



Active

Abandoning this PR following creation of  [Rework inline delete query](#) | ☒ [New](#) to rework



Write a reply...

Resolve

1 Aug

Resolved

not comfortable with this approach. Can we reuse the Spring Data delete approach which hides this from us.

An alternative could be

```
@Repository
public class DealFeedBatchDelete {

    private EntityManagerFactory entityManagerFactory;

    public static final String DELETE_ALL_QUERY_BY_ID_STRING = "DELETE FROM dealfeed d WHERE trade_leg_id IN :trade_leg_ids";

    public DealFeedBatchDelete(EntityManagerFactory entityManagerFactory) {
        this.entityManagerFactory = entityManagerFactory;
    }

    @Transactional
    public void performBatchDelete(final List<String> idsToDelete) {
        Query query = entityManagerFactory.createEntityManager().createQuery(DELETE_ALL_QUERY_BY_ID_STRING);
        /**
         * Some JPA providers require {@code ids} to be a {@link Collection} so we must convert if it's not already.
         */
        if (idsToDelete instanceof Collection) {
            query.setParameter("trade_leg_ids", idsToDelete);
        } else {
            Collection<String> idsCollection = StreamSupport.stream(idsToDelete.splititerator(), false)
                .collect(Collectors.toCollection(ArrayList::new));
            query.setParameter("trade_leg_ids", idsCollection);
        }

        query.executeUpdate();
    }
}
// You will need to test as I don't have the DB setup locally
```

Also, for more information see `SimpleJpaRepository.deleteAllByIdInBatch`

From above, we don't have to manually build the ids as done in `createDeleteQuery(...)` plus we are using `:` in delete

1 Aug



It still will not be in a transaction, when the connection opened for insert batch item, it will be a separate transaction.

1 Aug



1 Aug



 do you understand the problem statement here?  
Let me explain,  wants to use batch insert or preferably somehow batch upsert.

- Even with using version, hibernate will make N select queries to see if the item is present in the DB or not, hence we have to write the batch statement ourselves, i.e. PreparedStatement.
- The delete, I tried to implement using PreparedStatement but there was an exception because of mismatch of types in postgres
- Now, for solution you suggested above, the transactional annotation happens in the perform batchDelete method
- I would imagine you will have some code like this on the service level

```
insertAll() { dealFeedBatchDeleteRepo. performBatchDelete(ids); dealFeedBatchRepo.insert(dealfeeds); }
```

Now, what will happen, when the insert fails()? Can we roll-back the transaction there?  
So we will be in a state where there some data deleted and we cannot rollback.

My question is this, I know you don't like this approach but this approach is out of scope for this ticket since the change went on in this [PR](#)


I have arranged a meeting to go through this.

I am keep repeating myself, this should not deal with this because there is already code in develop which does the batch-delete inline, either we revert all of that or nothing.



Aug



 if you are not comfortable with merging this, then we need to revert the commit made on the above.

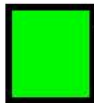
As mentioned on this comment 😊



Already tested it, please see these two gists for code

3

It does not rollback the transaction.



Write a reply...

Reactivate

Aug



Another option is:

```
private final DealFeedRepository repository; //spring-jpa but need to test the transactional stuff

public boolean saveAll(final List<DealFeed> dealFeedList) {
    return saveAllJdbcBatch(dealFeedList);
}

@Transactional
private boolean saveAllJdbcBatch(final List<DealFeed> dealFeedList) {
    repository.deleteAllByIdInBatch(dealFeedList.stream().map(x -> x.id).toList());
    try (final Connection connection = datasource.getConnection()) {
        connection.setAutoCommit(false); // Otherwise each statement is executed as transaction
        if (CollectionUtils.isEmpty(dealFeedList)) {
            log.info("no new deals");
            return true;
        }
        return insertBatch(dealFeedList, connection);
    } catch (SQLException e) {
        log.warn("failed to get data source connection", e);
        throw DealFeedException.from(e);
    }
}
```

Aug



`repository.deleteAllByIdInBatch(dealFeedList.stream().map(x -> x.id).toList())`  
this means we don't have to use entity manager.

The insert and delete will be in separate transaction because we are creating new connection for batch.

I am hoping that in batch-insert logic, we will propagate the exception and delete transaction will be rolled back automatically for us, but this requires a lot of testing which is out of scope for this ticket.

Aug



As mentioned on this comment 😊

Aug





[REDACTED] 29 Jul



if we are concerned about inline delete statement, we could do this but that comes with its own set of problems mainly transaction.

You mentioned that we can use default method on interfaces (which I did not know to be honest!) but this again have problem.

interfaces does not have any class variables, so the DataSource cannot initialised.

`datasource.getConnection`.

I believe the default method in interfaces is introduced for backward compatibility so any class which implements this interface does not need to implement the new interface methods, correct me if I am wrong. [REDACTED] [REDACTED]

```
dealFeedRepository.deleteAllByIdInBatch(dealFeedList.stream().map(x -> x.id).toList());
try (final Connection connection = datasource.getConnection()) {
    connection.setAutoCommit(false); // Otherwise each statement is executed as transaction
    if (CollectionUtils.isEmpty(dealFeedList)) {
        log.info("no new deals");
        return true;
    }
    // if insertBatch fails for any reason, we will not be able to rollback the
    // delete calls might not be rolled back. I am not sure adding @Transactional annotation
    // would solve the problem.
    return insertBatch(dealFeedList, connection);
} catch (SQLException e) {
    log.warn("failed to get data source connection", e);
    throw DealFeedException.from(e);
}
```

[REDACTED] 29 Jul



[REDACTED]

I have spent considerable amount of time, making delete statement to work using PreparedStatement.

Problems encountered was that there is a type mismatch between Java types and postgres types. Please let me know if we have solved this problem within Oils Dev Team.

The main concern is potential sql injection which I believe is fairly low, then can you please point me into resources which I can try since you are the team lead 😊

[redacted] 29 Jul



Please look at DealFeedBatchInsertRepository for the implementation detail. You were concerned about having TestDealFeedRepository, so I created one in main and renamed the original DealFeedRepository implementation class to DealFeedBatchInsertRepository . We do not use this repository implementation in the main code other than Test.

[redacted] 29 Jul



[redacted] You have asked me to forward the email from [redacted] which I did, when I spoke to you on Friday and asked if you had a chance to speak to [redacted] about it, you mentioned you did not. I am yet to get a feedback from you on this.

Available now

Yesterday 11:10

[redacted] let me know when you want to have the chat about PR we discussed yesterday. You mentioned you wanted to have a call with [redacted] and myself today, is this still on the table?

[redacted] Yesterday 14:02 **IMPORTANT**

[redacted] [redacted]



I have privately messaged you on Thursday stating I have forwarded [redacted] email.  
On Friday, when I came to the office after lunch time, I enquired about the above message, you said you do not check emails till 4pm every day.

Thursday 18:34

yes I am

Thursday 18:59

[redacted]

Please see this ticket. It has the screenshot of what [redacted] wanted

Please see this ticket. It has the screenshot of what [redacted] wanted

status as processing for this Petition.

5. When deals received from IM Data Transformer do the following
6. Create a batch of insert statements (using PreparedStatement) looping through data received and use the fetchId of the record created at step 4
7. In the same loop create a list of Tradelegids and add 1 Delete statement for the list of tradelegeids list created above. Make sure the delete statement is the first statement in the batch.

Thursday 19:28

forwarded the email, please speak to [redacted]



I am not sure if you had the chance to look at the email later on Friday but I am posting the requirement email I got from Jothi here.

With current datetime as value  
For fetchTime






Particularly point 6 and 7

 a 

 1 Aug



 you said you will look at this PR yesterday, and I am still waiting for it to be approved.



Write a reply...

Reactivate