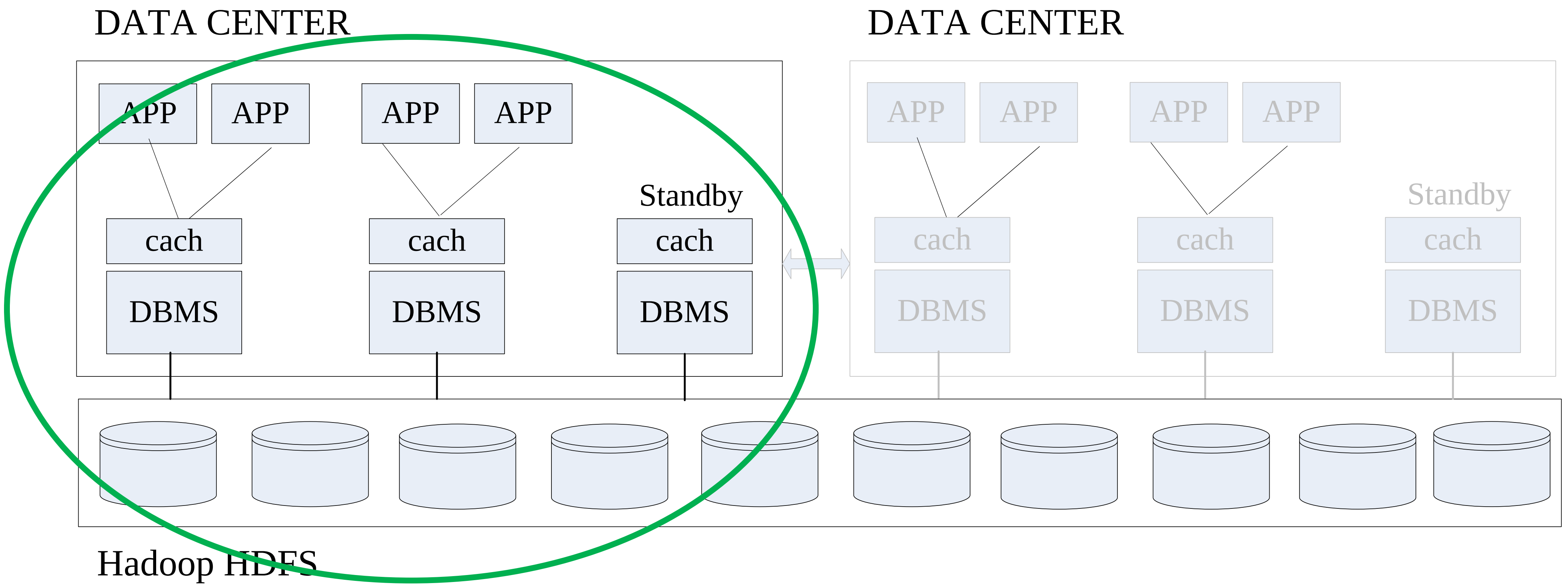


Figure 1. A data center in a distributed context

Data to be managed and processed include structured data (5 relational tables) and unstructured data (text, images, and video).

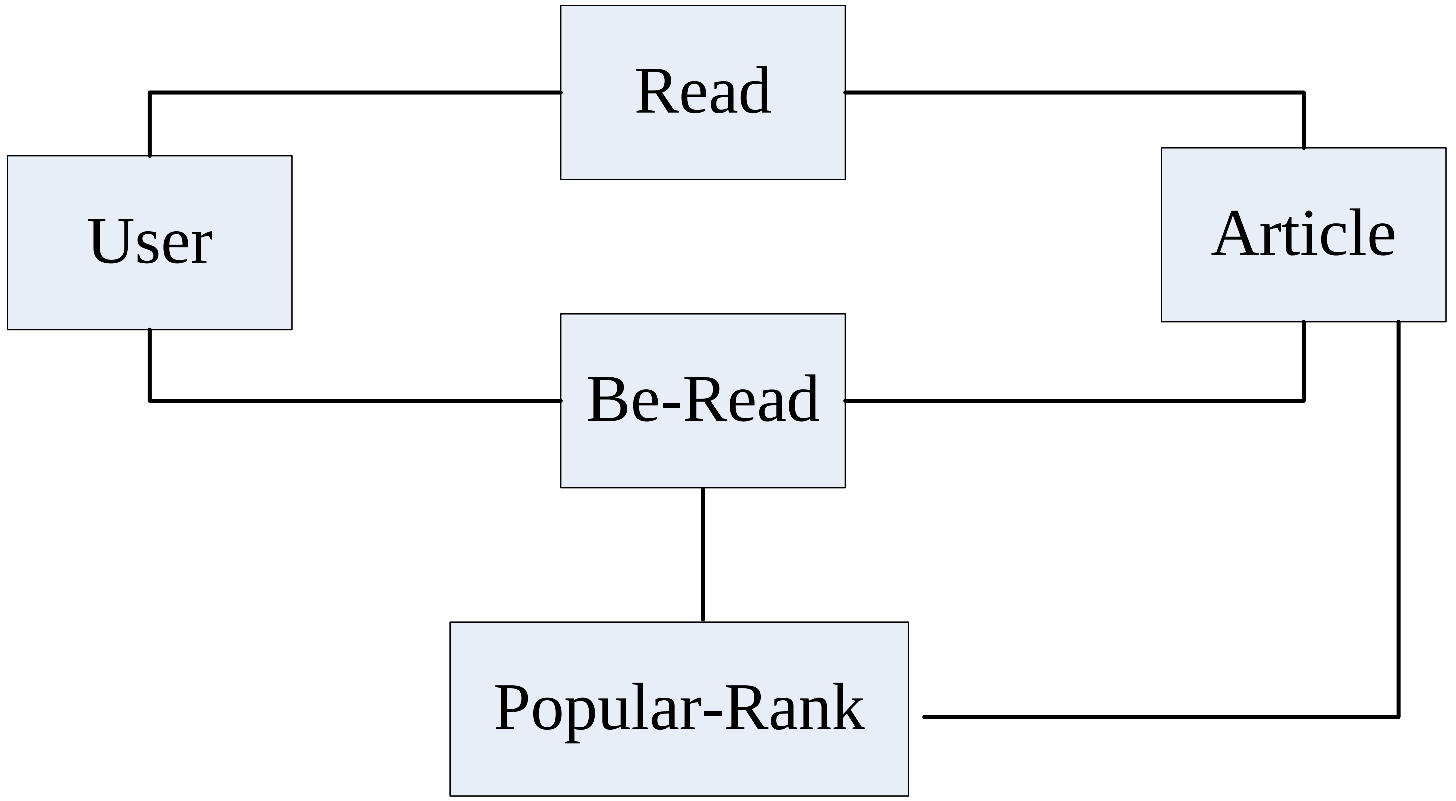


Fig. 2. Entity-Relation diagram

(1) **User** (id, timestamp, uid, name, gender, email, phone, dept. grade, language, **region,** role, preferTags,
obtainedCredits)

fragmented based on region attribute:
region="Beijing" allocated in DBMS1,   region= "HongKong" allocated in DBMS2.

(2) **Article** (id, timestamp, aid, title, **category,** abstract, articleTags, authors, language, text, image, video)

fragmented based on article category attribute:
category="science" allocated in DBMS1 and DBMS2,   category="technology" allocated in DBMS2.

(3) **Read** (id, timestamp, uid, aid, readTimeLength, aggreeOrNot, commentOrNot, commentDetail, shareOrNot)

fragmented based on User table without replica, and with the same allocation schema as User table.

(4) **Be-Read** (id, timestamp, aid, readNum, readUidList, commentNum, commentUidList, agreeNum, agreeUidList,
shareNum, shareUidList)

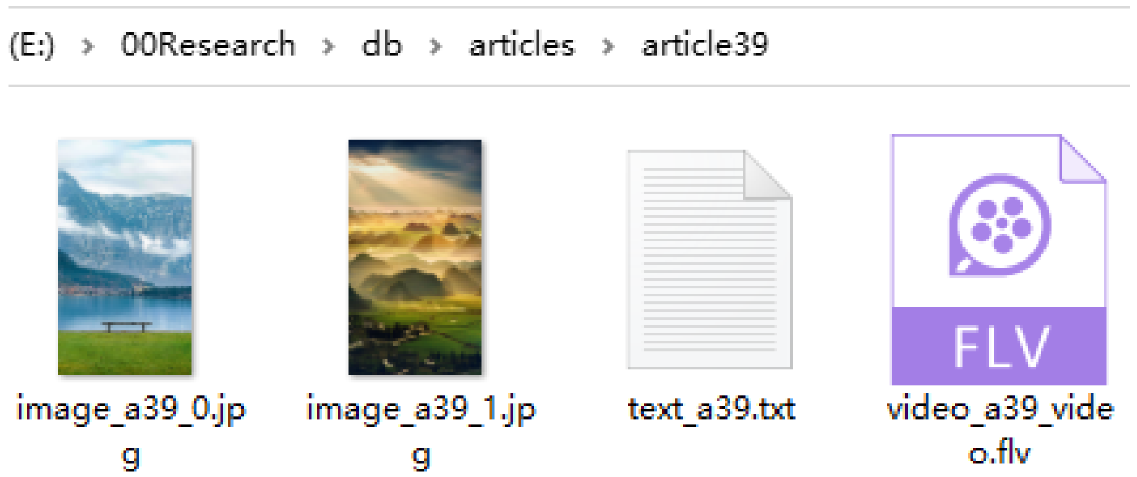
fragmented based on Article table with duplication:
category="science" allocated to DBMS1 and DBMS2,   category="technology" allocated to DBMS2.

(5) **Popular-Rank** (id, timestamp, temporalGranularity, articleAidList)
// temporalGranularity= "daily", "weekly", or "monthly"

fragmented based on temporalGranularity:
temporalGranularity="daily"allocated to DBMS1,
temporalGranularity="weekly"or "monthly" allocated to DBMS2.

Data Generator Program

- Run the Python program and generate Article table (10000 articles), User table (10000 users), and Read table (1 million records). Be-Read table and Popular-Rank table need to be derived yourself.
- Among the totally generated 10000 articles. Each article has a text file and 1-5 images. 5% of articles have a video each. In each article record, the attribute values of text, image, and video correspond to the filenames of text, image(s), and video, respectively, which are stored under the article directory



- Three different sizes (10G, 50G, or 100G) of data can be generated after running the corresponding program locally.

Implement a data center in a distributed context (Fig. 1) with functionalities：

1) Bulk load User table, Article table, and Read table with data partitioning and replica consideration into the data center
2) Efficient execution of data insert, update, and queries
    - Query users, articles, users' read tables (involving the join of User table and Article table) with and without query conditions
    - Populate the empty Be-Read table by inserting newly computed records into the Be-Read table.
    - Query the top-5 daily/weekly/monthly popular articles with articles details (text, image, and video if existing) (involving the join of Be-Read table and Article table)
3) Monitoring the running status of DBMS servers, including its managed data (amount and location), workload, etc.
4) (Optional) advanced functions
    - Hot / Cold Standby DBMSs for fault tolerance
    - Expansion at the DBMS-level allowing a new DBMS server to join
    - Dropping a DBMS server at will
    - Data migration from one data center to others

Reference Websites for the Project

- Container:
  - Docker (https://www.docker.com/)
- Cache:
  - Redis (https://redis.io/)
- DBMS:
  - MongoDB (https://www.mongodb.com/1)
  - MySQL (https://www.mysql.com/)
- Unstructured images and Videos:
  - Hadoop HDFS (https://hadoop.apache.org/)
- System Monitoring:
  - MongoDB Compass (https://www.mongodb.com/products/compass)
  - Robo 3T (https://robomongo.org/)

Project Evaluation

- Report and manual (30%)
- Demo (30%)