Lab4 – Experiment of Fetching strategies - Group 3G

Group members

1. Truong Hoang Son (Sonny) - 617315
2. Bui Ba Quyen (Mike) - 617282
3. Nguyen Huu Chi (Sean) - 617310

Sample Dataset

We use the for-loops to populate 100 Users, 1,000 Products and 1,000,000 Reviews. Each product will have 1000 reviews.

Report summary of getting all products.

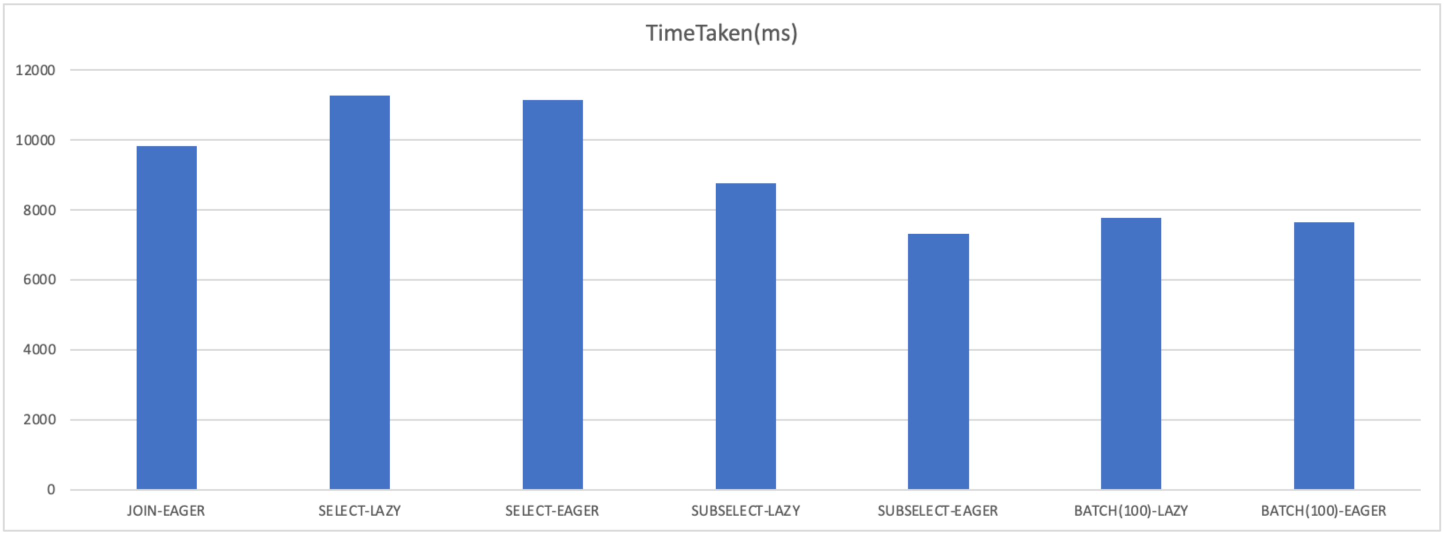
FetchMode	FetchType	NumberOfQuery	TimeTaken(ms)	Memory (Mb)
JOIN	EAGER	1	9822	661
SELECT	LAZY	1001	11260	383
SELECT	EAGER	1001	11141	379
SUBSELECT	LAZY	2	8751	431
SUBSELECT	EAGER	2	7305	417
BATCH (size=100)	LAZY	11	7759	502
BATCH (size=100)	EAGER	11	7634	410

Memory measurement before and after processing rest service.

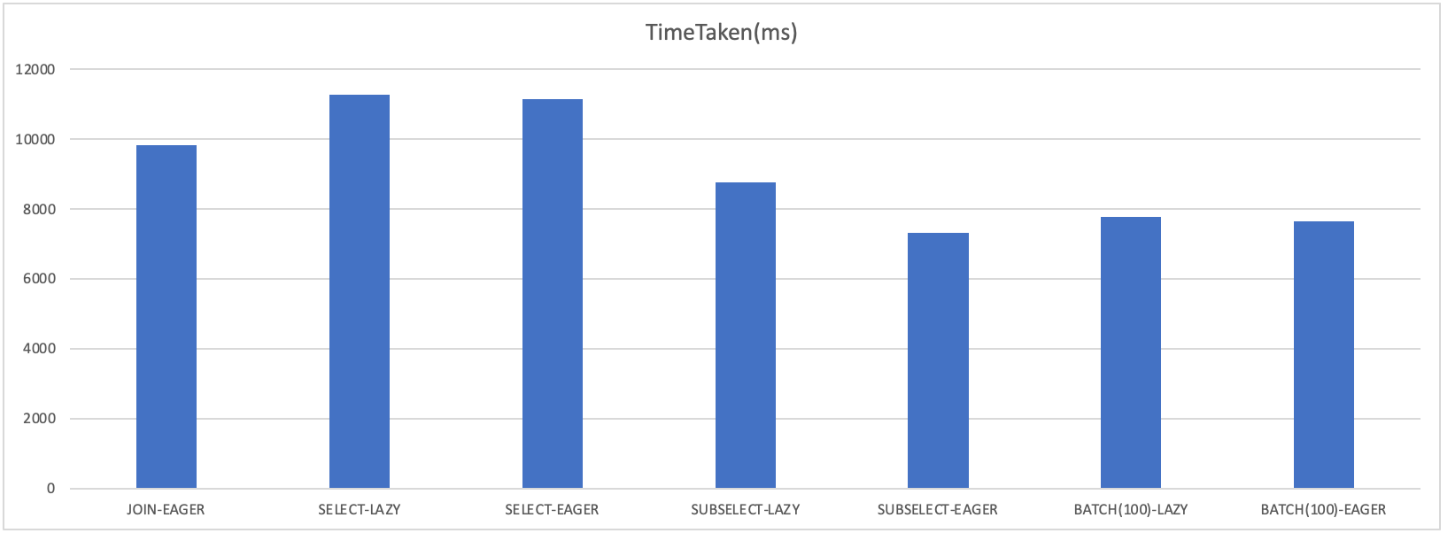
FetchMode	FetchType	Total Memory (Request)	Free Memory (Request)	Total Memory (Response)	Free Memory (Response)
JOIN	EAGER	1114	765	1234	224
SELECT	LAZY	1188	838	1108	375
SELECT	EAGER	1108	758	1220	491
SUBSELECT	LAZY	1194	844	1356	575
SUBSELECT	EAGER	1200	851	1624	858
BATCH (size=100)	LAZY	1194	844	1158	306
BATCH (size=100)	EAGER	1158	807	1274	513

Charts

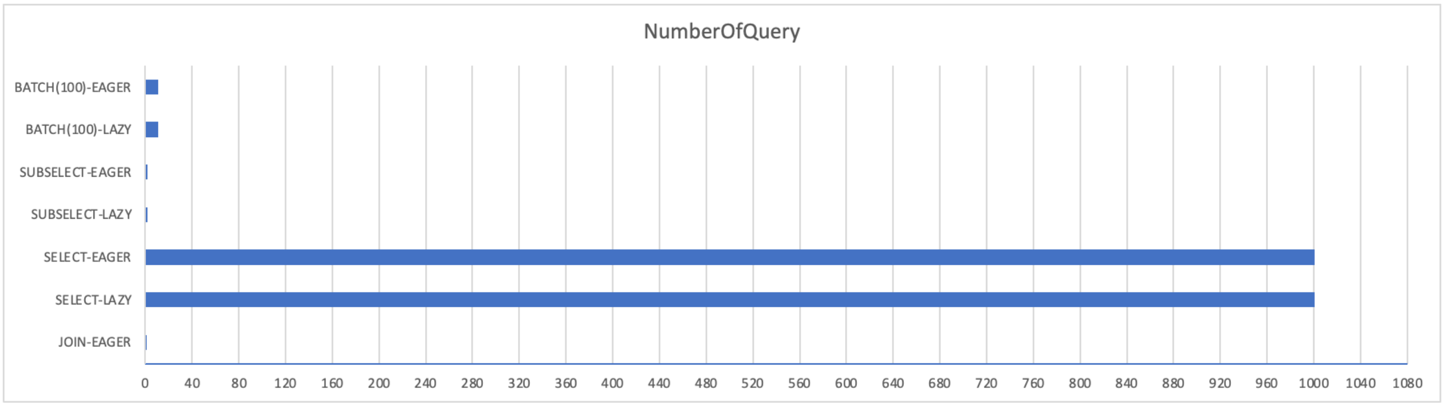
- Time Report



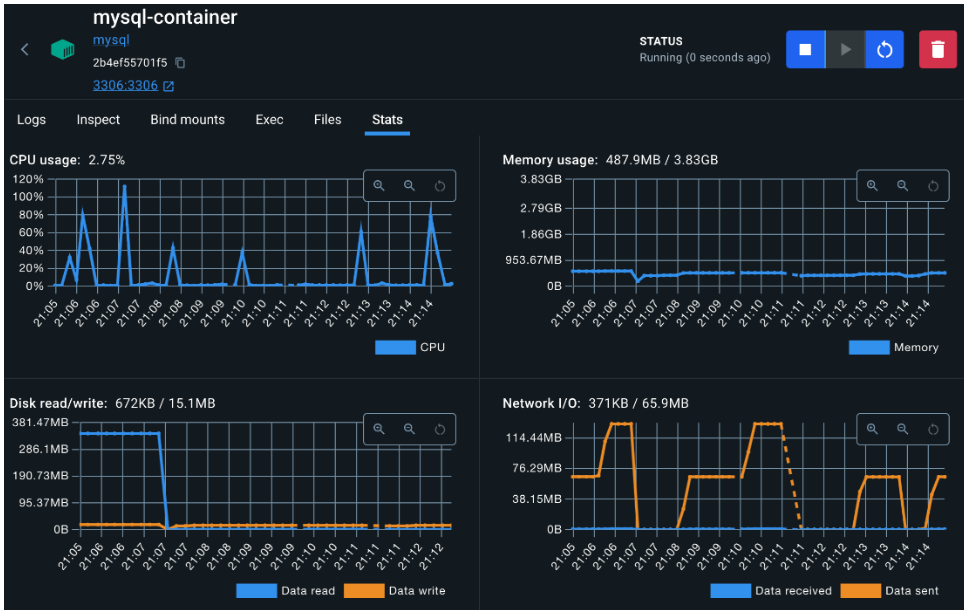
- Memory Report



- Number of SQL Queries



- Transfer Data Size : seem to be same (65 Mb) for all Fetch Strategies



FetchMode SELECT

Query generated:

Hibernate: select psl1_0.id,psl1_0.name,psl1_0.price from product psl1_0

(1000 times) Hibernate: select r1_0.product_id,r1_0.id,r1_0.comment from review r1_0 where r1_0.product_id=?

FetchMode JOIN

Query generated:

Hibernate: select pj1_0.id,pj1_0.name,pj1_0.price,r1_0.product_id,r1_0.id,r1_0.comment from product pj1_0 left join review r1_0 on pj1_0.id=r1_0.product_id

FetchMode SUBSELECT

Query generated:

Hibernate: select pssl1_0.id,pssl1_0.name,pssl1_0.price from product pssl1_0

Hibernate: select r1_0.product_id,r1_0.id,r1_0.comment from review r1_0 where r1_0.product_id in (select pssl1_0.id from product pssl1_0)

Fetch mode BATCH

Query generated:

```
Hibernate: select pbe1_0.id,pbe1_0.name,pbe1_0.price from product pbe1_0
```

```
(10 times) Hibernate: select r1_0.product_id,r1_0.id,r1_0.comment from review r1_0 where r1_0.product_id in
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)
```

<u>Practical Use:</u>

Comparing the performance of different fetch modes (SELECT, JOIN, SUBSELECT) involves considering various factors such as the size and complexity of your data model, the number of associations, and the specific use case of your application. Let's discuss the performance characteristics of each fetch mode:

FetchMode.SELECT:

- **Pros**:
  - Simple and straightforward fetching strategy.
  - Can be efficient for fetching small collections or when lazy loading is preferred.
- **Cons**:
  - Prone to the N+1 query problem, where each association fetches additional data through separate SELECT statements, leading to performance degradation.
  - Not suitable for fetching large collections or deep object graphs due to the potential for excessive database round-trips.

FetchMode.JOIN:

- **Pros**:
  - Reduces the number of database round-trips by fetching all associated entities in a single SQL SELECT statement with JOINs.
  - Can improve performance for fetching large collections or when eager loading is necessary.
- **Cons**:
  - May result in Cartesian product if associations have many-to-many relationships or if the fetched entities have deep object graphs, leading to excessive data duplication.
  - Can increase memory usage and network traffic due to fetching unnecessary data.

FetchMode.SUBSELECT:

- **Pros**:
  - Reduces the number of database round-trips compared to FetchMode.SELECT by grouping associated entities within a single subquery.
  - Suitable for scenarios where eager loading is necessary but fully eager fetching (as with JOIN) is not desired.
- **Cons**:
  - May still result in multiple SQL SELECT statements being executed, especially if there are many associations or if fetched entities have their own associations.
  - Performance can degrade if there are too many associations or if the subqueries fetch unnecessary data.

In general, the optimal fetch mode depends on your specific use case and performance requirements. For example:

- Use FetchMode.SELECT for lazy loading of small collections or when the N+1 query problem is manageable.
- Use FetchMode.JOIN for eager loading of large collections or when minimizing database round-trips is critical.
- Use FetchMode.SUBSELECT as a compromise between SELECT and JOIN, suitable for scenarios where eager loading is necessary but fully eager fetching is not desired.