**Strategies to Reduce Load on Main Database when Using MicroStrategy**

**1. Introduction**

This document outlines various approaches for reducing the workload on a main SQL Server database when MicroStrategy (MSTR) is used for reporting. It covers both **MSTR-focused solutions** and **database-level solutions**, with a special focus on offloading resource-intensive queries and handling daily data refreshes.

**2. MSTR-Focused Approaches**

| **Approach** | **Description** | **Key Benefits** |
| --- | --- | --- |
| **Filters & Prompts** | Apply filters/prompts at the highest necessary level (e.g., monthly vs. daily) in MSTR. | - Reduces unnecessary data retrieval - Improves query performance |
| **Intelligent Cubes** | Load data (aggregated or detailed) once per day into in-memory cubes for reuse in multiple reports. | - Minimal database hits during user queries - Faster response times |
| **Scheduling & Report Governance** | Run heavy or batch reports during off-peak hours. Configure MSTR Query Governor to limit execution time/rows. | - Less contention with critical daytime processes - Prevents runaway queries |
| **Limit Return Rows** | Use Query Governor settings to cap rows returned or set a maximum execution time. | - Avoids large result sets - Protects database from resource overload |

**3. Database-Level Approaches**

| **Approach** | **Description** | **Key Benefits** |
| --- | --- | --- |
| **Separate Data Mart (Reporting DB)** | Move or copy data (via ETL/ELT) into a dedicated reporting database once per day. | - Offloads read-intensive queries - Isolates main database workloads |
| **Pre-Aggregated/Materialized Tables** | Create summary or aggregated tables; refresh them daily. MSTR queries these smaller tables instead of raw facts. | - Reduces query complexity and scanning - Speeds up reporting |
| **Indexed Views (Materialized Views)** | Physically store aggregated or filtered data in a clustered index; ideal if data is relatively static. | - Faster repeat queries - Avoids on-demand calculations |
| **Targeted (Filtered) Views** | Regular views that filter data (e.g., last 90 days) or expose only essential columns. | - Enforces row reduction - Simplifies reporting logic |
| **Partitioning Large Tables** | Partition big fact tables by date or another key, so queries scan only the relevant partitions. | - Improves query performance - Helps manage massive datasets |
| **Resource Governor** | A SQL Server feature that caps CPU/memory usage for certain workloads (like MSTR queries). | - Prevents MSTR queries from monopolizing resources - Improves overall DB stability |
| **Proper Indexing & Statistics** | Maintain up-to-date statistics and create indexes on frequently used columns for filters, joins, or grouping. | - Reduces full table scans - Helps the optimizer choose efficient query plans |

**4. Offloading Load from the Main Database**

| **Strategy** | **How It Offloads** |
| --- | --- |
| **Daily-Refreshed Data Mart** | Shifts heavy read/report queries to a separate DB, reducing main DB load |
| **MSTR Intelligent Cubes** | Once cubes are built, queries largely stay in MSTR’s in-memory engine |
| **Pre-Aggregated Tables / Indexed Views** | Users query smaller, aggregated data instead of large fact tables |
| **Scheduling & Resource Throttling** | Runs large queries during non-critical times, controls resource consumption |

**5. Choosing the Right Approach**

| **Scenario** | **Best Fit** |
| --- | --- |
| Main DB is busy, must isolate reporting | **Separate Data Mart** to offload resource-intensive reports |
| One-stop solution, simpler maintenance | **Pre-Aggregated/Indexed Views** in the same DB with daily refresh |
| MSTR has enough memory, daily data requirements | **Intelligent Cubes** to push query load into MSTR’s in-memory engine |
| Moderate data volume, need a quick fix | **Filtered Views** + **Proper Indexing** to trim unnecessary data scans |

**6. Summary and Recommendations**

1. **MSTR-Focused Solutions**
   * **Filters/Prompts**: Reduce volume by specifying only necessary data.
   * **Intelligent Cubes**: Build once daily, then rely on in-memory data.
   * **Scheduling & Governance**: Prevent large queries from running during peak hours.
2. **Database-Level Solutions**
   * **Separate Data Mart**: ETL the data once a day for reporting, removing read load from the main DB.
   * **Pre-Aggregated Tables**: Store summary data for faster queries.
   * **Indexed/Filtered Views**: Either physically store data (indexed views) or logically limit data (filtered views).
3. **Offload Main Database**
   * **Partition large tables**, maintain **proper indexes/statistics**, and use **Resource Governor** if necessary.
   * Combine daily refresh cycles with optimized data structures to ensure the main DB remains efficient.