Jesse van Breda 0902729
01-10-2015

Query optimization for Assignment 1

Jesse van Breda 0902729
01-10-2015

The following tables each describe how a given SELECT query has been optimized using indexes. They state what queries are used to make these indexes, together with the execution time of the SELECT queries, both with and without indexes.

The project code contains a method which, with a click on the button ‘generate data’, generates 5000 rows. This is done a couple of times, so the database tables ‘User’ and ‘Character’ both contain around 15000 entries.

Optimized Query	SELECT * FROM "user" WHERE user_name = 'qw'
Index Statement	CREATE INDEX user_name_index ON "user" USING HASH (user_name)
Execution time	Without index: 32ms With Hash table index: 31ms
Motivation for the chosen index	Because this query contains an exact value that we are looking for, a Hash table index will work best here.

Optimized Query	SELECT user_name, password FROM “user” WHERE user_name = ‘qw’ AND password = ‘bjgqwqygobyxjx’
Index Statement	CREATE INDEX user_name_index ON "user" USING BTREE (user_name, password)
Execution time	Without index: 47ms With BTree index: 32ms
Motivation for the chosen index	Hash tables do not support multi-column indexes, so that’s why I chose for the Binary Tree index

Optimized Query	SELECT * FROM "user" u, "owns" o WHERE u.user_name = o.user_name AND u.user_name = 'qw'

Index Statement	CREATE INDEX user_name_index ON "user" USING HASH (user_name);
	CREATE INDEX owner_name_index ON "owns" USING HASH (user_name);

Execution time	Without index: 63ms With Hash table index: 32ms With BTree index: 31ms
Motivation for the chosen index	I chose the Binary tree index, because it is slightly faster than the Hash table index