SunGuide® Software System

Database Storage Guidelines

Version 3.0

October 6, 2011

Prepared for:

Florida Department of Transportation
Intelligent Transportation Systems Program
605 Suwannee Street, M.S. 90
Tallahassee, Florida 32399-0450
(850) 410-5600

DOCUMENT CONTROL PANEL		
File Name	2011-07-01_SunGuide_Database_Storage_Guidelines_v2-4.docx	
File Location	T:\Public\ITS\Software\SunGuide Software\Support\Database\Database Storage Guidelines\2011-07-01_SunGuide_Database_Storage_Guidelines_v2-4.docx	
Version Number	2.4	
Name		**Date**
Created By:	Steve Novosad, PBS&J	9/9/2010
Reviewed By:	TJ Hapney, PBS&J	9/28/2010
	Karen England, PBS&J	01/13/2011
	Clay Packard, Atkins	05/03/2011
	Arun Krishnamurthy, FDOT	05/23/2011
	Karen England, Atkins	05/24/2011
	Robert Heller, SwRI	06/23/2011
	Mark Laird, AECOM	08/04/2011
	Karen England, Atkins	01/22/2013
Modified By:	Clay Packard, PBS&J	9/27/2010
	TJ Hapney, PBS&J	9/28/2010
	Steve Novosad, PBS&J	9/29/2010
	TJ Hapney, PBS&J	9/29/2010
	Steve Novosad, PBS&J	11/29/2010
	Brian Ritchson, MCGI	3/31/2011
	Brian Ritchson, MCGI	05/03/2011
	Clay Packard, Atkins	05/03/2011
	Brian Ritchson, MCGI	05/23/2011
	Clay Packard, Atkins	05/23/2011
	Brian Ritchson, Atkins	06/28/2011
	Clay Packard, Atkins	07/01/2011
	Clay Packard, Atkins	10/06/2011
Completed By:	Karen England, Atkins	01/22/2013

Table of Contents

List of Tables

List of Appendices

List of Acronyms

CDW	Central Data Warehouse
DBMS	Database Management System
DMS	Dynamic Message Sign
FDOT	Florida Department of Transportation
FTE	Florida's Turnpike Enterprise
IT	Information Technology
LMT	Locally Managed Tablespace
SQL	Structured Query Language
TERL	Traffic Engineering Research Laboratory
TSS	Traffic Sensor Subsystem

1 Problem Statement and Purpose of Document

The database is a critical component of the SunGuide® software system. It fulfils several supporting activities including the primary role of persistent configuration data storage, such as the existence of devices and their communication parameters. Another activity is the computation and storage of operational data that may be reused in an operational fashion, such as Road Ranger locations and event details data. Archiving historical data is not as critical to operations; however, it is a part of the operations database. Thus, it is critical to maintain the health of the database and ensure that archiving data does not impact the other functions.

The database storage and archival analysis performed on the SunGuide software database is based on the existing District databases installed at the Traffic Engineering Research Laboratory (TERL). These databases were used to:

- Examine the database schemas design,
- Identify inefficiencies and their cause, and
- Provide recommendations that can be implemented to help eliminate these inefficiencies, reduce the probability of poor database performance (SunGuide software system and reporting) and data loss, and reduce the complexity of adding additional data elements to the database schemas.

There are several existing issues that have already been identified, including significantly large District database file sizes that cause issues in day-to-day operations and significant problems during a SunGuide software upgrade as well as slow running reports due to inefficient database design and extremely large table sizes.

Table 1.1 shows the total District database file sizes as of the beginning of 2011.

Table 1.1: Database Physical Sizes by District

District	Size (in Gigabytes)[1]
District 1	24.2
District 2	41.8
District 3	6.97
District 4	79.5
District 5	250.0
District 6	63.2
District 7	10.5
Florida's Turnpike Enterprise (FTE)	44.6

Note: [1] refers to sizes of databases. These values were measured from a reproduced database environment which may introduce variations from the production environment.

The highest known risk to the SunGuide software database, which has not been sufficiently addressed, is table growth; other more frequent, but less critical, issues occur that are handled by SunGuide software support. However, table growth is a non-urgent but very important issue for the long term that warrants investigation and mitigation to avoid any impact to the overall SunGuide software system. This issue is discussed in more detail in this document; strategies and considerations are proposed to address it. One specific area of interest is the storage of travel times. This function alone is a major contributor of the overall size of the SunGuide software database. There is also another specific issue related to travel times that is discussed due to its severe impact on reporting travel times.

After addressing the specific size and travel time-related issues, the overall health of the SunGuide software database is discussed in terms of backup and archiving strategies. This topic is addressed in a way to provide helpful guidance while respecting the operational differences in the information technology (IT) management groups amount the Districts. This document concludes with the solicitation of feedback in order to more fully address the size, safety, and health of the SunGuide software database throughout the state.

2 Database Issues

There are several issues that were investigated as a part of this analysis. Firstly, the schemas and their sizes were looked at and analyzed first, providing both a good overview of the database structure as well as some issues to address. Then, there were three additional, more specific issues that were also looked at.

2.1 Database Schemas Analysis

An analysis of a snapshot of the District databases housed at the TERL was performed to determine which tables have the greatest impact on the database due to size. These snapshots were obtained from the Districts from March to May of 2010. Both the FDOT_OWN and FDOT_ODS schemas were analyzed.

FDOT_OWN is the database used by SunGuide software in its normal day-to-day operations. This database stores data such as configurations, travel times, events/incidents, traffic sensors, etc. Many of the tables are static and rarely change in size since they store configuration data. Other tables, such as those related to events, grow significantly due to the large number of transactions that take place.

FDOT_ODS is the schema in the SunGuide software database used to archive data for future analysis and reporting. This database stores data such as traffic sensors, travel times, and dynamic message sign (DMS) data. Many of these tables grow rapidly, likely due to the number of traffic sensors deployed by a District and/or the frequency that travel times are calculated.

2.1.1 FDOT_OWN Analysis

Table 2.1 illustrates the table sizes from FDOT_OWN for each District. District 5 was chosen as the baseline for the table size due to the large amount of equipment it maintains. The ten largest tables for District 5 were found, for the most part, to be the same as those in other Districts, but in a different order according to size. Due to each District's different focus for SunGuide software, the order of the tables is different.

The top number in each cell is the number of rows that are contained in the tables. The bottom number is the estimated size of the table in bytes. The size was calculated based on the average row length reported from the Oracle Database Management System. The largest table for each District is highlighted in yellow.

Table 2.1: FDOT_OWN Largest Deployment Database Tables

Deployment / Table Name	District 1	District 2	District 4	District 5	District 6	District 7	FTE
DA_DEVICE_STATUS (Rows)	183,352	1,345,530	302,408	1,301,774	1,175,118	772,779	223,505
(Bytes)	18,518,552	137,244,060	31,752,840	131,479,174	113,986,446	80,369,016	22,797,510
EM_EVENT_CHRONO	144,584	486,991	2,667,685	515,605	1,794,633	1,124,233	108,353
	28,193,880	84,736,434	485,518,670	93,840,110	326,623,206 (see note below)	203,486,173	21,020,482
EMAUDT_EVENT_RESPONDER	235,214	651,588	5,999,455	836,743	3,447,138	1,809,192	189,785
	17,405,836	48,217,512	455,958,580	61,082,239	261,982,488	139,307,784	15,752,155
IDS_INCIDENT_ALARM	3,479	1,241	128,567	449,648	3,451	13,383	444,845
	313,110	134,028	15,556,607	51,259,872	393,414	856,512	50,267,485
EM_EVENT_RESPONDER	206,925	577,709	5,714,454	755,286	3,190,652	1,634,024	174,774
	12,001,650	33,507,122	348,581,694	45,317,160	85,057,816	101,309,488	12,059,406
CCTV_LOCK_USAGE	100,210	341,168	2,819,604	612,203	2,901,675	478,483	325,958
	6,814,280	18,081,904	140,980,200	39,793,195	145,083,750	27,752,014	21,187,270
EMAUDT_EVENT	66,140	175,881	1,061,294	221,931	739,168	400,616	47,695
	9,987,140	25,150,983	163,439,276	32,623,857	116,049,376	60,493,016	7,297,335
EMAUDT_EVENT_LOCATION	25,171	83,900	576,639	119,588	258,673	194,168	20,556
	7,073,051	21,981,800	167,801,949	31,092,880	73,463,132	50,677,848	5,981,796
IDS_TSS_ALARM_DATA	0	27	23	418,896	0	13,383	418,863
	0	1,323	1,357	30,579,408	0	1,070,640	30,158,136
AVLRR_VEHICLE_HISTORY	16,299	12,453	355,701	184,489	30,097	1,728,832	961,376
	847,548	560,385	19,919,256	16,972,988	1,414,559	124,475,904	75,948,704

Note: The largest table for District 6 was EVENT_HISTORY_ENTRY, which was not used by the other Districts with the exception of District 7. This implies that there is an inconsistency in the database schema for the Districts. In fact, the number of tables in the database for each District varies from 246 to 370 with no two Districts having the same number of tables. Some of this variability is due to tables created by Districts for use by local applications developed and supported by the District.

Table 2.1 provides a ranking of the largest tables for each deployment based on the total number of bytes (number of rows times average row length) for a table. The tables utilized were the ten largest tables from the District 5 database. Analysis shows that the largest five tables on average for all deployments are:

- EM_EVENT_CHRONO
- EMAUDT_EVENT_RESPONDER
- DA_DEVICE_STATUS
- EMAUDT_EVENT
- CCTV_LOCK_USAGE

The largest table for each District, highlighted in yellow, ranges in size from about 30 megabytes to almost 500 megabytes for each deployment. There are additional tables with similar byte volumes for several deployments. While none of the tables are overly large, there are some potential pitfalls that could develop over time if nothing is done to maintain the database. For example, the database could reach capacity. As a database reaches its maximum thresholds, commands will fail and produce errors that would not normally occur. If the database is not periodically optimized (i.e., indexes updated, tables reorganized), its performance will degrade over time. For example, if a table has millions of rows and has not been reorganized and the index recreated, it is possible that the database would have to perform a sequential row-by-row scan to find requested data. This type of scan could result in extremely long delays in obtaining the requested information.

The inconsistent growth of the tables is due to District operational differences. The Districts use SunGuide software in a variety of ways to manage their roadways and provide meaningful data to the operators. The tables in FDOT_OWN that are continually growing are tables that are not archived or periodically purged. The Districts need to examine these tables and determine the length of time they need the data to be maintained either for their operations or for data retention requirements and initiate a process to purge data that is older than the defined timeframe. Alternatively, the Districts could request that these tables be added to the Data Archive and stored long-term in a remote location such that the tables of the operational database could be kept as small as possible and older data could be moved to an archive database.

2.1.2 FDOT_ODS Analysis

Table 2.2 illustrates the table sizes from FDOT_ODS for each District. District 6 was selected as the baseline for the table sizes. Since most Districts have only 15 tables in the FDOT_ODS databases, with the exception of Districts 4 and 5 and since the table sizes reduce drastically after the first five to six tables, the analysis focuses on these first six tables.

This difference in table sizes again demonstrates how each District uses SunGuide software for its specific needs. The top number in each cell is the number of rows contained in the tables. The bottom number is the estimated size of the table in bytes. The size was calculated based on the average row length reported from the Oracle Database Management System.

Table 2.2: FDOT_ODS Largest Deployment Database Tables

Deployment / Table Name		District 1	District 2	District 5	District 6	District 7	FTE
ODS_TSS_LANES	**(Rows)**	960,519	1,198,737	1,203,055	4,370,421	63,999	1,037,541
	(Bytes)	79,723,077	62,334,324	91,432,180	288,447,786	4,479,930	70,552,788
ODS_TSS_LANE_POLL_DATA		17,940,645	33,744,246	51,752,139	71,993,380	19,926,722	34,671,294
		1,847,886,435	3,205,703,370	5,175,213,900	6,695,384,340	2,012,598,922	3,155,087,754
ODS_TSS_ROLLUP_DATA		83,313,781	207,818,606	100,378,962	52,086,928	26,943,366	16,599,054
		4,249,002,831	10,598,748,906	5,119,327,062	2,656,433,328	1,347,168,300	780,155,538
ODS_TRAVEL_TIME_INFO		6,640,227	44,508,891	646,904,627	30,267,630	48,837,584	3,014,310
		398,413,620	2,581,515,678	31,698,326,723	2,209,536,990	3,272,118,128	207,987,390
ODS_TSS_ROADWAY_LINKS		496,471	374,985	655,746	1,422,761	27,932	437,518
		34,256,499	21,749,130	46,557,966	88,211,182	2,039,036	31,938,814
ODS_DMS_MESSAGES		93,000	654,903	8,939,137	656,455	1,194,344	7,489
		14,136,000	80,553,069	1,206,783,495	59,080,950	181,540,288	561,675
ODS_TRAVEL_TIME_LINK		529	1,536	26,720	13,435	4,465	1,091
		60,306	90,624	2,217,760	1,222,585	339,340	79,643
ODS_TSS_DETECTOR_CONFIGS		217	839	1,383	1,464	2,181	2,176
		14,756	115,782	182,556	184,464	287,892	295,936
ODS_DMS_IDS		53	66	239	109	83	182
		2,385	1,980	10,516	3,161	2,988	6,734
ODS_DMS_LINKS		32	52	151	45	144	10
		576	884	3,020	765	2,592	180

Note: District 4 has no tables in the FDOT_ODS tables. District 6 does not include the ODS_TSS_TAG_READS table. All other Districts have the same schema for the FDOT_ODS database.

Table 2.2 shows similar statistics as the tables in the FDOT_ODS schema. Sizes for the largest of these tables range from two or three gigabytes upward to 30 gigabytes for each deployment.

Should these tables contain duplicate data or should tables contain duplicate rows, the performance of FDOT_ODS will be affected. The already large volume of data and the addition of complex queries that perform multiple joins, combined with duplicate data could cause the database performance to be extremely poor. Footprints issue 1589, discussed later in this document, is an example of the type of problem that can occur when data is duplicated.

The Data Archive Subsystem is the active SunGuide software module responsible for populating and purging data in these tables. Data Archive performs purging based on configuration parameters. The default and typically used configuration parameters for maintaining this data are as follows:

- For raw traffic sensor subsystem (TSS) data, the default value for maintaining the data is 14 days.
- For roll up data, the default value for maintaining the data is three years.

These default values accumulate large amounts of data. Maintaining this database will require that regular maintenance procedures be performed. Since the raw data is being constantly inserted into the database as well as having one day’s worth of data deleted each day, these tables will need to be reorganized and re-indexed on a periodic basis. Performing these activities will allow the database engine to more effectively manipulate the data as the tables are modified.

For data stored in the roll up tables, very large amounts of data will be accumulated. Some of the challenges of storing this volume of data are:

- Accessing the data to generate reports in a reasonable time frame,
- Providing logical space,
- Providing physical space,
- Deleting large amounts of data, and
- Maintaining (reorganizing) the database.

With the large volume of data potentially stored in the roll up tables, queries and reports will have to be carefully designed to take advantage of the ability of the database engine to optimize the queries. Complex queries, such as joining multiple tables, are not recommended due to the potential size of the dataset that will be returned. It is possible that the database engine could require more memory or temporary table space than is available to determine what dataset is to be returned.

When designing the logical layout of the database, one must consider how much logical space will be required and how often the database will have to extend its logical space to accommodate the amount of data it is storing. As the amount of data grows, the database engine may have to extend itself, thus adding more logical space. As tables grow in size, it is possible that a table may reside on more than one data file. While this is not a critical issue in and of itself, it can cause the database engine to have the table reside on different locations on the hard drive requiring more physical seeks on the disk and potentially affecting performance.

As part of the database planning and design, the physical space allocated for the database is crucial. In many cases, if the database runs out of physical size (i.e., can no longer logically extend), then the database most likely will have to be rebuilt, requiring the data from the existing database to be exported and then imported into the new database. This process is time consuming and expensive to perform. Careful planning must be done when allocating the database's physical size.

When large amounts of data are deleted, the transactions must be carefully planned. Deleting thousands or millions of rows of data may cause the database to perform poorly or cease to function. The result of these types of large transactions may affect the performance of the database following the completion of the transaction. The affected tables should be reorganized to create a situation where these tables are optimized for query execution. During mass deletion, if logging is not turned off, the log may fill up causing the database to potentially halt execution or return an error that it could not delete the data. This condition can occur because every delete would be recorded in the log; typically, log files are not designed to handle this type of large transaction. In addition, databases use temporary (undo) space to record the activity until it is committed. If the commits are not properly placed in the overall transaction, the temporary space could fill up causing the database engine to return an error and all of the transaction would be rolled back and no work committed.

There are several solutions to this issue. Instead of deleting millions of rows, the administrator could insert the rows to be kept into a temporary table. Once the temporary table is created, the original table could be truncated. The truncate command is functionally identical to delete except it deletes all rows and does so by de-allocating the associated data blocks. After the original table is truncated, the administrator would insert the rows from the temporary table back into the original table. This method is quicker, reduces logging, and puts much less data into the undo table. There are some cases where this method cannot be used, such as when the table is referenced by a foreign key restraint. In this case, an alternative method is to use a script that deletes rows in batches rather than all at once. This has the advantage of reducing the amount of data put into the undo table, but it is not nearly as fast as the first solution.

Within a database, if there are tables that are constantly changing (i.e., data being inserted and deleted) over time, these tables will become fragmented and perform poorly. The applications interfacing with the database should be examined to determine what impact they will have on tables. If tables are identified that are constantly modified, then these tables are good candidates for optimization/reorganization. Tables that are constantly modified with a large number of rows can cause poor performance even if they have indexes built on them. The reason for the poor performance is that tables are initially built in index sequential order; meaning that the rows are organized in sequential order based on the index. As rows are deleted, holes start to appear in the order, and as data is inserted it may be placed at the end of the table or it may be placed into one of the holes left from a deletion. This action is determined by the database engine. Eventually the table will become fragmented and queries executed against the table will take much longer because of the method by which the database engine has to search for the requested data. In order to minimize this fragmentation, these tables should be identified and regularly optimized/reorganized. When an optimization/reorganization is performed, the index is dropped; the data is unloaded from the table; the table is dropped and recreated; the data is reloaded in index sequential order; and the index is recreated. Depending on the volatility of the changes to a table, an optimization/reorganization may need to be performed as frequently as daily or as infrequently as once a month. Typically, reorganization requires that the database be taken offline because of the intensive database operations required to perform the reorganization.

Although these actions are necessary, there is an automated solution that comes close to removing all Oracle-based fragmentation. The District databases use locally managed tablespaces (LMT). This is the preferred setting for tablespace management. Currently, the extent allocation for almost all tables is controlled by the Oracle system, meaning that when a table needs to allocate a new extent, the system performs some calculations and chooses what it determines to be the correct size. By changing this setting to uniform extent allocation, there is only one size for every extent in that tablespace. Uniform extent allocation, in combination with LMT, eliminates almost all Oracle-based fragmentation. It is important to specify "Oracle-based fragmentation" because once Oracle outputs to the disk it is up to the operating system to decide how to store the data. In order to implement this solution all tablespaces would need to have their data preserved, contents dropped and recreated, and data reinserted as there is no way to alter this setting once a table is created. Database maintenance, optimization, and reorganization will still be necessary, but implementing this change will reduce the need significantly.

There is no current data showing this fragmenting issue occurring in SunGuide software currently; however, it is important for the system maintainer to keep this in mind when configuring and checking the health of the database periodically.

2.2 Configuration Records: Footprint 1589

The Current Data Archive system is responsible for managing how data is stored in the FDOT_ODS tables for historical data. Data Archive initially creates a configuration record for each detector, detector link, and each lane within the detector link. This configuration information describes the configuration, but does not include detected values each time (typically every 20 or 30 seconds) the detection is performed. As each set of detection values are inserted, the record storing these values references the records storing the configuration data. The first time a detection value is inserted, these configuration records are created. For subsequent detection values inserts, the configuration records are intended to be reused and referenced by each subsequently inserted detection record. However, this process is failing and a new, redundant set of configuration records are inserted rather than referencing the original configuration records. This is causing a tremendous number of redundant configuration records to be inserted. Some detection lanes have over 17,000 redundant configuration records associated with them, causing a size issue and, thus, performance issue as described throughout the earlier sections of this document. Moreover, as a table join is required to extract detection data for reporting, this issue is causing severe performance problems when running reports that utilize this data. Currently, when a report is generated for detection or travel time data for a period greater than three to five days, the time it takes to generate the report overwhelmingly exceeds the required maximum of one minute. This report generation can take well over 30 minutes, and in some cases, several hours.

In order to solve this problem, the following needs to be completed sequentially:

1. The defect of creating many, redundant configuration record needs to be resolved in the Data Archive Subsystem. Once resolved, this issue will be corrected moving forward, but the historical data will not be corrected.

2. The historical detector data needs to be corrected to properly reference a single, unique configuration record for each detector, link, and lane (there are three separate configuration tables for these three types of configuration items). This would require a database script to perform the following for each detector, line, and lane configuration record:

   A. Find the complete set of identical configuration records;
   B. Select one of them to represent the item; and
   C. Re-associate all other detection and configuration data records from that item to use the same configuration record rather than a redundant copy.

3. The purge script should be enhanced to remove all configuration records not being referenced by other configuration and detection data records. This will remove the redundant records immediately following the completion of the prior step as well as remove the configuration records no longer used as old data is purged in the future. After completion of duplicate entry removal, the tables should also be reorganized as well due to the many deletions.

2.3 Travel Time

Travel time data storage in the FDOT_ODS is the largest contributor to the SunGuide software database sizing issues. It would be reasonable to consider the need for storing this data into the FDOT_ODS and to consider not storing travel time data. Currently, the only purpose of the travel time data in the FDOT_ODS is to provide a data source for the Crystal Report templates that display travel times. The need for archiving the travel time data is satisfied via the data archive flat file. This flat file is much smaller, consumes no Oracle resources, and is much easier to maintain. A year's worth or more of the data would fit onto a single DVD for off-site storage.

The need to generate travel time reports must also be considered. The Central Data Warehouse (CDW) production system is in the concept and high-level design stage and will eventually serve this function. Specifically, the CDW will retrieve the travel time link configuration from the Districts and will recreate the travel time calculations based on the same configuration used by the Districts. A compromise could be to wait until the CDW is online and has proven its ability to meet this need. Another compromise would be to reduce the amount of data stored by decreasing the length of time for which data is kept. Typically, three years of travel time data is kept in the FDOT_ODS for Crystal Reporting; however, one full year - necessary for generating annual reports - may be sufficient and could be kept while the other two years could be purged.

2.4 Data Table Purge Scripts

Some portions of the SunGuide software data are purged by a script that is launched from the Data Archive Subsystem. This addressed some of the heavy contributors (as identified in section 2.1); however, there are some tables that are not purged. Table 2.3 shows the list of database tables that grow indefinitely and whether or not they are currently being purged by the purge script. The current purge script is executed from a location that is local to the ODS schema and does not purge any tables in the FDOT_OWN schema, but this concept could easily be applied to the FDOT_OWN schema as well.

Table 2.3: SunGuide Software Tables Purge Configuration

Table Name	Purged?
DA_DEVICE_STATUS	No
EM_EVENT_CHRONO	No
EMAUDT_EVENT_RESPONDER	No
IDS_INCIDENT_ALARM	No
EM_EVENT_RESPONDER	No
CCTV_LOCK_USAGE	No
EMAUDT_EVENT	No
EMAUDT_EVENT_LOCATION	No

Table Name	Purged?
IDS_TSS_ALARM_DATA	No
AVLRR_VEHICLE_HISTORY	No
ODS_TSS_LANE_POLL_DATA	Yes
ODS_TSS_ROLLUP_DATA	Yes
ODS_TRAVEL_TIME_INFO	Yes
ODS_TSS_LANES	No
ODS_TSS_ROADWAY_LINKS	No
ODS_DMS_MESSAGES	No
ODS_TRAVEL_TIME_LINK	No
ODS_511_INFO	Yes
ODS_TSS_TAG_READS	Yes

3 Backup and Archiving

3.1 Archive Guidelines

Archiving data is an important aspect of maintaining a manageable database. Archiving is the method of moving data out of the production database and physically storing it somewhere else. The data is removed from the database, reducing the size of the overall database without compromising the integrity of the database. Archiving is different from backups because the data is physically removed from the database; thus, optimizing the database and improving its performance. There are several questions to consider when planning data archival:

- How large is the database now?
- What are the largest tables?
- Will the tables continue to grow?
- At what rate will the tables grow?
- What are the long-term data storage requirements?

Moving large amounts of data out of the database and then reorganizing the database will improve the overall performance of the database. Regular monitoring of database performance will provide insight as to when the database needs attention. However, the most effective way to maintain database performance is to establish regular maintenance windows. Performing regular maintenance will provide a more consistent overall database performance.

3.1.1 Database Size

The overall size of the database is a factor in determining how often to archive data. If the database has never been archived and its size is reasonable, then archiving may not be necessary more than annually; whereas, if the database is very large, then more regular archiving (i.e., monthly, quarterly) should be considered.

3.1.2 Tables

Identifying the largest tables in the database is an important step in the analysis for archiving. There is no magic number of tables that should be archived; rather a threshold size should be established and the entire set of tables should be reviewed against that threshold. A typical threshold could be a table that exceeds a million or more rows. In fact, the threshold could be several million rows depending on how much data is stored in each row. After comparing the table sizes to the threshold, the number of tables to archive could be as little as one or as many as the entire set of tables. While a scenario of having an extremely large static table is rare, if such a table exists in the database, it likely does not need to be archived because it does not typically have a lot of transactions (inserts and deletes).

Tables that are archived should also be optimized (reorganized). The rationale for optimizing an archived table is that the primary index may no longer be effective when the table is queried. In order to return a table to an optimized state, the table must be reorganized. By performing this optimization, any queries on this table will return data in as efficient a manner as possible. Tables that have many inserts and deletes may not necessarily require archiving, but due to the large amount of activity in the tables, it will be necessary to regularly optimize the table. Regularly optimizing these tables will provide consistent performance for large queries and reports.

Performing database reorganization is not an Oracle-specific maintenance activity. The need for database reorganization has existed since databases were first developed, even before data management systems. As data is stored in some defined relationship to other data, links are established to facilitate the retrieval of elements throughout the structure. The database management system, which in this case is Oracle, must employ some algorithm to search for space when new elements are added and establish these links. Likewise, the links must be redefined when elements are deleted. The deletion of elements results in fragmentation of the disk space, which may or may not be efficiently reused when Oracle adds new elements. This fragmentation results in both decreased database performance and an increase in the amount of storage required. Over time, databases become increasingly less efficient as data is added and deleted. Reorganization is the method that reorders the data and re-establishes the links in the data in such a way that the fragmentation is eliminated. It results in optimized database performance and disk space. Oracle Tuning Pack, as one of it functions, assists with optimizing databases. Using the wizard you step through setting up the database reorganization to create scripts that can be executed using Oracle Tuning Pack to perform the reorganization. Oracle Tuning Pack analyzes and rebuilds fragmented indexes and tables, relocates objects to another tablespace if necessary, and recreates objects with optimal storage attributes. It is recommended that the database be taken offline to perform the reorganization as there will be a large number of disk inputs/outputs to locate the data optimally on the disk. Oracle Tuning Pack also provides the ability to optimize structured query language (SQL) statements by analyzing the current database schema design and providing recommendations to improve the performance of long running queries. SQL Access Advisor is another aspect of Oracle Tuning Pack that provides recommendations to optimize the schema design for maximum performance. SQL Access Advisor would be used when the schema is initially being designed.

All tables should be analyzed for their potential growth rate. If a table is identified as one that grows in size frequently, then it will be a candidate for archiving and optimization. If a table is identified as one that does not grow with regularity, then the table may not need to be archived. This analysis should be performed for each table that is known to not be static.

As part of the analysis for archiving tables, long-term data requirements must be identified. As part of this analysis, the amount of data that must be stored online and available for querying and reporting needs to be identified. If this information is not directly available, then a duration for storing the data needs to be identified. From the duration information, the amount of data can be estimated for that timeframe. This analysis should be performed for each table from which the approximate maximum database size can be estimated. From this analysis, the tables that will need to be archived can be identified. A strategy for archiving the data will be developed so that these tables maintain the appropriate data while the older data is archived. Some possible strategies are archiving data on a daily rolling basis, weekly rolling basis, monthly rolling basis, etc. The frequency with which the archive is performed will be based on how much data is added over a specified timeframe.

Based on the data taken for 20 days from Districts 4, 5, and 6, the SunGuide software FDOT_OWN database average growth will range from 2.5 gigabytes to 5 gigabytes per year. The variability in growth is due to the different operational differences that the Districts utilize.

3.1.3 Backup Guidelines

There are many options for backing up a database. In order to determine the right backup option, certain questions must be answered. These answers will shape the backup strategy that best fits the needs of the TMC.

- How much downtime is acceptable?
- How much are you willing to invest in hardware, software, and backup media?
- Do you require the backup to be automated?

- Backups can affect database performance. Are you willing to accept degraded performance on occasion?
- Is data primarily persistent with very little change?

As these questions are answered, the database strategy for the TMC will take shape. Answers to each of these questions are explored in the following sections.

3.1.3.1 ACCEPTABLE DOWNTIME

The amount of acceptable downtime is directly related to the type of backup that should be performed. If the goal is to have zero downtime, then a hot database backup is a solution to consider. A hot database backup produces the same results as a cold back up, but the database is allowed to run normally during this process. This is accomplished by freezing the system change number of each data file and writing all changes during a backup to the redo logs while the file is being copied. When the backup completes, Oracle sees that the data file is out of date and proceeds to apply all necessary updates in the redo log. If a day of downtime is acceptable, then one might consider performing a monthly database dump (full backup) and perform daily incremental (backup the log file) backups. Other backup options are:

- Establish a cold backup on standby. A cold backup is one where a backup database has been established on a backup server. Should the production database server fail, the latest backup and all incremental backups are restored to the backup database and the system is pointed to the cold backup database.
- Perform a database dump weekly with six incremental backups.
- Perform a database dump periodically (monthly or weekly) with no incremental backups.
- Create no database backup.

Establishing acceptable downtime is an important factor for database management and is inversely proportional to cost. The less downtime allowed, the more expensive the database backup strategy will be to implement.

3.1.3.2 INVESTMENT

When determining a database backup strategy, one must understand the cost involved when deciding the approach for backing up and restoring a database. If there is a high uptime requirement and the chosen solution is to maintain a mirrored database, then the hardware and software costs could be significant. For example, a tolling system requires essentially 100 percent uptime. Each transaction is executed on a primary and secondary database management system (DBMS). Should the primary fail, the system is automatically shifted over to the secondary DBMS. This implementation requires a significant investment in hardware and software. Hot database backups require no downtime and incur no extra costs. If the database contains primarily static data, a cold backup may be performed once a month; if the database fails between backups, it is acceptable to restore the previous month's database and accept the potential loss of several days of data.

3.1.3.3 AUTOMATED BACKUPS

Backups are time consuming and are typically performed in off-peak hours. Assigning staff to perform the backup ties up resources that could be used for other activities. Most backups have some level of automation to reduce the amount of staff interaction required. The more automated the backup is, the more the investment is in hardware, software, and media. A completely automated system could consist of hardware that provides a jukebox of recording media that is used to backup the database and the software required to perform the backups. This type of backup strategy would require a large investment in hardware, software, and backup media, but reduce the staff interaction required to periodically monitor the backups and replace media.

3.1.3.4 BACKUP TIME

A completed database backup can be a resource and time consuming process. During a complete backup, database performance can be degraded or the database may not be available at all. For these reasons, complete database backups are usually performed in off-peak hours (late night) when the demand on the database is low. Incremental backups are less intrusive on database performance, since these types of backups only create a backup of the current log file. The log file is closed out, a new log file is started, and the closed out log file is backed up. If incremental backups do affect performance to the point that the database is a bottleneck, these backups can also be performed in off hours.

3.1.3.5 DATA VOLATILITY

It is important to understand how, why, and for how long data is stored in the database. There are three types of data volatility discussed:

- Persistent and rarely changed data,
- Critical and regularly changed data, and
- Nomadic data (data that may change frequently, but is not critical to the operational success of the system).

Most databases have some combination of the data volatility discussed above. The backup strategy will be developed based on the database's primary function for storing the data. Data that is critical and regularly changes will require a more frequent backup strategy, while data that rarely changes or is nomadic may require a backup strategy that is periodic in nature.

SunGuide software databases are primarily persistent with small modifications (add new equipment) and nomadic data, such as speed information on links (periodically updated).

However, it also contains critical and regularly changing data with regards to event management which is constantly changing and is used in performance measures reporting.

4 Compression

One possible tool for reducing database size is compression. Compression reduces the size of a portion of the database, but incurs more overhead. Oracle compression can reduce the disk space used by up to one third. Although throughput is reduced with compression, full table scan speeds are increased because less data must be pulled into memory. Many reports in SunGuide software use full table scans, so it is safe to say that compression may benefit SunGuide software operations in the future. Careful consideration of these performance/storage tradeoffs is required before any modification to the SunGuide software database can be made.

5 Conclusion/Summary/Recommendations

Throughout this document, recommendations for improving the SunGuide software database are provided directly or implied. This section compiles the recommendations for improving the overall health and performance of the database. The recommendations are provided in a list format for the reader's convenience:

5.1 Central Office Recommendations

- Consider reducing the retention of travel time data stored in the FDOT_ODS from three years to one. The data could be preserved by sending it to the CDW or other long term storage. This change could reduce the size of relevant FDOT_ODS tables by 66 percent.
- Review the need for storing travel time data in the FDOT_ODS and consider eliminating this function. Archiving travel time data is accomplished by writing the data to an archive flat file. Also, consider whether the travel times can be reliably recreated from historical speed data. ODS_TRAVEL_TIME_LINK and ODS_TRAVEL_TIME_INFO are the two tables that store travel times. Table 2.2 shows the number of rows and size in bytes in these tables. This information can be referenced to show the database savings if travel time storage was reduced or eliminated. District 5 has the most travel time data with over 0.6 billion rows occupying over 31 GB of storage in their travel times, while FTE only has about 3 million rows occupying 0.2 MB of storage. The variability of different District's needs must be considered as well, as the solution may be required to include this as a configurable option at each SunGuide software deployment.
- Review the storage configuration. One option for older archived data is to move the data to less expensive storage such as a hard disk directly connected to the server rather than an expensive SAN drive. Table partitioning could be used such that all of

the data was available fro querying, while a smaller amount of data is in use in the production system.

- Change the extent allocation for LMTs from system controlled to uniform.
- Remove the daily and hourly intervals from ods_tss_rollup_data since they can be derived from the 15 minute rollups. This will reduce the amount of data stored in this table by 20.6 percent while maintaining current data granularity.
- Reduce the number of days maintained in ods_tss_lane_poll_data from 14 to two days. This change would reduce the table size by 86 percent.
- Enhance the purge scripts to add missing tables from section 2.4. The length of time to keep data for each table could be individually configured by each District based on their particular needs.
- Enhance purge scripts for the ability to automatically or manually purge/archive data based on filters such as on a per detector bases.
- Develop a script or manual procedure to periodically review the health and status of the database, the amount of available free space in the physical storage, and other metrics including the frequency of extending tables, etc.

5.2 District Recommendations

- Districts are recommended to have staff with database administration expertise to support, operate, and maintain the SunGuide software system's database.
- Tables with frequent inserts and deletions should be reorganized on a periodic basis to ensure that performance is maximized. Oracle Tuning Pack can be used to assist in determining which tables require reorganization and in building scripts to perform the reorganization. Since the database would be unavailable during the reorganization, the time and frequency of the reorganization should be carefully planned. The reorganization should be performed in off peak hours, such as late night, and no more frequently that once a month. This recommendation applies to both database schemas, FDOT_OWN and FDOT_ODS.
- If a District is not using logging, a full database backup should be performed, logging initiated, and the log backed up on a daily basis.
- If feasible, in coordination with the reorganization, a complete database backup should be performed. The previous six months or more of full backups and daily log backups should be maintained.
- Set detector poll cycles to a minimum of 30 seconds. If the poll cycle was previously set to 20 seconds before this change, then the size of the ods_tss_lane_poll_data table would be reduced by 33 percent.

Appendix A

Deployment Questionnaire

Appendix A: Deployment Questionnaire

1. What are your current backup methods? For example, do you perform a full backup once a week and then incremental backups the remaining six days?
2. Where do you store your backups?
3. What do you do with the data archive flat comma separated value files? How long do you keep the flat files?
4. How long do you keep your data?
5. Would you be interested in storing data off-site, long-term (e.g., CDW)?
6. Are you currently using Oracle Tuning Pack for optimization or reorganization within your database?
7. What is the maximum acceptable time (which would imply lost data) for your operations during any of the aforementioned maintenance tasks?