Assignment / Explore Query Planning and Indexing

Zongyu WU

Spring 2024

Setup connection.

```
library(RSQLite)
dbcon <- dbConnect(RSQLite::SQLite(), "sakila.db")
```

Question 1

```
# Drop index
dbExecute(dbcon, "drop index if exists TitleIndex")
```

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

```
query <- "select l.NAME as LanguageName, count(f.FILM_ID) as NumberOfFilms
  from LANGUAGE l join FILM f on l.LANGUAGE_ID = f.LANGUAGE_ID
  group by l.LANGUAGE_ID"
res <- dbGetQuery(dbcon, query)
print(res)
```

```
##   LanguageName NumberOfFilms
## 1      English          1000
```

Question 2

```
query <- "EXPLAIN QUERY PLAN
  select l.NAME as LanguageName, count(f.FILM_ID) as NumberOfFilms
  from LANGUAGE l join FILM f on l.LANGUAGE_ID = f.LANGUAGE_ID
  group by l.LANGUAGE_ID"
res <- dbGetQuery(dbcon, query)
print(res)
```

```
##   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

```
before3 <- Sys.time()
query <- "select f.TITLE, c.NAME, f.LENGTH
  from FILM f join FILM_CATEGORY fc on f.FILM_ID = fc.FILM_ID
  join CATEGORY c on c.CATEGORY_ID = fc.CATEGORY_ID
  where TITLE =  \"ZORRO ARK\""
```

```r
res <- dbGetQuery(dbcon, query)
after3 <- Sys.time()
print(res)
```

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

Question 4

```r
query <- "EXPLAIN QUERY PLAN
  select f.TITLE, c.NAME, f.LENGTH
  from FILM f join FILM_CATEGORY fc on f.FILM_ID = fc.FILM_ID
  join CATEGORY c on c.CATEGORY_ID = fc.CATEGORY_ID
  where TITLE =  \"ZORRO ARK\""
res <- dbGetQuery(dbcon, query)
print(res)
```

```
##   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

```r
query <- "create index if not exists TitleIndex on FILM(TITLE)"
res <- dbExecute(dbcon, query)
print(res)
```

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

Question 6

```r
before6 <- Sys.time()
query <- "select f.TITLE, c.NAME, f.LENGTH
  from FILM f join FILM_CATEGORY fc on f.FILM_ID = fc.FILM_ID
  join CATEGORY c on c.CATEGORY_ID = fc.CATEGORY_ID
  where TITLE =  \"ZORRO ARK\""
res <- dbGetQuery(dbcon, query)
after6 <- Sys.time()
print(res)
```

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

```r
query <- "EXPLAIN QUERY PLAN
  select f.TITLE, c.NAME, f.LENGTH
  from FILM f join FILM_CATEGORY fc on f.FILM_ID = fc.FILM_ID
  join CATEGORY c on c.CATEGORY_ID = fc.CATEGORY_ID
  where TITLE =  \"ZORRO ARK\""
```

```r
res <- dbGetQuery(dbcon, query)
print(res)
```

```
##   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

It's not the same. After adding the index, the search on f is changed from using integer primary key to using index TitleIndex. The scan on fc using covering index is also changed to a search using covering index. The query plan will show in the detail that it used the index. By adding an index, related plans are changed from scan to search. And search on that specific column is using that index for searching.

Question 8

I added the time calculation function in Question 3 and Question 6 to save the code repeat. Here I will print the results from the above two questions.

```r
cat("Question 3 time elapsed: ", round((after3 - before3) ,3), " sec\n")
```

```
## Question 3 time elapsed:  0.002  sec
```

```r
cat("Question 6 time elapsed: ", round((after6 - before6) ,3), " sec")
```

```
## Question 6 time elapsed:  0.001  sec
```

As we can see, the time is reduced. The amount of reduce varies though. It may be the same query is parsed in different ways.

Question 9

```r
query <- "select f.TITLE, l.NAME, f.LENGTH
  from FILM f join LANGUAGE l on f.LANGUAGE_ID = l.LANGUAGE_ID
  where lower(f.TITLE) like '%gold%'"
res <- dbGetQuery(dbcon, query)
print(res)
```

```
##                    title    name 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

```
query <- "EXPLAIN QUERY PLAN
  select f.TITLE, l.NAME, f.LENGTH
  from FILM f join LANGUAGE l on f.LANGUAGE_ID = l.LANGUAGE_ID
  where lower(f.TITLE) like '%gold%'"
res <- dbGetQuery(dbcon, query)
print(res)
```

```
##   id parent notused                                       detail
## 1  3      0       0                                       SCAN f
## 2  9      0       0 SEARCH l USING INTEGER PRIMARY KEY (rowid=?)
```

It doesn't use the index. It's because pattern match LIKE is used here.

Clean

```
dbDisconnect(dbcon)
```