Teradata

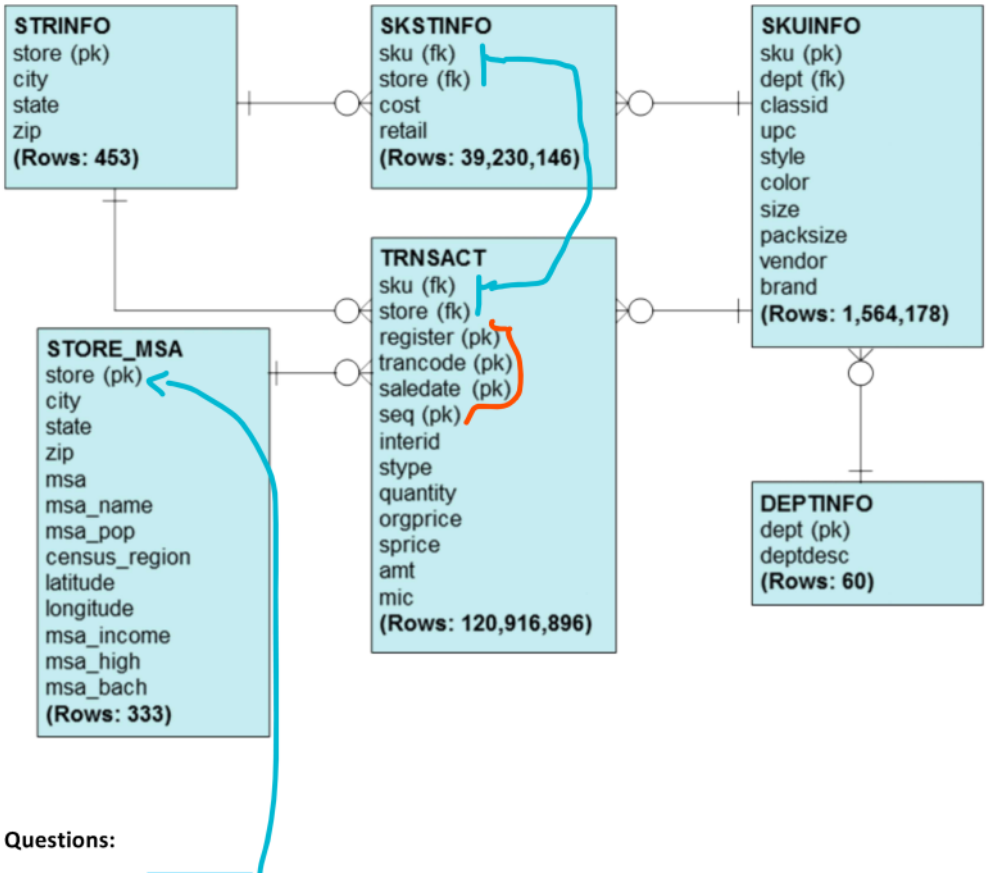
2019年6月6日 18:02

Dillards_relational_schema_practice_questions

Managing Big Data with MySQL — Dr. Jana Schaich Borg, Duke University

Dillard's Database Relational Schema Practice Questions

The relational schema we were given for the Dillard's database that we will be using in upcoming weeks of this course is provided below. The diagram has a slightly different format than other relational schemas we've seen so far, because it includes cardinality information between the primary key in one table, and the foreign key(s) in another table. Inspect the diagram, and answer the questions at the bottom of the page.



Questions:

1. What column(s) would you need to use to link the TRNSACT and STORE_MSA tables?
2. There is not a line connecting the TRNSACT and SKSTINFO tables in the diagram, but it is still possible to combine information from the two tables via their shared columns. What column(s) would you need to use to link the TRNSACT and SKSTINFO tables?
3. What column(s) comprise(s) the primary key of the TRNSACT table?
4. The TRNSACT and SKSTINFO tables have foreign keys to both the STRINFO (store info) and SKUINFO (sku info) tables. What can you infer about the relationship between stores and sku numbers?

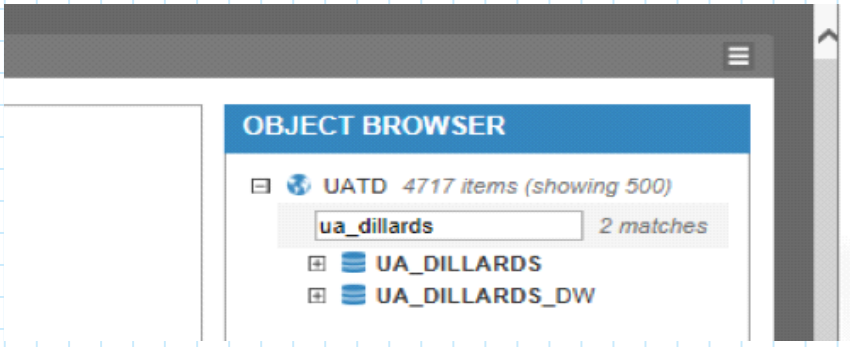
4. The TRNSACT and SKSTINFO tables have foreign keys to both the STRINFO (store info) and SKUINFO (sku info) tables. What can you infer about the relationship between stores and sku numbers?
5. How would you interpret the information provided by the cardinality symbols in the diagram?

The single line symbol can be interpreted as "must have a minimum of, but can have no more than, 1". Therefore, transactions can occur in exactly one store with exactly one metropolitan statistical area ("store_msa"). A store in the store info ("strinfo") table will not necessarily have a store_msa associated with it (because there is no line connecting the two tables). Each transaction has exactly one sku, which will be housed in exactly one department. Each store, store_msa, sku, and department can appear in many transactions, but does not have to appear in any. Each department can have many skus, but does not have to have any.

Answers:

1. Store
2. Sku and Store (you would need them both)
3. Register, trancode, and saledate, and seq (that means that you need all four to identify a unique transaction!)
4. Stores and skus have a many-to-many relationship: stores have many different skus in their merchandise, and skus can be present in many different stores
5. The single line symbol can be interpreted as "must have a minimum of, but can have no more than, 1". Therefore, transactions can occur in exactly one store with exactly one metropolitan statistical area ("store_msa"). A store in the store info ("strinfo") table will not necessarily have a store_msa associated with it (because there is no line connecting the two tables). Each transaction has exactly one sku, which will be housed in exactly one department. Each store, store_msa, sku, and department can appear in many transactions, but does not have to appear in any. Each department can have many skus, but does not have to have any.

Teradata



Sample queries using the SQL Scratchpad:

1. To establish ua_dillards as the default database so you will not have to qualify (prefix) queries and table names with ua_dillards, enter the following SQL statement in the SQL Scratchpad query panel (top left panel):

```
Database ua_dillards;
```

Instead of using **SHOW** or **DESCRIBE** to get a list of columns in a table, use:
HELP TABLE [name of table goes here; don't include the brackets when executing the query]

To get information about a single column in a table, you could write:
HELP COLUMN [name of column goes here; don't include the brackets when executing the query]

whether or not a column is a primary or foreign key. In order to get that information, use a **SHOW** command:
SHOW table [insert name of table here; don't include the brackets when executing the query];

Terdata uses a **TOP** operator instead of a **LIMIT** operator to restrict the length of a query output. Whereas LIMIT comes at the end of a MySQL query, **TOP** comes immediately after **SELECT** in a Teradata query.

The documentation for the **TOP function** can be found here:
http://www.teradatawiki.net/2013/10/Teradata-SAMPLE-Function.html

retrieve 10 random rows:

```
SELECT *
FROM strinfo
SAMPLE 10
```

Teradata has a different rule for GROUP BY statements than does MySQL.
In Teradata (and many other database systems):
Any non-aggregate column in the SELECT list or HAVING list of a query with a GROUP BY clause must also listed in the GROUP BY clause.
This rule makes it less convenient to write some queries, but ensures that outputs that aggregate across groups of records aren't mixed with outputs that only represent single records. To understand the aggregation rule better, consider this query:

```
SELECT sku, COUNT(sku), retail, cost
FROM skstinfo
GROUP BY sku
```

- This query would be executed in MySQL (if the Dillard's dataset was a MySQL database), but a random value in the retail column and the cost column would be outputted for each output row that reports a count of a given sku number.
- Teradata, on the other hand, will return this error if you tried to execute the query above:
  Error Code - 3504 Error Message - [Teradata Database] [TeraJDBC 15.10.00.05] [Error 3504] [SQLState HY000] Selected non-aggregate values must be part of the associated group.

To resolve the error, either **retail** and **cost** both need to be included in the GROUP BY clause (but note that this query would **output a separate row** for each **distinct** combination of **sku, retail, and cost** rather than a row for each sku, on its own):

```
SELECT sku, retail, cost, COUNT(sku)
FROM skstinfo
GROUP BY sku, retail, cost
```

...or **retail** and **cost** need to be **aggregated** to match the sku aggegration:

```
SELECT sku, COUNT(sku), AVG(retail), AVG(cost)
FROM skstinfo
GROUP BY sku
```