Database Systems - Assignment A05

Justin Lépine
Michael Zilber
Rémi Ducourtieux
Yanis Alioua

Task 1: Query Optimization

AUTHOR(ID, Name, Age, Email)
DOCUMENT(DOI, Title, Abstract, Text, Pages)
WRITE(ID, DOI)

```
SELECT A1.Name, A1.Email
FROM AUTHOR A1, DOCUMENT D1, WRITE W1
WHERE A1.ID = W1.ID
AND D1.DOI = W1.DOI
AND A1.Age < 30
AND A1.ID NOT IN (SELECT A2.ID
                  FROM AUTHOR A2, DOCUMENT D2, WRITE W2
                  WHERE A2.ID = W2.ID
                  AND D2.DOI = W2.DOI
                  AND D2.Pages < 10);
```

1 EN: Give the 5 steps to process a query in a database system.

1 : Query translation from high level language (ie SQL) to the system's internal form, via parser or compiler, resulting in a tree or graph data structure called “query tree”.
2 : Query optimization : the system finds an equivalent but more efficient expression to execute, and selects a detailed strategy for processing the query, called “execution strategy”.
3 : Generation of the query code from the relational algebra form
4 : The database runs the query code (interpreted or compiled)
5 : the result of the query is returned

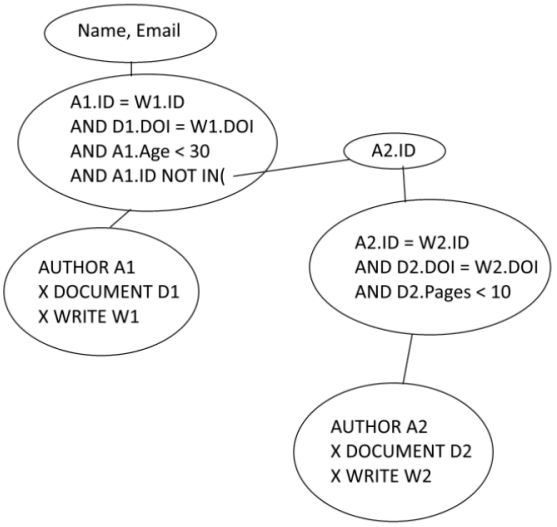
2 EN: Give three general ways of optimizing database queries.

1 : By performing a selection operation on the query, to reduce the number of entries needed, in a large join for example, select only some entries with a particular characteristic beforehand to reduce the final join size.

2 : Doing a projection can also reduce the size of relations and of temporary results. This allows to transform an equivalent relation to a dependent one, reducing the number of temporary results needed.

3 : Making use of other equivalence relations like distribution is useful, for example in the case of natural joins as in (`r1 JOIN r2) JOIN r3` is equivalent to `r1 JOIN (r2 JOIN r3)`. This can be used for example to optimize a query with multiple chained JOIN statements : depending on the situation, it can be beneficial to join some tables before others.

3 EN: Give a (non-optimized) query tree for the query above.



4 EN: Give an optimized query tree for the query above. Explain each optimization step.

Step 1 : apply selections earlier :

`A1.Age < 30` -> move to scan, of `AUTHOR A1`
`D2.Pages < 10` -> move to scan of `DOCUMENT D2`

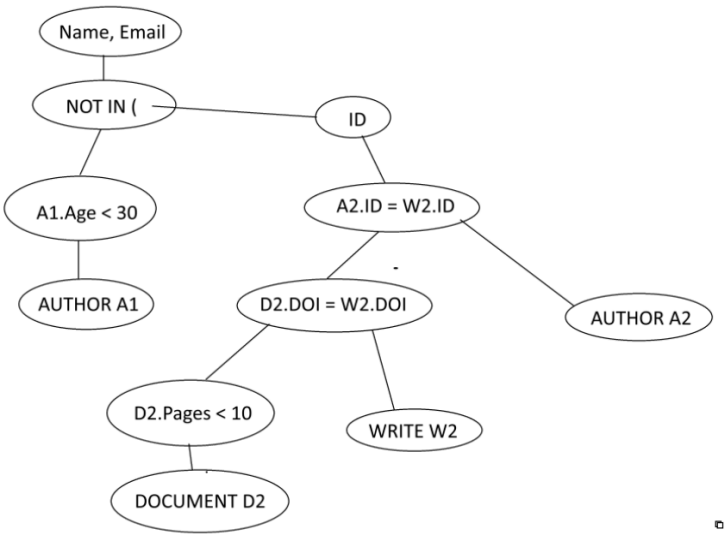
Step 2 : reorder joins

for the first query :

- start with `AUTHOR A1` (after filtering Age < 30)
- join with `WRITE W1` first on `ID`
- then join with `DOCUMENT D1` on `DOI`

for the nested query :

- start with `DOCUMENT D2` (after filtering Pages < 10)
- join with `WRITE W2` on `ID`
- then join with `AUTHOR A2` on `ID`



Task 2: Online Analytical Processing (OLAP)

1. ROLAP: better scaling and space efficiency but slow
   MOLAP: Fast but does not scale well and no space efficient
   HOLAP: Combination of ROLAP and MOLAP

2. Dispositive systems use data that is summarized to help with decision making and data analysis. While productive systems use data more directly for transactions.

3. It is very efficient but also does not lose precision. It is very good for summarization. It is very good for OLAP

4. Data cube describes a table with up to N dimensions with each dimension representing objects. It is implemented with a star or snowflake schema.

   Slicing: Slices of some of the data. For example if we were looking at sales depending on the year a slice could remove some of the years.

   Dicing: Makes the data more precise by removing some features. For example we can dice away the sales info we don't need for years.

   Pivoting: Rotates the cube so that the key changes to a different feature. For example we rotate to use the total profit instead of the year so we can look at the total profit and see the year as a feature.

Drill: Making more or less precise. For example changing the year into months. Or day into months.

Task 3: Embedded SQL

1) Explain the difference between dynamic and static SQL in the context of embedded SQL. Provide an example for each.

Static SQL is optimized during the precompilation phase because the SQL statements are fixed and known in advance. In contrast, dynamic SQL is constructed at runtime based on program conditions, so it is parsed and optimized during execution.
For example, if a search function is predefined and always generates the same type of report, static SQL is suitable. However, if users can apply different filters or customize the query results at runtime, dynamic SQL is more appropriate.

2) Explain what SQL Injection is and why it is considered a major security vulnerability in the context of databases and web applications. Give at least three specific risks that can result from this vulnerability.

SQL Injection is a security vulnerability that enables attackers to insert malicious SQL code into a query, potentially leading to unauthorized data access, data manipulation, or data loss.

3) Outline and explain the key steps, in terms of code structure, involved in updating a table using embedded SQL. Your explanation should cover all necessary elements from initiation to conclusion of the process.

Inside the code, there's 6 major steps involved in updating a table using embedded SQL. In order:

- Including a SQL Communication Header to provides necessary SQL structures
- Establishing connection with the database, required to begin interacting with the database
- Preparing the SQL statement (only if dynamic)
- Executing the SQL statement: the actual update operation is performed
- Checking for errors (not mandatory but recommended)
- Committing and disconnecting the connection with the database: saves the changes and safely ends the database session

Task 4: NoSQL

Was ist NoSQL und wofür steht der Begriff?:

NoSQL steht für Not Only SQL und bezeichnet nicht-relationale Datenbanksysteme, die Schemafrei arbeiten(Felder können dynamisch hinzugefügt werden), horizontal skalierbar sind (Datenverteilung erfolgt über Cluster), Verschiedene Daternmodelle bieten.
Im Prinzip ergänzen NoSQL Datenbanken SQL Datenbanken

Beschreibe die vier Haupttypen von NoSQL-Datenbanken, die in der Vorlesung erwähnt wurden. Nenne für jeden Typ ein Beispiel für eine Datenbanktechnologie.

Document Stores:
Speichern semistrukturelle Dokumente mit flexiblem Schema, ist ideal für Inhalte mit variabler Struktur
Beispiel: MongoDB

Graphdatenbanken
Modellieren Daten als Knoten und Kanten für effiziente Traversals und Netzwerkanalysen
Beispiel: Neo4j

Key-Value Stores
Speichern einfache Schlüssel-Werte-Paare ohne Schema, extrem performant für Lese- und Schreiboperationen
Beispiel: Redis

Wide-Column Stores
Organisieren Daten in Tabellen, Zeilen und frei definierbaren Spalten Familien für große verteilte Datensätze
Beispiel: Apache Cassandra

Beschreibe die wichtigsten Merkmale von Dokumentenspeichern. Wie werden die Daten verwaltet, und in welchen Fällen sollten sie den traditionellen relationalen Datenbanken vorgezogen werden?

Merkmale:
Schemafrei: Dokumente können je nach Eintrag unterschiedliche Felder tragen
Self-contained Entities: Alle zusammengehörigen Daten eines Objekts liegen in einem einzigen Dokument
Flexible Indizierung: Auch verschachtelte Felder und Arrays lassen sich gezielt indexieren
Eingebaute Skalierung: Automatisches Sharding und Replica Sets für hohe Verfügbarkeit

Datenverwaltung
Gruppierung gleichartiger Dokumente
CRUD Operationen greifen immer auf ganze Dokumente zu
Offizielle Treiber mappen Dokumente nahtlos auf native Objekte Ihrere Programmiersprache

Sharding verteilt Dokumente transparent auf mehrere Knoten, Replica Sets halten synchronisierte Kopien

Wann vorziehen
Variable Datenstrukturen: Häufige Schemaänderungen oder ungleichförmige Objekte
Verschachtelte Daten: Hierarchien, die sich in einer Tabelle nur mit komplizierten Jons abbilden ließen