

This document will contain the performance tuning suggested by the student to help enhance the database created for Cycles to Share, an expanding bike share company, based in Wales.

Performance Tuning

Using performance tuning to increase the speed and efficiency of the Cycles to Share database.

Contents

1. Introduction	2
2. Indexes	2
2.1 Clustered Indexes.....	2
2.2 Non-Clustered (Secondary) Indexes	2
2.3 Three Indexes Identified to Implement in Cycles to Share Database.....	3
3. Distributed Database	5
Bibliography	7

1. Introduction

In this document the student will discuss the importance of performance tuning in a database. This will include the use of secondary indexes and data distribution to improve the performance of queries. After the student has stated the importance of the use of secondary indexes and distributed databases, three secondary indexes will be identified and added into the database. Lastly, the student will suggest a possible solution for how the database could be fragmented and distributed to areas that require certain information more so than others.

2. Indexes

An index is a structure that when implemented can improve efficiency of data retrieval from a table in a database by creating a key that is made up of one or multiple columns from a table. When creating an index, the file contains the index key value and a row address to the record. The uses of indexes are beneficial for the daily operations of a database because it can drastically speed up the searching of a table, based on size. This is because without the use of indexes, when running a query the database engine will have to look at each row to find the relative information and when found will have to continue its search until the end of the rows as data might be similar elsewhere in the table. However, when using an index, the rows are sorted in either ascending or descending order, allowing the database manager can go straight to the row addresses of the relative information and pick them out directly. There are many types of different index types but this document will only cover two, clustered and non-clustered (also known as secondary indexes).

2.1 Clustered Indexes

A clustered index states how rows should be ordered within a table based on a defined column or columns. When a clustered index is created, it defines what order the physical data should be stored within the table [1], thus meaning there can be only one clustered index on each table [2].

An advantage of using a clustered index is that when the user wants to retrieve data, it can be faster to do it using this method because when performing a query the data is already ordered, so it's just the case of going to the correct clustering key and then to the page in which the row/ rows are stored. A common analogy for this approach is a phonebook, the data will be sorted by last name and in a phonebook there will be pages with multiple entries, so when a person wants to look for someone with the last name of 'Daniels', they know that record will be before 'Simmons'.

However, although it might result quicker to use a clustering index within a database, it does come with some disadvantages. A big disadvantage of using this method is data entry into a table can become slower in larger tables. This is because the rows have to be put into the tables in order, meaning the database manager will have to search for the correct place to page and position to place the row.

2.2 Non-Clustered (Secondary) Indexes

A non-clustered index is another approach that someone could adopt when performance tuning their database. Unlike the clustered index, the ordered data is not stored on the index key but instead holds a pointer back to the base data. This means that because the index key does not contain the actual data rows themselves, the query engine must perform an extra step to locate the data.

An advantage of using non-clustered indexes is that the creator of the database is not confined to one, unlike clustered indexing. However, non-clustered indexes also come with disadvantages, one

of these being extra disk space being used. This is because, as mentioned, the index is stored separately to the data.

2.3 Three Indexes Identified to Implement in Cycles to Share Database

In this section the student will identify three good candidates that can be used as a secondary index and then implement them in the Cycles to Share.

The student must take some considerations before deciding what indexes will be implemented into the database as certain things could cause the database to not run effective and efficiently. An example of where an index would not be appropriate to use is on a table with only a small amount of fields as this could be redundant and would waste disk space. Another time where an index can become a hinderance is where the column is frequently updated.

Using these guidelines the student has come up with three indexes that they believe would be a good choice to implement on the database.

The first index that the student has chosen to implement is on the account table and has decided on these based on what columns are unlikely to change and queries that the company may wish to run frequently.

```
CREATE INDEX idx_Account_FirstName_Surname_AccountNo_Password ON Account(FirstName ASC, Surname, AccountNumber, Password)

/*-----Check To ensure index works
SELECT FirstName, Surname, AccountNumber, Password FROM Account
--DROP INDEX idx_FirstName_Surname_AccountNo ON Account

DROP INDEX idx_Account_FirstName_Surname_AccountNo_Password ON Account
-----*/
```

Figure 1

Figure 1 shows the index that the student has created, this index will sort the required rows by first name in ascending order. The reasoning behind why the company may want to implement this index into the database is because name and surname are less likely to change and are often a popular choice for WHERE clauses. Also, by using this it could be easier for someone using the database to verify a user's account.

```
CREATE INDEX idx_Account_FirstName_Surname_AccountNo_Password ON Account(FirstName ASC, Surname, AccountNumber, Password)

SELECT FirstName, Surname, AccountNumber, Password FROM ACCOUNT WHERE FirstName LIKE 'Mike'
```

Figure 2

Results		Messages		
	FirstName	Surname	AccountNumber	Password
1	Mike	Green	1005	myplssW0r0

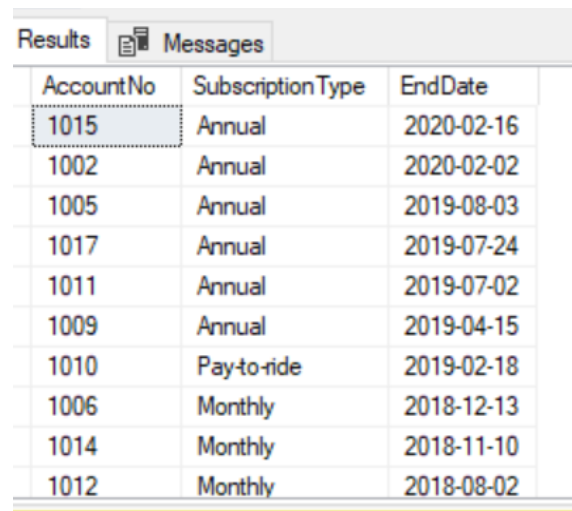
Figure 3

Figures 2 and 3 demonstrate how this may be used in a working environment.

The second index that the student has chosen to implement, seen in figure 4, is used to check subscription type, end date and account number. The student has added this index to the database as 'EndDate' could be used a lot by using 'ORDER BY' and could be used in WHERE clauses, e.g. WHERE EndDate = GetDate();

```
CREATE INDEX idx_AccountSubscription_AccountNo_SubscriptionType_EndDate ON AccountSubscription (EndDate DESC, AccountNo, SubscriptionType)
SELECT AccountNo, SubscriptionType, EndDate FROM AccountSubscription
```

Figure 4



AccountNo	SubscriptionType	EndDate
1015	Annual	2020-02-16
1002	Annual	2020-02-02
1005	Annual	2019-08-03
1017	Annual	2019-07-24
1011	Annual	2019-07-02
1009	Annual	2019-04-15
1010	Pay-to-ride	2019-02-18
1006	Monthly	2018-12-13
1014	Monthly	2018-11-10
1012	Monthly	2018-08-02

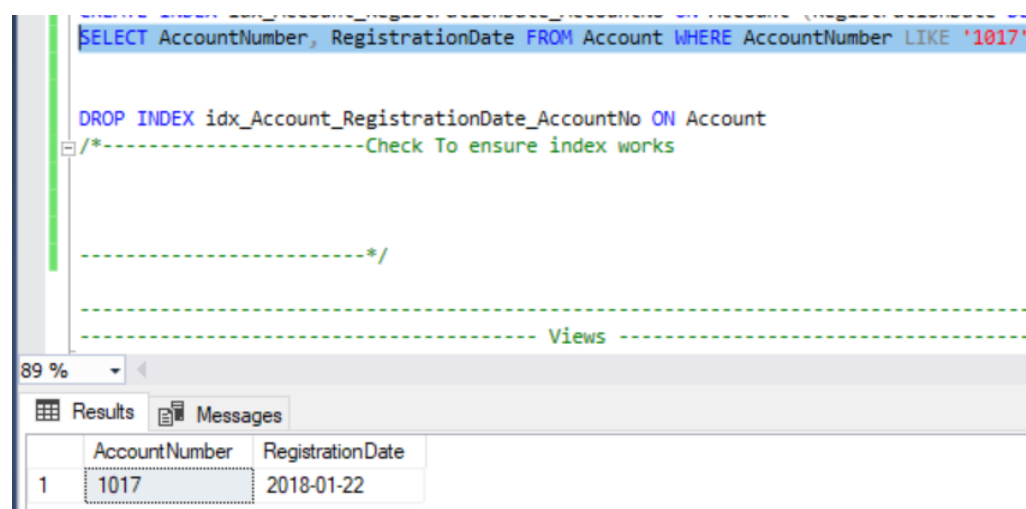
Figure 5

Figure 5 shows that when the student run the query, the information was displayed as expected, with the end date descending.

Lastly, the student has created another index that will order the registration date in a descending order. This can be used practically as it could be used in a WHERE clause to check certain parameters, see figure 7.

```
CREATE INDEX idx_Account_RegistrationDate_AccountNo ON Account (RegistrationDate DESC)
```

Figure 6



```
SELECT AccountNumber, RegistrationDate FROM Account WHERE AccountNumber LIKE '1017'
```

```
DROP INDEX idx_Account_RegistrationDate_AccountNo ON Account
```

/*-----Check To ensure index works

-----*/

-----Views

89 %

	AccountNumber	RegistrationDate
1	1017	2018-01-22

Figure 7

3. Distributed Database

Database distribution is the method of fragmenting a database which can then be stored in multiple locations, inside or outside of the original network [3]. This allows for data to be stored locally to where it's used most which could result in fast data retrieval. However, if a user needs to access information kept at a separate location, it could result in a much slower retrieval of data as it will have to travel further.

The main advantage of using a distributed database is, unlike a centralised database, if/ when the components fails, the system can still run on limited functionality rather than completely failing [3].

A database can be fragmented into many sections, these include horizontal, vertical, mixed and derived fragmentation. The student has identified these fragmentation strategies and briefly mentioned why Cycles to Share may want to adopt these for their database.

3.1 Horizontal Fragmentation

Horizontal fragmentation is where data is separated by rows which allows the user to fragment the data based on attributes. The following is a recommendation that the student will make to the company on how they may want to fragment their 'Station' table.

S ₁ : County = 'West Glamorgan' (County)
S ₂ : County = 'Bridgend County Borough' (County)
S ₃ : County = 'Cardiff County' (County)
S ₄ : County = 'Dyfed' (County)
S ₅ : County = 'Gwent' (County)

Doing this would result in 5 total fragments which are all based on the county attribute. The table below shows what the resulting fragments will look like.

S₁

CityCode	Name	County	Region
SWA	Swansea	West Glamorgan	South
NEA	Neath	West Glamorgan	South

S₂

CityCode	Name	County	Region
BRD	Bridgend	Bridgend County Borough	South

S₃

CityCode	Name	County	Region
CRD	Cardiff	Cardiff County	South

S₄

CityCode	Name	County	Region
CAM	Carmarthen	Dyfed	South
TBY	Tenby	Dyfed	Southwest

S₅

CityCode	Name	County	Region
NWP	Newport	Gwent	Southeast

If the user wanted to reconstruct the database, they can do so with the union operation.

$S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 = (\text{County})$.

The student has used horizontal fragmentation in this way so the data is split by county and each row is at the relevant county.

3.2 Vertical Fragmentation

Fragmenting vertically means that whole columns are going to be distributed rather than just individual rows, like the previous example.

The student suggests that the account table be fragmented vertically as it allows fragments to be stored at the sites that need them. An example of this is because the Swansea site will still handle the payments, we can fragment the payment information columns and store these at the Swansea site.

A₁: AccountNumber, PaymentDate, PaymentCard, PaymentCardCVC(Account)

A₂: AccountNumber, RegistrationDate, FirstName, Surname, DateOfBirth, Email, Password, PhoneNumber, Address, CityCode, PostCode(Account)

Fragment S₁

AccountNumber	PaymentDate	PaymentCard	PaymentCardCVC
1001	12-Jan-2018	Visa Debit 4659-4123-5352-1252	834
1002	03-Feb-2019	Direct Debit 4659-4412-5364-1934	324
1003	17-May-2018	Direct Debit 4659-4412-5364-1934	213
1004	18-May-2018	Visa Debit 4659-4228-4321-1937	973
1005	04-Aug-2018	Direct Debit 4659-2214-1882-1924	452

Above is the first five tuples taken from the Cycles to Share database.

Andrew Cooper

Fragment S₂

AccountNumber	RegistrationDate	FirstName	Surname	DateOfBirth	Email	Password	PhoneNumber	Address	CityCode	PostCode
1001	12-Jan-2018	John	Peacock	2-Oct-1989	John-Peacock@yahoo.com	john1data2	07921242538	16 Llantwit Road, Neath	NEA	SA11 8DP
1002	03-Feb-2017	Mary	Saunders	14-Jan-1992	Student.Mary@hotmail.co.uk	passwordz12	07263594756	33 Mansel Road, Bonymaen	SWA	SA1 7JS
1003	03-Apr-2018	Simon	Gates	14-Feb-1995	GatesLtd@hotmail.co.uk	mfeAfeEr1z*	01792633416	33 Jersey Road, Bonymaen	SWA	SA1 7BS
1004	20-May-2016	Tim	Wells	17-Sep-1985	Tim.Wells@msn.co.uk	asdfghjkl;	07663538238	99 Brookdale Road	SWA	SA1 7QS
1005	05-Aug-2018	Mike	Green	11-Jul-1995	MikeyGreen@hotmail.co.uk	Myp!ssw0r0	01792234152	38 Alister Street	NEA	SA11 2JS

Fragment S₂ can be distributed to areas that do not require any of the payment information, seeing as that will all be handled in Swansea.

Bibliography

- [1] B. Richardson, "What is the difference between Clustered and Non-Clustered Indexes in SQL Server?," SQLShackSkip to content, 28 Aug 2017. [Online]. Available: <https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/>. [Accessed 09 Mar 2019].
- [2] B. Wagner, "Clustered vs Nonclustered: What Index Is Right For My Data?," HackerMoon, 18 Dec 2017. [Online]. Available: <https://hackernoon.com/clustered-vs-nonclustered-what-index-is-right-for-my-data-717b329d042c>. [Accessed 09 Mar 2019].
- [3] M. Rouse, "Distributed Database," September 2018. [Online]. [Accessed 17 Mar 2019].