

# iMIS DB Maintenance

Doug Morris, Computer System Innovations



# Agenda

- SQL Server
  - Memory
    - x32 vs. x64
  - Processors
    - Single/Dual/Multi-Core
  - Drive Subsystems
    - Raid Levels

# Agenda

- Backups/Best Practices
  - Recovery Model
  - Doug's backup plan
  - Log backup files on separate drive array
- Maintenance Plans

# Agenda (continued)

- Indexes
  - Details/Background Info
  - iMIS Name\_Indexes vs. SQL
  - Manually re-indexing
  - Getting Dirty with Bruce Wilson

# SQL Server 2005

Configuration	VAS	Max physical memory	AWE/locked pages support
Native 32-bit on 32-bit OS	2 GB	64 GB	Yes
with /3GB boot parameter	3 GB	16 GB	Yes
32-bit on x64 OS (WOW)	4 GB	64 GB	Yes
32-bit on IA64 OS (WOW)	2 GB	2 GB	No
Native 64-bit on x64 OS	8 terabyte	1 terabyte	Yes
Native 64-bit on IA64 OS	7 terabyte	1 terabyte	Yes

VAS = Virtual Address Space

# 32-bit vs. 64-bit

- SQL 2005 can only effectively use 2-3GB of memory in 32-bit mode
- iMIS 10.6.29.02 and higher will run on SQL 2005 x64 on a Windows 2003 64-bit server
- Of course this is not supported by ASI....
- However, you can load up your SQL 2005 64-bit server with memory (on the cheap) and solve a LOT of performance problems....

# SQL Server Processors

- For most of us, a single processor is just fine
  - Today's processors are often dual and quad core, giving the performance effect of a multiple processor system
- Dual *Physical* processors rarely hurt performance, however, web accessed SQL Servers must be licensed per Physical processor

# SQL Server Processors

- A SQL 2005 Physical Processor license costs \$6,000 USD
- So, that “free” extra processor you got from Dell, will cost you upwards of \$6,000
- That’s one MEEELLION AUS Dollars!!! and TWO MEELLION Kiwi Dollars!
- That money is better spent in drive subsystems and iMIS Consultants

# Drive Subsystems

- Drive Subsystems/IO are also one of the three critical factors in selecting a SQL Server
- SQL Server writes data to a LOG file and then the data is written to the Data file
- So that record you just saved in iMIS, wrote to the log file and then LATER wrote to the data file

# Drive Subsystems

- If the server can write and read data from a log file at the same time it writes and reads data from a data file, it performs best
- Therefore, your log and data files should be on separate PHYSICAL drives/drive systems

# Drive Subsystems

- **RAID 0**
  - Data is striped across two or more drives
    - The more drives the faster
- **RAID 1**
  - Data is mirrored across two or more drives
- **RAID 5**
  - Data is striped across three or more drives with parity
    - The more drives the faster
- **RAID 10**
  - Data is striped and mirrored across four or more drives
    - The more drives the faster

# Drive Subsystems

RAID Levels	RAID0	RAID1	RAID5	RAID10
<b>Reliability</b>	Lowest.  Lack of fault tolerance results in data loss.	Very good.  Even better with duplexing.	Good.  Can tolerate single machine fault.	Excellent.
<b>Storage Efficiency</b>	100%	50%	>50%, <100%  (#drives - 1/#drives)	50%
<b>Random Read</b>	Excellent	Fair  Worst of the RAID levels but better than a single drive.	Excellent.	Excellent.
<b>Random Write</b>	Excellent.	Fair.  Worse than a single drive but better than some RAID levels.	Fair.  Generally better with larger stripe sizes.	Very good.
<b>Sequential Read</b>	Excellent.	Fair.  Comparable to a single drive.	Very good.  Generally, better with smaller stripe sizes	Excellent.
<b>Sequential Write</b>	Excellent.	Good.  Better than other RAID levels.	Fair.	Very good.

# Drive Subsystems

RAID Levels	RAID0	RAID1	RAID5	RAID10
<b>Cost</b>	Lowest.	Moderate.  Relatively high cost due to redundant drives; however, no expensive controller required	Moderate.	High.
<b>Recommended use</b>	Good for non-critical data or stagnantly updated data that gets backed up regularly or any data requiring fast write performance at very low cost. Great for testing.	Good for data that requires high fault tolerance at relatively low hardware cost (redundancy using parity requires more expensive hardware). Best for log files.	Very good for Read only data.	Data requiring high performance for both read and write and excellent reliability while trading off storage efficiency and cost.

# Drive Subsystems

- A baseline performance orientated SQL Server has both a RAID 5 (for Data) and a RAID 1 (for Logs)
- A better SQL Server has each of those on it's own separate controller
- A REALLY KICK BUTT Server, has a RAID 10 for the Data and a RAID 1 for the Logs *each with its own separate controller!*

# Drive Subsystems

- A single RAID 10 drive subsystem performs very well
- A RAID 5 drive system performs fair at best – but performs better with more drives
- A single drive or drives system is not so good

# Drive Subsystems - Summary

- Simply upgrading and/or changing your SQL Server drive subsystem can have a profound effect on the performance of your server.

# Recovery Model

- There are three recovery models the database can be in:
  - Simple
    - Log file is truncated every 15 seconds or so
  - Bulk Logged
    - Bulk operations such as import/insert are not logged
  - Full
    - All operations are logged

# Recovery Model

- Each model has its advantages
  - Simple
    - No maintenance is required
  - Bulk Logged
    - Log file does not grow like crazy on large inserts
  - Full
    - Ultimate recoverability

# Recovery Model

- Each model has its disadvantages
  - Simple
    - No ability to recover to a point in time since last backup
  - Bulk Logged
    - Ability to recover is lost during large insert/import
    - Maintenance on log file (truncation/backup) is required
  - Full
    - Maintenance on log file (truncation/backup) is required

# Recovery Model

- From a performance standpoint
  - Full and Bulk Logged are faster
  - Simple is actually slower since the data in the log file has to be cleaned out on a regular basis

# Backup Plan

- A DB Maintenance Plan can be configured to do all of your work
- However, MY preference is as follows
  - No use of ARCServe/Backup Exec agents
  - Nightly DB Backup to Data backup device *prior* to tape backup
  - Morning Backup of Log File to Log backup device located on a separate server if possible
    - That allows best chance of recovery

# Backup Plan

- Hourly backup of log to Log Device up until the nightly backup occurs
  - This prevents the log file from growing and filling up your server!
- My plan gives you ultimate recoverability and the ability to know where your last data backup is....

# Maintenance Plan

- A maintenance plan will allow you to keep more backups and have them automatically delete after a certain number of days
- Alas sometimes it is difficult to determine what the last backup is and you have less control
- So my way is better.....

# DB Backup/Main Plan Walk Thru

- Using the maintenance plan, create a maintenance plan that not only backs up the database but checks the database for minor issues

# SQL Indexes

- SQL Indexes help SQL return MUCH data faster
- However they actually slightly slow down inserts because of their overhead
- A clustered index sorts the data in order of the index

# SQL Indexes

Raw Data  
before  
Index

ID
900
800
990

Raw Data before  
after Clustered  
Index created on  
ID

ID
800
900
990

# iMIS Indexes

- ASI has determined the best indexes for your iMIS tables
- You should not mess with them – unless there is a major issue (normally ASI will instruct you on what to create)
- However, for any of your custom tables, you should create indexes as required

# Indexes on your table

- Your clustered index should be on the most frequently used field(s) for lookups
- Other indexes should be created for performance issues (when you have a lookup)
- **WITHOUT** an index, SQL does a table scan (it looks at every darn record in the db)

# How Indexes Work

An index on  
**Activity\_SEQN** might  
store lookup data  
(internally) like this

Trans Table Might Look Like This

ID	Trans Date	Product Code	Amount	ACTIVITY_SEQN
1000	4/1/2005	DUES	100	7992
1001	31/12/2006	MEETING	400	84922
1002	15/1/2005	ORDER	300	12321
1003	11/11/2006	MEETING	300	238492
1004	1/1/2005	ORDER	800	3299
1005	10/12/2004	CALL		234
1006	20/1/2005	DUES	200	5923
1007	1/12/2005	DUES	200	69239
1008	12/12/2006	MEETING	400	912311
1009	12/11/2006	ORDER	10	9262
1010	14/10/2006	CALL		4812
1011	30/10/2006	MEETING	300	57923
1012	12/12/1004	ORDER	10	1999

ID	ACTIVITY_SEQN
1005	234
1012	1999
1004	3299
1010	4812
1006	5923
1000	7992
1009	9262
1002	12321
1011	57923
1007	69239
1001	84922
1003	238492
1008	912311

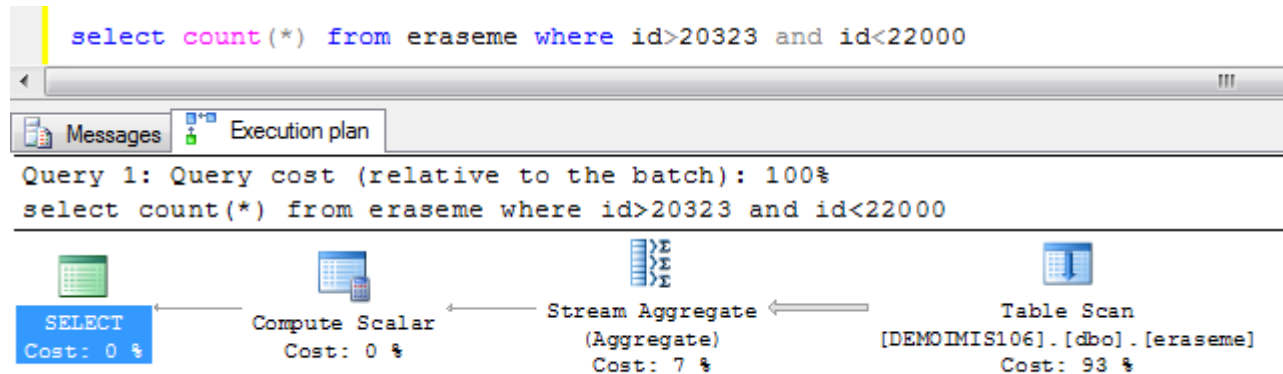
This allows SQL to find any record based on  
**ACTIVITY\_SEQN** FAST!

# How Indexes Work

```
-- Create Table with 300,000 IDs
create table eraseme(ID int)
go
declare @ID int
set @ID=1
while @ID<300000
begin
insert eraseme(ID) select @ID
select @ID=@ID+1
end
```

# How Indexes Work

Enter the following in Query Analyzer and press Ctrl + L



Note the table scan – THIS IS BAD

# How Indexes Work

Now run the following Query

```
create clustered index iID on eraseme(ID)
```

Now try your query again – note the index is used!

```
select count(*) from eraseme where id>20323 and id<22000
```

Query 1: Query cost (relative to the batch): 100%

```
select count(*) from eraseme where id>20323 and id<22000
```

Execution Plan:

- SELECT (Cost: 0 %)
- Compute Scalar (Cost: 0 %)
- Stream Aggregate (Aggregate) (Cost: 13 %)
- Clustered Index Seek ([DEMOIMIS106].[dbo].[eraseme].[iID]) (Cost: 87 %)

# Try this to see the improvement

```
Drop index eraseme.iID
go
dbcc freeproccache
dbcc dropcleanbuffers
go
declare @now as datetime
select @now=getdate()
select count(*) from eraseme where id>20323 and id<22000
select getdate()-@now
```

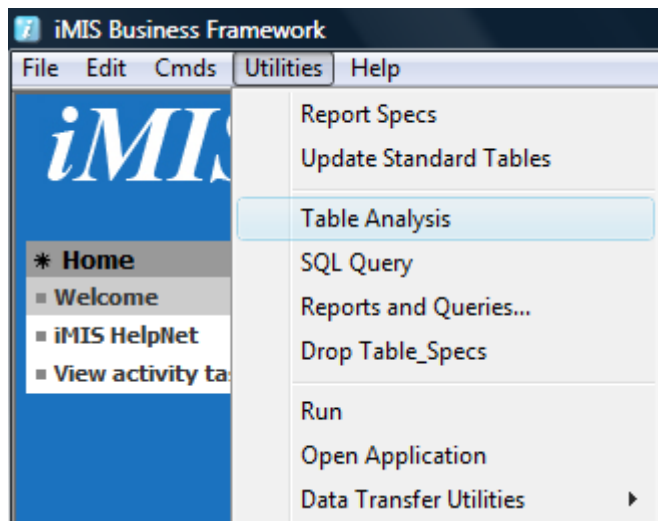
# Now with the Index on

```
create clustered index iID on eraseme(ID)
go
dbcc freeproccache
dbcc dropcleanbuffers
go
declare @now as datetime
select @now=getdate()
select count(*) from eraseme where id>20323 and id<22000
select getdate()-@now
```

# SQL Index Maintenance

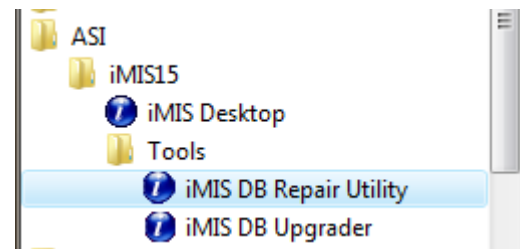
10.6

Login as Manager!sa  
Utilities – Table Analysis



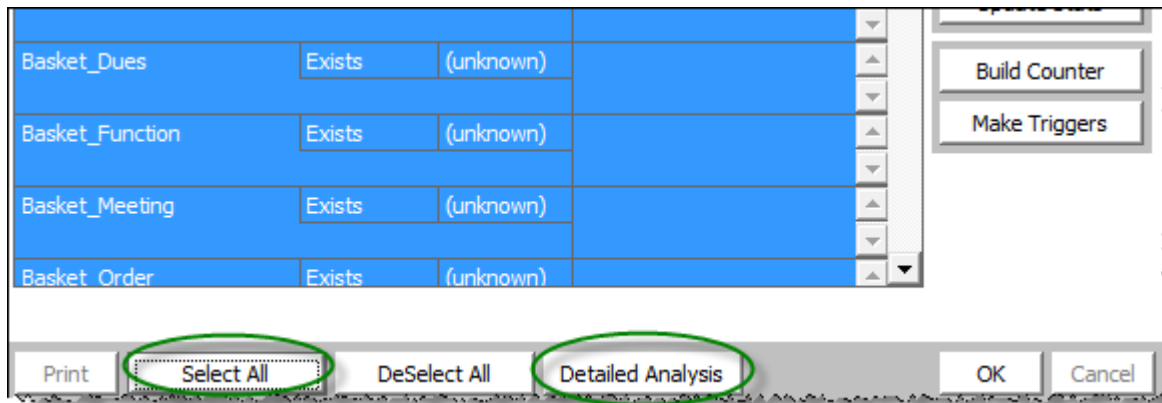
15.x

Start – Programs – ASI –  
iMIS15 – Tools – iMIS DB  
Repair Utility



# SQL Index Maintenance 10.6

1. Select All Tables
2. Detailed Analysis





# SQL Index Maintenance 10.6

Correct the issue by dropping any existing indexes and then recreating them  
Make sure you have the appropriate table highlighted  
And people SHOULD be out of iMIS (but they don't have to be)

The screenshot shows the SQL Index Maintenance tool interface. A table is listed with columns for Name, Status, and Error. The 'Name' row is highlighted in blue and shows an error: 'missing index: pkNameID, missing index: ...'. To the right of the table is a vertical toolbar with several buttons: 'Make Indexes', 'Drop Indexes', 'Update Stats', 'Build Counter', and 'Make Triggers'. The 'Make Indexes' and 'Drop Indexes' buttons are circled in green.

Name	Status	Error
Meet_Resources	Exists	(unknown)
Meet_Std_Resrc	Exists	(unknown)
Member_Types	Exists	(unknown)
Mem_Trib_Code	Exists	(unknown)
Name	OK	Error missing index: pkNameID, missing index: ...
Name_Address	Exists	(unknown)

# Name\_Indexes vs. SQL

- Do not confuse the Name\_Indexes table (used by iMIS for lookups) with SQL Indexes.
- Name\_Indexes is a table in iMIS used for lookups – it functions somewhat similar in concept to SQL indexes, yet, it is an iMIS maintained one and needs to be populated by YOU, not SQL.

# Name\_Indexes

1. Identify the field you want your users to be able to search on. Tell iMIS about this field by going to Customers – Set up module – Indexes

The screenshot shows the iMIS software interface. The main window is titled "Customers" and has a navigation bar with "Home", "Customers", "Billing", "Events", "Fund Raising", "AR/Cash", "Certification", and "Expos". The "Customers" module is selected, and the "Set up general options" menu is open. The "Indexes" option is selected, and the "Customer Setup - Indexes" dialog box is displayed. The dialog box has a table with the following data:

	Prompt	Field/Formula
Index 1	State	Name.STATE_PROVINCE
Index 2		
Index 3		
Index 4		

Below the table, the text "Record was modified." is displayed. The dialog box also has "Print", "Save", and "Cancel" buttons.

# Name\_Indexes

2. Enter the field you want to be able to search on – below we are replacing State with Email and adding in Name\_Demo.SPOUSE so we can look up individuals by their spouse name

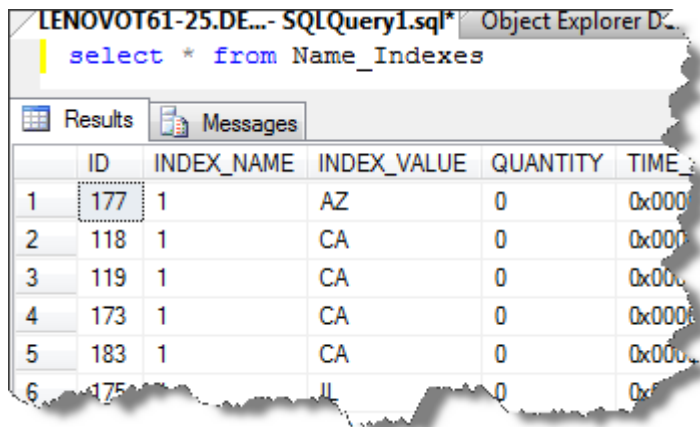
	Prompt	Field/Formula
Index 1	Email	Name.EMAIL
Index 2	Spouse	Name_Demo.SPOUSE
Index 3		
Index 4		

Record was modified.

Note: only single instance fields (from Name and Demographic tables can be used)

# Name\_Indexes

Now, any *new record or updated record* can be found by email. But what about all of the others? Must create script to back populate Name\_Indexes table for existing records. It helps to first understand how Name\_Indexes stores data for lookups.



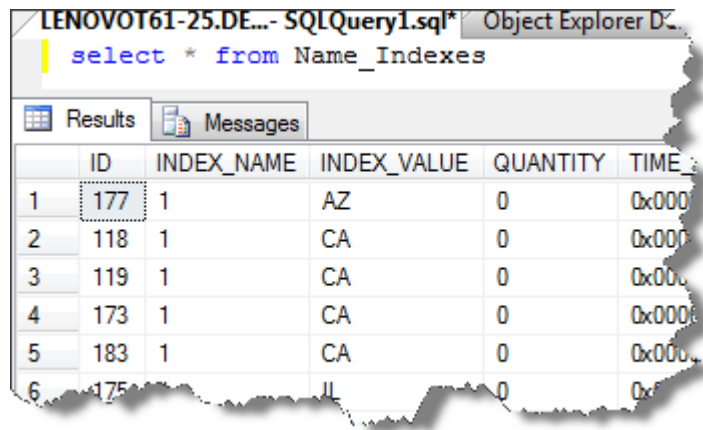
The screenshot shows a SQL query window with the following text: `select * from Name_Indexes`. Below the query, a table of results is displayed with the following columns: ID, INDEX\_NAME, INDEX\_VALUE, QUANTITY, and TIME. The table contains six rows of data.

	ID	INDEX_NAME	INDEX_VALUE	QUANTITY	TIME
1	177	1	AZ	0	0x0000
2	118	1	CA	0	0x0000
3	119	1	CA	0	0x0000
4	173	1	CA	0	0x0000
5	183	1	CA	0	0x0000
6	175	1	JL	0	0x0000

- ID is the ID of the record
- INDEX\_NAME Corresponds to the Index# (State was 1, now Email will be 1 and Spouse will be 2)
- INDEX\_VALUE is the actual data (the state in this case)

# Name\_Indexes

So when I search for everyone in AZ, I get all of the IDs back that have that value in INDEX\_VALUE



The screenshot shows a SQL Server Object Explorer window with a query window titled "LENOVOT61-25.DE...- SQLQuery1.sql\*" and "Object Explorer D...". The query window contains the SQL statement: `select * from Name_Indexes`. Below the query window, there are tabs for "Results" and "Messages". The "Results" tab is active, displaying a table with the following data:

	ID	INDEX_NAME	INDEX_VALUE	QUANTITY	TIME
1	177	1	AZ	0	0x0000
2	118	1	CA	0	0x0000
3	119	1	CA	0	0x0000
4	173	1	CA	0	0x0000
5	183	1	CA	0	0x0000
6	175	1	JL	0	0x0000

# Name\_Indexes

3. Now let's back populate the data – we want to put the emails in INDEX\_NAME 1 and the Spouse Names in INDEX\_NAME 2

# Name\_Indexes

- Utilities – Report Specs – Select SYSTEM-System Setup –Rebuild Name\_Indexes
- Press Clone and change the name to Rebuild Name\_Indexes (your org name)
- Press Parameters

Report Specifications

New Open Edit Delete Find

AR-Account Status- A/R Statements  
AR-Account Status- A/R Statements (Crystal)  
AR-Account Status- A/R Statements (continuous)  
AR-Account Status- A/R Statements (laser)  
AR-Account Status- A/R Statements PO (Crystal)  
AR-Account Status- A/R Statements SFC ASI-010-C  
AR-Account Status- A/R Statements SFL ASI-010  
AR-Account Status- Selected Invoices

Hide Omnibus Duplicates  
Show Omnibus Duplicates

System SYSTEM Category System Setup  Hide from Menu  
 Request Printer Setup

Title Rebuild Name\_Indexes (your org name)

Description  
This will be our custom Name\_Indexes population utility

\*This is a User Defined Report

Print Clone Parameters Options Save Cancel

# Name\_Indexes

- Click in the Parameters box and press Ctrl + Z
- Find the following code and fix it!

```
insert Name_Indexes
  ( INDEX_NAME ,
    INDEX_VALUE ,
    ID )
select '1' ,
       upper ( Name.STATE_PROVINCE ) ,
       Name.ID
from Name
where  Name.STATE_PROVINCE <> ''
go
```

# Name\_Indexes

- New Code below

```
insert Name_Indexes
(INDEX_NAME,
INDEX_VALUE,
ID)
select '1',
upper(Name.EMAIL),
Name.ID
from Name
where Name.EMAIL <> ''
go
```

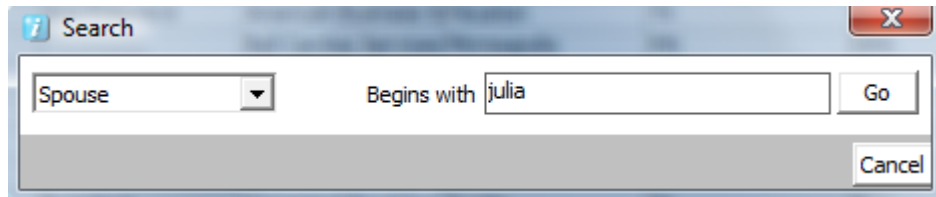
```
insert Name_Indexes
(INDEX_NAME,
INDEX_VALUE,
ID)
select '2',
upper(Name_Demo.SPOUSE),
Name_Demo.ID
from Name_Demo
where Name_Demo.SPOUSE <> ''
go
```

# Name\_Indexes

- Click OK, Save
- Now run your new script – Utilities – Reports & Queries – System Setup
- Select your Rebuild Name\_Indexes (your org name)
- Click Run
- Try your handy work out!

# Name\_Indexes

Check your work (don't forget to exit and restart *iMIS* first)



A screenshot of a search dialog box titled "Search". The dialog box has a close button (X) in the top right corner. It contains a dropdown menu with "Spouse" selected, a text input field with "Begins with" and "julia", a "Go" button, and a "Cancel" button at the bottom right.

# Now let's get down and dirty

Special Thanks to Bruce Wilson of RSM McGladrey

- Dbcc showcontig()
  - Scan Density is the most useful statistic. 75% or above is acceptable.
  - It's very common for the [Best Count] number to decrease dramatically after a reindex. This means disk space was reclaimed.
  - Avg. Page Density (full) after a reindex will be somewhere near fillfactor. May be lower than expected if the table has wide rows.

# Manually Re-Indexing Tables

- Set a fill factor for each table thoughtfully according to its usage. Set most tables to 90 (rarely insert or grow in the middle), 75 (sometimes grow), 50 (frequently grow) or 10 (new table with lots of additions expected).
  - Bigger fill factor = less disk space used = less disk activity, but also less room for new rows in the middle or for rows to grow. Smaller fill factor = more disk used but less impact from large, scattered inserts.
  - Name family is clustered on ID, which increases monotonically, and rows rarely get fatter. Fill factor 95.
  - Trans records never change size after write, but occasionally meetings transactions get more lines. Fill factor 95+.
  - Activity clusters on ID, but amount of growth before next reindex probably won't grow too significantly. Usually around 75.

# Manually Re-Indexing Tables

- Set a fill factor for each table thoughtfully according to its usage. Set most tables to 90 (rarely insert or grow in the middle), 75 (sometimes grow), 50 (frequently grow) or 10 (new table with lots of additions expected).
  - GUID clustered tables 50 or 75 depending on current size relative to expected short-term growth.
  - System\_Params, Gen\_Tables, Product: Some growth by editing descriptions, but generally high like 90.

# Manually Re-Indexing Tables

- `dbcc dbreindex(Name, “”, 90)`

or

- `alter index all on Name rebuild with (fillfactor=90)`

or (less invasive...)

- `dbcc indexdefrag (0,Name)`

# Other Bruce Tips..

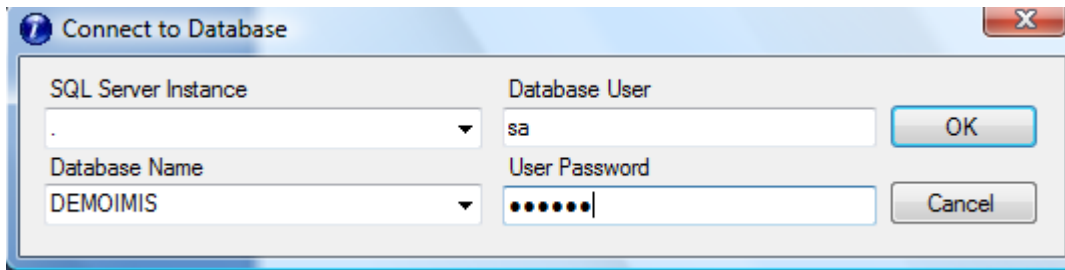
- Make sure every disk volume is 15% free or more. Standard Microsoft recommendation. NTFS performance goes in the toilet above 85% full.
- Use the highest db compatibility level supported by your version of iMIS.
  - When moving to 90, remember to `dbcc updateusage(dbname)` and `sp_updatestats` to take advantage of index improvements.
- Disable autoshrink on every database. In that ever-so-rare case when a database shrinks, you'll know it and can shrink it yourself.
- Disable autoclose on every database. Your production databases will never use it, and your non-production databases won't matter. The option is meant for embedded SQL use, which iMIS is not.

# Other Bruce Tips..

- Set autogrow size to a fixed amount for each file (e.g. 100MB), then check your database often enough so it never has to autogrow.
- It's faster to command your database to grow than let it autogrow, because autogrow will always make the query that needed space wait.
- Rule of thumb: keep 20% free space for normal operations in each filegroup. 50% before an upgrade to iMIS15 or loading .NET on 10.x.

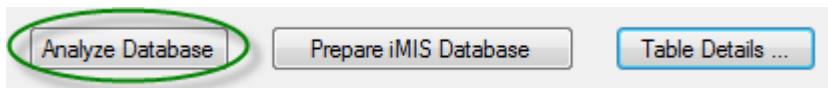
# SQL Index Maintenance 15.x

## 1. Login to Server



A screenshot of the 'Connect to Database' dialog box. The dialog has a title bar with a question mark icon and a close button. It contains four input fields: 'SQL Server Instance' (a dropdown menu), 'Database User' (a text box containing 'sa'), 'Database Name' (a dropdown menu containing 'DEMOIMIS'), and 'User Password' (a text box with masked characters). There are 'OK' and 'Cancel' buttons on the right side.

## 2. Analyze Database



A screenshot of a toolbar with three buttons: 'Analyze Database', 'Prepare iMIS Database', and 'Table Details ...'. The 'Analyze Database' button is highlighted with a green oval.

# SQL Index Maintenance 15

Look for tables that have issues – such as

## Output

```
* Index Activity.iActivityID is missing
* Index Activity.iActivityPRODUCT_CODE is missing
* Index Activity.iActivityTICKLER_DATE is missing
* Index Activity.iActivityCO_ID is missing
* Index Activity.PK_Activity is missing
* Index Activity.iActivityADDITIONAL_NAME_ID is missing
* Index Activity.iActivityORIGINATING_TRANS_NUM is missing
* Index Activity.iActivityACTIVITY_TYPE is missing
* Index Activity.iActivityTRANSACTION_DATE is missing
* Index Activity.iActivitySOURCE_CODE is missing
* Index Activity.iActivityORG_CODE is missing
* Index Activity.iActivityCAMPAIGN_CODE is missing
* Index Activity.iActivitySOLICITOR_ID is missing
* Index Activity.iActivitySOURCE_SYSTEM is missing
* Index Activity.iActivityUF_4 is missing
* Index Activity.iActivityUF_5 is missing
* Index Activity.iActivitySeqnActivityTypeID is missing
```

# SQL Index Maintenance 15

Identify the table at issue and click the table details button  
Next, select the table with an issue and click Drop Indexes  
then Create Indexes.

Table Details ...

