ECE30030/ITP30010 Database Systems

Term Project

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

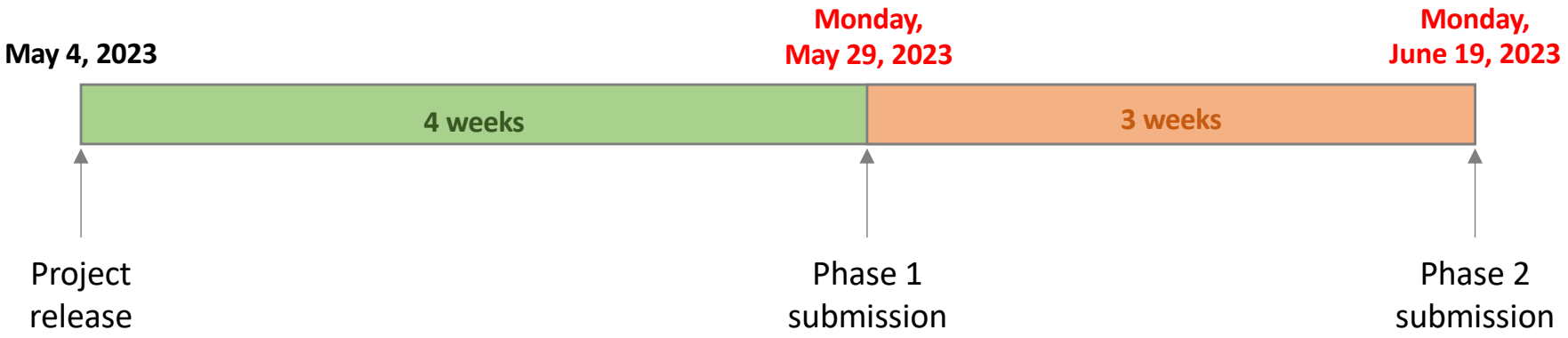
Term Project

- Goals
  - To practice the concepts and underlying mechanisms of database management system with an actual database instance
  - To represent database designs in modeling languages and analyze the designs with respect to given constraints
  - To articulate the relational database language (structured query language)
  - To exercise the optimization and evaluation of the database performance

- In this project, each team will be given a large chunk of data that is completely unnormalized
  - Your objective is to design a "good" database schema that can accommodate the provided data without any loss of information
    - "Good" in that…
      - Efficient in terms of space and time complexity

Term Project Overview

- Planned timeline



- Phase 1 “space” submission: Monday, May 29, 2023
- Phase 2 “time” submission: Monday, June 19, 2023

KUBiC: Korean Unification Bigdata Center

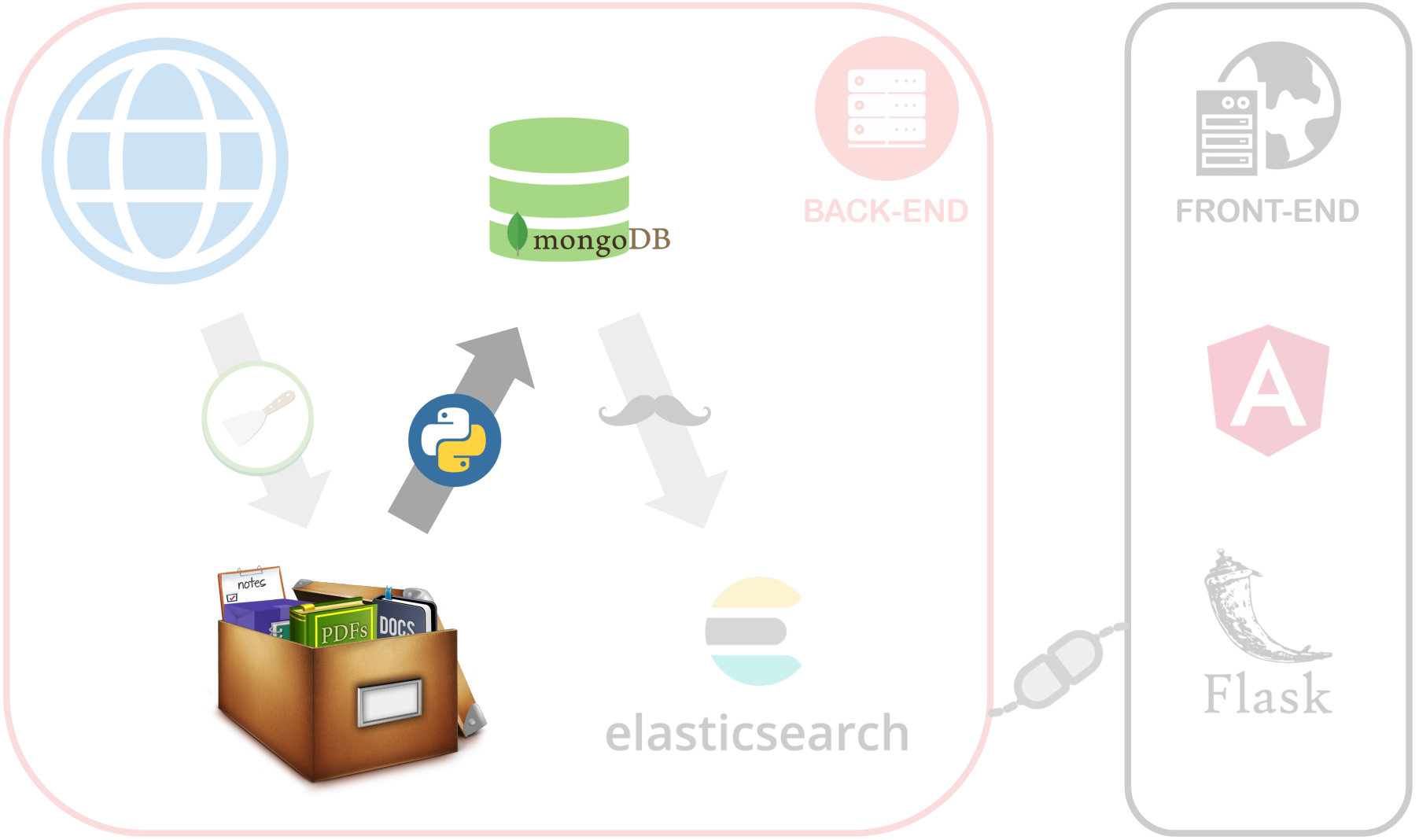
- **Term-Project data is provided by the KUBiC project team**
- A government-funded project on a data-center development focusing on the Korean unification
  - URL: https://kubic.handong.edu/
  - Data archive + search engine + web-based analysis tools, specialized on the Korean unification and North Korea research
  - Contains a lot of academic papers and government reports on the relevant topics

KUBiC
Korea Unification Bigdata Center

한동대학교
HANDONG GLOBAL UNIVERSITY

통일부
Ministry of Unification

KUBiC: Korean Unification Bigdata Center



Term Project

- Background
  - You will be given large chunks of data snapshot from the KUBIC database, that consist of one SQL dump file and two csv files
    - kubicdb
    - tfidf.csv
      - TF-IDF analysis of the service documents
      - 877,490 records, 4 columns (approx. 170.6 MB)
    - rcmd.csv
      - Cosince similarity analysis of the service documents
      - 1,000,000 records, 3 columns (approx. 126.8 MB)

Term Project

- *Phase 1 requirements*
  - *Design and implment a database that can effectively accommodate the entire data without any loss*
    - *You and your team will need to draw E-R diagrams and conduct a number of normalization processes*
  - *Import the data; there should be no missing portion*
    - *You will be asked to create and submit views*
  - *Make the database size as small as possible!*

- *Phase 2 requirements*
  - *Optimize the database using*
    - *Denormalization*
    - *Indexing*

Phase 2: Database Optimization

- Goal: Design and implement a database instance that is efficient in time
  - You may also want to go through denormalization processes on your database instance from Phase 1
  - You may need to add indexes to the database

Phase 2: Database Optimization

- Denormalization
  - Usually carried out to improve the read performance of the database
    - Write may become slower
  - “Normalize until it hurts, denormalize until it works”

Revisited: Denormalization for Performance

- We may want to use non-normalized schema for performance

- Example: displaying *prereqs* along with *course_id,* and *title* requires join of *course* with *prereq*
  - Alternative 1: Use denormalized relation containing attributes of *course* as well as *prereq* with all above attributes
    - faster lookup
    - extra space and extra execution time for updates
    - extra coding work for programmer and possibility of error in extra code
  - Alternative 2: Use a materialized view defined a *course* ⋈ *prereq*
    - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors

Heads-up: Index and Performance Improvement

	ICUSTAY_ID	DRUG	DOSE_VAL_RX	DOSE_UNIT_RX	ROUTE
1	[redacted]	Amoxicillin-Clavulanic Acid	250	mg	PO
2	[redacted]	Amoxicillin	1000	mg	PO
3	<null>	Amoxicillin-Clavulanic Acid	500	mg	PO
4	<null>	CefazoLIN	2	g	IV
5	[redacted]	CefazoLIN	2	g	IV
6	[redacted]	Cefazolin	2	gm	IV
7	<null>	CefazoLIN	2	g	IV

- A query: **SELECT** ICUSTAY_ID, DRUG, DOSE_VAL_RX, DOSE_UNIT_RX, ROUTE
  FROM PRESCRIPTIONS P
  WHERE P.DRUG **LIKE** 'amoxicillin%' **OR** P.DRUG **LIKE** 'cefazolin';

- Without indexes on DRUG

```
[2021-05-22 22:43:43] 500 rows retrieved starting from 1 in 3 s 935 ms
(execution: 3 s 841 ms, fetching: 94 ms)
```

- With an index on DRUG

```
[2021-05-22 22:53:41] 500 rows retrieved starting from 1 in 410 ms
(execution: 371 ms, fetching: 39 ms)
```

Adding Indexes (and resources)

- Creating an index
  - **CREATE INDEX** *index_name* **ON** *table_name* (*column_name(s)*);

- Resources
  - MySQL CREATE INDEX: https://dev.mysql.com/doc/refman/8.0/en/create-index.html
  - Adding indexes on DataGrip: https://www.jetbrains.com/help/datagrip/indexes.html
  - Adding indexes on Workbench: https://dev.mysql.com/doc/workbench/en/wb-table-editor-indexes-tab.html

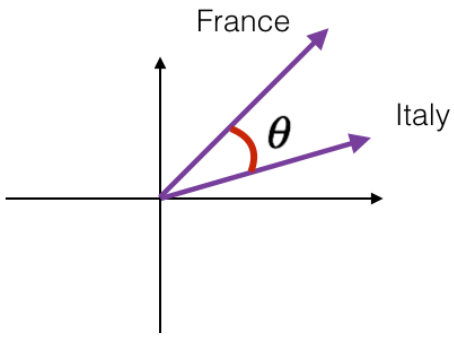
Background

- Background: similarity
  - Contains the similarity (cosine similarity) between each pair of data instances
  - We use it to measure the similarity between each pair of documents

docID	rcmdDocID	Score
10030990067319472539	10030990067319472539	1.0000000000000002
10030990067319472539	10043047191793211293	0.1329758471186704
10030990067319472539	10046263945557091965	0.03193380184794579
10030990067319472539	10046442708758033928	0.0987590617876676
10030990067319472539	10055720692083007959	0.1822967727883514
10030990067319472539	10056260562668352212	0.14230154872067327
10030990067319472539	10082680622925682	0.10731904324528363
10030990067319472539	10083204039233583851	0.18898456260537214
10030990067319472539	10090285731741476390	0
10030990067319472539	10110208171198839861	0.05863854143310857
10030990067319472539	10146231429070036509	0.06463178421565242
10030990067319472539	10193093611430635245	0.07202188938888371
10030990067319472539	10196528194641373645	0.010952017003927252
10030990067319472539	10238602817808319169	0.04562877191564491
10030990067319472539	10284149192682678031	0.03247219967020281

Background

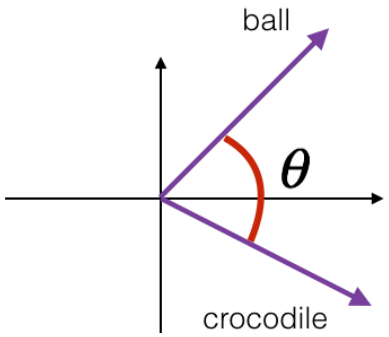
- Background: similarity
  - Contains the similarity (cosine similarity) between each pair of data instances



France and Italy are quite similar

θ is close to 0°

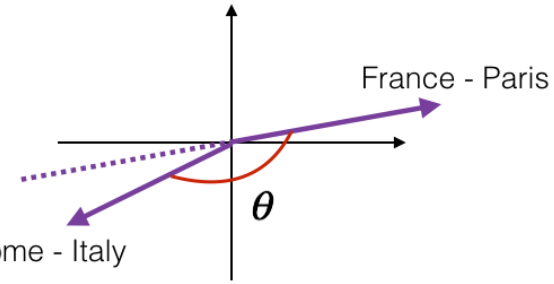
$\cos(\theta) \approx 1$



ball and crocodile are not similar

θ is close to 90°

$\cos(\theta) \approx 0$



the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)

θ is close to 180°

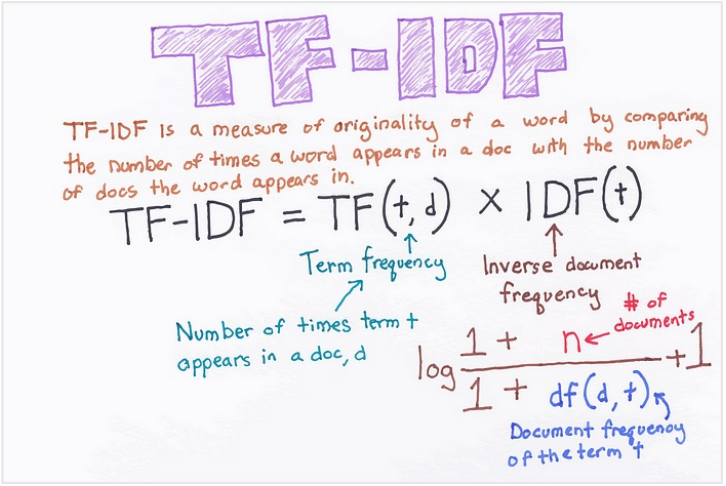
$\cos(\theta) \approx -1$

* Image src: https://datascience-enthusiast.com/DL/Operations_on_word_vectors.html

Background

- TF-IDF stands for term frequency-inverse document frequency
  - Quantifies the importance or relevance of string (words, phrases, lemmas, *etc*.) in a document amongst a collection of documents (corpus)

- TF-IDF breaks down into two parts: TF and IDF
  - TF (term frequency): The weight of a term in a document that is simply proportional to the term frequency
  - IDF (inverse document frequency): The additional factor based on a corpus, to diminish the weight of terms that occur very frequently in the document set and to increase the weight of terms that occur rarely



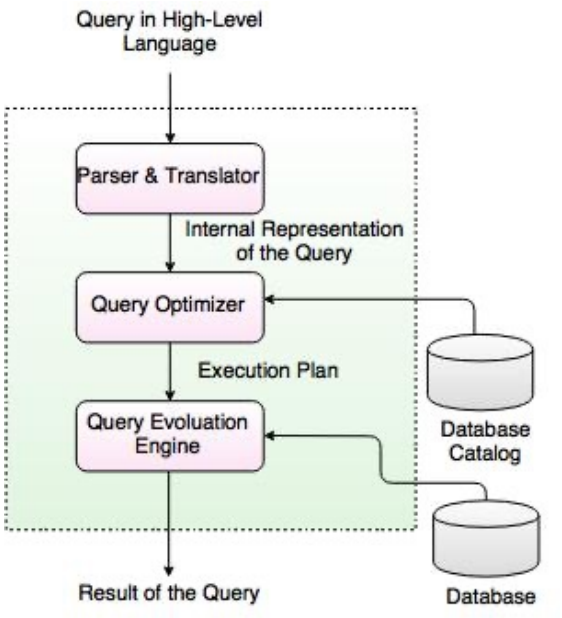
Background

- Background: frequency
  - Contains the TF-IDF (term frequency-inverse document frequency) scores for selected words for each document
  - Measures how relevant a word is to a document in a collection of documents
    - The higher the score, the more relevant that word is in that document
    - Example: https://sci2lab.github.io/ml_tutorial/tfidf/

docID	docTitle	tfidfWord	Score
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가공	0.004478425513887752
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가능	0.042000969424066226
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가속	0.005095251415320924
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가액	0.009833360305863292
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가임	0.009276335805335717
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가정	0.020639077897537134
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가족	0.057368035701677596
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가족법	0.008750184322130957
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	가치관	0.005245828309475467
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	각종	0.007061676068532896
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	간주	0.004251216381898961
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	갈등	0.02004565833769511
10011697067070999700	(2011) 북한이탈주민의 사회적응에 영향을 미치는 요인 분석 : 인천지역을 중심…	감금	0.009025864008681217

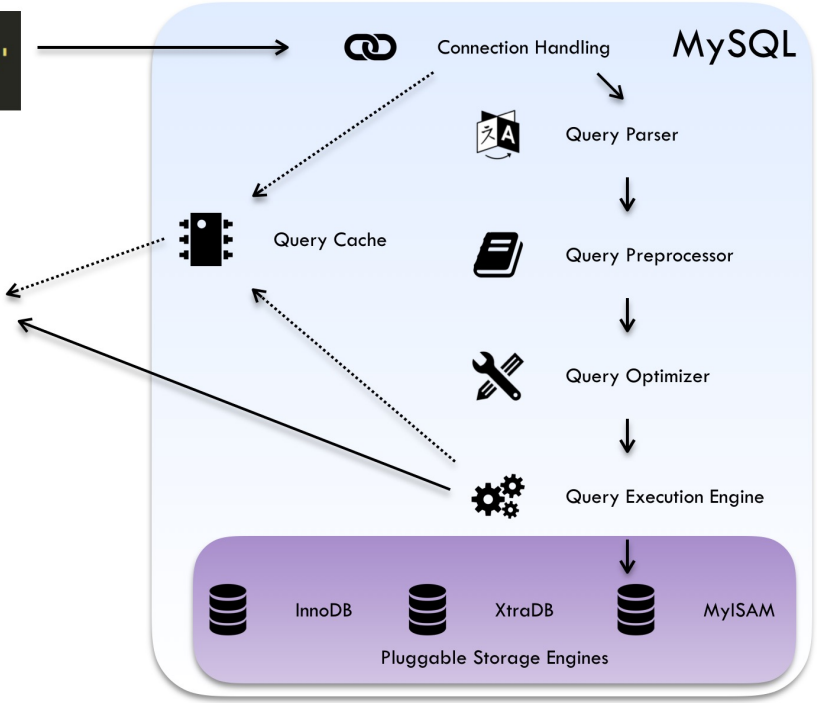
Background

- Query proceeing – key components
  - Query Parser: Uses the SQL grammar to interpret and validate the query
    - The query will be broken into tokens and a “parse tree” will be built based on the tokens




```
SELECT * FROM stock_table
WHERE trade_date = '2016-04-06'
AND symbol = 'AAPL';
```

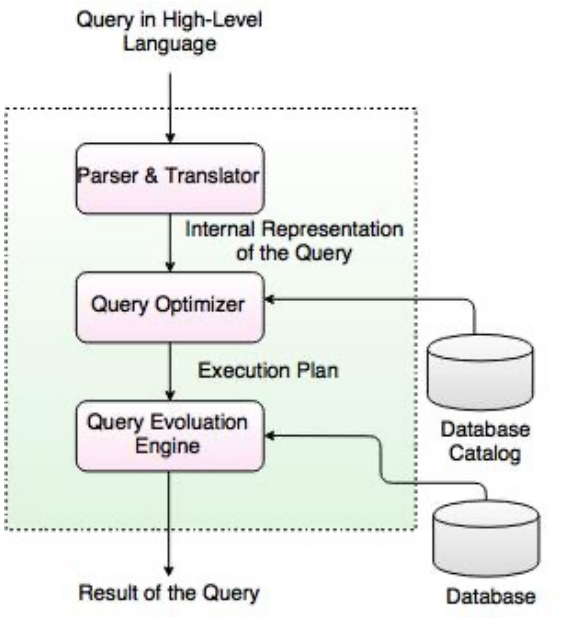
symbol	trade_date	price
AAPL	2016-04-06	110.96



* Sources: https://www.tutorialride.com/dbms/sql-query-processing.htm
https://azureqisite.wordpress.com/2017/01/24/mysql-introduction/

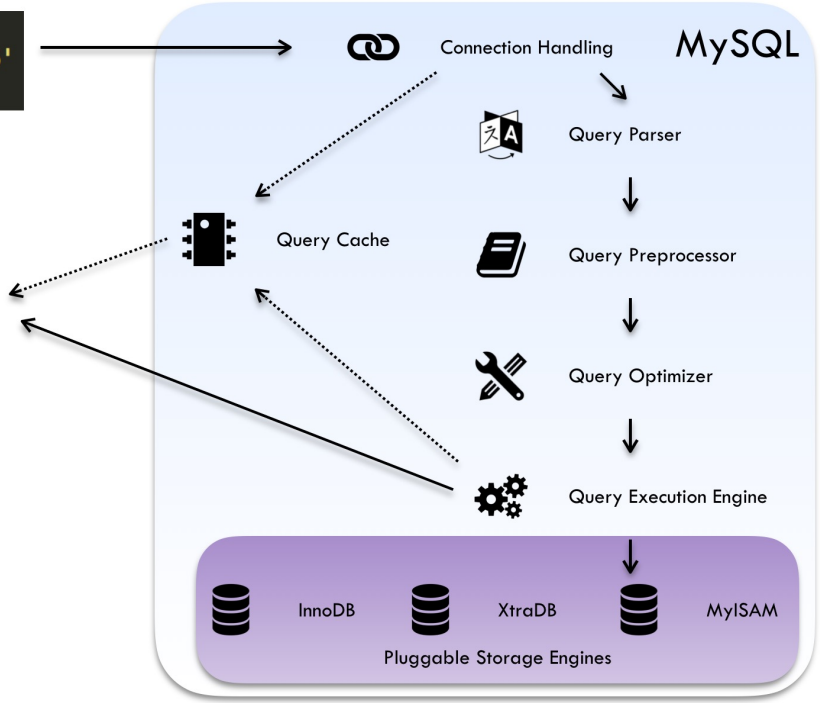
Background

- Query proceeing – key components
  - Query Preprocessor: Checks resulting parse tree for additional semantics that Query Parser cannot resolve, *e.g.*, existence of tables and columns, aliases, *etc*.




```
SELECT * FROM stock_table
WHERE trade_date = '2016-04-06'
AND symbol = 'AAPL';
```

symbol	trade_date	price
AAPL	2016-04-06	110.96



* Sources: https://www.tutorialride.com/dbms/sql-query-processing.htm
https://azureqisite.wordpress.com/2017/01/24/mysql-introduction/

Background

- Query proceeing – key components
  - Query Optimizer: A cost-based Query Optimizer will turn the valid parse tree into query execution plan
    - Various plans will be measured and the least expensive on will be chosen
    - However, the optimizer may not always choose the best plan for many reasons such as wrong statistics, ignorance of other running queries, user defined functions, *etc*.
  - Pluggable storage engines (DB engines)
    - InnoDB: The default transactional storage engine for MySQL
      - The most important and broadly useful engine overall
      - Designed for short-lived transactions that usually complete rather than being rolled back
    - MyISAM: The default storage engine for MySQL in version 5.1 or older
      - It is why MySQL still has the reputation of being a non-transactional database management system

* Sources: https://www.tutorialride.com/dbms/sql-query-processing.htm
https://azureqisite.wordpress.com/2017/01/24/mysql-introduction/

Tools

- EXPLAIN (= DESCRIBE = DESC)
  - Used throughout various SQL databases and provides information about how your SQL database executes a query
  - In MySQL, EXPLAIN can be used in front of a query beginning with SELECT, INSERT, DELETE, REPLACE, and UPDATE
    - **EXPLAIN**
      SELECT *
      FROM foo
      WHERE foo.bar = 'infrastructure as a service' **OR** foo.bar = 'iaas';
    - MySQL would then show its statement execution plan by explaining which processes take place in which order when executing the statement

Tools

- SHOW PROFILE / SHOW PROFILES
  - Display profiling information that indicates resource usage for statements executed during the current session
    - **SHOW PROFILES**: Shows summary profile information for recent queries
    - **SHOW PRFILE**: Shows detailed break down of the profile for recent queries
    - **SHOW PROFILE FOR QUERY** *n*: Shows the break down table for query #*n*
  - To control profiling, use the profiling session variable, which has a default value of 0 (OFF)
    - **SET** profiling = 1;

Phase 2: Database Optimization

- Tasks: Submit your own query and its result to answer below question
    1. *On average, in which month are the most publications released (posted)? Submit your solution along with the query that works on your database schema.*

Phase 2: Database Optimization

- Tasks: Submit your own query and its result to answer below question
    2. *Find the 5 most important keywords (in terms of TFIDF) in the document that is bookmarked (saved) by the most users.*

Phase 2: Database Optimization

- Tasks: Submit your own query and its result to answer below question
    3. *Give the title of the most similar document to the document that is saved least frequently by the users in the handong.ac.kr domain.*

Phase 2: Database Optimization

- Tasks: Submit your own query and its result to answer below question
    4. *Find the three most important keywords (in terms of tf-idf) in the second most frequently bookmarked (saved by the users) document amongst the articles authored by "조한범".*

Phase 2: Database Optimization

- Tasks: Submit your own query and its result to answer below question
    5. *For all words that are used in the frequency analysis, show how many times each word has been used in the analysis (how many times each words has been used in the frequency table).*

Phase 2: Database Optimization

- What to submit
  - A report including
    - For each task, submit your solution queries, their results, and the execution time (your own measures on COSS' DBMS)
    - Description of the re-normalization and indexing steps that the team has gone through, along with proper rationale and justification
    - Summary of the database size and table sizes (in Kilobytes)
  - A zipped MySQL dump file containing all the updated database implementations including the database schema, records, views, *etc*.

    - Reminder: SQL dump file is an ordinary text file, written in the SQL syntax
      - Contains a record of the table structure and/or the data from a database
      - Often used for backing up a database so that its contents can be restored in the event of data loss