# Performance Optimization Report

**For Licensing Self-Certification Portal of Buildings Department**

**Version 0.1**

**Jan 2025**

? The Government of the Hong Kong Special Administrative Region

The contents of this document remain the property of and may not be reproduced in whole or in part without the express permission of the Government of the HKSAR.

## Distribution

| Copy No. | Holder |
| --- | --- |
| 1 | Buildings Department (BD) |
| 2 | Master Concept (Hong Kong) Limited (MC) |

## Amendment History

| Change Number | Revision Description | Pages Affected | Revision/Version Number | Date |
| --- | --- | --- | --- | --- |
| 1 | 1st draft | All | 0.1 | 16/01/2025 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Table of Contents

1. [Introduction](#introduction)
   * [1.1 Goal of Performance Optimization](#goal-of-performance-optimization)
     + [1.1.1 Server Loading](#server-loading)
     + [1.1.2 Bandwidth Usage](#bandwidth-usage)
     + [1.1.3 Better User Experience](#better-user-experience)
   * [1.2 Performance Optimization Actions](#performance-optimisation-actions)
   * [1.3 Storage Allocation](#storage-allocation)
   * [1.4 Required Response Time](#required-response-time)
     + [1.4.1 Online Transaction](#online-transaction)
     + [1.4.2 Online Reports](#online-reports)
2. [Critical Online Transition Timing](#critical-online-transition-timing)
3. [Critical Batch Cycle Timing](#critical-batch-cycle-timing)
4. [Optimization Changes](#optimization-changes)
   * [4.1 Optimization Actions](#optimization-actions)
     + [4.1.1 Create Stored Procedures](#create-stored-procedures)
     + [4.1.2 Create Clustered Indexes](#create-clustered-indexes)

## 1. Introduction

The performance optimization of the system focuses on improving Online Transaction performance. This report outlines the goals, actions, storage allocation, and required response times for optimizing the Licensing Self-Certification Portal.

### 1.1 Goal of Performance Optimization

The primary goal is to improve the system's response time for users. This involves optimizing program implementation to consider server loading, bandwidth usage, and overall user experience.

#### 1.1.1 Server Loading

Server loading is a critical factor affecting response time. It is influenced by:

1. The number of system users.
2. The efficiency of the programming code.

Optimizing programming code to use fewer server resources reduces server loading and improves response time.

#### 1.1.2 Bandwidth Usage

Bandwidth usage also impacts response time. It is affected by:

1. The number of system users.
2. The size of transmitting resource.

Reducing the size of transmitted data minimizes bandwidth usage, allowing the system to serve more users simultaneously.

#### 1.1.3 Better User Experience

The application's performance directly impacts the department's image. Good performance reflects positively on the department, while poor performance can lead to negative perceptions.

### 1.2 Performance Optimization Actions

To improve response time and serve more users concurrently, the following optimization measures can be taken:

* Optimize programming and query logic to reduce server loading.
* Optimize page and image sizes to reduce bandwidth burden.
* Improve resource retrieval speed by indexing and hashing.
* Cache frequently used resources.
* Pre-generate resources that require heavy instant server loading.
* Reduce waiting time for third-party services.
* Archive expired records to minimize the size of the active datastore.

### 1.3 Storage Allocation

Database data will be stored as follows:

* System data: Stored in the "Database" server in the Integrated system.
* Textual data: Stored in the "Database" server in the Integrated system.

All required system and textual data will be logically stored in various filegroups of the Microsoft SQL Server database.

The filegroup growth size is set to meet the recommendation of below 256 MB for data files.

### 1.4 Required Response Time

Required response times are defined for two categories: Online Transactions and Online Reports. Programs and reports requiring immediate user interaction should respond within the specified timeframes.

95% of all online functions should meet the committed response time.

The committed response time is affected by the network status of the site. The testing environment should meet the following network health level:

* Maximum number of concurrent users: 100.
* Minimal bandwidth: 2Mb/s per testing machine.
* Maximum network round-trip latency (ping) to the Integrated system: 200ms.
* Remote testing sites will have a **50% mark-up** time to the committed response time.

#### 1.4.1 Online Transaction

The following are classified as online transactions:

* User Account Program
* Form and Record Management Program

Online transactions are classified into the following groups:

| Transaction Complexity | Number of Concurrent Users |  |  |
| --- | --- | --- | --- |
|  | 40 | 80 | 100 |
| Simple | < 1 sec | < 2 sec | < 3 sec |
| Medium | < 2 sec | < 3 sec | < 4 sec |
| Complex | < 2.5 sec | < 3.5 sec | < 5 sec |

Online transactions can also be classified as follows:

| Online transactions | Description |
| --- | --- |
| Online Update Transactions | Used to update records in LSCP, for example, create supervision plan |
| Online Enquiry Transactions | Used to retrieve records from LSCP, for example, filter site monitoring records |
| Full-text Search | Used to search for records with given key words, for example, search assigned TCP |

#### 1.4.2 Online Reports

The programs in the following category are classified as online reports:

* [Report Types - To be defined based on system functionality]

As the report is a standalone internal system for BD, the measurement would only estimate when the concurrent users are not more than 20.

| On-line Reports | Committed Response Time |
| --- | --- |
|  |  |
|  |  |

## 2. Critical Online Transition Timing

The following matrix lists programs with their complexity and transaction type.

Offline modules allow users to save information locally when internet connectivity is unavailable. Once connectivity is restored, data is submitted and stored in the database. The asynchronous handling logic is included in functional testing. The data sending logic is similar to other APIs, so a separate section for offline modules is not included.

| **Module ID** | **Program ID** | **Program Name** | **Complexity (Simple / Medium / Complex)** | **Online Transaction Program** | **Report Program** | **Type of transactions** | **Mobile Apps** |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## 3. Critical Batch Cycle Timing

The following table lists batch programs and their cycle timings. These programs differ from the modules in the previous section as they run frequently (potentially thousands of times daily), while the modules in the previous section run less often (once or twice daily). Most of these modules are scheduled jobs running in the backend, so performance fluctuations typically do not affect end users. The generate report module has a higher impact and requires optimization.

| **Program ID** | **Program Name** | **Online Web** | **Mobile** | **Update Batch** | **Require Optimization?** | **Cycle Timing** |
| --- | --- | --- | --- | --- | --- | --- |
| **BATCH PROGRAM** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## 4. Optimization Changes

Optimization will **only** focus on the performance of programs and reports.

### 4.1 Optimization Actions

#### 4.1.1 Create Stored Procedures

All reports are created using stored procedures. Stored procedures are precompiled, unlike dynamic prepared statements that are compiled each time they are invoked. Once executed, a stored procedure remains in the cache, saving execution time. Stored procedures primarily use primary keys for searching.

#### 4.1.2 Create Clustered Indexes

When a primary key constraint is created, a unique clustered index is automatically created on the column(s) if one does not already exist. The primary key column cannot allow NULL values. A unique non-clustered index is created by default to enforce a unique constraint. When designing a clustered index, the data types used as clustering keys are considered. For example, primary keys are BIGINT data type, which is a good choice for a clustered index key.

| **Table ID** | **Table Name** | **LSCP Entity** | **Key Nature** | **Index Field** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Database Analysis Summary and Recommendations

Based on the provided database\_schema.md file, here's a summary of the database and potential optimization areas:

**Database Overview:**

* **Database Name:** bd
* **Size:** 88.10 MB
* **Collections (Tables):** 12
* **Total Documents (Rows):** 1,278,983
* **Total Data Size:** 371.24 MB

**Key Observations and Recommendations:**

1. **Large Collections:** sysfilerefs and adrblkfilerefs are significantly larger than other collections, both in document count and size.
   * **Recommendation:** Analyze the usage patterns of these collections. Consider archiving older data to reduce their size. Ensure appropriate indexes are in place for common queries. Review the data retention policies for these collections.
2. **Submissions Collection:** This collection is empty.
   * **Recommendation:** Investigate if this collection is intended to be empty or if there's an issue with data population. If it's unused, consider removing it.
3. **Mixed Data Types:** Some fields have mixed data types (e.g., tasks.user, eminutes.from, eminutes.to).
   * **Recommendation:** Standardize data types within fields to improve query performance and data integrity. Consider using a single data type (e.g., objectId or string) and converting data accordingly.
4. **Indexing:** The field analysis doesn't explicitly mention indexes.
   * **Recommendation:** Review query patterns for each collection and create indexes on frequently queried fields, especially those used in WHERE clauses, JOIN conditions, and sorting. Pay particular attention to indexing fields used in the sysfilerefs and adrblkfilerefs collections. Consider compound indexes for queries that use multiple fields.
5. **Average Document Size:** The cases collection has a relatively high average document size (2.65 KB).
   * **Recommendation:** Examine the structure of documents in this collection. If possible, normalize the data by moving some fields to related tables.
6. **Application Collection:** Contains many fields with null values and mixed types.
   * **Recommendation:** Review the schema and consider breaking down this collection into related tables for better normalization and to reduce the number of null values.
7. **Eminutes Collection:** Contains fields sysFileRefId that could be related to the sysfilerefs collection.
   * **Recommendation:** Consider creating a foreign key relationship and indexing the sysFileRefId field for faster lookups.
8. **Tasks Collection:** Contains application and submissionCase fields that could be related to the applications and cases collections.
   * **Recommendation:** Consider creating foreign key relationships and indexing these fields for faster lookups.

**Relating Database Analysis to Performance Optimization Report:**

* **Storage Allocation:** The database analysis provides insights into the size and structure of the data, which is crucial for planning storage allocation. The recommendations to archive data in large collections directly relate to optimizing storage usage.
* **Optimization Actions:** The recommendations for indexing and standardizing data types directly support the "Optimize programming and query logic to reduce server loading" action. Using stored procedures (as mentioned in the report) is also a good practice.
* **Critical Online Transition Timing:** Understanding the data structure and query patterns helps in identifying the most critical online transactions that need optimization. For example, if a complex query involves joining sysfilerefs and adrblkfilerefs, optimizing those collections and the join operation is crucial.

**Next Steps:**

1. **Query Analysis:** Use database profiling tools to identify the slowest queries.
2. **Index Optimization:** Implement the recommended indexes and monitor their performance.
3. **Data Archiving:** Implement a data archiving strategy for large collections.
4. **Schema Refinement:** Refine the database schema based on the analysis and recommendations.
5. **Performance Monitoring:** Continuously monitor database performance and adjust optimization strategies as needed.

<< End of Document >>