

2019

Cycle To Share Database

ASSIGNMENT 1

JOSHUA FORD (1701602)

LECTURER: KEMI ADEMOYE

Contents

1. Abstract.....	2
2. Performance tuning	2
2.1 Data queries	2
2.2 Indexes created.....	2
3 Distributed database solution.....	3
3.1 suitable fragmentation	3
References	5

1. Abstract

The report looks at the effects of indexes in a database and how efficient it can make a database. It also looks at what the indexes needed to be on so that they were effective on the database rather than using an index on a table which was not being used as much as something else. It also covers the types of fragmentation and how it is used in database management systems, and how the different types work.

2. Performance tuning

2.1 Data queries

Data queries that would be frequently used in the Cycle to share database would be the columns that would be important for booking and renting a bike. The requirements are important because if the author does not know the main requirements for the database, he could be creating indexes for functions that would not be used as much, or indexes that are too small. Indexes are a very efficient way of finding data inside of a database. 'A database index allows a query to efficiently retrieve data from a database' [1], the [cite site](#) states that it is a good way to efficiently retrieve data. The requirements would be to be able to rent a bike which would include the attributes pickup date and return date, so that the user would know when to pickup and return the bike. Another requirement that the database would require would be to view the Tariff types, these would be the subscription type and subscription price, this would be to view the current the current tariff types available and the prices. Another requirement would be to locate and get information on a bike, this could improve the performance because there would be lots of bikes and this would make a faster and more efficient way of finding and bike on the database. The author chose to use these secondary indexes because they were the most important functions in the database and when sifting through lots of data such as the bike information table, tariff information table and the rental bike information table, it is easier t use a secondary index to look for the data in each field which is specified in the index description.

2.2 Indexes created

After analysis done by the author, the indexes that were chosen to be done was firstly on the Rental bike table, it finds the rental bike table and also specifically finds the pickupdatetime and returndatetime collumn rather than going through every table. This is so that when a user wants to have information on their bike rental, if can find the data [efficiently](#).

```
--RentalBike : pickupdate , returnDate  
CREATE INDEX RenatlBikeInfo ON RentalBike (PickupDateTime DESC, ReturnDateTime DESC);
```

The second index that was created after analysis was on the tarrif information, this is so that the user can find information on the subscription type and price without having to go through every [table](#).

Commented [KA1]: Take care with similar sounding words/phrases.

Reference(s):<http://www.grammarbook.com/homonyms/confusing-words-letter-b.asp>
<http://www.oxforddictionaries.com/words/commonly-confused-words>
<https://copywriterscrucible.wordpress.com/2009/09/01/41-commonly-misused-english-words/>
<http://www.scholastic.com/parents/blogs/scholastic-parents-raise-reader/top-20-most-commonly-confused-homophones>

Commented [KA2]: This is okay, but it would be better if it also included the BikeCode as well to work out the hire charge/ check if a bike has been returned.

Commented [KA3]: Remember to add page numbers to your report.

```
--Tariff : subscriptionType , subscriptionPrice
CREATE INDEX TariffInfo ON TariffRate (SubscriptionType DESC, SubscriptionPrice DESC);
```

The third index that was created was on the bike information, this was so that the author could find the data on what specific bike it was and where it was.

```
--Bike : BikeCode , StationCode
CREATE INDEX BikeInfo ON Bike (BikeCode DESC, StationCode DESC);
```

Commented [KA4]: Acceptable, but no need to sort in descending order.

3 Distributed database solution

3.1 ~~suitable~~ Suitable fragmentation

Horizontal fragmentation is the distribution of rows. It is often used to make the queries run faster, as cited below.

‘Allows a relation or class to be partitioned into disjointed tuples or instances.....should be fragmented so the queries of the site run faster.’ [2]

This would be the fragmentation of a column so that the data in each column can be seen in a different way, this would be for the account table, it would be split into rows, this would mean that one row would have the data from one person. Another table that would do this would be bike. The user could use a where statement so show certain information for a bike. It would split it into columns to show the information for one bike. Another case would be, in the bikes table. The table would be split so that the bikes in a certain station would be in one station then divided, then another table with the details of the bikes in that station and it would be the same for the bikes in the other stations.

Vertical fragmentation is the fragmentation (dividing) of a table vertically so that it can be seen by each attribute, when divided, each table should have the primary key,

‘...allows a relation or class to be partitioned into disjoint sets of columns or attributes except the primary key’ [2]

This could be done by looking at the information about the bikes in a certain location, the user could want to see bikes at one station, this means it would be split into vertically into columns so show each data in a specific field, rather than the users data, you can see all of the different users on the system data in one specific field. For example, in the station table, if you wanted to see where each station is, you could fragment the citycode column and by looking at the citycode column, find what city the station is in.

the table is divided into arbitrary blocks, based on the needed requirements.

Commented [KA5]: You have only described the different types of fragmentation but not really shown how you would implement a distributed database for Cycles to Share. For example, identify which tables are not suited for fragmentation indicating why and whether they are to be centralised or replicated at the different sites. For horizontal fragmentation, identify what tables you will apply this strategy and how you would implement this. For example, fragment based on what region the data belongs to, determined by the city code or station code and make sure you use derived fragmentation to place related child records at the same location as the parents. For vertical fragmentation, there is a requirement that payment is to be processed at the head office. You can apply vertical fragmentation to the Account and RentalBike tables and for the fragments that do not contain payment information, further fragment horizontally meaning you are applying a mixed strategy.

Mixed fragmentation is the fragmentation of a table in which the table is split up mixed depending on the needs of the user.

‘...the table is divided into arbitrary blocks, based on the needed requirements.’ [2]

It states that mixed fragmentation depends on how the user will need to split the tables depending on how they need them. Mixed fragmentation would be, in the accounts table, it could be split horizontally so that the user can see the city code of each account, to see where the owner of each account is from, each account from one station would be in one table and then divided from the next table which is full of accounts from the next city. It could then be split vertically for the users account details like, her name, address, date of birth and other personal information. Then it would be divided into another table for the account details like the payment details and the station information and information on the bikes.

Derived fragmentation is when relations are in many tables. If the user wanted to see all the details for a rental, it would mean joining the columns from the rental table which would be the account number, pickup station and rental number. Then because of that table having the account number, you can then fragment the account table which would have the names and other details of the user. This would mean seeing both details of the user from different tables.

References

- [1] K. Wenzel, "database indexes explained," essential sql, [Online]. Available: <https://www.essentialsql.com/what-is-a-database-index/>. [Accessed 12th march 2019].
- [2] M. P. Bhuyar, D. Gawande and P. A. Deshmukh, "Horizontal Fragmentation Technique in Distributed," *International Journal of Scientific and Research Publications*, vol. 2, no. 5, p. 2, 2012.
- 3. Data generator tool used: Mockaroo
<https://www.mockaroo.com/>

MARKING CRITERIA AND STUDENT FEEDBACK

This section details the *extent* to which the assessment criteria are demonstrated by you, which in turn determines your mark. The marks available for each category of skill are shown. Lecturers will use the space provided to comment on the achievement of the task(s), including those areas in which you have performed well and areas that would benefit from development/improvement.

Common Assessment Criteria Applied (Assignment 1)	Marks available	Marks awarded
1. Engagement with Literature Skills		
Basic evidence of reference sources. For additional improvements, expand the range of reference sources consulted. (Grade mark: 30.00%)	5	1.5
2. Knowledge and Understanding Skills		
A very good range of queries to create sample data for the database has been provided with additional, appropriate queries to test the constraints added to the tables. A fair attempt at the transaction with some sample data to test it, but unfortunately your logic is not quite correct. The table you need to check is the RentalBike table as this is the table that stores the bikes taken out on each rental. As such, your transaction does not work. (Grade mark: 75.00%)	20	15
3. Cognitive and Intellectual Skills		
Good and appropriate secondary indexes have been chosen, and you have given some good reasons for your choices but see your report for additional comments. For the distributed database design strategy, you have simply described the different types of fragmentation but not demonstrated how you would implement a distributed design for the case study. (Grade mark: 40.00%)	30	12
4. Practical Application Skills		
A very good attempt overall. There are, however, some things to note. First, why did you comment out all the code? Monetary attributes should be a numeric type that allows fractional values to be specified for pence and not INT which only allows whole numbers. You have some missing FKs e.g. the PickupStation in the Rental table. For attributes requiring a date and time element, set the data type to DATETIME and not just DATE. Note the difference between VARCHAR and CHAR. VARCHAR is for variable character length string and is more appropriate for the comments columns instead of CHAR. Also note that some cards have a CVC number with 4 characters. RegionalStation in the station table is just a field to indicate if this is the region's main station office and would probably work best as a BIT field. (Grade mark: 66.25%)	40	26.5
5. Transferable Skills for Life and Professional Practice		
The report is of a good standard. (Grade mark: 60.00%)	5	3

Assignment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)	Late Submission Penalties (tick if appropriate)	58%
	Up to 1 week late (40% Max)	
	Over 1 week late (0%)	