

# Advanced Feature Query Optimization and Indexing

## *MainePad Finder*

**Sophia Priola**

For my advanced database feature I implemented a combination of database indexing and query optimization to improve the performance of common operations in MainePadFinder, especially property search, listing details, and reviews. In addition to the core indexes our group already defined, I added four targeted indexes:

- “CREATE INDEX IDX\_PROPERTY\_BEDS\_BATHS ON PROPERTY (BEDROOMS, BATHROOMS);”
- CREATE INDEX IDX\_RENT\_ADDR ON PROPERTY (RENT\_COST, ADDR\_ID, PROPERTY\_ID);
- “CREATE INDEX IDX\_SESSIONS\_USER ON SESSIONS (USER\_ID);”
- “CREATE INDEX IDX\_SESSIONS\_EXPIRES ON SESSIONS (EXPIRES\_AT);”

The two SESSIONS indexes support our authentication flow. IDX\_SESSIONS\_USER speeds up operations that look up or delete sessions by USER\_ID, such as logout and cleanup of older sessions for a specific user. IDX\_SESSIONS\_EXPIRES makes it more efficient to find and remove expired sessions, which is important for keeping the SESSIONS table from growing without bound over time.

The two PROPERTY indexes support the property search endpoint (/api/properties), which joins PROPERTY to ADDRESS and then filters on fields such as BEDROOMS, BATHROOMS, and RENT\_COST. Without indexes, MySQL would need to scan the entire PROPERTY table to apply those filters. With IDX\_PROPERTY\_BEDS\_BATHS and IDX\_RENT\_ADDR , the database can jump directly to rows in the relevant ranges (for example, “at least 2 beds and 1 bath”), making search significantly faster and more scalable as the number of listings grows.

I also optimized queries in the properties backend API to reduce roundtrips and make better use of these indexes. The /api/properties endpoint builds a single SELECT that joins PROPERTY and ADDRESS and then appends optional filters (city, min/max rent, beds, baths) in the WHERE clause, allowing MySQL to apply the appropriate indexes rather than running multiple separate queries. For the listing details endpoint (/api/listing/<id>), we use the GET\_LISTING\_WITH\_LANDLORD stored procedure, which performs all necessary joins (property, address, landlord) inside the database in one call instead of making multiple app level queries. For reviews, we use INSERT ... ON DUPLICATE KEY UPDATE on the indexed

(USER\_ID, PROPERTY\_ID) primary key in the REVIEW table, which lets us handle “create or update review” in a single statement.

Together, indexing and query optimization have a positive effect on the performance and scalability of our application. Listing details and session creation are handled in stored procedures instead of multiple application-side queries, which reduces database roundtrips. Property search benefits directly from the new indexes on BEDROOMS, BATHROOMS, CAN\_RENT, and RENT\_COST, allowing MySQL to avoid full table scans on common filters. As the PROPERTY, REVIEW, and SESSIONS tables grow, index-based access keeps query performance predictable and efficient, which is especially important for frequently used paths like property search, loading listings, and authenticating users.