​**PROJECT -2**

**DATABASE FOUNDATIONS FOR BUSINESS ANALYTICS**

**Project Team:**

Yeshasvini Reddy Tirumala (ytr210001)

Anudeep Chowdary Boyapati (axb210083)

Vanisha Rodrigues (vxr210022)

Pawar Raybhan Sanjay (rxp210020)

Nokki John Paul (jnx190014)

**Professor:** Dr. Farzad Kamalzadeh

University of Texas at Dallas

Texas 75080

May 12, 2022

Table of Contents

[Part 1: Indexing and Query Timing 3](#_Toc103297058)

[Part 2: MongoDB and MQL 7](#_Toc103297059)

[Appendix: 11](#_Toc103297060)

# Part 1: Indexing and Query Timing

* 1. **List all the current indexes in your database and the columns they are associated with along with the index type**

**Table Airbnb-hosts:**

1. Key Name: host\_id – Primary Key, Unique: Yes, Index Type: BTREE (Clustered)
2. Key Name: host\_neighbourhood, Index Type: BTREE; NOTE: BTREE is generally the default index type.

**Table Airbnb\_host\_location:**

1. Key Name: host\_neighbourhood, Index Type: BTREE( Clustered )

**Table listings\_location:**

1. Key Name: zipcode, Unique: Yes, Index Type: BTREE (clustered)

**Table listings:**

1. Key Name: id, Index Type: BTREE( Clustered )
2. Key Name: listing\_id, Unique: Yes, Index Type: BTREE
3. Key Name: host\_id, Unique: Yes, Index Type: BTREE
4. Key Name: zipcode, Unique: Yes, Index Type: BTREE

**Table Property\_availability:**

1. Key Name: id, Index Type: BTREE ( Clustered )
2. Key Name: listing\_id, Unique: Yes, Index Type: BTREE

**Table property\_reviewer\_id:**

1. Key Name: reviewer\_id, Index Type: BTREE ( Clustered )

**Table property\_reviews\_id:**

1. Key Name: review\_id, Unique: Yes, Index Type: BTREE (Clustered)
2. Key Name: reviewer\_id, Index Type: BTREE

**Table property\_reviews:**

1. Key Name: review\_id, Unique: Yes, Index Type: BTREE (Clustered)
   1. **Explain what is in common between these columns (why these columns are indexed automatically by the database management system)**

We’ve noticed that all the primary keys and foreign key constraints are automatically indexed, including the attributes marked unique.

It’s our understanding that primary keys are automatically indexed by the database system, they create an implicit index. However foreign keys are necessarily not, they are considered. Foreign keys do not create indexes. Only alternate key constraints (UNIQUE) and primary key constraints create indexes automatically.

* 1. **Make a copy of your database and delete all the indexes there (you might need to delete foreign keys before you can delete some of the indexes) – now you have two databases: database A with indexes and database B without any indexes.**

We have made a copy of the initial database ‘londonan’ called ‘airbnblondon db’

We now have two databases. In the database B however, we had to delete the foreign keys to remove the indexes.

* 1. **Write at least 5 queries (with JOINs between your tables**

Queries attached to the appendix

* 1. **Execute and time these queries on both databases and report your findings (repeat timing for each query at least 10 times and average the times**

We have executed and timed these queries on both databases. Our findings are:

The average time of A: 0.021

The average time of B: 0.0314

Table Database A:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Queries** | 1st Run | 2nd Run | 3rd Run | 4th Run | 5th Run | 6th Run | 7th Run | 8th Run | 9th Run | 10th Run | Avg of 10 Runs |
| **Q1** | 0 | 0.016 | 0 | 0.016 | 0 | 0 | 0 | 0 | 0 | 0.015 | 0.0047 |
| **Q2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.016 | 0 | 0.015 | 0.0031 |
| **Q3** | 0 | 0 | 0 | 0 | 0 | 0.016 | 0.016 | 0 | 0 | 0.015 | 0.0047 |
| **Q4** | 0 | 0 | 0 | 0 | 0 | 0.016 | 0 | 0.016 | 0 | 0 | 0.0032 |
| **Q5** | 0 | 0 | 0 | 0.016 | 0.016 | 0 | 0.016 | 0 | 0.016 | 0 | 0.0064 |

Table Database B:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Queries** | 1st Run | 2nd Run | 3rd Run | 4th Run | 5th Run | 6th Run | 7th Run | 8th Run | 9th Run | 10th Run | Avg of 10 Runs |
| **Q1** | 0 | 0.016 | 0 | 0.016 | 0 | 0 | 0 | 0 | 0.016 | 0 | 0.0048 |
| **Q2** | 0 | 0 | 0.015 | 0.016 | 0.015 | 0 | 0 | 0.016 | 0 | 0 | 0.0062 |
| **Q3** | 0 | 0 | 0.016 | 0.016 | 0 | 0.016 | 0 | 0.015 | 0 | 0 | 0.0063 |
| **Q4** | 0.015 | 0 | 0.016 | 0 | 0.016 | 0 | 0 | 0.015 | 0 | 0.015 | 0.0077 |
| **Q5** | 0 | 0 | 0 | 0.016 | 0.016 | 0 | 0.016 | 0 | 0.016 | 0 | 0.0064 |

* 1. **Select some columns from database A (columns that are not already indexed) and create index on them**

Some columns that were selected from database A that are not already indexed are:

Key Name: host\_name, Type: BTREE, Unique: No

Key Name: host\_listings\_count, Type: BTREE, Unique: No

Key Name: host\_location, Type: BTREE, Unique: No

Key Name: property\_type, Type: BTREE, Unique: No

Key Name: roomt\_type, Type: BTREE, Unique: No

Key Name: reviewer\_name, Type: BTREE, Unique: No

Key Name: price, Type: BTREE, Unique: No

* 1. **Write a query for each column – the query should include the column in the WHERE clause in a condition**

Queries attached to the appendix

* 1. **Execute and time these queries on both databases and report your findings (repeat timing for each query at least 10 times and average the times)**

We have executed and timed these queries on both databases. Our findings are:

The average time of A: 0.0143

The average time of B: 0.0477

Table Database A:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Queries** | 1st Run | 2nd Run | 3rd Run | 4th Run | 5th Run | 6th Run | 7th Run | 8th Run | 9th Run | 10th Run | Avg of 10 Runs |
| **Q1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.015 | 0.016 | 0 | 0.0031 |
| **Q2** | 0.016 | 0 | 0 | 0 | 0.016 | 0 | 0 | 0 | 0 | 0 | 0.0032 |
| **Q3** | 0 | 0 | 0 | 0.016 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0016 |
| **Q4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.016 | 0 | 0 | 0.0016 |
| **Q5** | 0.016 | 0 | 0 | 0 | 0 | 0 | 0.016 | 0 | 0 | 0.016 | 0.0048 |

Table Database B:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Queries** | 1st Run | 2nd Run | 3rd Run | 4th Run | 5th Run | 6th Run | 7th Run | 8th Run | 9th Run | 10th Run | Avg of 10 Runs |
| **Q1** | 0.017 | 0 | 0.017 | 0.015 | 0.019 | 0 | 0.017 | 0.015 | 0.019 | 0.015 | 0.0115 |
| **Q2** | 0.016 | 0 | 0.016 | 0 | 0.017 | 0 | 0.017 | 0 | 0.017 | 0 | 0.0083 |
| **Q3** | 0.018 | 0.016 | 0 | 0.016 | 0.018 | 0.016 | 0.018 | 0.016 | 0 | 0 | 0.0118 |
| **Q4** | 0.017 | 0 | 0.017 | 0 | 0 | 0 | 0 | 0.017 | 0 | 0.017 | 0.0068 |
| **Q5** | 0.016 | 0 | 0.015 | 0 | 0.015 | 0 | 0.015 | 0 | 0.016 | 0.016 | 0.0093 |

* 1. **Make a conclusion based on your findings in this part**

1. Indexes are **special lookup tables** that database search engines employ to speed up data retrieval. An index is a reference to information in a table. It's like using an index in the document to find information quickly.
2. **SELECT** queries and **WHERE** clauses are sped up with an index. The columns indexed are faster when SELECT or WHERE clauses are used, as seen in the queries file. However, when we used the JOIN and INSERT commands, we found that they had no influence on other queries like **JOIN** or **INSERT**.
3. Indexes, like the UNIQUE constraint, can be unique in that they preclude duplicate entries in the column or combination of columns on which the index is applied. We've indexed columns with unique keys and/or names to see if there's a tiny speed boost, and we can estimate the impact on huge databases with more than 1GB of data. When an object is created, the database server creates implicit indexes. The database management system creates indexes automatically for main key and unique constraints.
4. While indexes are intended to speed up data retrieval, they can also be avoided. On small tables, indexes have no substantial impact. Indexes on columns with a lot of NULL values have a smaller impact.

# Part 2: MongoDB and MQL

**2.1. Explore your dataset and familiarize yourself with the dataset and its content**

The dataset is extracted from bikez.com. The website has one of the most comprehensive database of specifications of various motorcycles old and new. Although specifications of a motorcycle (statistics of each model) are the primary data available, motorcycle enthusiasts post questions regarding the motorcycle and others answer them. You can in some cases, find pictures, comments posted for each listing. You also can find financial information, find insurance quotes etc. As for the data, it has 38624 documents in the json file.

**2.2. Explain why it is better to use non-relational databases such as MongoDB**

Not all data is structured. And one such case is with the dataset we are using. Our data is stored in documents and each document has a unique identifier. The data doesn’t have a set schema. It contains a range of information and in different formats. We believe it is better to use non-relational databases such as MongoDB to work with because, it is easy to modify the values in specific fields without rewriting the entire document and without disrupting the existing structure. And since, they do not have a standard structure, it is easy to store even a new document type. These are shown in action in the queries.

**2.4. List some of the attributes of your database which are common among all documents.**

Model: Specifies the model of the motorcycle the document contains information about.

Year: The year in which the model is manufactured.

Category: Type of Motorcycle.

Displacement: The volume of all the cylinders in the engine. (Generally denoted in cc)

Engine type: The type of engine used in the motorcycle.

Torque: The rotational force that is supplied to the wheels from the engine.

Bore x Stroke: Diameter of the piston and the depth it travels.

Cooling system: The type of cooling system employed.

Gear Box: The type or the number of gears employed.

Front Suspension: The type of suspension used in the front.

Rear Suspension: The type of suspension used in the rear.

Front Brake: The type of the brake used and/or the size and the number of the disc(s) used in the front.

Rear Brake: The type of the brake used and/or the size and the number of the disc(s) used in the rear.

Seat height: The height of the seat from the ground.

Wheelbase: The distance between the center pf the front wheel to the center of the rear wheel.

Fuel Capacity: The size of the fuel tank.

Color options: The colors in which the motorcycle is available in

**2.4.1. For these fields, provide some of the values they contain in the database**

Model: “Kawasaki Estrella Special Edition”, “AJS Sorvio 125”, “Honda CBR 650F”.

Year: “2016”, “2017”, “2018”, “2019”, “2020”

Category: “Naked bike”, “Custom/ Cruiser”, “Enduro/Offroad”, “Scooter”, ”ATV”, “Sport”.

Displacement: “249.0 ccm (15.19 Cubic Inches)”, “998.0 ccm (60.90 cubic inches)”

Engine type: “Single Cylinder, four-stroke”, “V2, four-stroke”, “In-line four, four stroke”.

Torque: “66.0 Nm (6.7 kgf-m or 48.7 ft.lbs) @ 5500 RPM”, “55.0 Nm (5.6 kgf-m or 40.6 ft.lbs) @ 5500 RPM”, “39.3 Nm (4.0 kgf-m or 29.0 ft.lbs)”

Bore x Stroke: “76.0 x 55.0 mm (3.0 x 2.2 inches)”, “95.0 x 63.4 mm (3.7 x 2.5 inches)”.

Cooling system: “Liquid”, “Air”, “Oil & Air”.

Gear Box: “6-speed”, “5-speed”, “Automatic”, “4-speed”.

Front Suspension: “Telescopic fork”, “Telescopic, coil spring, oil damped”

Rear Suspension: “Monoshock”, “Twin shocks”, “Dual shocks”, “Swing arm”

Front Brake: “Single Disc”, “Expanding brake (drum brake)”, “Single disc. Two piston calipers”

Rear Brake: “Single Disc”, “Expanding brake (drum brake)”, “Single disc. ABS”

Seat height: “810 mm (31.9 inches) If adjustable, lowest setting.”, “790 mm (31.1 inches) If adjustable, lowest setting.”, “770 mm (30.3 inches) If adjustable, lowest setting.”

Wheelbase: “1689 mm (66.5 inches)”, “1380 mm (54.3 inches), “1280 mm (50.4 inches)

Fuel Capacity: “12.00 litres (3.17 gallons), “13.00 litres (3.43 gallons)”, “14.00 litres (3.70 gallons)”

Color options: “Red”, “Orange/White/Black”, “Lime Green”, “Black”, “Blue”, “Black/Silver”, “Yellow”.

**2.5. List some of the attributes of your database which are not common among all the documents**

Valves per cylinder: The number of valves used in each cylinder

Exhaust System: The type of exhaust system used.

Factory Warranty: The standard warranty offered by the manufacturer

Carrying Capacity: If there is any built-in storage capacity.

Oil Capacity: The capacity of engine oil that it can hold.

**2.5.1. For these fields, provide some of the values they contain in the database**

Valves per cylinder: “4”, ”3”, ”2”, ”5”.

Exhaust System: “Titanium silencer”, “Black megaphone steel”, “Exhaust TEC enhanced centrally mounted”.

Factory Warranty: “One year, Unlimited Mileage, Transferable”, “12 month Unlimited mileage limited warranty”, “24 month Unlimited mileage”

Carrying Capacity: “Saddle bag”, “Under Seat Storage”

Oil Capacity: “1.00 litres (0.07 quarts)”, “3.50 litres (0.23 quarts), “4.90 litres (0.32 quarts)

**2.6. Write at least 5 queries using the key-value pairs you found in the previous steps to narrow down the result**

1. {$and:[{"Category" : 'Sport'}, {$or:[{"Cooling system": {$eq: "Oil & air"}}, {"Cooling system": {$eq: "Liquid"}}]},{"Frame type":"Diecast aluminium twin-spar"}]}
2. {$and:[{$or : [{"Category" : "Enduro / offroad"}, {"Category": "Sport"}]}, {$or:[{"Compression": {$eq: "12.5:1"}}, {"Compression": {$eq: "13.6:1"}}]}, {'Cooling system':'Liquid'}, {'Emission details': 'Euro 4'}]}
3. {$and:[{$and : [{"Fuel consumption":"6.50 litres/100 km (15.4 km/l or 36.19 mpg)"}, {'Ignition':'Electronic'}]}, {"Starter":"Electric & kick"}]}
4. {$and :[{$or: [{"Fuel capacity":"14.50 litres (3.83 gallons)"}, {"Fuel capacity":"12.00 litres (3.17 gallons)"}]},{"Front brakes":"Single disc. Hydraulic"},{"Year":"2013"}]}
5. {$and: [{$or: [{"Frame type":"Diamond"}, {"Frame type":"Aluminium dual beam chassis with removable seat support subframe"}]},{"Gearbox":"6-speed"}, {"Starter":"Electric"}, {"Year":"2021"}]}

**2.7. Write at least 5 update queries to update some of the values in your database**

1. db.bikez.insert({"Model":"Kawasaki Ninja","Year":"2022","Category":"Super","Price as new": "$5000","Gearbox":"6-speed manual","Color options":"Black/White"})
2. db.bikez.insert({"Model":"Pulsar 250","Year":"2021","Price as new": "$2500","Gearbox":"6-speed manual","Color options":"Black and Red","Starter": "Kick"})
3. db.bikez.insert({"Model":"Yamaha YZF-R3","Year":"2020","Category":"Super","Price as new": "$3800","Gearbox":"6-speed Automatic","Color options":"Blue","Starter":"Electric","Ignition":"CDI"})
4. db.bikez.insert({"Model":"KTM Duke","Year":"2022","Category":"Super","Price as new": "$5000","Color options":"Black and white/Blue and white"})
5. db.bikez.insert({"Model":"Iron 883","Year":"2022","Category":"Luxury","Price as new": "$11249","Gearbox":"6-speed manual","Color options":"Black","Engine type":"Air Cooled, Evolution"})

**2.8. Write at least 5 queries to insert new documents into your database**

1. db.bikez.updateMany({"Year":"2022","Category":"Luxury" },{$set: { "Price as new":"$12000"}})
2. db.bikez.updateMany({"Year":"2022","Gearbox":"6-speed manual"},{$set: { "Starter":"Electric"}})
3. db.bikez.updateMany({ "Engine type": "Single cylinder, four-stroke", "Gearbox":"5-speed"},{$set: { "Fuel control": "Overhead Valves (OHV)"}})
4. db.bikez.updateMany({"Year":"2021",Model:"AJP PR7" },{$set: { "Color options":"White, Black/Blue"}})
5. db.bikez.updateMany({"Year":"2021","Category":"Sport"},{ $set: { "Engine details":"Revelation engine"} })

**2.9. Write at least 5 delete queries to remove documents from your database**

1. db.bikez.deleteMany ({"Model":"Iron 883","Year":"2022","Category":"Luxury","Price as new": "$12000"})
2. db.bikez.deleteOne({"Model":"KTM Duke"})
3. db.bikez.deleteOne({"Year":"2021","Category":"Sport"})
4. db.bikez.deleteMany({"Year":"2021","Category":"Sport"})
5. db.bikez.deleteMany({"Model":"Aprilia RX 125","Engine details":null})

# Appendix:

**1.4**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application, table

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**1.7**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

* 1. Graphical user interface, text, application, email

     Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated