MAKE YOUR DATABASE DREAM OF ELECTRIC SHEEP DESIGNING FOR AUTONOMOUS OPERATION
Part #1 – Background
Part #2 – Engineering
Part #3 – Oracle Rant
**INDEX SELECTION IN A SELF-ADAPTIVE DATABASE MANAGEMENT SYSTEM**

Michael Hume

Laboratory for Computer Science, M.I.T., Cambridge, Massachusetts 02139

We address the problems of automatically adjusting the physical organization of a database to satisfy changing database access patterns. Database administrators and database designers have faced these problems for many years. Our approach is to identify the essential characteristics of a database management system and then to develop an algorithm for selecting indexes that can adapt to changing access patterns. The algorithm is designed to be efficient and scalable, and it has been tested on a variety of real-world databases.

**INTRODUCTION**

The efficient utilization of a database depends on the optimal matching of the database organization to the access requirements and other characteristics of the database. The problem of automatically adapting the physical organization of a database to changing access patterns has been studied for many years. Our approach is to identify the essential characteristics of a database management system and then to develop an algorithm for selecting indexes that can adapt to changing access patterns. The algorithm is designed to be efficient and scalable, and it has been tested on a variety of real-world databases.

**INDEX SELECTION IN ADAPTIVE DATABASE SYSTEMS**

We are currently developing a self-adaptive database management system that monitors the access patterns and the characteristics of a database and then adapts its organization to satisfy those access patterns. The system is based on the principles of self-adaptive database management, which involves the dynamic reconfiguration of the database schema and the physical organization of the database to optimize performance. The system is designed to be efficient and scalable, and it has been tested on a variety of real-world databases.

**INFORMATION**

- **Self-Adaptive Databases**: 1970-1990s
- **Algorithm**: Tuning Algorithm
- **Components**: Admin, Tuning Algorithm, Index Selection, Partitioning / Sharding, Data Placement
- **Keywords**: Autonomic DBMSs, Self-Adaptive Databases
**ABSTRACT**

In this paper, we present a taxonomy of self-tuning database systems and analyze the challenges and opportunities for automating database management. The paper is based on extensive experience with self-tuning database systems. The taxonomy is based on the types of knobs that can be self-tuned, such as index selection, partitioning, data placement, and algorithm selection. The challenges include the complexity of the knobs, the interactions between them, and the need for effective monitoring and control.

1. **RESEARCH OF AUTOADMBS PROJECT**

The goal of the AutoAdmin project was to develop a self-tuning database management system that automatically adapts to changing workloads and environments. The project was funded by the National Science Foundation and involved researchers from several universities. The system was implemented in the open-source database management system PostgreSQL. The project's main contributions included:

- **Algorithm Selection:** The system automatically selects the best algorithm for a given query, based on the current workload and system state.
- **Index Selection:** The system automatically selects the best index for a given query, based on the current workload and system state.
- **Partitioning:** The system automatically partitions the data to optimize query performance.
- **Data Placement:** The system automatically places the data on the best storage device to optimize query performance.
- **DML Tuning:** The system automatically tunes the database management language (DML) to optimize query performance.

2. **AN INTRODUCTION TO PHYSICAL DATABASE DESIGN**

2.1 **Importance of Physical Design**

A good physical design is crucial for optimizing database performance. A poorly designed database can lead to slow query response times and wasted storage space. The importance of physical design is highlighted by the fact that a well-designed database can achieve significant improvements in query performance and reduce the cost of data storage.

2.2 **State of the Art in 1993**

At the end of the 1990s, physical design was still an emerging field. Many database systems did not have good physical design capabilities. The focus was on query processing and optimization, rather than on physical design.

---

**SELECT**

**FROM**

**A**

**JOIN**

**B**

**ON**

**A.ID = B.ID**

**WHERE**

**A.VAL > 123**

**AND**

**B.NAME LIKE 'XY%'

---

**Tuning Algorithm**

**Admin**

**Knob Configuration**

**Data Placement**

**Partitioning / Sharding**

**Index Selection**

**AutoAdmin**
Why is this Previous Work Insufficient?
Problem #1: Human Judgements

Problem #2: Reactionary Measures

Problem #3: No Transfer Learning
What is **Different** this Time?
OtterTune
Existing Systems

Peloton
New System
Database Tuning-as-a-Service

→ Automatically generate DBMS knob configurations.
→ Reuse data from previous tuning sessions.

OtterTune
ottertune.cs.cmu.edu
Throughput (txn/sec)
Self-Driving Database System

→ In-memory DBMS with integrated planning framework.

→ Designed for autonomous operations.

Peloton
pelotondb.io
Design Considerations for Autonomous Operation
Configuration Knobs
Internal Metrics
Action Engineering
Anything that requires a human value judgement should be marked as off-limits to autonomous components.

– *File Paths / Network Info*
– *Durability / Isolation Levels*
– *Hardware Usage*
– *Recovery Time*
The autonomous components need hints about how to change a knob.

- Min/max ranges.
- Separate knobs to enable/disable a feature.
- Non-uniform deltas.
If the DBMS has sub-components that are tunable, then it must expose separate metrics for those components.

Bad Example: RocksDB
INTERNAL METRICS

ROCKSDB COLUMN FAMILY KNOBS

```
rocksdb_override_cf_options=
    cf_link_pk={prefix_extractor=capped:20}
```

GLOBAL FAMILY METRICS

```
mysql> SHOW GLOBAL STATUS;
```

```
+---------------+----------+----------+
| METRIC_NAME   | VALUE    | STAT     |
|---------------+----------+----------|
| ABORTED_CLIENTS | 0        |          |
| ROCKSDB_BLOCK_CACHE_BYTES_READ | 295700537 |          |
| ROCKSDB_BLOCK_CACHE_BYTES_WRITE | 709562185 |          |
| ROCKSDB_BLOCK_CACHE_DATA_HIT   | 64184    |          |
| ROCKSDB_BLOCK_CACHE_DATA_MISS  | 1001083  |          |
| ROCKSDB_BYTES_READ            | 5573794  |          |
| ROCKSDB_BYTES_WRITTEN         | 5817440  |          |
| ROCKSDB_FLUSH_WRITE_BYTES     | 2906847  |          |
| UPTIME_SINCE_FLUSH_WRITE      | 5996     |          |
```
No action should ever require the DBMS to restart in order for it to take affect.

The commercial systems are much better than this than the open-source systems.
Allow replica configurations to diverge from each other.
What About Oracle's Self-Driving DBMS?
LE - I saw your announcement about Oracle putting out the "self-driving" DBMS. What's up with that? You know that my squad been working on our self-driving DBMS for the last two years:

http://pelotondb.io

How can you do this to me after all that we've been through together? This is like that time we were together in Guatemala back in 1999. Do you remember when you asked me whether I had a spare condom? I gave my last one to you and then I found out the next night that you took all of them in your suitcase. You told me that...
True autonomous DBMSs are achievable in the next decade.

You should think about how each new feature can be controlled by a machine.
QUERY-BASED WORKLOAD FORECASTING FOR SELF-DRIVING DATABASE MANAGEMENT SYSTEM

SIGMOD 2018
QUERY-BASED WORKLOAD FORECASTING FOR SELF-DRIVING DATABASE MANAGEMENT SYSTEM

SIGMOD 2018

PELOTON
BUS TRACKING APP WITH ONE-HOUR HORIZON

Ensemble (LR+RNN)

Actual

Predicted

Queries Per Hour

0 15000 30000 45000 60000

9-Jan 11-Jan 13-Jan 15-Jan 17-Jan
Provide a notification callback to indicate when an action starts and when it completes.

Harder for changes that can be used before the action completes.
Thank you!