Assignment/Explore Query Planning and Indexing

Balakrishna,D

Summer Full 2023

Establishing DB Connection

```r
library(DBI)
library(RSQLite)

# Connect to the SQLite database
conn <- dbConnect(RSQLite::SQLite(), "sakila.db")
```

Question 1

```r
# Drop user-defined Indexes
dbExecute(conn, "DROP INDEX TitleIndex;")
```

```
## [1] 0
```

```r
# Get the number of films per language
query <- "SELECT l.name, COUNT(*) AS num_films FROM film f LEFT JOIN language l ON
f.language_id=l.language_id GROUP BY f.language_id;"
results <- dbGetQuery(conn, query)

# Print the results
print(results)
```

```
##      name num_films
## 1 English      1000
```

Question 2

```r
# Get the query plan for the query
query <- "EXPLAIN QUERY PLAN SELECT l.name, COUNT(*) AS num_films
          FROM film f LEFT JOIN language l ON f.language_id=l.language_id
          GROUP BY f.language_id;"

query_plan <- dbGetQuery(conn, query)

# Print the query plan
print(query_plan)
```

```
##   id parent notused                                                  detail
## 1  7      0       0                                                  SCAN f
## 2  9      0       0 SEARCH l USING INTEGER PRIMARY KEY (rowid=?) LEFT-JOIN
## 3 14      0       0                            USE TEMP B-TREE FOR GROUP BY
```

Question 3

The following SQL query will return the title, category name and length of the film with the title “ZORRO ARK”:

```
# Get the title, category name and length of the film with the title "ZORRO ARK"
query3 <- "SELECT f.title AS title, c.name AS category, f.length AS length
          FROM film f JOIN film_category fc ON f.film_id=fc.film_id JOIN
          category c ON fc.category_id=c.category_id WHERE title = 'ZORRO ARK';"
results <- dbGetQuery(conn, query3)

# Get the time taken for execution
query_3_time <- system.time(dbGetQuery(conn, query3))

# Print the results
print(results)
```

```
##       title category length
## 1 ZORRO ARK   Comedy     50
```

Question 4

The query plan for the query in step 3 is as follows:

```
# Get the query plan for the query
query <- "EXPLAIN QUERY PLAN SELECT f.title AS title, c.name AS category,
          f.length AS length FROM film f JOIN film_category fc
          ON f.film_id=fc.film_id JOIN category c ON
          fc.category_id=c.category_id WHERE title = 'ZORRO ARK';"
query_plan <- dbGetQuery(conn, query)

# Print the query plan
print(query_plan)
```

```
##   id parent notused
## 1  4      0       0
## 2  6      0       0
## 3  9      0       0
##                                                           detail
## 1 SCAN fc USING COVERING INDEX sqlite_autoindex_film_category_1
## 2                  SEARCH c USING INTEGER PRIMARY KEY (rowid=?)
## 3                  SEARCH f USING INTEGER PRIMARY KEY (rowid=?)
```

Question 5

The following SQL statement will create a user-defined index called “TitleIndex” on the column TITLE in the table FILM:

```
# Create a user-defined index called "TitleIndex" on the column `TITLE`
# in the table `FILM`

query <- "CREATE INDEX TitleIndex ON film (title);"
dbExecute(conn, query)
```

```
## [1] 0
```

Question 6

```
# Re-run the query from step 3 now that you have an index

query6 <- "SELECT f.title AS title, c.name AS category, f.length AS length
            FROM film f JOIN film_category fc ON f.film_id=fc.film_id
            JOIN category c ON fc.category_id=c.category_id
            WHERE title = 'ZORRO ARK';"
results <- dbGetQuery(conn, query6)

# Get the time taken for execution
query_6_time <- system.time(dbGetQuery(conn, query6))

# Print the results
print(results)
```

```
##       title category length
## 1 ZORRO ARK   Comedy     50
```

```
query <- "EXPLAIN QUERY PLAN SELECT f.title AS title, c.name AS category,
          f.length AS length FROM film f JOIN film_category fc
          ON f.film_id=fc.film_id JOIN category c
          ON fc.category_id=c.category_id
          WHERE title = 'ZORRO ARK';"
query_plan <- dbGetQuery(conn, query)

# Print the query plan
print(query_plan)
```

```
##   id parent notused
## 1  5      0       0
## 2 10      0       0
## 3 14      0       0
##                                                                         detail
## 1                                   SEARCH f USING INDEX TitleIndex (title=?)
## 2 SEARCH fc USING COVERING INDEX sqlite_autoindex_film_category_1 (film_id=?)
## 3                                SEARCH c USING INTEGER PRIMARY KEY (rowid=?)
```

Question 7

The query plans for both query(4) and query(6) are slightly different. The difference is notably scene in the detail column of the plans. In query(4), the table was searched using the primary key, whereas after creating the index, we can see in the query plan of query(6) that the table is searched using the index we defined.

Question 8

Execution times for both query(3) and query(6):

```
# Print the execution times
print("Query 3: ")
```

```
## [1] "Query 3: "
```

```
print(query_3_time)
```

```
##    user  system elapsed
##       0       0       0
```

```
print("Query 6: ")
```

```
## [1] "Query 6: "
```

```
print(query_6_time)
```

```
##    user  system elapsed 
##   0.001   0.000   0.001 
```

Here we can clearly see that indexing improves execution times for the same results which are obtained from the same queries.The elapsed time has clearly decreased.

Question 9

```
# Get the title, language and length of all films with the word "GOLD" with any
# capitalization in its name

query <- "SELECT f.title AS title, l.name AS language, f.length AS length
          FROM film f LEFT JOIN language l ON f.language_id=l.language_id
          WHERE title LIKE '%GOLD%';"
results <- dbGetQuery(conn, query)

# Print the results
print(results)
```

```
##                     title language length
## 1         ACE GOLDFINGER  English     48
## 2   BREAKFAST GOLDFINGER  English    123
## 3             GOLD RIVER  English    154
## 4 GOLDFINGER SENSIBILITY  English     93
## 5        GOLDMINE TYCOON  English    153
## 6             OSCAR GOLD  English    115
## 7   SILVERADO GOLDFINGER  English     74
## 8             SWARM GOLD  English    123
```

Question 10

Here we get the query plan for query(9):

```
# Get the query plan for the query
query <- "EXPLAIN QUERY PLAN SELECT f.title AS title, l.name AS language,
          f.length AS length FROM film f LEFT JOIN language l
          ON f.language_id=l.language_id WHERE title LIKE '%GOLD%';"
query_plan <- dbGetQuery(conn, query)

# Print the query plan
print(query_plan)
```

```
##   id parent notused                                                   detail
## 1  3      0       0                                                   SCAN f
## 2  8      0       0 SEARCH l USING INTEGER PRIMARY KEY (rowid=?) LEFT-JOIN
```