

# Flexible Indexing Using Signatures

David Holmes  
9517 Linden Avenue  
Bethesda, MD 20814  
(240)426-1658  
E-mail: holmesdo@aol.com

## ***Abstract***

*This paper discusses an implementation of database signatures. Previous work demonstrates how to use signatures to reduce base table storage requirements in a data warehouse thereby improving performance of database administration tasks as well as complex queries.*

*This work builds upon prior work by showing how signatures enable efficient indexing. This efficient indexing strategy relies partly on abstraction and also on shifting the initial query access path to smaller tables serving as parents with foreign keys into larger transactional tables. The design shrinks the size of transactional tables and their possible indices. Moving database indexing to smaller tables reduces storage requirements and processing overhead associated with the indexes.*

## **1 Introduction**

This paper presents a database design technique called signatures, which applies to data warehouses and specifically to large transactional tables. As described in other work, a signature is a warehouse assigned surrogate key that uniquely identifies a set of details pertaining to a master. In other words, signatures are an alternative for supporting data related to set membership.

Consider healthcare claims as an example, where each row in a claim master table represents a single claim. Assume that some non-prime columns exist for each claim master row, such as a claim type, claim from date, claim through date, claim paid date, etc.

Party is an abstraction, creating a super-type / sub-type model for all professionals and organizations participating in healthcare claims. The super-type has a few generic columns of data applicable to all providers and the sub-types have details relevant to specific types of parties.

A claim party table associates each claim with some number of parties. With one row per party per claim, the claim party table is a deep table. If the claim table is the master, then claim party is a detail table with one claim having many parties.

In very large databases, with billions of rows in the larger tables, shrinking the problem at hand has benefits. Signatures have two applications in the claims model that reduce storage cost. First, signatures allow the designer to remove columns from the claim master, by migrating them to a smaller signature member table. Second, signatures replace some detail tables with smaller tables while retaining functional equivalency.

The signature columns belonging to the master table provide flexible indexing. Rather than indexing on three types of claim dates, the designer creates an index on a single date signature column. The cost of additional indexes on the non-prime columns is mitigated because of the smaller number of rows in the signature tables.

When a detail table is eliminated from the design, we add a signature to the master. The index is smaller because the cardinality of the detail table exceeds the cardinality of the functionally equivalent master table with a signature.

Using signatures as the foundation for indexing tables may use less storage and lower the index maintenance burden on extract, transformation and load (ETL) processes while maintaining comparably indexed access paths to base tables.

## 2 Prior Work

The concept of signatures in computing is well established. In [1], Bloom filters allowed for approximate membership queries for sets using hashing functions and a bit vector of a given length. Hashing function(s) applied to members determine which bits in the vector are set to ON. Bloom filters are built for query performance and have predictable false positive error rates dependent upon the number of hash functions and filter / vector size.

An application of Bloom filters, called signature files was described in [2] as an alternative to inverted indexes for information retrieval. In this example, a document signature represents approximate terms found within documents. Document signatures are much smaller than original documents and easier to maintain than inverted indexes, but maintaining high performance demands keeping the signatures in memory because of the need to perform sequential scans. Imagine a collection of three terms and their hash value / bit vectors  $\{t_1$  (0101),  $t_2$  (1010),  $t_3$  (0011) $\}$  occurring in documents  $d_1$   $\{t_1\}$ ,  $d_2$   $\{t_1, t_3\}$  and  $d_3$   $\{t_1, t_2\}$ . The Bloom filters / signatures for the documents are  $\{d_1$  (0101),  $d_2$  (0111),  $d_3$  (1111) $\}$ . In this example, queries searching for term  $t_3$  find a false positive hit on document  $d_3$ .

To optimize Web cache applications, [3] introduced compressed Bloom filters, to minimize proxy memory and transmission file size of objects transferred between proxies.

In retail data warehouse applications, market basket analysis requires n-way self joins of transaction line items and [4] described how to assign a market basket signature to simplify the problem to single pass SQL. The work is similar to our work with respect to simplifying SQL and enhancing query performance, but did not explore how to reduce storage capacity requirements and improve database administration windows.

In [4] the Fundamental Theorem of Arithmetic, or FTA, provided a means to determine absolute set membership, unlike Bloom filters, or IR signature files which provide approximate set membership. While FTA is not mandatory to generate signatures for absolute membership functionality, it easily implemented with standard SQL.

In [5], there is an extension of the FTA based set membership from [4]. The focus of this work is database compression through signatures. Specifically, signatures allow designers to eliminate detail tables from master-detail data models and retain functional equivalency.

## 3 Design Details

This work proposes signature based design patterns as a means for providing efficient access paths to transactional data in a data warehouse. Signatures improve efficiency by:

- Decreasing row width of transactional tables
- Collapsing rows of repetitive data extracted from the transactional tables into smaller lookup tables, and;
- Reducing indexing overhead

### 3.1 Claims Model

Figure 1 shows a simplified claims model we will use for comparison purposes with a signature model. In this example, claims have multiple non-prime date columns. Also, claims are associated with one or more business parties, such as physicians and institutions. It is understood that there may be additional columns in each of the tables. The additional columns are extraneous to this discussion and therefore omitted. [5] Has information on designing supporting columns into signature tables.

Assume user access paths into the claims are via any combination of FromDate, ThroughDate, PaidDate and PartySK. In the case of party searches, users probably search by a natural key existing in a sub-type table, such as a license number, but ultimate access into the claim table is by way of PartySK, the surrogate key for parties.

Assume that query volume by each access path sufficiently warrants a non-unique secondary index (NUSI) on the column. This translates to three NUSIs on the master Claim table and one NUSI on the detail ClaimParty table.

The sizing of NUSIs depends upon data demographics and database software. Sizing estimations follow later in this work.

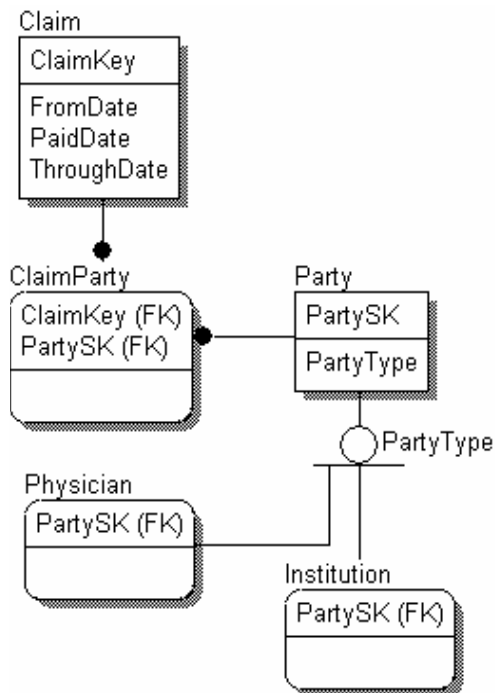


Figure 1. Simplified Traditional Model

A sample business query might be to provide a list of claims, parties and paid dates for claims occurring on April 1<sup>st</sup>. The relational algebra would use two entities Claim (C) and ClaimParty (CP) and would be:

$$\pi_{\text{ClaimKey, PaidDate, PartySK}} \left( \sigma_{\text{FromDate}=2006-04-01} (C \triangleright \triangleleft CP) \right)$$

To avoid a full table scan of the Claim table, the physical data model would require an index on the FromDate column. If the query profile for the system includes a mix of filters each type of date, the indexing requirement could become problematic with three secondary indices on one of the largest tables.

### 3.2 Signature Model

Signatures require less storage capacity than traditional master-detail models for a couple of reasons. Repetitive data in the master Claim table migrates to smaller supporting tables, such as DateSignature in Figure 2. DateSignature contains one row per distinct set of dates. In other words, no two rows would have the same

dates in DateSignature. The assertion that DateSignature is smaller assumes that many claims share the same sets of dates.

There may be as many as four NUSIs to support date searches, three on DateSignature date columns and one on the Claim DateSignatureSK column. In practice, the indexes on the DateSignature table may not be necessary, provided the table is small. Also, the DateSignatureSK NUSI on Claim may be better as part of a non-unique primary index (NUPI), eliminating the NUSI storage and maintenance overhead altogether.

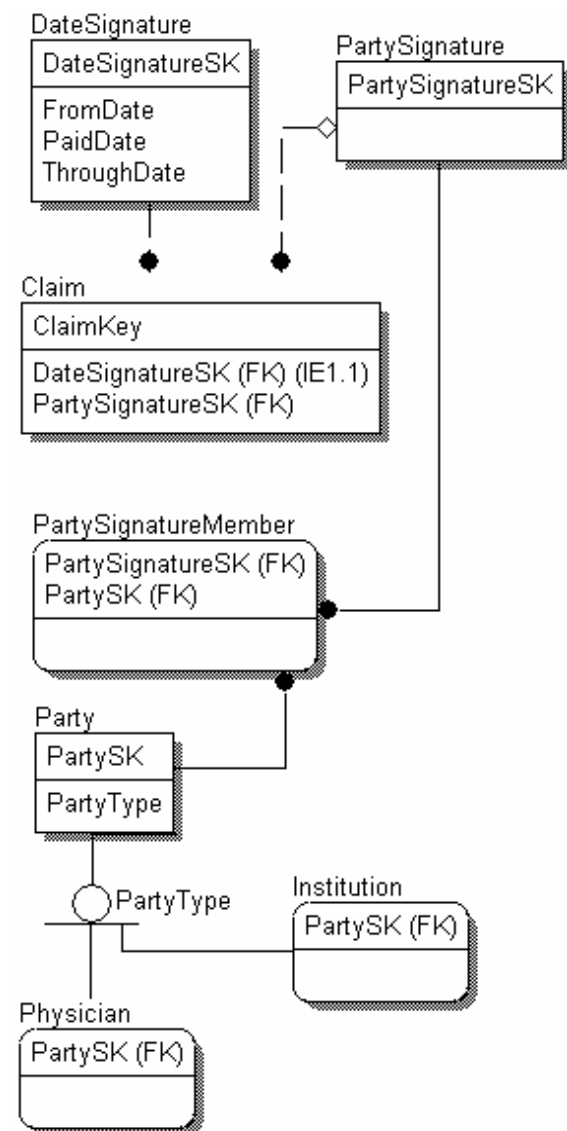


Figure 2. Signature Model

A second optimization is the elimination of the detail ClaimParty table from Figure 1. Instead of this table, the proposed design includes signature and signature member tables. Signature uniquely identifies each distinct combination of parties and the member table identifies each party associated with each signature.

The signature model differs from the traditional design, but remains functionally equivalent with Claim (C), DateSignature (DS) and PartySignatureMember (PSM):

$$\pi_{ClaimKey, PaidDate, PartySK} \left( \sigma_{FromDate=2006-04-01} (C \triangleright \triangleleft DS \triangleright \triangleleft PSM) \right)$$

To avoid a full table scan of the claim table, an index is required on the DateSignatureSK column. This single construct enables indexed access of the Claim table on any of the claim dates. Indices on the date columns in the DateSignature table may help performance, depending upon data demographics, but this table should be much smaller than Claim and their overhead correspondingly less.

## 4 Capacity Estimation

The following capacity estimations assume a Teradata V2R5 database environment and are shown in Table 1 to compare the efficiency of secondary indexes for traditional and signature models.

Table 1. Configuration Parameters

Number of AMPS (A)	40
Cardinality (C)	
- Claim	1,000,000,000
- ClaimParty	2,000,000,000
Index Value Size (V)	4 in each example, but is variable
Distinct Index Values (D)	variable
Rows per Value (R)	variable

### 4.1 NUSI Sizing Estimation

The following formula estimates the number of bytes of storage,  $N$ , needed for a non-unique secondary index (NUSI) in Teradata.

$$N = 8C + (D(V + 17)(MIN(A, R)))$$

Table 2 estimates the NUSI storage requirement for the three claim date columns using the traditional model. The estimation assumes the master table covers a period of one year and details may exist for any of the days within the year.

Table 2. Traditional Date NUSI Estimation

NUSI	Distinct Values	Rows per Value	Projected GB
<i>Claim – Master Indexes</i>			
FromDate	365	2,739,726	7.5
PaidDate	365	2,739,726	7.5
ThroughDate	365	2,739,726	7.5
<i>Total</i>			22.4

Table 3 estimates the NUSI storage requirements for the three date columns using a signature model. The estimate includes an index on the master table and three indexes on the date signature table, one for each type of date.

Three estimates are provided. The optimistic estimate assumes that combinations of dates tend to occur repetitively. In this context, this implies that patterns exist within the data. For example, a claim from date tends to precede the claim through date, which tends to precede the claim paid date. A pessimistic estimate assumes that no combination of dates ever occurs on more than one claim.

Table 3. Signature Date NUSI Estimation

NUSI	Distinct Values	Rows per Value	Projected GB
<i>Claim – Master Indexes</i>			
DateSignature	10K	100K	7.5
	10M	100	15.3
	1B	1	27.0
<i>DateSignature Table (Cardinality 10M)</i>			
FromDate	365	27,397	0.1
PaidDate	365	27,397	0.1
ThroughDate	365	27,397	0.1

Optimistically, signature indexing reduces indexing storage requirements of the example by 67%. Pessimistically, signatures can increase

storage when the data demographics of the date columns yield no repetition of values across rows. This is consistent with the base table sizing estimates in [5].

Table 4 estimates the NUSI storage for the traditional model ClaimParty column PartySK. Predictably, the size varies based upon the data demographics. The maximum projected size for the NUSI on the traditional model is 60GB, which is reached with 50 million distinct values.

Table 4. Traditional Party NUSI Estimation

NUSI	Distinct Values	Rows per Value	Projected GB
<i>Claim – Detail Index</i>			
PartySK	100K	20K	15.0
PartySK	1M	2K	15.7
PartySK	10M	200	24.8

Table 5 estimates the NUSI storage for the signature model for party signatures. Once again, the size of the index depends upon data demographics, with a maximum possible size of 29 GB.

Table 5. Signature Party NUSI Estimation

NUSI	Distinct Values	Rows per Value	Projected GB
<i>Claim – MasterPartySignatureSK Index</i>			
PartySignature	100K	10K	7.5
PartySignature	1M	1K	8.2
PartySignature	10M	100	15.3

## 5 Conclusions

Previous work demonstrated how signature based data models reduce table storage capacity requirements and thereby improving query performance and database administration activities. The savings resulted from replacing detail tables from master-detail data models with functionally equivalent signature tables.

This work examined how two applications of signatures improve indexing capacity requirements without sacrificing access path performance. First, multiple non-prime columns in the master table are replaced by a single signature. This decreases the row width of the master table and also minimizes the number of

secondary indexes needed on the master table without sacrificing efficient query access.

Second, the technique allows for the elimination of the detail table and provides functional equivalence in the master table. This widens the master table by one column and four to eight bytes, but does not increase the depth. Since the master has fewer rows than the original detail table, the signature approach is frequently more efficient because indexing is done on a table with fewer rows.

In each example, NUSI size varies based upon the number of distinct values for the indexed column. As the number of distinct values increases, so does storage usage. It is reasonable to expect more distinct values for a signature column. As long as the signatures don't have "too many" distinct values, the signature approach is more efficient for indexing and base table storage. The best way of determining "too many" is by profiling the data.

Signatures have applicability beyond storage and performance optimization. Future work will examine the use of signatures as a row level security mechanism and how they enable generic, multi-dimensional aggregates.

## References

- [1] Bloom, B., Space/time tradeoffs in hash coding with allowable errors, *Communications of the ACM*, 13(7):422-426, 1970.
- [2] Grossman, D., Frieder, O., Information retrieval algorithms and heuristics, *Kluwer Academic Publishers*, 1997.
- [3] Mitzenmacher, M., Compressed bloom filters, *Symposium on Principles of Distributed Computing*, 2001.
- [4] Maxwell, D., Holmes, D., Newton, M.. N-way Market Basket Analysis. *International Conference on Information and Knowledge Engineering*, 2002.
- [5] Holmes, D., Grossman, D., Frieder, O.. *Database Compression Using Signatures*. 2006