CS 4604: Introduction to Database Management Systems

Midterm Review

Virginia Tech CS 4604 Sprint 2021
Instructor: Yinlin Chen
Midterm Exam Topics

• Relational Algebra
• Entity-Relationship (E/R)
• SQL
• Storing and Indexing
• Hashing and Sorting
• Query Processing
Relational Algebra

• Selection: $\sigma_{\text{condition}}(R)$
• Projection: $\pi_{\text{att-list}}(R)$
• Cartesian product: $R \times S$
• Set union: $R \cup S$
• Set difference: $R - S$
• Intersection $\cap$
• Joins ($\Join\bowtie$)
• Rename ($\rho$)
E/R Diagrams: Relationships

• Show a many-one relationship by an arrow entering the “one” side. Many → One

• Show a one-one relationship by arrows entering both entity sets. One ← One

• In some situations, we can also assert “exactly one,” i.e., each entity of one set must be related to exactly one entity of the other set. To do so, we use a rounded arrow. Exactly One → )
E/R Example

• Each department teaches multiple courses. Each course has a number.
Converting E/R Diagrams to Relational Designs

• Entity Set → Relation
  – Attribute of Entity Set → Attribute of a Relation
• Relationship → relation whose attributes are
  – Attribute of the relationship itself
  – Key attributes of the connected entity sets
• Several special cases:
  – Weak entity sets
  – Combining relations (especially for many-one relationships)
  – ISA relationships and subclasses
• Also note how referential integrity comes in (foreign keys)
Basic SQL Query

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification;
```

- **Relation-list**: A list of relation names (possibly with range-variable after each name)
- **Target-list**: A list of attributes of relations in relation-list
- **Qualification**: conditions on attributes
- **DISTINCT**: optional keyword for duplicate removal
  - Default = no duplicate removal!
- **ORDER BY**: for sorting values
Boolean operators

• NOT, AND, and OR

SELECT name, num dogs FROM Person
WHERE age >= 18
AND num dogs > 3;
NULL

- SELECT name, num dogs FROM Person
  WHERE age <= 20 OR num dogs = 3;

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>num_dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ace</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>Ada</td>
<td>NULL</td>
<td>3</td>
</tr>
<tr>
<td>Ben</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Cho</td>
<td>27</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Aggregate Functions

- SUM, AVG, MAX, MIN, and COUNT
- The input to an aggregate function is the name of a column, and the output is a single value that summarizes all the data within that column.
- Every aggregate ignores NULL values except for COUNT(*).

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>num dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ace</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>Ada</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>Ben</td>
<td>7</td>
<td>NULL</td>
</tr>
<tr>
<td>Cho</td>
<td>27</td>
<td>3</td>
</tr>
</tbody>
</table>
Group By and Having

- SELECT age, AVG(num dogs) FROM Person
  WHERE age >= 18
  GROUP BY age
  HAVING COUNT(*) > 1;

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>num_dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ace</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>Ada</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>Ben</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Cho</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>Ema</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Ian</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>Jay</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>Mae</td>
<td>33</td>
<td>8</td>
</tr>
<tr>
<td>Rex</td>
<td>27</td>
<td>1</td>
</tr>
</tbody>
</table>
Illegal Queries

• SELECT age, AVG(num dogs) FROM Person;

• SELECT age, num dogs FROM Person GROUPBY age;
You Should Already Know

- SELECT <columns>
  FROM <tbl>
  WHERE <predicate>
  GROUP BY <columns>
  HAVING <predicate>
  ORDER BY <columns>
  LIMIT <num>;

- https://github.com/VTCourses/CS4604_Labs/tree/master/2.select
- https://github.com/VTCourses/CS4604_Labs/tree/master/3.more_queries
Other SQL Functions

- DATEDIFF()
- ROUND(), Sum(), min(), max(), count()
- IFNULL()
- IF()
- ABS(), avg()
- MOD()
- Between…and
- CASE…WHEN
- A lot more: https://www.w3schools.com/sql/sql_ref_mysql.asp
Join Variants

- Inner Joins
- Outer Joins
- Natural Join
- [https://github.com/VTCourses/CS4604_Labs/tree/master/4.joins](https://github.com/VTCourses/CS4604_Labs/tree/master/4.joins)

```
SELECT <column expression list>
FROM table_name
  [INNER | NATURAL
   | {LEFT | RIGHT | FULL } {OUTER}] JOIN table_name
  ON <qualification_list>
WHERE …
```
More SQL

- Sub-queries
- Correlated Subqueries
- SQL DDL
- Constraints
- Triggers
- Functions
- Note how referential integrity can be enforced (foreign key; on delete cascade etc.)
Tree Indexes

• B+-Trees
  – Carefully understand the Definition!
  – Searching
  – Inserting
  – Deleting
Each interior node is at least partially full:
- \( d \leq \text{#entries} \leq 2d \) (* root: \( 1 \leq \text{#entries} \leq 2d \) *)
- \( d \): order of the tree (max fan-out = \( 2d + 1 \) )

Data pages at bottom need not be stored in logical order
- Next and prev pointers

Height: the length of a path from the root to a leaf
Hashing/Sorting

• Hashing
  – Static Hashing
  – Extendible Hashing
  – Linear Hashing

• Sorting
  – Two-way merge sort
  – External merge sort
  – B+ trees for sorting

• How to search and build, internalize the structure
• Understand the process, how to cost it, how many passes it takes etc.
Hashing Summary

• B-trees and variants: in all DBMSs
• Hash indices: in some DBMSs
  – Hashing is useful for joins
• Hashing performs well on exact match queries
• B+ tree performs well on:
  – Search:
    • exact match queries
    • range queries
    • nearest-neighbor queries
  – Insertion and deletion
  – Smooth growing and shrinking
Sorting Summary

• External sorting is important
• External merge sort minimizes disk I/O cost:
  – Pass 0: Produces sorted runs of size $B$ (# buffer pages)
  – Later passes: merge runs.
• Clustered B+ tree is good for sorting
• Unclustered B+ tree is usually very bad
Join techniques

• Nested-loops join
  – Simple Nested Loop Join
  – Page Nested Loop Join
  – Block Nested Loop Join

• Index-nested loops join

• Sort-merge join

• Hash join
  – Naive Hash Join
  – Grace Hash Join
Reading and Next Class

• Midterm Review
• Next: Project Interim presentation