TEAM 106 - DATA BUDS
STAGE 3
Database Implementation and Indexing

Part 1: Data Definition Language (DDL) Commands and Queries

Courses

```sql
CREATE TABLE Courses (
    CourseId INT PRIMARY KEY,
    Topic VARCHAR(100) NOT NULL
);
```

```
mysql> SELECT COUNT(*) FROM Courses;
+----------+
| COUNT(*) |
+----------+
|      433 |
+----------+
1 row in set (0.00 sec)
```

User

```sql
CREATE TABLE User (
    UserId INT  PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    Password VARCHAR(50) NOT NULL
);
```

```
mysql> SELECT COUNT(*) FROM User;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

Tutor

```sql
CREATE TABLE Tutor (
    TutorId INT PRIMARY KEY,
    Topic VARCHAR(50) NOT NULL
);
```

```
mysql> SELECT COUNT(*) FROM Tutor;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.01 sec)
```

Problems:

```sql
CREATE TABLE Problems (
    Questions VARCHAR(50) PRIMARY KEY,
    Answers VARCHAR(50) NOT NULL,
    Difficulty INT,
    Topic VARCHAR(50) NOT NULL
);
```

```
mysql> SELECT Count(*) From Problems;
+----------+
| Count(*) |
+----------+
|    44515 |
+----------+
1 row in set (0.00 sec)
```

Info:

```sql
CREATE TABLE Info (
    InfoID INT PRIMARY KEY ,
    Topic varchar(50) NOT NULL,
    WebURL varchar(250),
    ResourceType varchar(50)
);
```

```
mysql> SELECT Count(*) FROM Info;
+----------+
| Count(*) |
+----------+
|       65 |
+----------+
1 row in set (0.01 sec)
```

Advanced Queries:

```
mysql> SELECT C.Topic, AVG(P.Difficulty) AS AvgDifficulty
    -> FROM Courses C
    -> JOIN Problems P ON C.Topic = P.Topic
    -> GROUP BY C.Topic
    -> LIMIT 15;
+-----------------------------------------------------------------+---------------+
| Topic                                                           | AvgDifficulty |
+-----------------------------------------------------------------+---------------+
| Advanced Topics in Natural Language Processing                  |        6.6071 |
| Advanced Topics in Security, Privacy, and Machine Learning      |        5.2308 |
| Applied Machine Learning                                        |        5.5588 |
| Artificial Intelligence                                         |        4.9310 |
| Computer Vision                                                 |        4.5758 |
| Data Science Discovery                                          |        5.9737 |
| Deep Learning                                                   |        5.7333 |
| Deep Learning for Computer Vision                               |        5.1071 |
| Introduction to Computer Science II                             |          NULL |
| Introduction to Data Mining                                     |        5.6154 |
| Machine Learning                                                |        6.0800 |
| Machine Learning for Bioinformatics                             |        5.7500 |
| Machine Learning for Signal Processing                          |        6.1143 |
| Modeling and Learning in Data Science                           |        5.2647 |
| Natural Language Processing                                     |        5.3030 |
+-----------------------------------------------------------------+---------------+
15 rows in set (1.00 sec)
```

```
mysql> SELECT c.CourseId, c.Topic, q.Question
    -> FROM Courses c
    -> JOIN Problems q ON c.Topic = q.Topic
    -> WHERE q.Question LIKE '%A%'
    -> LIMIT 15;
+----------+-----------------------------------------------+-----------------------------------------------------------------------------+
| CourseId | Topic                                         | Question                                                                    |
+----------+-----------------------------------------------+-----------------------------------------------------------------------------+
|    74740 | Advanced Topics in Natural Language Processing | What is the definition of standardized problem?                             |
|    74740 | Advanced Topics in Natural Language Processing | What is a grid world problem?                                               |
|    74740 | Advanced Topics in Natural Language Processing | What does the King Midas problem inspire us?                                |
|    74740 | Advanced Topics in Natural Language Processing | What is the difference of omniscience from rationality?                     |
|    74740 | Advanced Topics in Natural Language Processing | If A is a child node of B                                                   |
|    74740 | Advanced Topics in Natural Language Processing | What is node?                                                               |
|    74740 | Advanced Topics in Natural Language Processing | What does the book Preceptrons (1969) mentioned?                            |
|    74740 | Advanced Topics in Natural Language Processing | What is the benefits of Iterative-deepening A* search?                      |
|    74740 | Advanced Topics in Natural Language Processing | How transition model and sensor model function in model-based reflex agent? |
|    74740 | Advanced Topics in Natural Language Processing | What does Space complexity evaluates specificly on an algorithm's performance? |
|    74740 | Advanced Topics in Natural Language Processing | How can independence be ascertained?                                        |
|    74740 | Advanced Topics in Natural Language Processing | What is knowledge engineer?                                                 |
|    74740 | Advanced Topics in Natural Language Processing | What is search?                                                             |
|    74740 | Advanced Topics in Natural Language Processing | What are dead ends?                                                         |
|    74740 | Advanced Topics in Natural Language Processing | When is a sentence valid?                                                   |
+----------+-----------------------------------------------+-----------------------------------------------------------------------------+
15 rows in set (0.00 sec)
```

```
mysql> SELECT I.WebURL FROM Info I natural join (SELECT C.Topic FROM Courses C natural join Problems P  GROUP BY C.Topic HAVING ROUND(AVG(P.Difficulty), 3) = 4.931) AS AvgDiffTopics;

+-----------------------------------------------------------------------------------------------------------------+
| WebURL                                                                                                          |
+-----------------------------------------------------------------------------------------------------------------+
| https://www.youtube.com/watch?v=XXqr0cPQMZA&ab_channel=Staffbase                                                |
| https://www.youtube.com/watch?v=Yq0QkCxoTHM                                                                     |
| https://www.analyticsvidhya.com/blog/2021/06/a-beginners-guide-to-artificial-intelligence/                      |
| https://towardsdatascience.com/the-future-of-artificial-intelligence-ff64f4a5bce7                               |
| https://builtin.com/artificial-intelligence                                                                     |
| https://www.forbes.com/sites/bernardmarr/2020/07/06/the-top-5-business-benefits-of-artificial-intelligence-ai/ |
| https://www.ibm.com/cloud/learn/what-is-artificial-intelligence                                                 |
| https://www.sas.com/en_us/insights/analytics/what-is-artificial-intelligence.html                               |
| https://www.microsoft.com/en-us/ai/ai-lab                                                                       |
| https://www.nvidia.com/en-us/deep-learning-ai/what-is-deep-learning/                                            |
| https://www.techradar.com/news/what-is-ai-artificial-intelligence-explained                                     |
| https://www.mckinsey.com/featured-insights/artificial-intelligence                                              |
| https://www.technologyreview.com/2021/10/27/1038498/artificial-intelligence-explained/                          |
| https://www.datacamp.com/community/blog/what-is-artificial-intelligence                                         |
| https://hbr.org/2019/07/artificial-intelligence-for-the-real-world                                              |
+-----------------------------------------------------------------------------------------------------------------+
15 rows in set (0.90 sec)
```

```
mysql> SELECT DISTINCT T.TutorId FROM Tutor T natural join Problems P  JOIN (SELECT C.Topic, ROUND(AVG(P.Difficulty), 3) AS AvgDifficulty FROM Courses C JOIN Problems P ON
 BY C.Topic HAVING AvgDifficulty = 4.9310) AS AvgDiffTopics ON P.Topic = AvgDiffTopics.Topic;
+---------+
| TutorId |
+---------+
|     104 |
|     384 |
|     900 |
+---------+
3 rows in set (0.90 sec)
```

Indexing:

Indexing for Query:

SELECT c.CourseId, c.Topic, q.Question FROM Courses c JOIN Problems q ON c.Topic = q.Topic WHERE q.Question LIKE '%A*%';

Base Cost:
| -> Filter: (c.Topic = q.Topic) (cost=2212.38 rows=2165) (actual time=0.430..0.551 rows=48 loops=1)
    -> Inner hash join (<hash>(c.Topic)=<hash>(q.Topic)) (cost=2212.38 rows=2165) (actual time=0.428..0.537 rows=48 loops=1)
        -> Table scan on c (cost=0.10 rows=433) (actual time=0.017..0.089 rows=433 loops=1)
        -> Hash
            -> Filter: (q.Question like '%A*%') (cost=46.75 rows=50) (actual time=0.073..0.385 rows=5 loops=1)
                -> Table scan on q (cost=46.75 rows=450) (actual time=0.020..0.193 rows=450 loops=1)
|

Indexes I tried:

CREATE INDEX question_topic ON Problems(Question(100));

CREATE INDEX question_topic ON Problems(Topic)

CREATE INDEX question_topic ON Courses(Topic)

Explanation:

I tried a variety of different indexes for this particular query. I first attempted to Create an index for Problems(Questions(100)). The logic behind this was to minimize the cost of performing the LIKE clause for where the questions is accessed because I thought the database would be able to quickly "look up" A*. This was an incorrect assumption though as it had no effect on the actual cost likely because the individual entries still have to be fully scanned every time to find whether there exists the substring A* anywhere in it, not just at the beginning or at a specific point in the string.

I then attempted to index on the Problems(Topic), but this also led to no performance increase. My hypothesis was that, since a large portion of the cost (in fact most of it) came from the join table clause, the cost would be heavily reduced by creating an index that the database would be able to immediately look up rather than having to scan the full tables every single time. This turned out to be false though, and when I examined the table sizes I realized that the problems table is many times larger than the Courses table. This likely means that rather than the database trying to join Problems to Courses; it was joining courses to problems in order to avoid excessive comparisons. This meant that indexing Courses rather than problems would likely save time, which is what I did next.

There was an immediate performance improvement (as can be seen below) when I created the index for the Courses(topic). This meant that by adding indexes to the Courses table the database was able to find matches faster. This change ended up being the only index that I added. When I attempted to add other indexes on top of this one,

such as the previous two it only led to performance decreases, either in terms of cost or time.

Final Index:

```
mysql> CREATE INDEX question_topic ON Courses(Topic);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT c.CourseId, c.Topic, q.Question  FROM
Courses c  JOIN Problems q ON c.Topic = q.Topic  WHERE q.Question LIKE '%A*%';
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------+
| EXPLAIN
|
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------+
| -> Nested loop inner join  (cost=85.87 rows=243) (actual time=0.091..0.473 rows=48
loops=1)
    -> Filter: ((q.Question like '%A*%') and (q.Topic is not null))  (cost=46.75 rows=50)
(actual time=0.075..0.422 rows=5 loops=1)
        -> Table scan on q  (cost=46.75 rows=450) (actual time=0.020..0.215 rows=450
loops=1)
    -> Covering index lookup on c using question_topic (Topic=q.Topic)  (cost=0.31
rows=5) (actual time=0.006..0.009 rows=10 loops=5)
 |
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set (0.00 sec)
```

Indexing for Query:

Select C.Topic, AVG(P.Difficulty) AS AVGDifficulty FROM Courses C Join Problems P on C.topic = P.Topic Group By C.Topic

```
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=2.156..2.157 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=2.155..2.155 rows=15 loops=1)
        -> Filter: (P.Topic = C.Topic)  (cost=19531.96 rows=19485) (actual
time=0.196..1.156 rows=2303 loops=1)
            -> Inner hash join (<hash>(P.Topic)=<hash>(C.Topic))  (cost=19531.96
rows=19485) (actual time=0.194..0.513 rows=2303 loops=1)
                -> Table scan on P  (cost=0.02 rows=450) (actual time=0.027..0.125 rows=450
loops=1)
                -> Hash
                    -> Table scan on C  (cost=44.05 rows=433) (actual time=0.032..0.095
rows=433 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
```

---+
1 row in set (0.00 sec)

Indexes attempted
CREATE INDEX question_topic ON Courses(Topic);
CREATE INDEX question_topic2 ON Problems(Topic);

Create Index difficulty ON Problems(Difficulty);

I started out by trying a similar tactic to the last query by first creating an index for the Courses topic in order to reduce the cost of the join. This turned out to be very fruitful as it reduced the massive cost of 19531 for the join clause down to only 398. This likely meant that there were far more problems being pulled than in the previous query thus magnifying the impact of the cost saving "look up table" indexing

Once again I attempted to index on the Problems(Topic), but this led to no performance increase. Double indexing on the same operation seems to have no real benefit for the actual performance. As it is only performing one comparison from the smaller table to the larger table, rather than doing two comparisons in both directions. This makes sense as it would be very cost inefficient to do some form of dual checking.

Lastly I attempted to index on the difficulty of the problems which turned out to have a small positive impact on time but had no impact on cost. My only rational is that while it might not improve the cost as all difficulties have to be accessed and scanned anyway, it could be attributed to faster lookups of specific difficulty values. Alternatively it might just be a thing with google cloud

Final Index:
CREATE INDEX question_topic ON Courses(Topic);
CREATE INDEX question_topic3 ON Problems(Difficulty);

```
mysql> CREATE INDEX question_topic ON Courses(Topic);
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> CREATE INDEX question_topic3 ON Problems(Difficulty);
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Explain Analyze Select C.Topic, AVG(P.Difficulty) AS AVGDifficulty
FROM Courses C Join AI_Questions P on C.topic = P.Topic Group By C.Topic
    -> ;
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------+
| EXPLAIN
|
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=3.019..3.021 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=3.018..3.018 rows=15 loops=1)
        -> Nested loop inner join  (cost=398.89 rows=2189) (actual time=0.060..2.007
rows=2303 loops=1)
```

```
            -> Filter: (P.Topic is not null)  (cost=46.75 rows=450) (actual
time=0.042..0.161 rows=450 loops=1)
                -> Table scan on P  (cost=46.75 rows=450) (actual time=0.041..0.136
rows=450 loops=1)
            -> Covering index lookup on C using question_topic (Topic=P.Topic)
(cost=0.30 rows=5) (actual time=0.002..0.004 rows=5 loops=450)
 |
+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```