 1 frame. If sufficient space exists, all the needed pages are kept in the buffer, and the cost becomes linear in the number of pages of the two relations. In this case, the entire behavior of the plan in which S is inner is completely characterized by two *hot points* and their corresponding fault rate:

- hp1 = 1, faults(hp1) = |R|(1 + pages(S))
- hp2 = 1 + pages(S), faults(hp2) = pages (R) + pages(S)

The number of frames needed by a plan is called its hot set size, and usually is the largest hot point no larger than the total buffer size. Note that the length of the loop on the inner relation is exactly known at the database level, but it is very hard to discover in the VM approach.

The basic idea of the hot set model is that, given an access plan and a replacement policy, the number of faults as a function of the available buffer space can be predicted. Three main types of reusal, modeled upon primitive strategies, are considered:

1. Simple reusal. It models the sequential scan of a relation: the current page is accessed several times in order to read all the tuples on it, but once the last tuple in the page is read, the page will never be reaccessed again. There is only one hot point at 1 frame.
2. Loop reusal. Used in the example in Fig. 1. There are as many hot points as there are loops.
3. Index reusal. It includes both clustered and unclustered index access. Reusal can occur when indices are used in a join, or when an index is accessed by several processes. In unclustered index access, data pages are accessed randomly, and the hot set is estimated through Yao's function.

It must be stressed that LRU replacement is not required, and that other replacement strategies can be adopted [2,13]. As a matter of fact, both simple and loop reusal do not benefit from LRU replacement at all. The *query locality set model* (QLSM) [2] extends the hot set model by determining the hot set size on a file-instance basis rather than on a primitive operation basis, and by adopting replacement policies appropriate for each type of reusal. As an example, a nested loop join between R and S (S inner) is characterized by two different access patterns: a simple reusal (sequential scan) on R, requiring 1 frame, and a loop reusal on S, requiring 1 to pages(S) frames. MRU (most recently used) replacement can be used for loop reusal, since it is optimal for this type of reference string.

Since the reference string of the application is known, optimal replacement policies can be used. This opportunity is exploited in *OLRU* [14], which derives an optimal replacement policy for clustered index reusal, e.g., repeated access to clustered B+ trees. In this case, the Independent Reference Model (IRM) [3] is used. IRM was originally proposed as a theoretical evaluation model for VM systems, and assumes, in extreme synthesis, that the probability of reaccess for each page is known and stationary. Under this assumption, it can be proven that the optimal buffering strategy for a buffer of B frames consists of ordering the pages by decreasing reaccess probabilities, fixing the first B-1 high-probability pages in the buffer and using the remaining frame to access all the other low-probability pages. Since the optimal strategy only requires a ranking among pages, it can be straightforwardly applied to a B+ tree of order m and height h by assuming a uniform distribution of access to the leaf level. In this case, the probability of reaccessing a page at level j $(0 \leq j < h)$ is $1/m^j$. The optimal policy is then an allocation by levels, i.e., fixing the first levels in the tree in the buffer. LRU replacement allocates the buffer by traversal stacks and is therefore suboptimal, unless severe deviations from uniformity in data page access occur, in which case OLRU and LRU are similar in performance. LRU-K, which assumes IRM as well, can be seen as a run-time approximation of OLRU.

A complete characterization of query buffer requirements and corresponding access costs is required for two major reasons. First, query optimizers need to have a precise *cost estimate* as a function of the available buffer size, in order to discriminate execution plans. This is especially important because (i) cost curves can intersect as shown in Fig. 1: R inner is cheaper in the interval [1,30], whereas S inner is better or no worse for buffer sizes of at least 31 frames; (ii) many query evaluation strategies (e.g., nested loops) do not explicitly account for memory, and they must be compared with other strategies (such as fragmentation/recursive hash partitioning join) in which available buffer space is fully accounted for and directly managed.

Second, *thrashing* phenomena [3] can occur in database buffers as in virtual memory systems, and become potentially more frequent as the number of concurrent active users increases. Thrashing occurs when the available memory is insufficient to keep all the pages each active process needs. Consequently, processes steal pages from each other and, if the available memory is severely overcommitted, the system collapses because all activity is devoted to swapping pages to and from main-memory. In VM systems, thrashing can be avoided by (i) monitoring the pagination device for excessive utilization or (ii) using the working set model [3] in order to estimate the memory requirements of active processes. In buffer management, there is no pagination device and accesses may be directed to a high number of secondary devices. In addition, the working set model, which is an expensive run-time estimator, cannot be efficiently used in database systems [13].

Thrashing avoidance, and the more general problem of *scheduling* queries for execution in order to optimally use buffer resources, is considerably simplified in the predictive approach because buffer requirements for each query are known before execution. The simplest policy [13] schedules queries for execution in such a way that the sum of their hot set sizes does not exceed the total buffer space. Each query can then be run in isolation in its own buffer partition, and this, by definition, avoids thrashing. However, page sharing, which is relatively frequent in database systems, is not accounted for. If two different processes request the same page, two faults occur. If sharing is considered these additional faults can be avoided, and the actual required buffer size can decrease because a page shared by several processes requires only one frame. For these reasons, hot set scheduling [12,13] maintains an additional global LRU chain to manage free pages, and, on a local page fault, scans the entire buffer for potential shared pages. In addition, a measure of buffer consumption is used in order to avoid unnecessarily inflating hot set sizes, which would result in serializing small queries behind queries with high buffer requirements. Variations of this scheduling algorithm include DBMIN [2] (where different local replacement policies can be used), scheduling with marginal gains [10], and scheduling with prefetching [1].

The VM Versus the Predictive Approach

When compared to the predictive approach, the VM approach has the advantage of placing all concerns on buffering into a single system component that can be seen as a black box from the rest of the system. In addition, the VM approach is more generally applicable since it does not require non-procedural interactions, and inherently implements page sharing.

However, for non-procedural systems, the predictive approach solves a number of important problems:

- Uniformity of estimated costs for query optimization. Query plans can be compared, regardless of whether methods directly manage memory or not.
- Thrashing avoidance, and efficient, low cost query scheduling. The predictive approach characterizes requirements before execution and does not require expensive run-time estimators.
- Resource planning for self-tuning databases. Predictive characterization of buffer requirements can be used to determine the optimal buffer size for actual workloads.
- Better performance. Since reference strings are known, optimal replacement policies can be used and buffer requirements carefully tuned.

There is a duality between detecting sequential scans and detecting page sharing. The VM approach has no problems in implementing page sharing. VM strategies work reasonably well for index access, but, despite correctives, they tend to break on simple and loop reusal. Conversely, sequential scan detection and inner loop size detection is trivial in the predictive approach, but, since prediction is based on a query run in isolation, run-time corrections are required for page sharing. This duality suggests that a combination of the two approaches might be beneficial.

Key Applications

The trend towards cheaper and larger main-memories does not make buffer management less important. In fact, the increase in available main-memory has been so far matched by a larger increase in secondary storage capacity and in the amount of data to be managed, in the complexity of queries, and in the number of users. Consequently, buffer management continues to be a fundamental topic for database systems, and indeed its results carry over to different areas, such as web caching.

Current research areas in database systems include automatic buffer sizing in self-tuning databases [15], extendibility of buffer management strategies [6], specific buffering strategies for object databases [8], XML databases and P2P data architectures, multidimensional databases, real-time databases where process priorities must be considered in scheduling, and, of course, new index structures or evaluation strategies for which buffer analysis is required [4]. In addition, challenging applications managing very large amounts of data such as sensor data, search engines, large digital libraries, etc. require high-performance buffer management.

Cross-references

► Evaluation of Relational Operators
► Indexing
► Recovery Techniques

Recommended Reading

1. Cai F.F., Hull M.E.C., and Bell D.A. Buffer management for high performance database systems. In Proc. High-Performance Computing on the Information Superhighway (HPC-Asia'97). Seoul, Korea, 1997, pp. 633–638.
2. Chou H.-T. and DeWitt D.J. An evaluation of buffer management strategies for relational database systems. In Proc. 11th Int. Conf. on Very Large Data Bases, 1985, pp. 174–188.
3. Coffman Jr. and Denning P.J. Operating Systems Theory. Prentice-Hall, Englewood Cliffs, NJ, 1973.
4. Corral A., Vassilakopoulos M., and Manolopoulos Y. The impact of buffering on closest pairs queries using R-trees. In Proc. Fifth East European Conference on Advances in Databases and Information Systems, 2001, pp. 41–54.
5. Effelsberg W. and Haerder T. Principles of database buffer management. ACM Trans. Database Syst., 9(4):560–595, 1984.
6. Goh L., Shu Y., Huang Z., and Ooi C. Dynamic buffer management with extensible replacement policies. VLDB J., 15(2): 99–120, 2006.
7. Johnson T. and Shasha D. 2Q: a low overhead high performance buffer management replacement algorithm. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 439–450.
8. Kemper A. and Kossmann D. Dual-buffering strategies in object bases. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 427–438.
9. Lee D., Choi J., Kim J.-H., Noh S.H., Min S.L., Cho Y., and Kim C.S. On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies. In Proc. Int. Conf. on Measurement and Modeling of Computer Systems, 1999, pp. 134–143.
10. Ng R., Faloutsos C., and Sellis T. Flexible buffer allocation based on marginal gains. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991, pp. 387–396.
11. O'Neil E.J., O'Neil P.E., and Weikum G. The LRU-K page replacement algorithm for database disk buffering. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 297–306.
12. Sacco G.M. and Schkolnick M. A mechanism for managing the buffer pool in a relational database system using the hot set model. In Proc. 8th Int. Conf. on Very Data Bases, 1982, pp. 257–262.
13. Sacco G.M. and Schkolnick M. Buffer management in relational database systems. ACM Trans. Database Syst., 11(4):473–498, 1986.

14. Sacco G.M. Index access with a finite buffer. In Proc. 13th Int. Conf. on Very Large Data Bases, 1987, pp. 301–309.
15. Storm A.J., Garcia-Arellano C., Lightstone S.S., Diao Y., and Surendra M. Adaptive self-tuning memory in DB2. In Proc. 12th Int. Conf. on Very Large Data Bases, 2006, pp. 1081–1092.

Buffer Manager

Goetz Graefe
Hewlett-Packard Laboratories, Palo Alto, CA, USA

Synonyms

Cache manager

Definition

If a buffer pool is employed in a database management system, the associated software must provide appropriate services for efficient query processing, correct transaction execution, and effective sharing and reuse of database pages. It must provide interfaces for page access including pinning and latching pages, and it must invoke primitives for disk I/O and synchronization.

Historical Background

Database buffer pool management was studied heavily in the 1970s and 1980s as the new relational database management system posed new challenges, in particular non-procedural queries with range scans. Reliance of virtual memory and file system buffer pool was investigated but rejected due to performance issues (read-ahead, prefetch) and correctness issues (transaction management, write-behind, write-through).

Buffer pool management is currently not a very active research area. It may be revived in order to serve deep storage hierarchies, e.g., slow disk, fast disk, flash memory, main memory, and CPU cache. It may also be revived in the context of very large memories, e.g., query optimization that considers residence in the buffer pool or physical database design that includes temporary or partial indexes that exist only in the buffer pool.

Foundations

This section describes a database system's buffer pool and its management by focusing on management of individual pages in the buffer pool, replacement policies, asynchronous I/O, and the requirements imposed by concurrency control and recovery.

Buffer Pool Interfaces

The principal methods provided by the buffer pool manager request and release a page. A page request fixes or pins a page, i.e., it protects it from replacement as well as movement within the buffer pool, thus permitting access to the page by in-memory pointers. Variants of requesting and releasing a page apply to disk pages immediately after allocation (no need to read the page contents from disk) and immediately after deallocation (no need to save the page contents).

In addition, a buffer pool may provide methods for concurrency control among threads in order to protect page contents, also known as latching. Pragmatically, the methods to pin and to release a page include parameters that control latching.

Concurrency control among transactions is usually not provided by the buffer pool. In aid of logging and recovery, a buffer pool must support a method to force a page to disk and may support a method to retain one page until another page has been written.

For performance, a buffer pool may support methods to hint asynchronous prefetch, read-ahead, and write-behind. If the buffer pool serves a memory pool for other software layers, it must support methods for memory allocation and deallocation.

The principal methods upon which a buffer pool manager relies are disk reading and writing, including asynchronous operations, and scheduling primitives to implement latching. If the buffer pool size is dynamic, memory allocation and deallocation are also required. If the buffer pool supports multiple page sizes, it requires fast methods for moving (copying) page contents from one memory address to another.

Replacement Policies

The goal of the replacement policy (or retention policy) is to speed up future page accesses. Prediction of future accesses can be based on past accesses (how recent, how frequent, whether read or write) or on hints from higher software layers within the database management system. Standard policies include LRU (least recently used, implemented using a doubly-linked list), LRU-K (least recent K uses), LFU (least frequently used), second chance (usually implemented following a clock metaphor), generalized clock (using counters instead of a single bit per page frame), and combinations of those. Many combinations have been proposed, including the hot set model, the query locality set model, adaptive replacement cache, etc.

They differ in their heuristics to separate pages used only once, e.g., in a large sequential scan, from pages likely to be reused, e.g., pages containing the database catalog or root pages of B-tree indexes. Alternative designs let higher software layers hint the likelihood of reuse, e.g., love/hate hints or keep/toss hints.

Dirty pages (containing recent updates) may be retained longer than clean ones because their replacement cost and delay are twice as high (write plus read instead of merely a read operation) and because correct preparation for recovery may impose restrictions on the order in which pages are written. For example, write-ahead logging requires writing the relevant log page to stable storage before overwriting old database contents. On the other hand, non-logged operations (e.g., index creation) require flushing dirty pages as part of transaction commit.

Asynchronous I/O

Rather than merely responding to requests from higher software layers, a buffer pool may employ or enable asynchronous I/O, in three forms:

Write-behind cleans the buffer pool of dirty pages in order to complete update transactions as fast as possible yet enable quick page replacement without needing a write operation prior to a read operation. However, most write-behind is driven by checkpoints rather than page faults, based on typical checkpoint intervals (very few minutes) and retention intervals as calculated or optimized using the five-minute rule (many minutes or even hours). A write-behind operation may leverage a disk seek forced by another read or write operation. Alternatively, a write-behind operation may move the data to achieve this effect, e.g., in log-structured file systems and write-optimized database indexes.

Read-ahead speeds up large scans, e.g., a range scan in a B-tree or a complete scan of a heap structure. The appropriate amount of read-ahead is the product of bandwidth and latency, i.e., the smallest of I/O bandwidth and processing bandwidth multiplied with the delay from initiation to completion of a read operation. Read-ahead may be triggered by observation of the access pattern or by a hint from a higher software layer, e.g., a table scan in a query execution plan.

Prefetch accelerates fetch operations by loading into the buffer pool precisely those pages that contain needed data records. Prefetch in heap structures is quite straightforward. In B-tree indexes, prefetch may apply to all tree levels or merely to the leaf level, combined with synchronous read operations or large read-ahead for interior B-tree nodes. Prefetch speeds up not only to ordinary forward processing but also to transaction rollback as well as system recovery.

Concurrency Control and Recovery

The buffer pool may participate in concurrency control, e.g., if multi-version concurrency control requires multiple versions of individual pages. The mechanisms for pinning and latching are essential for coordination of multiple threads accessing the same in-memory data structures, including in-memory images of on-disk pages.

The buffer pool always participates in the preparation for recovery including transaction rollback, media recovery, and system recovery. For example, by retaining all modified pages in the buffer pool until transaction commit, one can avoid logging *undo* information in the persistent log – this is called a "no steal" policy. By forcing merely log pages to stable storage, one can avoid writing all modified database pages back to disk – this is called a "no force" policy. Most database management systems use "steal – no force" by default. The log volume of index creation and similar operations is often minimized using a "force" policy.

Logging volume can be reduced by ensuring specific write sequences. For example, when a B-tree node is split and some records are moved to a new node, one can avoid logging the moved records by writing the new page before writing the modified old page.

The buffer pool also participates in checkpoint processing. In addition to recording active transactions, a checkpoint must write all dirty database pages to the log or to the database. Proactive asynchronous write-behind may lessen the number of write operations during the checkpoint.

Cooperative Buffer Pool Management

Multiple buffer pools may cooperate. Prototypical examples include client-server operation (in which the buffer pools form a hierarchy) and shared-disk database systems (in which the buffer pools are peers). Those environments require optimizations for both data traffic among buffer pools and control messages, in particular for concurrency control and lock management.

Key Applications

A buffer pool and its management software are required in any database management system that employ multiple levels in a memory hierarchy, process and store data at different levels, and do not rely other means for moving data between those levels. The main example is main memory and disks – the buffer pool manager manages which data pages are immediately available for access, e.g., from the query execution engine. A CPU cache is a level in the memory hierarchy above the main memory, but data movement between main memory and CPU cache are automatic. A database management system could rely on a file system and its buffer pool manager but usually does not due to performance issues (e.g., read-ahead, prefetch) and due to correctness issues (write-behind, write-through). A database management system could rely on virtual memory provided by the operating system but typically does not for the same reasons.

Future Directions

While basic buffer pool management in traditional database management systems is well understood, there are many developments that build upon it. For example, integration of database cache, mid-tier cache, and web cache may become imperative in order to maximize efficiency and thus minimize costs for hardware, management, power, and cooling.

Buffer pools and buffer management will become more pervasive with the increased virtualization of storage and processing. At the same time, it will become more complex due to missing information on the true cost and location of data. It will also become more pervasive and complex due to increasing use of peer-to-peer storage, communication, and processing.

Any buffer pool becomes more effective with data compression and co-location. Compression reduces the space required locally (in the buffer pool) and remotely. Co-location techniques such as master-detail clustering enable access to multiple related records or pieces of information within a single frame in the buffer pool and with a single access to the remote location.

In a very large buffer pool, one might create temporary on-disk structures that never even exist on disk. For example, a temporary index on a permanent table may be created yet retained in the buffer pool. During contention in the buffer pool, the index is dropped. Ideally, such an index is partitioned, created and dropped incrementally, and left behind as a free side effect of query execution.

The data structures that manage a buffer pool, both its contents descriptors and its replacement policy, can be complex yet require very high concurrency, in particular in forthcoming many-core processors. Hardware-assisted transactional memory may simplify the software implementation effort as well as increase concurrency and performance.

In deep memory hierarchies, e.g., a three-level hierarchy of traditional memory, flash memory, and disk, contents descriptors and data structures in aid of the replacement policy may be separate. For example, the contents descriptors may need to be persistent if the flash memory is part of the persistent database, but all data structures for the replacement policy (between flash memory and disk) might be in the traditional memory.

Cross-references

- ▶ Buffer pool
- ▶ B-tree locking
- ▶ Concurrency control and recovery
- ▶ Flash memory
- ▶ Lock manager
- ▶ Storage hierarchy
- ▶ Storage layer
- ▶ Storage manager

Recommended Reading

1. Bansal S. and Modha D.S. CAR: Clock with adaptive replacement. In Proc. 3rd USENIX Conf. on File and Storage Technologies, 2004, pp. 187–200.
2. Chou H-T. and DeWitt D.J. An evaluation of buffer management strategies for relational database systems. Algorithmica, 1(3):311–336, 1986.
3. Effelsberg W. and Härder T. Principles of database buffer management. ACM Trans. Database Syst., 9(4):560–595, 1984.
4. Gray J. and Putzolu G.R. The 5 minute rule for trading memory for disk accesses and the 10 byte rule for trading memory for CPU Time. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1987, pp. 395–398.
5. Ramamurthy R. and DeWitt D.J. Buffer-pool Aware Query Optimization. In Proc. 2nd Biennial Conf. on Innovative Data Systems Research, 2005, pp. 250–261.

6. Stonebraker M. Operating System Support for Database Management. Commun. ACM, 24(7):412–418, 1981.

Buffer Pool

Goetz Graefe
Hewlett-Packard Laboratories, Palo Alto, CA, USA

Synonyms

I/O cache, Page cache

Definition

Cost constraints (dollars per gigabyte) prohibit in-memory databases in most cases, but processors can access and manipulate data only while it is in memory. The in-memory buffer pool holds database pages currently in use and retains those deemed likely to be used again soon.

The buffer pool and the buffer management component within the storage layer of a database management system provide fast access and fast recall of on-disk pages using in-memory images of those pages. In addition to the size of individual pages and of the entire buffer pool, key issues are (i) page replacement in response to buffer faults, and (ii) page retention and update in aid of database recovery.

A database buffer pool differs from virtual memory as it contributes to correctness and efficiency of query and update processing, e.g., by pinning pages while in use and by ensuring a write sequence that guarantees the ability to recover. In some systems, the buffer pool permits "stealing" memory for query processing operations such as sorting and hash join, for utilities such as reorganization and consistency checks, etc.

Historical Background

Gray and Putzolo's paper introducing the five-minute rule makes a strong case for using multiple levels of the memory hierarchy for data collections that include both hot and cold data, i.e., data that are accessed with different frequencies. Hot data are kept at a high level of the memory hierarchy, whereas cold data are fetched as needed and buffered temporarily.

A buffer pool is also needed if the provided granularity of access is too coarse. Disks permit random page accesses, whereas query execution, predicate evaluation, etc. require accesses to individual records, fields, and bytes.

Early research demonstrated that virtual memory is not appropriate for use in database management systems, as observed by Härder, Stonebraker, Traiger, and their research teams for early relational database management systems. Reasons include both correctness, specifically with respect to recovery, and performance, specifically asynchronous read-ahead and write-behind.

In order to avoid double page faults, a buffer pool should not be subject to page replacement by virtual memory provided by the operating system. One means to achieve this is to vary the size of the buffer pool in response to paging rates of the virtual memory. The five-minute rule gives guidance for the appropriate sizing of memory and buffer pool.

Techniques to map on-disk database contents into virtual memory advanced in the context of object-oriented databases and persistent programming languages but did not result in wide adoption.

Flash memory still requires an in-memory buffer pool for access to bytes within pages, yet it can also serve as a buffer pool for pages that require faster access than rotating disks can provide. In fact, buffer pool management techniques such as replacement policies apply to all levels in a multi-level memory hierarchy, e.g., CPU caches, RAM, flash memory, disk caches, rotating disk, tape media, etc. In an extreme case, a fast disk may serve a buffer pool for a slower disk; alternatively, both disks may serve as permanent storage and buffer replacement policies may be adapted for page placement on those disks, possibly including frequent page migration.

Foundations

This section describes a database system's buffer pool, replacement policies, and the requirements imposed by concurrency control and recovery.

Buffer Frames

A buffer pool contains many frames, each capable of holding the image of an on-disk page. Pages can be fixed-length or variable-length, i.e., multiple base pages, typically a power of 2. Space management is complex for variable-length pages; one technique employed commercially relies on multiple buffer pools with a single page size in each, i.e., a fixed-length page frame in each buffer pool.

In addition to being idle or unused, buffer frames can be pinned to protect the page from replacement, latched (locked) to protect against concurrent readers or writers, or in transit from or to disk. Pinning and latching are important for performance; they enable the database's query execution software to inspect or update records directly in the buffer pool.

A buffer pool may hold multiple versions of the same on-disk page in aid of transaction isolation, compression, asynchronous write-behind, or protection from partial writes. Write-behind with a single copy inhibits further updates until the write operation completes. The copy step may compact free space between variable-length records.

A small descriptor data structure is used for each frame in the buffer pool. It identifies the on-disk page and its status with respect to pinning, latching, versioning, etc. The descriptor also participates in a look-up scheme, typically a hash table, and in data structures used for page replacement.

Buffer Pool Data Structures

In addition to page frames and their descriptors, a buffer pool needs data structures to locate buffered pages and to manage page replacement including page frames currently unused. Locating a buffered page, i.e., mapping from a disk address (page number) to an in-memory address, usually is implemented with a hash table.

The data structures needed for page replacement depend on the page replacement policy – one method that is simple but not ideal is to employ a doubly linked list of page descriptors with pages ordered the time since they were last used. When a page is pinned, it is removed from the list. When it is unpinned, it is inserted at the head of the list. When a page is needed for replacement, the page at the tail of the list is chosen.

Replacement Policies

The goal of the replacement policy (or retention policy) is to speed up future page accesses. Prediction of future accesses can be based on past accesses (how recent, how frequent, whether read or write) or on hints from higher software layers within the database management system. Standard policies include LRU (least recently used, implemented using a doubly-linked list), LRU-K (least recent K uses), LFU (least frequently used), second chance (usually implemented following a clock metaphor), generalized clock (using counters instead of a single bit per page frame), and combinations of those. Many combinations have been proposed, including the hot set model, the query locality set model, adaptive replacement cache, etc.

They differ in their heuristics to separate pages used only once, e.g., in a large sequential scan, from pages likely to be reused, e.g., pages containing the database catalog or root pages of B-tree indexes. Alternative designs let higher software layers hint the likelihood of reuse, e.g., love/hate hints or keep/toss hints.

Dirty pages (containing recent updates) may be retained longer than clean ones because their replacement cost and delay are twice as high (write plus read instead of merely a read operation) and because correct preparation for recovery may impose restrictions on the order in which pages are written. For example, write-ahead logging requires writing the relevant log page to stable storage before overwriting old database contents. On the other hand, non-logged operations (e.g., index creation) require flushing dirty pages as part of transaction commit.

Key Applications

A buffer pool and its management software are required in any database management system that employ multiple levels in a memory hierarchy, process and store data at different levels, and do not rely other means for moving data between those levels. The main example is main memory and disks – the buffer pool manager manages which data pages are immediately available for access, e.g., from the query execution engine. A CPU cache is a level in the memory hierarchy above the main memory, but data movement between main memory and CPU cache are automatic. A database management system could rely on a file system and its buffer pool manager but usually does not due to performance issues (e.g., read-ahead, prefetch) and due to correctness issues (write-behind, write-through). A database management system could rely on virtual memory provided by the operating system but typically does not for the same reasons.

Future Directions

In deep memory hierarchies, e.g., a three-level hierarchy of traditional memory, flash memory, and disk, contents descriptors and data structures in aid of the replacement policy may be separate. For example, the contents descriptors may need to be persistent if the flash memory is part of the persistent database, but all data structures for the replacement policy (between flash memory and disk) might be in the traditional memory.

Cross-references

► B-tree locking
► Concurrency control and recovery
► Lock manager

▶ Flash memory
▶ Storage hierarchy
▶ Storage layer
▶ Storage manager

Recommended Reading

1. Bansal S. and Modha D.S. CAR: Clock with adaptive replacement. In Proc. 3rd USENIX Conf. on File and Storage Technologies, 2004, pp. 187–200.
2. Chou H-T. and DeWitt D.J. An evaluation of buffer management strategies for relational database systems. Algorithmica, 1(3):311–336, 1986.
3. Effelsberg W. and Härder T. Principles of database buffer management. ACM Trans. Database Syst., 9(4):560–595, 1984.
4. Gray J. and Putzolu G.R. The 5 minute rule for trading memory for disk accesses and the 10 byte rule for trading memory for CPU time. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1987, pp. 395–398.
5. Ramamurthy R. and DeWitt D.J. Buffer-pool aware query optimization. In Proc. 2nd Biennial Conf. on Innovative Data Systems Research, 2005, pp. 250–261.
6. Stonebraker M. Operating system support for database management. Commun. ACM, 24(7):412–418, 1981.

Business Intelligence

Stefano Rizzi
University of Bologna, Bologna, Italy

Definition

Business intelligence is a business management term that indicates the capability of adding more intelligence to the way business is done by companies. More precisely, it refers to a set of tools and techniques that enable a company to transform its business data into timely and accurate information for the decisional process, to be made available to the right persons in the most suitable form. Business intelligence systems are used by decision makers to get a comprehensive knowledge of the business and of the factors that affect it, as well as to define and support their business strategies. The goal is to enable data-based decisions aimed at gaining competitive advantage, improving operative performance, responding more quickly to changes, increasing profitability and, in general, creating added value for the company.

Key Points

Though business intelligence has its roots in reporting systems, it was born as a term within the industrial world in the early 1990's, to indicate a set of technologies aimed at satisfying the managers' request for efficiently and effectively analyzing the enterprise data in order to better understand the situation of their business and improving the decision process. In the mid-1990's business intelligence became an object of interest for the academic world, and ten years of research managed to transform a bundle of naive techniques into a well-founded approach to information extraction and processing that led to defining the modern architectures of data warehousing systems. Currently, business intelligence includes not only the tools to gather, provide access to, and analyze data and information about company operations, but also a wide array of technologies used to support a closed decisional loop (known as *Business Performance Management*) where the company performance is measured by a set of indicators (commonly called *Key Performance Indicators, KPI*s) whose target values are determined by the company strategy, and where the actions taken are aimed at matching current and target values for these indicators.

From an architectural point of view, the core of a business intelligence system is usually a data warehouse that stores the corporate historical data in a consistent and integrated form. A number of applications may be built around the data warehouse, for instance aimed at supporting OLAP analysis, data mining, what-if analysis, forecasting, balanced scorecards preparation, geospatial analysis, click-stream analysis. The architecture may be completed by a reactive data flow, more suited for monitoring the time-critical operational processes by supporting real-time applications.

Cross-references

▶ Data Mining
▶ Data Warehouse Applications
▶ Data Warehousing Systems: Foundations and Architectures
▶ On-Line Analytical Processing
▶ What-if Analysis

Recommended Reading

1. Eckerson W. Performance dashboards: Measuring, monitoring, and managing your business. Wiley, 2005.
2. Golfarelli M., Rizzi S., Cella I. Beyond data warehousing: What's next in business intelligence? In Proc. ACM 7th Int. Workshop on Data Warehousing and OLAP, 2004, pp. 1–6.
3. Moss L.T. and Atre S. Business Intelligence Roadmap: The complete project lifecycle for decision-support applications. Addison-Wesley Information Technology Series, 2003.

Business Process Execution Language

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Synonyms

BPEL; BPEL4WS

Definition

The *Business Process Execution Language for Web Services* (BPEL) has emerged as a standard for specifying and executing processes. It is supported by many vendors and positioned as the "process language of the Internet." BPEL is XML based and aims to enable "programming in the large," i.e., using BPEL new services can be composed from other services.

Key Points

BPEL [2,3] supports the modeling of two types of processes: executable and abstract processes. An *abstract*, (not executable) *process* is a business protocol, specifying the message exchange behavior between different parties without revealing the internal behavior for any one of them. This abstract process views the outside world from the perspective of a single organization or (composite) service. An *executable process* views the world in a similar manner. However, things are specified in more detail such that the process becomes executable, i.e., an executable BPEL process specifies the execution order of a number of *activities* constituting the process, the *partners* involved in the process, the *messages* exchanged between these partners, and the *fault* and *exception handling* required in cases of errors and exceptions.

A BPEL process itself is a kind of flow-chart, where each element in the process is called an *activity*. An activity is either a primitive or a structured activity. The set of *primitive activities* contains: `invoke`, invoking an operation on a web service; `receive`, waiting for a message from an external source; `reply`, replying to an external source; `wait`, pausing for a specified time; `assign`, copying data from one place to another; `throw`, indicating errors in the execution; `terminate`, terminating the entire service instance; and `empty`, doing nothing.

To enable the presentation of complex structures the following *structured activities* are defined: `sequence`, for defining an execution order; `switch`, for conditional routing; `while`, for looping; `pick`, for race conditions based on timing or external triggers; `flow`, for parallel routing; and `scope`, for grouping activities to be treated by the same fault-handler. Structured activities can be nested and combined in arbitrary ways. Within activities executed in parallel the execution order can further be controlled by the usage of `links` (sometimes also called control links, or guarded links), which allows the definition of directed graphs. The graphs too can be nested but must be acyclic.

The terminology above is based on BPEL 1.1 which was introduced in 2003 [3]. A new version of the standard [2] was published in 2007. This version has been approved as an OASIS Standard. This new version resolves many semantical issues [1,5]. Moreover, new activity types such as `repeatUntil`, `validate`, `forEach` (parallel and sequential), `rethrow`, `extensionActivity`, and `compensateScope`, have been added and some of the existing activities have been renamed (`switch/case` renamed to `if/else` and `terminate` renamed to `exit`). Currently, many extensions are under development, including BPEL4People which enables BPEL activities to be executed by human resources [4].

Cross-references

- ▶ BPMN
- ▶ Business Process Management
- ▶ Choreography
- ▶ Composition
- ▶ Orchestration
- ▶ Web Services
- ▶ Workflow Management
- ▶ Workflow Patterns

Recommended Reading

1. Aalst van der W.M.P., Dumas M., ter Hofstede A.H.M., Russell N., Verbeek H.M.W., and Wohed P. Life after BPEL? In WS-FM, 2005, pp. 35–50.
2. Alves A., Arkin A., Askary S., Barreto C., Bloch B., Curbera F., Ford M., Goland Y., Guzar A., Kartha N., Liu C.K., Khalaf R., Koenig D., Marin M., Mehta V., Thatte S., Rijn D., Yendluri P., and Yiu A. Web services business process execution language, version 2.0 (OASIS Standard). WS-BPEL TC OASIS. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, 2007.
3. Andrews T., Curbera F., Dholakia H., Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I., and Weerawarana S. Business process execution language for web services, version 1.1. Standards Proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.

4. Kloppmann M., Koenig D., Leymann F., Pfau G., Rickayzen A., von Riegen C., Schmidt P., and Trickovic I. WS-BPEL extension for people BPEL4People. In Proc. 22nd Int. Conf. on Conceptual Modeling, 2005.
5. Wohed P., van der Aalst W.M.P., Dumas M., and ter Hofstede A.H.M. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In I.Y. Song, S.W. Liddle, T.W. Ling, and P. Scheuermann, editors. In Proc. 22nd Int. Conf. on Conceptual Modeling, 2003, pp. 200–215.

Business Process Management

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Synonyms

Workflow management; Process management; Case handling

Definition

Information technology has changed business processes within and between enterprises. Traditionally, information technology was mainly used to support individual tasks ("type a letter") and to store information. However, today business processes and their information systems are intertwined. Processes heavily depend on information systems and information systems are driven by the processes they support [1]. *Business Process Management* (BPM) is concerned with the interactions between processes and information systems. An important element of BPM is the modeling and analysis of processes. Processes can be designed using a wide variety of languages ranging from BPMN to Petri nets. Some of these languages allow for analysis techniques (e.g., model checking and simulation) to answer questions related to correctness and performance. Models can be used to configure generic software tools, e.g., middleware, workflow management systems, ERP systems, etc. These systems, also referred to as *Business Process Management Systems* (BPMS), are used to enact relevant business processes. A business process management system can be defined as: *a generic software system that is driven by explicit process designs to enact and manage operational business processes*. The system should be "process-aware" and "generic" in the sense that it is possible to modify the processes it supports. The process designs are often graphical and the focus is on structured processes that need to handle many cases. Workflow management systems are typical examples of such "process-aware" systems. An important technological enabler for business process management systems is the Service Oriented Architecture (SOA). The partitioning of processes into services makes it easier to isolate the process-logic.

Historical Background

Traditionally, information systems are viewed from either a *process-centric* or an *information-centric* perspective. The information-centric view focuses on the information managed by the system. Database management systems provide the functionality required to store and retrieve data. Since the 1970's, there have been consensus on the modeling of data. Although there are different languages and different types of database management systems, the fundamental concepts are quite stable for the information-centric view of information systems. The process-centric view on information systems on the other hand can be characterized by the term "divergence." There is little consensus on the fundamental concepts. Despite the availability of established formal languages (e.g., Petri nets and process calculi) industry has been pushing ad-hoc/domain-specific languages. As a result there is a plethora of systems and languages available today.

An good starting point from a scientific perspective is the early work on office information systems. In the 1970's, people like Skip Ellis, Anatol Holt, and Michael Zisman already worked on so-called office information systems, which were driven by explicit process models [6]. It is interesting to see that the three pioneers in this area independently used Petri-net variants to model office procedures. During the 1970's and 1980's, there was great optimism about the applicability of office information systems. Unfortunately, few applications succeeded. As a result of these experiences, both the application of this technology and research almost stopped for a decade. Consequently, hardly any advances were made in the 1980's. In the 1990's, there again was a huge interest in these systems [7]. The number of workflow management systems developed in the period 1995–2005 and the many papers on workflow technology illustrate the revival of office information systems. Today workflow management systems are readily available. However, their application is still limited to specific industries such as banking and insurance. In fact, workflow technology is often hidden inside other systems. For example, ERP systems like SAP and Oracle provide workflow engines. Many other platforms include workflow-like functionality. For example, integration and application infrastructure software such as IBM's Websphere provides extensive process support.

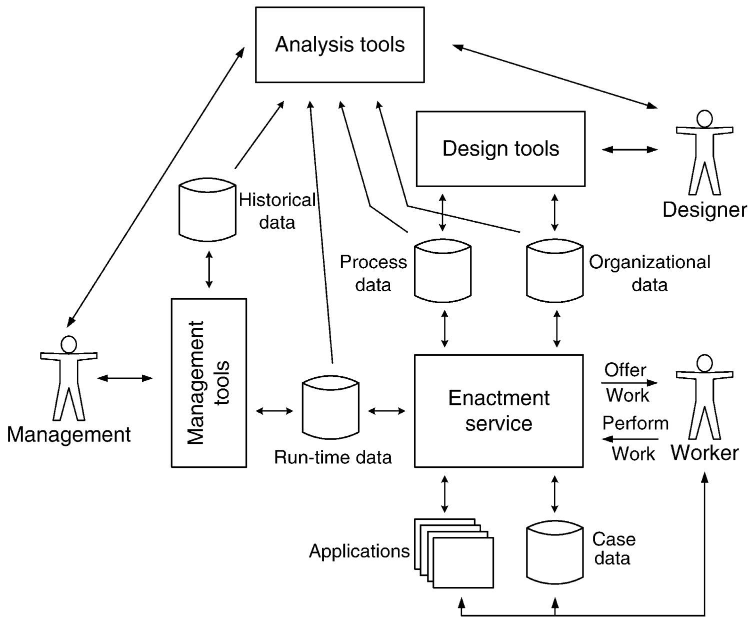
When comparing today's business process management systems to the workflow management systems of the nineties two things can be noted. First of all, the focus is no longer exclusively on automation and enactment, e.g., process analysis (simulation, process mining, verification, etc.) is increasingly important. Second, the use of web technology makes it easier to realize such systems even if processes are scattered over multiple organizations.

Foundations

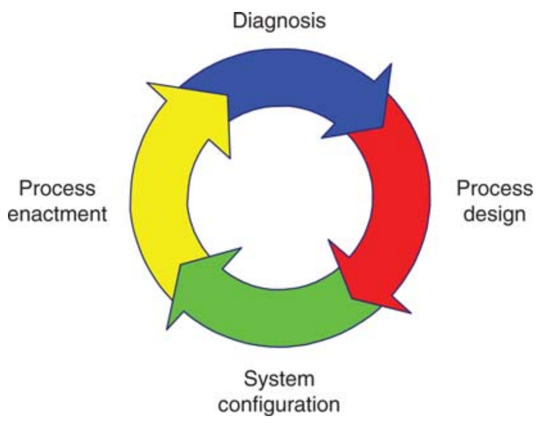
Business process management looks at the relationships between business processes and information systems. Using information systems in an innovative way enables new types of business processes. For example, making paper documents electronic may enable the concurrent execution of tasks thus shortening flow times. Moreover, characteristics of business processes lead to requirements for business process management systems (cf. workflow patterns). Business process management is *not* limited to the automation of business processes. For example, it is vital to *analyze* processes before and after they are enacted. During the design phase it is vital to use verification techniques to assess the correctness of the process design. Moreover, simulation techniques can be used to estimate the performance of the process once it is realized. While the processes is running, the information system needs to record information about actual events and realized performance. Using process mining techniques and other types of business intelligence, the event logs of systems can be analyzed. Based on such a diagnosis, the process can be improved.

The modeling and analysis of processes plays a central role in business process management. Therefore, the choice of language to represent an organization's processes is essential. Three types of languages can be identified:

1. *Formal languages*: Processes have been studied using theoretical models. Mathematicians have been using Markov chains, queueing networks, etc. to model processes. Computer scientists have been using Turing machines, transition systems, Petri nets, and process algebras to model processes.



Business Process Management. Figure 1. Architecture of a business process management system.



Business Process Management. Figure 2. BPM life-cycle.

All of these languages have in common that they have *unambiguous semantics* and allow for *analysis*.

2. *Conceptual languages*: Users in practice have problems using formal languages. They prefer to use higher-level languages. Examples are BPMN (Business Process Modeling Notation), EPCs (Event-Driven Process Chains), UML activity diagrams, etc. These language are typically *informal*, i.e., they do not have a well-defined semantics and do not allow for analysis. Moreover, the lack of semantics makes it impossible to directly execute them.
3. *Execution languages*: Formal language typically abstract from "implementation details" (e.g., data structures) and conceptual languages only provide an approximate description of the desired behavior. Therefore, more technical languages are needed for enactment. An example is the BPEL (Business Process Execution Language) language. Most vendors provide a proprietary execution language.

The existence and parallel use of these three types of languages causes many problems. The lack of consensus makes it difficult to exchange models. The gap between conceptual languages and execution languages leads to re-work and a disconnect between users and implementers. Moreover, both types of languages are not supported by advanced analysis tools.

Figure 1 shows the typical architecture of a business process management system. The figure also shows three roles of people involved: management, designer, and worker. The "heart" of the business process management system is the enactment service also known as "workflow engine" [4,7]. This engine is offering the right pieces of work (work-items) to workers at the right point in time. In order to do this, it needs to have detailed descriptions of the processes and organizations involved. Using design tools one can model processes and organizations. Note that Fig. 1 presents an idealized view. As indicated before there may be different languages (formal, conceptual, and execution languages) involved. The designer may first model the process in an informal manner. This model is then converted into a model that can be enacted or analyzed. The enactment service is driven by models in order to offer the right piece of work to the right persons at the right time. Moreover, the enactment service is starting applications and provides access to case data. During execution all kinds of information are recorded and at any time there is a (partial) history (i.e., audit trails, event logs, etc.) and a current state (run-time data). This information can be used for all kinds of analysis. Some types of analysis focus on the process design (e.g., verification and simulation). Other types of analysis focus on the actual behavior of the process (e.g., process mining).

As indicated before, different types of people are involved (management, designers, and workers). Moreover, business process management systems have a characteristic *life-cycle*. Figure 2 shows the four phases of such a life-cycle [7]. In the *design phase*, the processes are (re)designed. In the *configuration phase*, designs are implemented by configuring a process aware information system (e.g., a BPMS). After configuration, the *enactment phase* starts where the operational business processes are executed using the system configured. In the *diagnosis phase*, the operational processes are analyzed to identify problems and to find things that can be improved. The focus of traditional workflow management (systems) is on the lower half of the BPM life-cycle. As a result there is little support for the diagnosis phase. Moreover, support in the design phase is limited to providing an editor while analysis and real design support are missing. It is remarkable that few systems provide good support simulation, verification, and validation of process designs. Another problem of conventional workflow systems is the lack of flexibility. Fortunately, the emphasis is shifting from automation of highly structured processes to issues such as flexibility and analysis. Case handling systems such as FLOWer and academic prototypes such as DECLARE, YAWL/worklets, and ADEPT offer innovative ways of supporting flexible processes. Process mining tools such as ProM, ARIS PPM, etc. allow for the analysis of actual behavior. This supports the diagnosis phase and triggers process improvement.

The architecture demonstrated in Fig. 1 does not show any organizational boundaries. In the traditional setting it was very difficult to support inter-organizational processes. Web services and the Service Oriented Architecture (SOA) simplify the distribution of processes over different organizations [8]. Moreover, the paradigm shift towards services has also changed the architecture within a single organization. When focusing on processes, two terms are important: (i) choreography and (ii) orchestration.

Choreography is concerned with the exchange of messages between those services. *Orchestration* is concerned with the interactions of a single service with its environment. While choreography can be characterized by reaching an agreement and the monitoring of the overall progress, the focus of orchestration is more on the implementation of a particular service by describing the process logic and linking this to neighboring services. Orchestration languages are close to traditional workflow languages (BPMN, BPEL, Petri nets, etc.). An important characteristic of such languages is the ability to compose a service by using other services. The role of choreography languages (e.g., WS-CDL) is less clear.

Business process management is clearly related to management science. For example, topics such as operations research, operations management, business process re-engineering are highly relevant [9]. There are also clear links with coordination languages and theory. Coordination can be defined as "managing dependencies between activities." Process modeling languages and concepts such as choreography and orchestration are obviously related to coordination.

Key Applications

Banking

The financial industry has changed dramatically using both business process re-engineering and workflow-like technologies. Many processes have been rationalized using business process management techniques. The rise of e-banking led to a dramatic reduction of people and offices. Banking processes are supported and monitored by business process management systems.

Government

Government organizations need to react quickly to new legislation, i.e., the corresponding processes need to be modified based on changes in tax laws, customs procedures, immigration laws, corporate governance, safety regulations, etc. Business process management assists in dealing with these changes and further improving the processes.

Business-to-Business

Mergers and virtual enterprises trigger the need for cross-organizational workflows. Web services and business process management techniques can assist in connecting process fragments from different organizations. The SOA combined with languages like BPEL provides a good basis for cross-organizational workflows.

Health-care

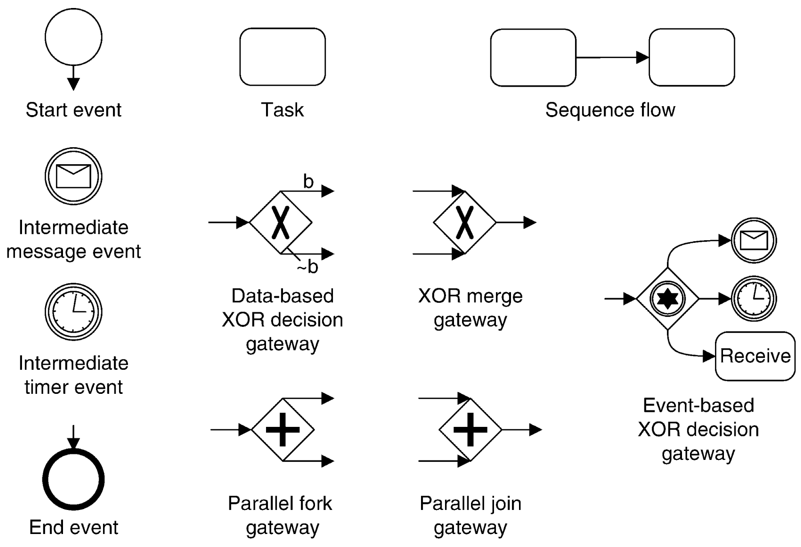
Business process management techniques have mainly been applied to structured processes. Given the nature of care processes, it is not easy to streamline these processes and to support them with workflow-like systems. However, they only way to reduce costs and improve effectively is to provide better support for such processes. Hence, the health-care domain poses an interesting and relevant challenge for business process management.

Cross-references

▶ BPEL
▶ BPMN
▶ Composition
▶ Choreography
▶ Orchestration
▶ Process Mining
▶ Web Services
▶ Workflow Management
▶ Workflow Management and Workflow Management Systems
▶ Workflow Model Analysis
▶ Workflow Patterns

Recommended Reading

1. Dumas M., van der Aalst W.M.P., and ter Hofstede A.H.M. Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley, New York, NY, USA, 2005.
2. Georgakopoulos D., Hornick M., and Sheth A. An Overview of Workflow Management: From Process Modeling to Workflow



Business Process Modeling Notation. Figure 1. BPMN notation.

Automation Infrastructure. Distrib. Parallel Databases, 3:119–153, 1995.
3. Jablonski S. and Bussler C. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer, London, UK, 1996.
4. Leymann F. and Roller D. Production Workflow: Concepts and Techniques. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.
5. Reijers H. Design and Control of Workflow Processes: Business Process Management for the Service Industry. LNCS 2617. Springer, Berlin Heidelberg New York, 2003.
6. van der Aalst W.M.P. Business process management demystified: a tutorial on models, systems and standards for workflow management. In Lectures on Concurrency and Petri Nets, J. Desel, W. Reisig, G. Rozenberg (eds.). LNCS 3098. Springer, Berlin Heidelberg New York, 2004, pp. 1–65.
7. van der Aalst W.M.P. and van Hee K.M. Workflow Management: Models, Methods, and Systems. MIT, Cambridge, MA, 2004.
8. Weske M. Business Process Management: Concepts, Languages, Architectures. Springer, Berlin Heidelberg New York, 2007.
9. zur Muehlen M. Workflow-Based Process Controlling: Foundation, Design and Application of Workflow-Driven Process Information Systems. Logos, Berlin, 2004.

Business Process Model

► Workflow Model

Business Process Modeling

► Workflow Modeling

Business Process Modeling Notation

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Synonyms

BPMN

Definition

The Business Process Modeling Notation (BPMN) is a graphical notation for drawing business processes. It is proposed as a standard notation for drawing models understandable by different business users. BPMN aims to bridge the communication gap that frequently occurs between business process design and implementation. The language is similar to other informal notations such as UML activity diagrams and extended event-driven process chains.

Key Points

BPMN was initially developed by Business Process Management Initiative (BPMI). It is now being maintained by the Object Management Group (OMG) who released a "Final Adopted Specification" in 2006 [3]. The intent of BPMN is to standardize a business process modeling notation in the face of many different modeling notations and viewpoints.

A model expressed in terms of BPMN is called a Business Process Diagram (BPD). A BPD is essentially a flowchart composed of different elements. There are four basic categories of elements: (i) *Flow Objects*, (ii) *Connecting Objects*, (iii) *Swimlanes*, and (iv) *Artifacts* [3]. Flow objects are the main graphical elements to define the behavior of a process. There are three types of flow objects: *Events*, *Activities*, and *Gateways*. Events are comparable to places in a Petri net, i.e., they are used to trigger and/or connect activities. There are different types of activities. Atomic activities are referred to as tasks. Gateways are used to model splits and joins. The flow objects can be connected to establish a control-flow. Swimlanes are just a means to structure processes. Artifacts are used to add data or to further annotate process models.

Figure 1 shows the basic set of symbols used by BPMN. These symbols can be combined to construct a BPD. On the left hand side, four types of events are shown. Three type of splits are shown. The data-based XOR gateway split passes control to exactly one of its output arcs. The parallel fork gateway passes control to all output arcs. The event-based XOR gateway selects one output arc based on the occurrence of the corresponding event. Note that Fig. 1 shows only a subset of all possible notations. The complete language is rather complex. The specification itself [3] is 308 pages without providing any formal semantics.

BPMN is an informal language aiming at the communication and not directly at execution. Therefore, the language is positioned as the design language for BPEL, i.e., BPMN diagrams are gradually refined into BPEL specifications. Given the lack of formal semantics this is not a trivial task [2,4]. Therefore, several attempts have been made to provide semantics for a subset of BPMN [1].

Cross-references

- ▶ BPEL
- ▶ Business Process Management
- ▶ Composition
- ▶ Orchestration
- ▶ Petri Nets
- ▶ Web Services
- ▶ Workflow Management
- ▶ Workflow Patterns

Recommended Reading

1. Weske M. Business Process Management: Concepts, Languages, Architectures. Springer, Berlin Heidelberg New York, 2007.
2. White S. Using BPMN to model a BPEL process. BPTrends, 3(3):1–18, March 2005.
3. White S.A. et al. Business Process Modeling Notation Specification (Version 1.0, OMG Final Adopted Specification), 2006.
4. Wohed P., van der Aalst W.W.P., Dumas M., ter Hofstede A.H.M., and Russell N. On the Suitability of BPMN for Business Process Modelling. In Proc. Int. Conf. Business Process Management 2006, pp. 161–176.

Business Process Monitoring

▶ Event-Driven Business Process Management

Business Process Optimization

▶ Process Optimization

Business Process Redesign

▶ Business Process Reengineering

Business Process Reengineering

Chiara Francalanci
Politecnico di Milano University, Milan, Italy

Synonyms

Business Process Redesign

Definition

Business process reengineering refers to a substantial change of a company's organizational processes that (i) is enabled by the implementation of new information technologies that were not previously used by the company, (ii) takes an interfunctional (or interorganizational) perspective, i.e., involves multiple organizational functions (or organizations) that cooperate along processes (iii) considers end-to-end processes, i.e., processes that deliver a service to a company's customers, (iv) emphasizes the integration of information and related information technologies to obtain seamless technological support along processes.

Historical Background

In 1993, Hammer and Champy [3] introduced the concept of business process reengineering as a radical and fast change of organizational processes that leverages information technology. Their work was the first of a wave of research contributions that analyzed the role and impact of information technologies in business process reengineering. This research was accompanied by a widespread use of the term business process reengineering in the industry to indicate large projects that involved the implementation of client-server architectures and the concurrent redesign of core business processes. Client-server enabled a more extensive sharing of organizational data within organizations by making server data accessible to personal computers through more user friendly client applications. In turn, this enabled a redesign of end-to-end business processes towards greater interfunctional cooperation. These changes primarily involved service companies, such as insurance companies, banks, research institutions, and so on, since client server had much broader application in service industries due to a more widespread use of personal computers by all employees and in all activities, both clerical and operating. More recently, the term has been extended to indicate any substantial redesign of business processes that is enabled by information technologies.

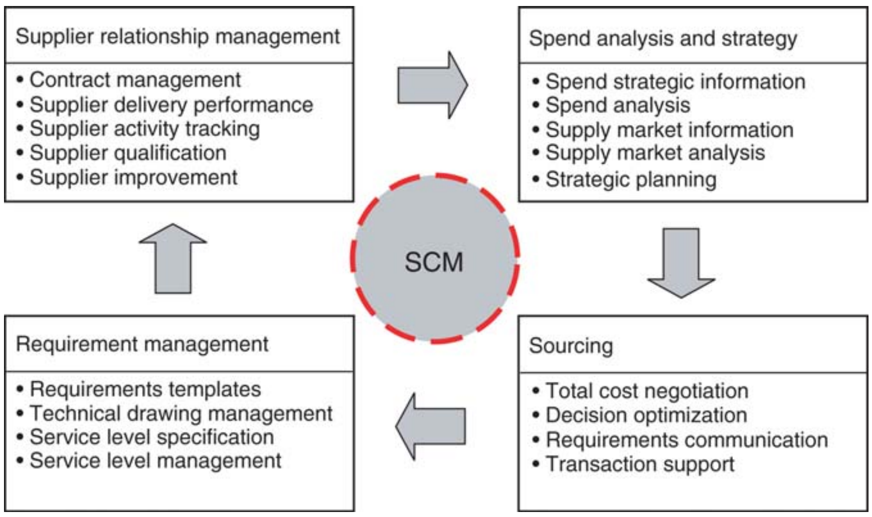
Foundations

The interfunctional integration of business processes has been recognized as a fundamental lever to improve organizational performance ever since the early studies within the information perspective of organizational theory. Going back to Galbraith's analysis [5], two complementary methods for increasing the information processing capacity of an organization are proposed. The first improves the communication of information within organizations along hierarchies of authority through *vertical* information systems. Information is gathered from lower hierarchical levels, consolidated, and conveyed to higher decision-making centers. While very effective for routine decision making, vertical information systems can quickly become overloaded in conditions of increasing uncertainty. In this case, firms can resort to direct *lateral* communication, Galbraith's second method. Liaison roles, task forces, project teams, and matrix structures are increasingly powerful mechanisms to facilitate this lateral interchange of information. While the first method reinforces the communication of information across levels of authority, the second consists of a set of organizational solutions allowing information exchanges orthogonal to the hierarchy.

Solutions for direct lateral communication permit higher organizational efficiency when hierarchies become inadequate information processors because of increasing coordination needs. This happens when dependencies between agents are difficult to foresee, as they continually change over time. In principle, within a hierarchy, lateral communication can take place only through levels of authority. Any two agents communicating through hierarchical levels experience an efficiency that is inversely proportional to their distance in the hierarchy. If dependencies between agents are stable and cause recurring paths of lateral communication, hierarchies of authority can be built by grouping agents who require most frequent interaction. By minimizing the distance between interacting agents, hierarchical coordination can be efficient. On the contrary, when organizations operate in conditions of high uncertainty, lateral information exchange becomes essential. Since demands for responsiveness and flexibility in today's business environment are raised as a result of increased uncertainty, the lateral exchange of information becomes critical to support overall information processing needs. Business process reengineering involves the redesign of organizational processes towards a higher degree of lateral communication.

Traditional Intra-Organizational Reengineering

Historically, mainframe-based architectures could support the implementation of vertical information systems, but were inherently inadequate for lateral communication. In centralized architectures, no distinction was made between data and applications. Data belonged to the application creating them and could be accessed only through that same application. Consistent with Galbraith's recommendations for the use of IT to support vertical communication, design methodologies witnessed a focus on the hierarchical conception and implementation of information systems. For example, the Normative Application Portfolio [9]


Supplier relationship management
• Contract management
• Supplier delivery performance
• Supplier activity tracking
• Supplier qualification
• Supplier improvement
Spend analysis and strategy
• Spend strategic information
• Spend analysis
• Supply market information
• Supply market analysis
• Strategic planning
SCM
Requirement management
• Requirements templates
• Technical drawing management
• Service level specification
• Service level management
Sourcing
• Total cost negotiation
• Decision optimization
• Requirements communication
• Transaction support

Business Process Reengineering. Figure 1. The supply chain management (SCM) learning cycle.

adopted a layered view of organizations, following Anthony's hierarchical framework of planning and control systems. Three main classes of applications were distinguished, supporting operations, management, and strategy, respectively. The application portfolio was created by building vertically from one level to the next, in order to guarantee that activities on lower levels supported higher level activities.

In the mid-1970s, databases and database management systems (DBMS) introduced significant changes in the design of IT architectures. Based on database technology, a new type of IT architecture was implemented, which can be broadly categorized as *centralized*. Centralized architectures allow the *logical separation* between data and applications. The layer of software services constituting the DBMS logically separated data and applications and permitted their independent design. Data common to different applications could be designed and managed as a unified resource. Database management systems exported data manipulation services that could be accessed by any application.

The management of data as a unified resource favored information sharing by integrating data common to different applications. Users running different applications could exchange information by storing and retrieving data in the central database. Conversely, designers could conceive new applications taking advantage of previously gathered and integrated information. A classical example is the use of accounting data by financial applications, allowing more precise financial analyses through detailed information on a firm's cash flows. Likewise, replenishment could be optimized through integration with order fulfillment data.

In the 1990s, client-server architectures allowed the *physical separation* between data and applications. Unlike the logical separation provided by databases in centralized architectures, the physical separation allowed the storage of data and the execution of applications on any computer. Within a client-server environment, shared data were typically stored on the server, but applications could be stored and run on local servers or personal computers.

The DBMS layer of centralized architectures was complemented by an additional layer of network services achieving the physical separation between data and applications. This made the number and the location of resources transparent to individual nodes in the architecture. By relying on peripheral processing and storage capacities of individual nodes, distributed architectures could grow incrementally. This provided the flexibility to continuously adjust to requirements and to implement a variety of applications according to individual needs. This greater flexibility enabled lateral organizational solutions that allowed different functions to take full advantage of organizational information with a variety of functionalities that could be more easily adjusted to changing requirements. These functionalities are now incorporated inside ERP (Enterprise Resource Planning) systems representing fully integrated software solutions that embed laterally integrated organizational processes.

Supply Chain Management Process Reengineering

More recently, the Web service paradigm is shifting reengineering activities towards interorganizational and interpersonal processes. Web service platforms are designed to wrap a company's information system and make selected functionalities available as web services to both internal and external users. This opens up a number of new opportunities. First, the Web service paradigm is causing a radical redesign of supply chain management processes (see Fig. 1). Supply chain management applications (SCM) support the integration of suppliers into a company's information system. This integration allows concurrent and efficient planning of production activities along the value chain. SCM involve a learning process, as shown in Fig. 1. This learning process starts from the monitoring of suppliers to measure their performance and, hence, optimize procurement activities. Then, a subset of efficient and reliable suppliers is selected for tighter integration, ranging from electronic orders to requirement management and electronic requirement management and codesign. If the supplier management process is effective, a company can build and evaluation and qualification system that can lead to official certifications that suppliers themselves can leverage as part of their brand equity. Overall, SCM involves a deep reengineering of supply management processes that represents the objective of a number of current projects.

Knowledge Management Process Reengineering

Knowledge management processes constitute a second important objective of current business process reengineering activities enabled by the Web and by the more recent Web service paradigm. Knowledge management systems (KMS) are "IT-based systems developed to support and enhance the organizational processes of knowledge creation, storage/retrieval, transfer, and application" ([1]: 114). KMS span a large and complex spectrum from help desk and customer care applications to those designed to develop employee skills. Virtual communities and collaborative environments are forms of KMS and KMS can also serve as corporate knowledge repositories and maps of expertise.

The literature on Knowledge Management Systems (KMS) largely assumes that an individual's knowledge can be captured and converted into group or organization-available knowledge. When individuals do not contribute to such systems, the knowledge creation capability of the firm is adversely affected. However, there is little clarity in the information systems literature on which mechanisms are necessary for conversion to take place. Many in the information systems literature (e.g., [6]) have argued for social influences (such as culture), hierarchical authority, and/or economic incentives, all of which are external influences that rely on the broader social context outside the KMS system. An alternative view to these external influences is internal or intrinsic motivation. This view assumes that there is little that a broader context outside of the person and the person's interactions with KMS can do to enhance contributions. "Creating and sharing knowledge are intangible activities that can neither be supervised nor forced out of people. They only happen when people cooperate voluntarily" [6]. Intrinsic motivation implies that the activity is performed because of the immediate satisfaction it provides in terms of flow, self-defined goal, or obligations of personal and social identity rather than some external factor or goal. The external factors can even undermine knowledge contributions if they interfere with intrinsic motivation. Although both views are likely to play a role in knowledge sharing, the internal view is important in organizations that take on characteristics of knowledge era (or postmodern) organizations. Such organizations require a more participative and self-managing knowledge worker compared to industrial era organizations. Knowledge-era organizations are associated with increased egalitarianism among positions, increased availability of information and knowledge resources, and increased self-management of knowledge workers. Wikipedia.org represents a typical example of postmodern self-organizing KMS, in which everyone in the world can contribute spontaneously with personal expertise and where control over contributions appropriateness is not based on any hierarchical structure, but is completely peer-based.

In a postmodern organization, an employee's commitment to his or her organization results from autonomous forms of organizing and the resulting self-expression and feelings of responsibility and control of the outputs of work. With knowledge workers, where work is associated with flows of knowledge and information, rather than flows of materials, self-expression, responsibility, and control are often targeted to knowledge outputs. Knowledge workers want to share their knowledge across the organization while preserving their association with these contributions. This type of psychological attachment, or emotional connection between the person and knowledge is well documented in open source initiatives. When given a choice to require or not require attribution to one's creative works (permit others to copy, distribute, display, modify the work but only if given credit without any economic implications), individuals invariably choose the requirement of future users of their knowledge to attribute knowledge to them (http://Creative commons.org/). The importance of psychological attachment to knowledge is no less important in commercial contexts.

KMS do not necessarily harness psychological attachment between knowledge embedded in the system and the individual who is the source of that knowledge. On one side, it is risky for the organization to let knowledge reside within the minds of individuals, in a form that can easily leak across the firm's boundaries and lead to a loss of competitive advantage. On the other hand, individual employees may perceive their personal goals to be poorly served by sharing knowledge unless the system helps to manage the knowledge worker's personal attachment to knowledge and make this attachment known in relevant organizational communities. In knowledge intensive organizations, people's distinctiveness depends upon their possessed knowledge. KMS that do not help construct, communicate, and defend the psychological attachment between the knowledge and the knowledge worker, and the rest of the organization can reduce the motivation to contribute knowledge to KMS. Fostering psychological attachment is likely to increase the knowledge workers quality of contributions, not only the quantity. In fact, the more complex and tacit the knowledge contributed to KMS, the higher the effort that knowledge workers are likely to put in their contributing activities. When the KMS fosters psychological attachment to contributions, knowledge workers increase their likelihood to engage also those knowledge sharing activities that require greater effort in order to be carried out.

Process Modeling Languages and Techniques

Reengineering initiatives typically involve a process modeling phase that supports the analysis of existing processes and the specification of new processes. UML (www.uml.org) represents a quasi-standard for process modeling and is widely used in business process reengineering.

Over the years, the scope of business processes and BPM has broadened. Initially, BPM, or workflow, was a technique that helped design largely human-based, paper-driven processes within a corporate department. For example, to handle a claim, an insurance claims process, taking as input a scanned image of a paper claims form, would pass the form electronically from the mailbox (or worklist) of one claims specialist to that of another, mimicking the traditional movement of interoffice mail from desk to desk. The contemporary process orchestrates complex system interactions, and is itself a service capable of communicating and conversing with the processes of other companies according to well-defined technical contracts. A retailer's process to handle a purchase order, for example, is a service that uses XML messages to converse with the service-based processes of consumers and warehouses.

A number of new modeling languages have recently been proposed to accommodate this complexity. For example, the i* model and its subsequent developments within the TROPOS project allow the representation of strategic relationships between organizations and their relationships with goals, resources and system components (cf.[10]). The impact of the structure of cooperation forms on the interactions among actors are modeled in the field of Multi-Agent Systems to determine an architectural solution based on typical software non-functional requirements (i.e., security, integrity, modularity, etc.). Moreover, the i* model supports the analysis of high-level goals together with non-functional requirements. The ability to move from high-level goals to sub-goals is also provided by KAOS (cf.[4]). KAOS is a formal approach for analyzing goals and for transforming goals into requirements for the software system. Finally, GBRAM (Goal Based Requirements Analysis Method, cf.[2]) proposes a method supporting the initial identification of high-level goals. Different from i* and KAOS, GBRAM does not assume that high-level goals are previously identified and provides a set of strategies to elicit goals from all available sources of information.

Key Applications

Enterprise Resource Planning (ERP), Web Services (WS), Knowledge Management Systems (KMS).

Cross-references

- ► Data architectures
- ► DBMS

► ERP
► IT architectures
► UML
► Web services

Recommended Reading

1. Alavi M. and Leidner D.E. Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS Q, 25(1):107–136, 2001.
2. Anton A. Goal Identification and Refinement in the Specification of Software-based Information Systems, Ph.D. Dissertation. Georgia Institute of Technology, Atlanta, 1997.
3. Champy J. and Hammer M. Reengineering the Corporation. Harper Collins, New York, NY, 1993.
4. Dardenne A., Lamsweerde vanA., and Fickas S. Goal-directed requirements acquisition. Sci. Comput. Program., 20(1–2):3–50, 1993.
5. Galbraith J.R. Organization Design. Addison-Wesley Publishing Company, Reading, MA, 1977.
6. Jarvenpaa S.L. and Staples D.S. The use of collaborative electronic media for information sharing: An exploratory study of determinants. J. Strateg. Inform. Syst., 9(2–3):129–154, 2000.
7. Kim W.C. and Mauborgne R. Procedural justice, strategic decision making, and the knowledge economy. Strateg. Manage. J., 19:323–338, 1988.
8. Malone T.W. and Crowston K. The Interdisciplinary study of coordination. ACM Comput. Surv., 26(1):87–119, 1994.
9. Nolan R.L. Managing the Data Resource Function. West Publishing Company, St. Paul, Minnesota, MN, 1982.
10. Yu E. and Mylopoulos J. Using goal, rules and methods to support reasoning in business process reengineering. Int. J. Intell. Syst. Account. Finance Manage., 5(1):1–13, 1996.