Database Systems I

Indexes (1)

Instructor: Ouldooz Baghban Karimi
CMPT 354 - Summer 2019
Indexes

✅ Views

• Indexes

• Storage Management
  • Secondary Storage *(Required Reading)*
  • Index Structures
(Virtual) Views

• Relations that are defined by a query over other relations
  • Not stored in the database
  • Can be queried as if they existed

• Constructed periodically from the database and stored
  • Can speed up the execution of queries
  • Special Case: **Indexes**
Views in SQL (1)

• Declaring Views

```
CREATE VIEW <view-name> AS <view-definition>;
```

The view definition is a SQL query

• Attributes of the view could be specified renaming table attributes
  Surrounded by parentheses, after the name of the view

```
<view-name (attribute list)>
```

• Querying Views: A view may be queried exactly as if it were a stored table

• Removing Views

```
DROP VIEW <view-name>;
```
Views in SQL (2)

• Modifying Views
  • Limited insertions, deletions, or updates to updatable views

• Updatable Views
  • Using `SELECT`, not `SELECT DISTINCT`
  • The `WHERE` clause must not involve `R` in a subquery
  • The `FROM` clause consists of only one occurrence of `R`, and no other relation
  • The list in the `SELECT` clause must include enough attributes that for every tuple inserted into
    the view, we can fill the other attributes out with NULL values or the proper default

• Instead-of Triggers on Views
  • Use instead-of instead of before and after
  • Trigger to replace an action on a view by an action on the underlying base table
Example: Instead-of Triggers on Views

CREATE TRIGGER ParamountInsert
INSTEAD OF INSERT ON ParamountMovies
REFERENCING NEW ROW AS NewRow
FOR EACH ROW

    INSERT INTO Movies(title, year, studioName)
    VALUES(NewRow.title, NewRow.year, 'Paramount');
Indexes

• Views

✓ Indexes

• Storage Management
  • Secondary Storage (Required Reading)
  • Index Structures
Index

• An index on an attribute A of a relation is a data structure that makes it efficient to find those tuples that have a fixed value for attribute A.

• Motivation: limit operation expense in large relations

• Index on an attribute
  • Speeding up queries and joins involving that attribute
  • Complicating insertions, deletions, and updates to the relation
Index Usefulness

• The most useful index on a relation: an index on its key.

  • Queries with a value for the key are common $\rightarrow$ an index on the key will get used frequently.

  • At most one tuple with a given key value $\rightarrow$ the index returns either nothing or one location for a tuple $\rightarrow$ at most one page must be retrieved to get that tuple into main memory.
Index Usefulness

• Other effective Indexes

  • Almost a key attribute: relatively few tuples have a given value for that attribute. Even if each of the tuples with a given value is on a different page, we shall not have to retrieve many pages from disk.

  • Tuples clustered on an attribute: We cluster a relation on an attribute by grouping the tuples with a common value for that attribute onto as few pages as possible. Then, even if there are many tuples, we shall not have to retrieve nearly as many pages as there are tuples.
Indexes in SQL (1)

• Declaring Indexes
  ```sql
  CREATE INDEX <index-name> ON <relation(<attributes>)>;
  ```

• Removing Indexes
  ```sql
  DROP INDEX <index-name>;
  ```
Acknowledgements

I have used materials from the following resources in preparation of this course:

• Database Systems: The Complete Book
• Database Systems (Kifer, Bernstein, Lewis)
• Database System Concepts: https://www.db-book.com
• Course offerings
  • W 4111 (Eugene Wu - Columbia): https://w4111.github.io/
  • CS 245 (Matei Zaharia - Stanford): http://web.stanford.edu/class/cs245/
  • CS 186 (Joe Hellerstein - Berkeley): https://sites.google.com/site/cs186fall17/