Assignment / Explore Query Planning and Indexing

Chandresh Lokesha

Spring 2024

```
library(RSQLite)

# Step 1: Connect to the SQLite database
conn <- RSQLite::dbConnect(RSQLite::SQLite(), dbname = "sakila.db")
```

```
res <- dbGetQuery(conn, "SELECT * from FILM")
print(res)
```

Question 1

```
# Step 2: Retrieve the list of user-defined indexes
indexes <- RSQLite::dbGetQuery(conn,
                               "SELECT name FROM sqlite_master
                               WHERE type = 'index'
                               AND tbl_name != 'sqlite_sequence'
                               AND sql NOT LIKE '%UNIQUE%'
                               AND sql NOT LIKE '%PRIMARY KEY%'")

# Step 3: Drop each user-defined index
if (nrow(indexes) > 0) {
  for (i in 1:nrow(indexes)) {
    RSQLite::dbExecute(conn, paste0("DROP INDEX IF EXISTS ", indexes$name[i]))
  }
}

res <- dbGetQuery(conn, "SELECT l.name, count(*) from FILM f, LANGUAGE l
                  where l.language_id = f.language_id GROUP BY l.language_id")

print(res)
```

```
##      name count(*)
## 1 English     1000
```

Question 2

```
query <- "SELECT l.name, count(*) AS film_count
          FROM FILM f
          JOIN LANGUAGE l ON l.language_id = f.language_id
          GROUP BY l.name"
plan <- RSQLite::dbGetQuery(conn, paste("EXPLAIN QUERY PLAN", query))
print(plan)
```

```
##   id parent notused                                       detail
## 1  7      0       0                                       SCAN f
## 2  9      0       0 SEARCH l USING INTEGER PRIMARY KEY (rowid=?)
## 3 12      0       0                 USE TEMP B-TREE FOR GROUP BY
```

Question 3

```
query3 <- "SELECT f.title, f.length, c.Name AS film_category
          FROM FILM f
          JOIN FILM_CATEGORY fc on fc.film_Id = f.film_Id
          JOIN Category c on fc.category_id = c.category_id
          where f.title = 'ZORRO ARK'"

res <- RSQLite::dbGetQuery(conn, query)
print(res)
```

```
##      name film_count
## 1 English       1000
```

Question 4

```
plan <- RSQLite::dbGetQuery(conn, paste("EXPLAIN QUERY PLAN", query3))
print(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

```
RSQLite::dbExecute(conn, "CREATE INDEX IF NOT EXISTS TitleIndex ON FILM (TITLE)")
```

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

Question 6

```r
plan <- RSQLite::dbGetQuery(conn, paste("EXPLAIN QUERY PLAN", query3))
print(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 Comments and differences

Yes, they are different.Looking at the query plan on the film table details we can say that indxing is being used in the later. As you can see before creating index SEARCH on f is performed using primary key, and after creating index search f is based on TitleIndex

Question 8

Measure Time without Index Title

```r
dbExecute(conn, "DROP INDEX IF EXISTS TitleIndex;")
```

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

```r
bt <- Sys.time()
execQuery <- function ()
{
 res <- RSQLite::dbSendQuery(conn, query3)
}

et <- Sys.time()

t.which <- et - bt

cat("Time elapsed: ", round((t.which),3), " sec")
```

```
## Time elapsed:  0.001  sec
```

Measure Time with Index title

```r
RSQLite::dbExecute(conn, "CREATE INDEX IF NOT EXISTS TitleIndex ON FILM (TITLE)")
```

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

```r
bt <- Sys.time()
execQuery <- function ()
{
 res <- RSQLite::dbSendQuery(conn, query3)
}

et <- Sys.time()

t.which <- et - bt

cat("Time elapsed: ", round((t.which),3), " sec")
```

```
## Time elapsed:  0  sec
```

Report: time taken for the entire expression to execute

As you can see the indexing has reduced time to execute

Question 9

```r
query1 <- "SELECT l.name, f.title, f.length
            from FILM f, LANGUAGE l
            where l.language_id = f.language_id and title LIKE '%[Gg][Oo][Ll][Dd]%';"


res <- dbGetQuery(conn, query1)
```

Question 10

```r
plan <- RSQLite::dbGetQuery(conn, paste("EXPLAIN QUERY PLAN", query1))
print(plan)
```

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

There is no difference i.e it did not use the index (as you can see in in the query plan deatils 'SCAN f' is performed).This is because in the query plan 'LIKE' operator is used and it will not make use of indexing