# View Reviews

**Paper ID**
262

**Paper Title**
Schema Selection for Block as a Value

**Track Name**
Research Paper First-Round

**Reviewer #4**

## Questions

**1. Overall Evaluation**
Weak Accept

**2. Confidence**
Knowledgeable

**3. Originality**
Medium

**4. Importance**
Medium

**5. Summary of the contribution (in a few sentences)**
This paper examines a database design problem in the context of BaaV databases (a recently proposed type of KV data store). Given a workload of parametric queries, the paper addresses how to design a BaaV database schema to optimize the evaluation of the queries within a certain space constraint. The proposed solution is formulated as an integer linear program. The experimental results are based on a SparkSQL over Cassandra implementation.

**6. List three or more strong points, labelled S1, S2, S3, etc.**
S1. The approach is presented formally and is technically sound.

S2. The paper has some nice ideas.

**7. List three or more weak points, labelled W1, W2, W3, etc.**
W1. The paper lacks a convincing discussion on why existing database design solutions can't be applied/adapted to solve the problem.

W2. Some of the proposed criteria for ranking BDSs are rather coarse-grained, and daccess is actually not used at all in the algorithm.

W3. Many important experimental details are missing from the paper.

**8. Detailed evaluation, labelled D1, D2, D3 etc.**
D1. The problem of selecting BDSs is basically a materialized view selection problem where each BDR (X,Y) in the BDS corresponds to a materialized view with an index on attributes X to efficiently retrieve attributes Y. The view selection problem is a well studied problem; in particular, the AutoAdmin Project at Microsoft Research has over the years published many papers on this topic and has developed a comprehensive solution to select indexes, materialized views, and indexes on materialized views that takes into account of space budget and maintenance cost. In fact, the ranking criteria in these papers are more realistic than the ones used in this paper (see D2) and are

targeted at more general query workload than the SPC queries studied in this paper. There're also other ILP-based solutions (e.g. Papadomanolakis & Ailamaki, An Integer Linear Programming Approach to Database Design, ICDE Workshop, 2007) developed for the database design problem. The paper ought to provide a more detailed discusssion on why such existing solutions cannot be applied/adapted to solve the database design problem for BaaV databases. May I suggest that the authors free up some space in the paper (e.g., by moving some proofs to a referenced technical report) to include such a discussion.

D2. In a view selection problem, there are three standard criteria to rank a set of views V: (C1) the query performance improvement of using V, (C2) the space cost of V, and (C3) the update maintenance cost of V.

The proposed ranking criteria for BaaV database schema uses two metrics usf and daccess for C1. The first usf criterion is a rather coarse-grained metric that is a function of the number of scan-free queries and the query frequencies; however, a schema that has a lower value of this metric is not necessarily more efficient than another schema with a higher usf value. The second daccess criterion is also a rather coarse-grained metric that is independent of the query workload and query frequencies. Together, these two coarse-grained metrics do not necessarily correlate with the efficiency improvement of the database schema. Indeed, the paper acknowledges that the daccess metric it actually not useful in practice as it is ignored in the algorithm described in Section V.

D3. The experiments were run using a cluster of four virtual machines. Since the focus of the paper is on the quality of the database schema, it would have been better to also include experimental results on a single machine to exclude other factors (e.g., data partitioning, replication) that could affect the performance comparison. The paper should include details on how the tables in Cassandra are partitioned and replicated.

D4. The experiments used specific default space budget values (0.7 times for TPC-DS and 2.5 times for TPC-H). On what basis are these values determined?

D5. The paper did not describe how each table is implemented in Cassandra. For the proposed approach, was an index built on X for each BRS (X,Y)? For the baseline approaches, were indexes used to improve query performance? These details should be included in the paper. The experimental graphs should show the actual space cost of each approach (including table and index space) rather than using the count of the number of table columns as a proxy for the storage cost. Without using a proper storage cost, it is not clear how the experiments ensure that each approach's storage cost actually fits within the allocated space budget.

D6. Since the paper introduces the "scan-free evaluability" criterion for ranking database schemas, it would be interesting to examine and comment on the impact of this critierion on query performance. For instance, in the experimental results shown, are the schemas selected by the proposed approach all have scan-free queries? If so, how would the methods compare as the space budget is reduced to reduce the number of scan-free queries for the proposed method?

D7. The experimental queries are modified queries from two benchmarks that are not presented in the paper; it would be good to made available the experimental queries (along with their queries) in a technical report.

**10. Candidate for a Revision? (Please note that the authors have only one month to revise)**
Yes

**11. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc**
Please address the issues in D1 to D7.

**Reviewer #5**

## Questions

**1. Overall Evaluation**
Weak Reject

**2. Confidence**
Knowledgeable

**3. Originality**
Medium

**4. Importance**
Medium

**5. Summary of the contribution (in a few sentences)**
The authors study the problem of efficient schema selection in the context of BaaV stores, which extend the concept of KV stores by allowing blocks of values to be mapped to every key. The paper builds on earlier work by the authors that showed that using the BaaV model can speed up relational queries over KV stores.
The current paper describes the problem of schema selection for a given workload and defines 4 metrics that need to be optimized for: scan-free evaluation, reducing data access, space requirements and cost of updates. The authors shows that this problem is NP-hard and suggest approximation algorithms that reduce the problem to Integer linear programming. Finally, the paper provides experimental evaluation over modified versions of TPC-H and TPC-DS and showed that the schemas computed by their algorithms lead to efficient query evaluation, and they can achieve 4-7x faster performance over SparkSQL-over-Cassandra.

**6. List three or more strong points, labelled S1, S2, S3, etc.**
S1. The paper is well written, with sufficient motivation of the problem and background
S2. The paper provides a solid theoretical base for the claims
S3. Initial experimental evaluation shows promising results
S4. Table 1 containing definitions is very useful: consider adding a few more of the abbreviations used throughout the code (e.g. SPC)

**7. List three or more weak points, labelled W1, W2, W3, etc.**
W1. Some of the claims made in the paper around the wide use of KV stores for SQL query processing are exaggerated (see detailed comments below).
W2. The experimental evaluation is limited: SPC is a very restricted class of queries and it's unclear how the algorithms can be extended to more realistic workloads
W3. There has been work on building indexes on top of KV stores, it would be interesting to see how your work compares and relates to those approaches (see detailed points below)

**8. Detailed evaluation, labelled D1, D2, D3 etc.**
D1. Some claims for the use of SQL on top of KV stores are far-fetched: for example Impala is not really a SQL layer built on top of a KV store: it supports a variety of underlying storage layers and while it can access some that are arguably KV stores (e.g. HBase), it is optimized for analytics over parquet files. It also supports a variety of other optimizations of the storage organization, such as partitioning which helps serve range queries well.
D2. While the paper has a solid theoretical foundation, the class of queries it considers are not representative of real customer workloads: TPC-H and TPC-DS are a good start, but the paper only considers the SPC portion of those queries, which is rather limiting. It is unclear how this can be extended beyond this simplistic class.
D3. There has been work on building indexes on top of KV stores, see e.g. "FoundationDB Record Layer: A Multi-Tenant Structured Datastore", SIGMOD 2019. It would be interesting to point out how your work compares or if it can be extended to fit these advances. Or is BaaV eliminating the need for indexes?
D4. I think Example 1 can be made a bit more realistic if you change "bank" relation for say "Walmart", and then the query will be: "cities with Walmart locations with customers (still) shopping at Amazon"

D5. Fix grammar in the first sentence of the abstract: "rather than to fetch" -> "rather than fetching"
D6: Typo in second paragraph of Introduction: "sequence of of"

**10. Candidate for a Revision? (Please note that the authors have only one month to revise)**
No

**Reviewer #6**

## Questions

**1. Overall Evaluation**
Weak Reject

**2. Confidence**
Knowledgeable

**3. Originality**
Low

**4. Importance**
Low

**5. Summary of the contribution (in a few sentences)**
The paper extends the block-as-a-value approach by the same authors, and uses an ILP solver to find the optimal block schema. That can indeed speed up the access, but the overall approach is not new, ILP solvers have been used for this problem before.

**6. List three or more strong points, labelled S1, S2, S3, etc.**
S1 improving the query performance of K/V stores is very useful
S2 formulating the schema selection as optimization problem is nice, as we can now give guarantees
S3 experiments based upon well known benchmarks TPC-H/DS (even though queries are very limited)

**7. List three or more weak points, labelled W1, W2, W3, etc.**
W1 using ILP solvers for access path selection is not new, this has been well studies for index selection
W2 the choice of block schema has implications for consistency and concurrency, which is ignored in the paper here
W3 the query support is quite limited

**8. Detailed evaluation, labelled D1, D2, D3 etc.**
Using an optimizer to decide about the access path is a good idea, but that is not really new. The whole area has been well studied in the context of index selection, which is very similar to the block selection problem studied here. See for example the following paper for for very similar approach that uses ILP solvers for index selection:

Stratos Papadomanolakis, Anastassia Ailamaki: An Integer Linear Programming Approach to Database Design. ICDE Workshops 2007: 442-449

And there are many, many other papers that use similar optimization steps to decide about physical design, usually different in how they capture the workload, how they predict the effects of indexes, or how they model updates. It is not clear how this paper here advances that already quite mature field.

What is perhaps even more problematic is that the block choice can effects on consistency and concurrency, but that is completely ignored by the paper. In a distributed K/V store, the schema choice affects the semantics (via atomicity and consistency guarantees of the underlying system). When we decide to group tuples differently, the semantic of the application is affected. How can we make such choices in an automated matter? The paper only aims to reduce access costs, but it is not clear of that is a viable approach.

Finally, the query functionality offered by the system is very limited. While the authors first talk about TPC-H and TPC-DS, they only execute the SPJ parts of the query workload, and they ignore FK-FK joins. That is overly restrictive. Of course SPJ queries are important. But first, non-PK/FK joins do happen in practice, even if they are more rare than PK/FK joins. And second, users do want to run more complex SQL, in particular they want to use aggregations. TPC-DS has some quite advanced query constucts, but TPC-H is more harmless. Seeing real TPC-H queries would been useful. And the experiments should have included a comparison with an alternative system, for example Greenplum (or even better a commercial system like Exasol).

**10. Candidate for a Revision? (Please note that the authors have only one month to revise)**
No