Consider the following Java-JDT plugin name in German:

Consider the following Java-JDT plugin name in German:

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

A BU THESIS LATEX TEMPLATE

by

JOE CANDIDATE

B.S., Some University, 2010
M.S., Another University, 2012

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2015

Approved by

First Reader

First M. Last, PhD
Professor of Electrical and Computer Engineering

Second Reader

First M. Last
Associate Professor of ...

Third Reader

First M. Last
Assistant Professor of ...

Facilis descensus Averni;
Noctes atque dies patet atri janua Ditis;
Sed revocare gradum, superasque evadere ad auras,
Hoc opus, hic labor est. *Virgil (from Don's thesis!)*

Acknowledgments

Here go all your acknowledgments. You know, your advisor, funding agency, lab mates, etc., and of course your family.

As for me, I would like to thank Jonathan Polimeni for cleaning up old LaTeX style files and templates so that Engineering students would not have to suffer typesetting dissertations in MS Word. Also, I would like to thank IDS/ISS group (ECE) and CV/CNS lab graduates for their contributions and tweaks to this scheme over the years (after many frustrations when preparing their final document for BU library). In particular, I would like to thank Limor Martin who has helped with the transition to PDF-only dissertation format (no more printing hardcopies – hooray !!!)

The stylistic and aesthetic conventions implemented in this LaTeX thesis/dissertation format would not have been possible without the help from Brendan McDermot of Mugar library and Martha Wellman of CAS.

Finally, credit is due to Stephen Gildea for the MIT style file off which this current version is based, and Paolo Gaudiano for porting the MIT style to one compatible with BU requirements.

Janusz Konrad
Professor
ECE Department

A BU THESIS LATEX TEMPLATE

JOE CANDIDATE

Boston University, College of Engineering, 2015

Major Professors: First M. Last, PhD
Professor of Electrical and Computer Engineering
Secondary appointment

First M. Last, PhD
Professor of Computer Science

ABSTRACT

Have you ever wondered why this is called an abstract*? Weird thing is that its legal to cite the abstract of a dissertation alone, apart from the rest of the manuscript.*

Contents

List of Tables

List of Figures

List of Abbreviations

The list below must be in alphabetical order as per BU library instructions or it will be returned to you for re-ordering.

CAD	. . . .	*Computer-Aided Design*
CO	. . . .	*Cytochrome Oxidase*
DOG	. . . .	*Difference Of Gaussian (distributions)*
FWHM	. . . .	*Full-Width at Half Maximum*
LGN	. . . .	*Lateral Geniculate Nucleus*
ODC	. . . .	*Ocular Dominance Column*
PDF	. . . .	*Probability Distribution Function*
$\mathbb{R}^2$	. . . .	*the Real plane*

Chapter 1

Introduction

1.1 Motivation

1.2 Problem at hand

Hello

1.3 Structure of thesis

Works

1.4 Conclusion

The next chapter gives an in-depth view of the pipeline used by the current state of art technology for query optimization in traditional data bases including the mathematical knowledge for simplification and the overall framework. The next chapter also introduces the reader to data stream and how data bases are used for them called DSMS and showcases an approach to optimize queries on data streams for the problem discusses above. The following chapter list out the details if implementation, challenges face, evaulation methods used, benchmark test case timings, followed by a summary of the paper.

Chapter 2

Related Work

2.1 Introduction, Query optimization

A database can be thought of as a list of tables, where in each table itself can be considered as a list of data points ordered initially in the sequence they are entered.

There are various tools which can be used to connect to a database, here we focus on structured query languages(SQL). A simple SQL query looks like this

```sql
SELECT column_name_1, column_name_2
FROM table_name
WHERE condition
```

This query is essentially asking to display the 2 columns from the table where the condition given is satisfied. This to particular query might be looking simple, but if the condition introduced is a complex one or if the table from which we need to return the output is complex, the question of how to execute the query optimally becomes difficult to answer.

2.2 Converting SQL queries to parse trees

We don't describe the exact grammar for the convertion to the parse tree. In these parse trees, there are 2 types of nodes, one the atoms, which are essentially keywords in SQL, operators, constants and attributes. The second is Syntactic categories, these are names for families of subqueries in triangular brackets. Each of the syntactic category has unique expansion into atoms and further syntactic categories.

2.3 Relational algebra

As we saw above, order of operations matters, if the order of operations is not thoughtout and done blindly alot of redundant steps are executed and memory is moved around unnecessarily. There are few ways to atleast look and analyse the operations and how they can be simplied.

Let R,S *be relations. Some simple laws, associativity and commutativity can easily be verified:-*

- $R \times S = S \times R$
- $(R \times S) \times T = R \times (S \times T)$
- $R \bowtie S = S \bowtie R$
- $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
- $R \cup S = S \cup R$
- $(R \cup S) \cup T = R \cup (S \cup T)$
- $R \cap S = S \cap R$
- $(R \cap S) \cap T = R \cap (S \cap T)$

When applying associative law on relations, need to be careful whether the conditions actually makes sense after the order is changed.

While the above identities work on both sets and bags(bags allow for repeatition). To show that laws for sets and bags do differ an easy way is to consider the distributive property.

$$A \cap_S (B \cup_S C) = (A \cap_S B) \cup_S (A \cap_S C)$$
$$A \cap_B (B \cup_B C) \neq (A \cap_B B) \cup_B (A \cap_B C)$$

We can simply show it with an example. Let $A = \{t\}, B = \{t\}, C = \{t\}$*. The LHS comes to be* $\{t\}$*, whereas RHS is* $\{t, t\}$

2.3.1 Select operator σ

First we start with simple properties of the σ *operator. Need to be careful about the attributes used in the select operator condition when pushing it down.*

- $\sigma_{C_1 \wedge C_2}(R) = \sigma_{C_1}(\sigma_{C_2}(R))$
- $\sigma_{C_1 \vee C_2}(R) = (\sigma_{C_1}(R)) \cup_S (\sigma_{C_2}(R))$
- $\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$
- $\sigma_C(R - S) = \sigma_C(R) - \sigma_C(S) = \sigma_C(R) - S$
- $\sigma_C(R \times S) = \sigma_C(R) \times S$
- $\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$
- $\sigma_C(R \underset{D}{\bowtie} S) = \sigma_C(R) \underset{D}{\bowtie} S$
- $\sigma_C(R \cap S) = \sigma_C(R) \cap S$

2.3.2 Projection operator π

While for the Select operator(σ) the identities were quite straight forward with not many things to consider, the identities for Projection operator (π) are bit more involved.

- $\pi_L(R \bowtie S) = \pi_L(\pi_M(R) \bowtie \pi_N(S))$*, where* M, N *are attributes required for the join or they are inputs to the projection.*
- $\pi_L(R \underset{D}{\bowtie} S) = \pi_L(\pi_M(R) \underset{D}{\bowtie} \pi_N(S))$*, similar to above identity/ law.*
- $\pi_L(R \times S) = \pi_L(\pi_M(R) \times \pi_N(S))$

- $\pi_L(R \cup_B S) = \pi_L(R) \cup_B \pi_L(S)$

- $\pi_L(\sigma_C(R)) = \pi_L(\sigma_C(\pi_M(R))$

2.3.3 Duplicate Elimination operator δ

The δ operator eliminates duplicates from bags.

- $\delta(R) = R$, *if R does not have any duplicates.*

- $\delta(R \times S) = \delta(R) \times \delta(S)$

- $\delta(R \bowtie S) = \delta(R) \bowtie \delta(S)$

- $\delta(R \underset{D}{\bowtie} S) = \delta(R) \underset{D}{\bowtie} \delta(S)$

- $\delta(\sigma_C(R)) = \sigma_C(\delta(R))$

- $\delta(R \cap_B S) = \delta(R) \cap_B S$

2.3.4 Aggregation operator γ

It is difficult to give identities for the aggregation operator, like done for the above operators. This is mostly due to how the details of how the aggregation operator is used.

- $\sigma(\gamma_L(R)) = \gamma_L(R)$

- $\gamma_L(R) = \gamma_L(\pi_M(R))$, *where M must at least contain the attributed used in L.*

-

2.4 Converting Parse trees into logical expression

2.5 Explain difficulties/ Time complexity

2.6 Optimzation using relation algebra

2.7 Introduction to Data Streams

2.8 Data stream windowing

2.9 Query Processing of data streams(Combine the DBMS and DSMS)

2.10 Challenges of query optimization on data streams

2.11 Conclusion and discussion

2.12 SQL Query compiler

The steps involved are

2.12.1 Parsing

In a very general sense, given an SQL query, SQL converts it into a parse tree based on SQL grammar.

2.12.2 Preprocessing

This step has several functions.

If a "view" is used in the query as a relation, then each instance has to be replaced by the parse tree.

The preprocessor also has to conduct semantic checking, that is, check if relations used exist, check for ambiguity, and type checking. If a parse tree passes the preprocessing then it is said to be ***valid***

2.12.3 Logical Query Plan

The first step is to modify the parse tree into using operators and operators of relational algebra.

The next step is to convert expression obtained from the above substitution and modify it into an expression which can be converted to most efficient physical query plan.

To improve the algebraic expression obtained, few common steps taken are pushing down selections and projections carefully, carefully placing duplicate eliminations, combining selections, showing associativity and commutivity in the expression to help with enumeration.

At the end when we have the expression ready, we enumerate the physical plans and calculate their cost of execution and select the method with the lowest cost.

2.12.4 Cost Estimation

We need to consider what algorithm each operator in the expression is going to use, such as join, sort, scanning and more. Also need to consider the order for the associative and commutative operators, because at the end the operators are binary and how the output of one operator is provided as an input to the next/ outter operator in the expression.

2.13 System R

2.14 Deep Reinforcement learning

Markov decision process(MDP) is used to formalize various types of stochastic processes. In MDPs, the goal of the agent is to make a sequence of actions to optimize/ maximize an objective function.
Formally a MDP is a $5-tuple$

$$\langle S, A, P(s,a), R(s,a), s_0 \rangle$$

$S \rightarrow$ *Set of all possible states the agent can be in.*

$A \rightarrow$ *Set of all possible actions the agent can take.*

$P(s,a) \rightarrow$ *A probability distribution of going to various states given current state and action.* $s^1 \sim P(s,a)$

$R(s,a) \rightarrow$ *Reward for taking action* a *on state* s.

$s_0 \rightarrow$ *Describes the initial state of the system/ agent.*

The performance of the agent is measured using the rewards collected along the way through various states. So the objective of an MDP is to find a policy $\pi : S \rightarrow A$, *a function that maps states to actions, in order to maximize the expected value:-*

$$\underset{\pi}{\operatorname{argmax}} \mathbb{E} \left[\sum_{t=0}^{T-1} R(s_t, a_t) \right]$$

$$\textit{subject to } s_{t+1} = P(s_t, a_t), a_t = \pi s_t$$

This method does not reduce the search space, and unlikely greedy solution, this will lead to an optimal solution. This method does not reduce the search space, and unlikely greedy solution, this will lead to an optimal solution.

Reinforcement learning(RL) is a technique which optimizes MDPs iteratively, by running a simulation in each iteration and changing the policy to find an optimal one based on the cumulative reward.

2.15 Relations

A common method/ data structure used to formalize joins

Query Graph $\rightarrow$ *A query graph* G *is an undirected graph, where each relation* R *is a vertex and each join predicate* ρ *defines an edge between 2 vertices. Let* κ_G

denote the number of connected components in G

A join of relation R_1, R_2, in the graph corresponds to remove the vertices v_{R_1}, v_{R_2}, replacing them with a vertex $v_{R_1+R_2}$, the edges of the form $(v_{R_1}, v)\&(v_{R_2}, v)$ are replaced by $(v_{R_1+R_2}, v)$. Note each reduction reduces number of vertices by one, so this process is repeated until there are κ_G number of vertices left.

Join Optimization Problem $\rightarrow$ *Let G be a query graph and J be a join cost model. Find sequence, $c_1 \circ c_2 \circ ... \circ c_n$ resulting in $|V| = \kappa_G$ to minimize*

$$\min_{c_1, c_2, ... c_n} \sum_{i=1}^{n} J(c_i)$$

$$\textit{subject to } G_{i+1} = c(G_i)$$

Using these definitions, we define a MDP.

$$\langle \{G_0, G_1, ...G_T\}, c, P(G, c), -J, G \rangle$$

We are still not certain about how the cost function J is structured. We are still not certain about how the cost function J is structured. We are still not certain about how the cost function J is structured.

We are still not certain about how the cost function J is structured.

Chapter 3

Stream Optimization

3.1 Query Optimization of Data Streams

Chapter 4

Implementation

4.1 Query Optimization of Data Streams

Chapter 5

Stream Optimization

5.1 Query Optimization of Data Streams

Chapter 6

Stream Optimization

6.1 Query Optimization of Data Streams

Appendix A

Proof of xyz

This is the appendix.

CURRICULUM VITAE

Joe Graduate

Basically, this needs to be worked out by each individual, however the same format, margins, typeface, and type size must be used as in the rest of the dissertation.