v3 Concurrency Phenomenon Write-Up, Music API

Cole Robinson, Geoff Blaylock

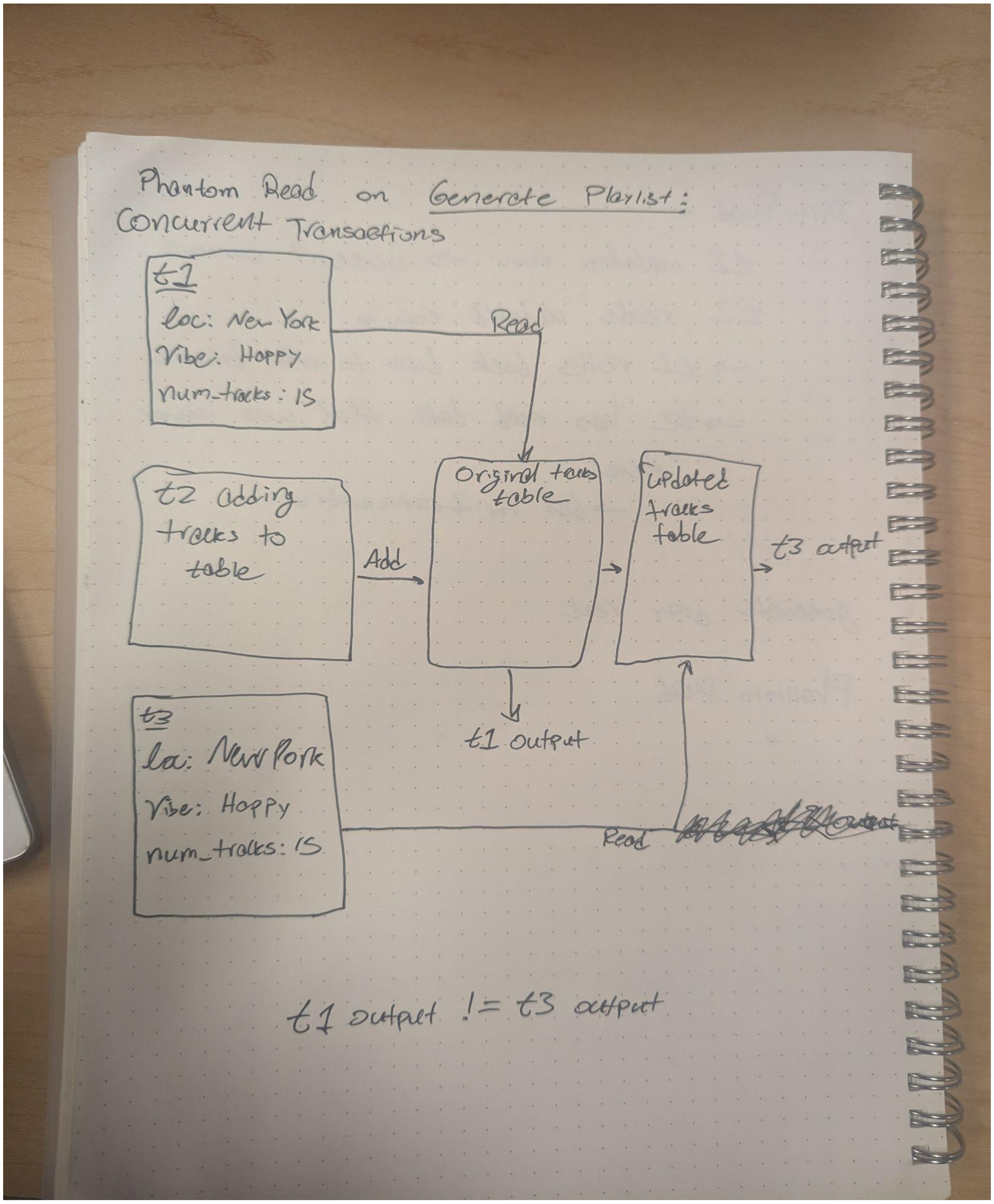
It should also be noted that many of these anomalies could occur due to the Weather API's potential to change as well. For instance, if two concurrent processes were to obtain different values from the Weather API, this would result in many potential anomalies that might be out of control.

1. **Phantom Read on Generate Playlist** - Consider a transaction, t1, that generates a playlist given the user's vibe, location, and number of tracks. Now consider a concurrent transaction,t2, that is adding tracks to the database. If another concurrent transaction, t3, with the exact same parameters as t1 runs, it is possible that these newly added tracks will appear in the result of t3 but not t1. In order to fully prevent this phantom read scenario, the generate function will need to be set to 'SERIALIZABLE'. This function likely wouldn't suffer from Dirty Reads or Non Repeatable Reads since the api does not support editing the actual music data once inserted.

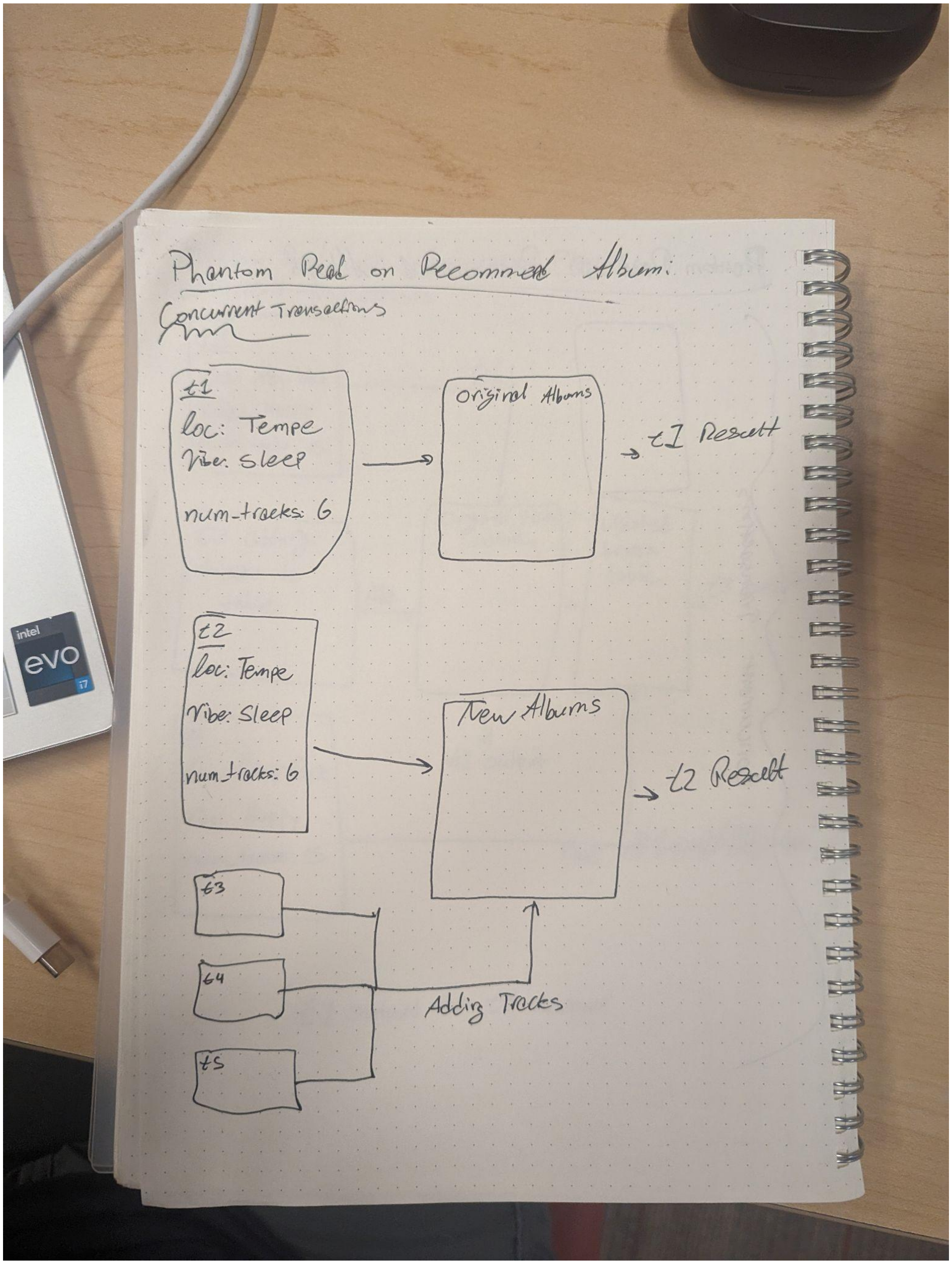
2. **Phantom Read on Reccomend Album -** In similar fashion to generate playlist, this endpoint suffers from potential phantom reads without any concurrency control. Consider a transaction t1, that would like an album recommendation with about 6 tracks. In this case t1 might get an album with roughly six songs. Now imagine that the artist of t1's album recently released the deluxe version of the album, and it just so happens that several users (t3, t4, t5) decided to concurrently add these new songs at the same time as t1. Finally consider another concurrent transaction t2, that uses the same parameters as t1. Due to the recent increase in songs from t1's result album, t2 might return with an entirely different album. This can also be solved by setting the Recommend Album's isolation level to 'SERIALIZABLE'.

3. **Delete Playlist -** Deleting a playlist could result in a dirty read with concurrent processes. A dirty read could occur in this situation when one transaction t1 is attempting to add/delete a track from a playlist or get the playlist. If another concurrent transaction t2 deletes the playlist in question, then t1 ends up with a dirty read of a playlist that does not exist anymore. This can be resolved by setting the other accessor (reading) endpoints to an isolation level of "REPEATABLE READS" as well as the delete_playlist to prevent concurrent processes from getting two different results when reading. This will ensure that concurrent processes are always using the same snapshot of the data.

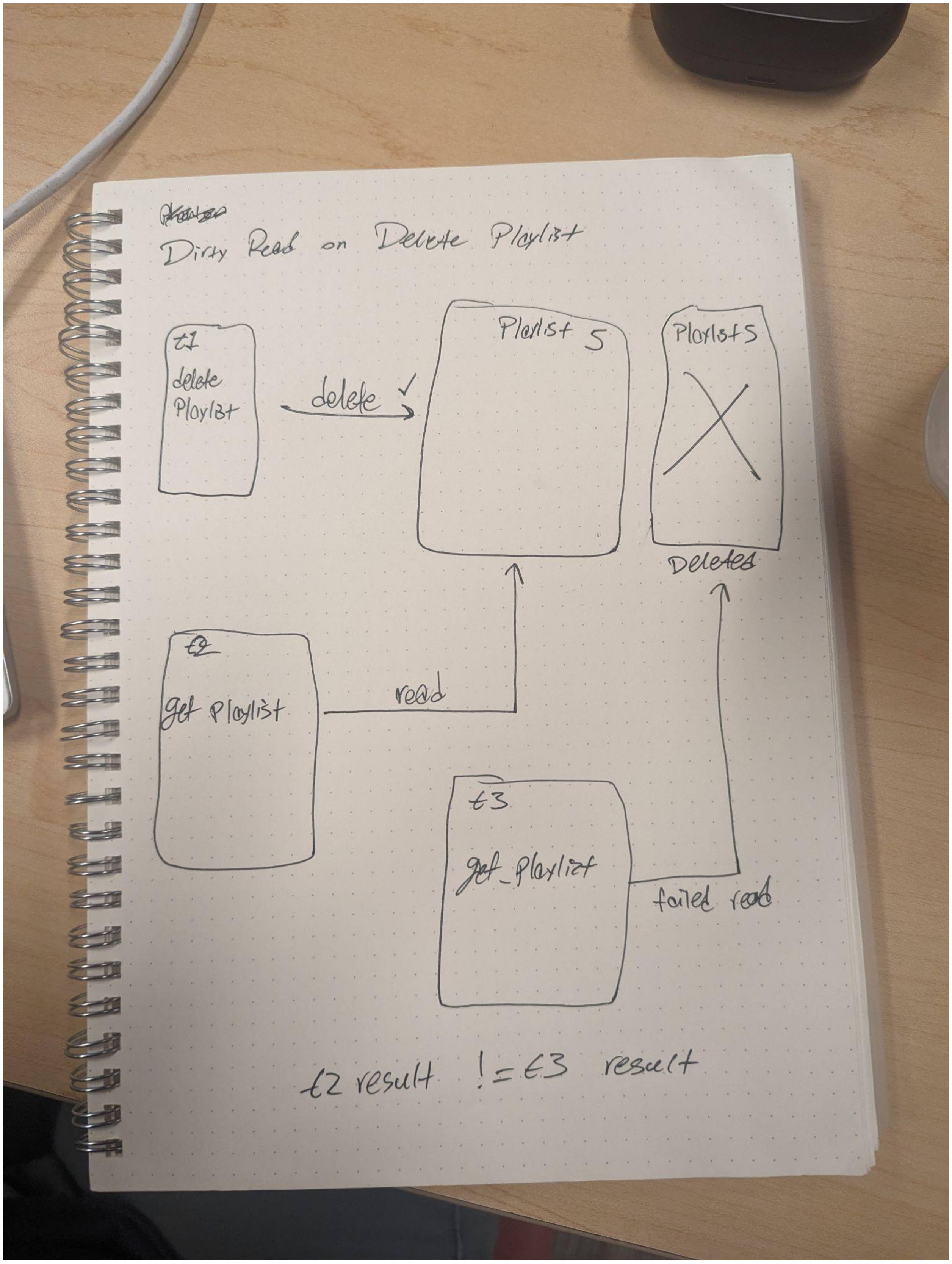
Diagrams for Each:
Phantom Read on Generate Playlist



Phantom Read On Reccomend Album:



Dirty Read on Delete Playlist:


~~Phantom~~
Dirty Read on Delete Playlist
t1
delete Playlist
delete ✓
Playlist 5
Playlist 5
Deleted
t2
get Playlist
read
t3
get_playlist
failed read
t2 result != t3 result