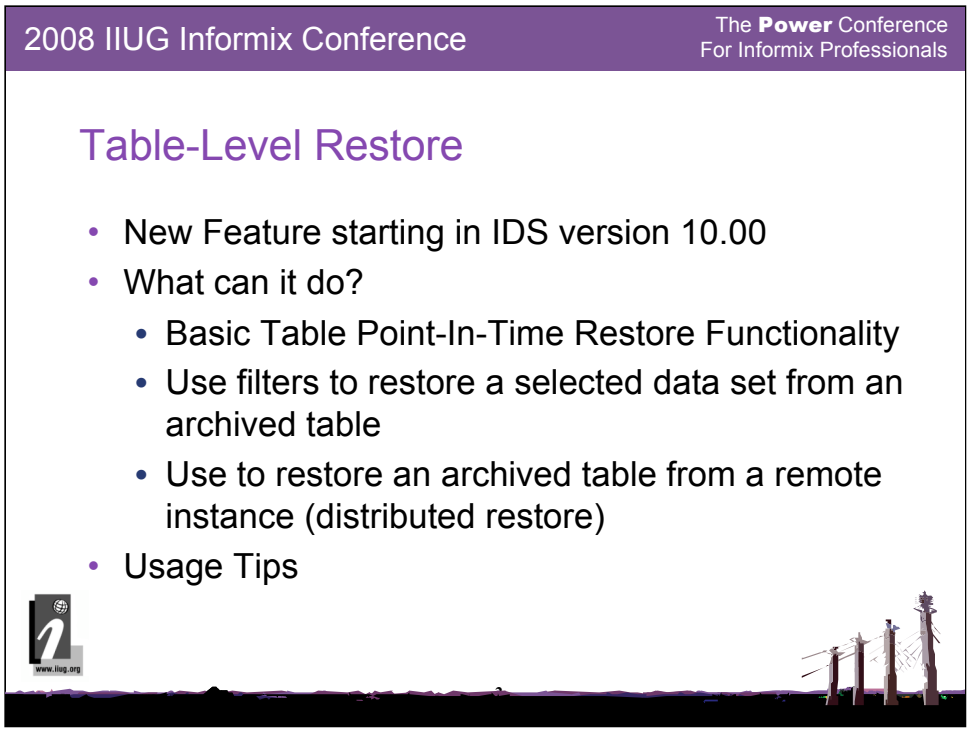
Using Point-In-Time Table-Level Restore

Hyun-Ju Vega
IBM

Session 122 (Alternate)
Monday, April 00, 2008 • 00:00 a.m. – 00:00 a.m.

2008 IIUG Informix Conference

www.iiug.org

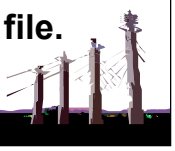


Find out how to use table-level restore to restore tables from IDS backups. Table-level restore (TLR) allows you to restore multiple tables from a level-0 backup and logs to a "live" system -- handy for restoring tables for testing or restoring an accidentally dropped table.

See a live demo,
get clear examples of what options to use and when, and tips
on debugging problems. Presentation includes a list of issues that people have run into while using TLR and information on how to avoid the same problems.

TLR Purpose

- **Provide the customer with the ability to easily extract a set of tables, a table or a portion of a table from a level 0 archive to a user specified point in time.**
- **The extracted data can be placed in an external table or on a table on the server of the user's choice regardless of server version or machine type as long as the database server is listed in the sqlhosts file.**



The point-in-time is compared to the transaction commit/rollback time recorded in the logical log records.

The extracted table data is restored via SQL inserts of the table data from the level 0 archive, then by replaying the relevant logical log records (via SQL insert/update/delete) for the table log records to the PIT indicated.

Benefits

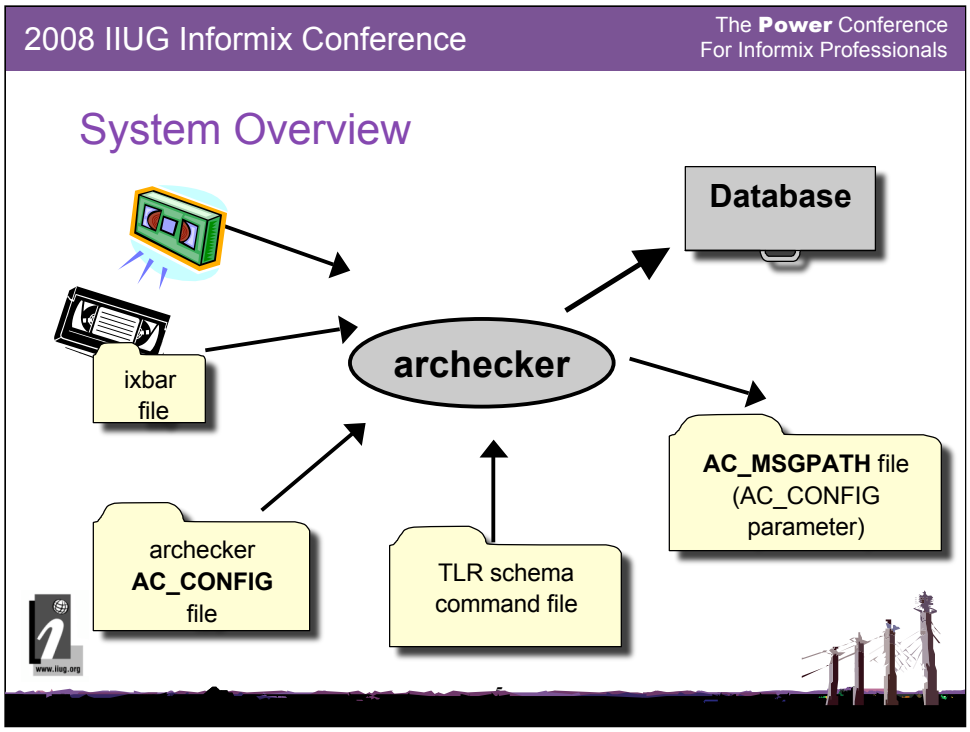
- **Extract a table or set of tables**
- **Filter the retrieved data**
- **Retrieve just a subset of columns**
- **Repartition the data**
- **Data may be placed in the same version database or on a different database version with a different machine architecture**

TLR is easy to use because most of the definition of the process is based on SQL. TLR data can even be restored to DB2 (need to use Gateway).

It is different from other restore products IBM provides for IDS, for example, onbar/ontape.

TLR Uses --
-- Application errors where you truncated a table or deleted rows from a table.
-- Works at table level
-- Can select a subset of all of the tables (for an application error for example).
-- Doesn't have to replace existing file
-- Can select a portion of the data
-- Can restore to any database server connected to existing server.

onbar/ontape --
-- To resolve physical issues w/ disk
-- Works at dbspace level
-- Used for full system restore
-- Provide full database consistency
-- Faster



2 places you input information

Archecker configuration file

Schema command file

In the archecker configuration file (AC_CONFIG file), these must be specified:

AC_MSGPATH

AC_STORAGE

AC_VERBOSE

All of the rest of the parameters, if not specified, will default to the values in the ONCONFIG. The TLR schema command file, if not specified in the AC_CONFIG file, must be specified on the command line.

Two types of restores

- **Physical Only**
  - Restore from level 0 archive only.
  - Can restore to an external table with delimited ASCII or Informix output.
  - Allow filter to be placed on the output data.

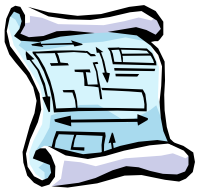
- **Physical & Logical Restore (Default)**
  - Restore from a level 0 archive, then logical logs to a point-in-time (PIT).
  - **During processing, temporarily adds two extra integer columns and an index to the destination table.**

Different Parts

- archecker executable
- Schema Command File
- archecker configuration file (AC_CONFIG)
  - AC_CONFIG is an environment variable
  - Default file is $INFORMIXDIR/etc/ac_config.std
- ixbar file (for restoring from onbar backups)
- Access to the storage manager or ontape device

Schema Command File

- The Table level restore feature is controlled by an SQL like language.
- The file the SQL commands are placed in is called a “schema command file”.
- SQL comments are allowed in the command file.
  - **Database**
  - **Insert into … Select**
  - **Create table**
  - **Set**
  - **Restore**
  - **SQL comments**



These commands are allowed.

Also, comments are allowed.

We’ll review each of these commands in the following slides.

Error checking happens to make sure you don’t enter incorrect values.

Schema Command File Example

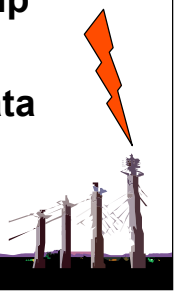
```
-- workspace CANNOT be temp dbspaces
set workspace to dbs1,dbs2;
-- source table
database test1;
create table tlr    (
        a_serial   serial,
        b_char     char(20)
      ) in dbspace1;

-- destination table
create table tlr_dest    (
        a_serial   serial,
        b_char     char(20)
     ) in dbspace2;
insert into tlr_dest select * from tlr;

restore to '2005-01-01 01:01:01';      -- If left out, default is "restore to current"
```

Database Statement

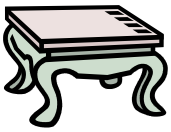
- **Set the current database statement**
- **All table names referenced following the database statement are associated with the current database unless fully qualified**
- **DATABASE <dbname> [LOG MODE ANSI]**

Create Table Statement

- **The full create table syntax is supported -- except the “same as” clause.**
- **This statement is used to specify both target and source tables**
- **Source Table**
  - **A table that is on the archive backup**
- **Target Table**
  - **The destination table, where the data should be placed**



Source Table



- **A table that is on the archive backup**
- **For each source table you must specify the storage options.**
  - **You must specify a dbspace name for the source table.**
  - The dbspace names in the fragmentation clause is used to find the table's data on the archive.
  - The exact fragmentation schema is not required.
- **Exact column layout is required**
  - Column types and locations must be exact
  - Not able to completely verify column layout
  - Wrong column layout will lead to unpredictable behavior.

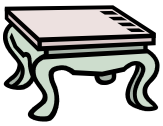
Source table is the table (data) on tape. archecker must be able to map the table name to its fragments so it can use the IXBAR file to find the right objects (for onbar backups). What matters is that archecker gets all of the names of fragments but really doesn't care about what data was stored in which fragment.

TIP: If you know a certain fragment will not contain any data you want to restore, don't specify it in the fragmentation schema.

TLR doesn't have a copy of the schema of the data on tape -- therefore, cannot validate the schema -- TLR relies on the user to supply an accurate schema of the table(s).

OnBAR and ontape don't require schema knowledge since they restore the physical pages of chunks, not specific data.

Target Table



- **If the target table exists then data is appended.**
- **If the target does not exist then the table is created.**
  - **All attributes specified in the schema file will be utilized.**

TIP: For logical restore, the target table is altered to temporarily add two integer columns, then altered again to drop those columns. If the destination table already exists and contains data, performance will decrease.

External Table

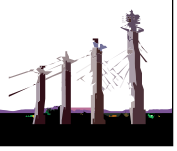
- **Used to specify a pipe or OS file as the location of the physical portion of the restore**
- **CREATE EXTERNAL TABLE ….**
  USING (‘<filename>’, ‘<format>’);
- **Format**
  - **INFORMIX or (ASCII) DELIMITED**
- ***Logical restore does not happen.***

Used only for target table, and only for physical restore.

Data Types Not Support for TLR

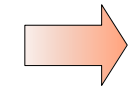
Extended Data Type
LISTS
MULTISET
SET
ROW
DISTINCT
OPAQUE
Built-in Data Type
CLOB -- only for physical restore *
BLOB -- only for physical restore *

- This mean these data types cannot be in any of tables being restored.
- If these data types do appear an error will be given when processing the TLR command file.
- CLOB and BLOB columns are **supported** for physical restore only starting in 10.00.xC3.



Insert into Select From Statement

- **Only a subset of the insert/select syntax is supported**
- **One table retrieved per statement**
- **Each statement requires a separate connection to the database**
- **A source table may only be extracted once per restore**
  - **No multi-siting**
  - **A table name may only exist once as a source table**



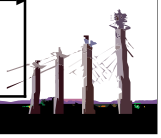
One table load per insert-select.

Each insert –select will create a connection.

No multi-siting – cannot retrieve one source table and send it to multiple target tables. If you try to do this, it will error.

Insert into Select From Statement

Feature	Supported	Unsupported
Insert table	fully qualified table name	views synonyms
Insert columns	column names	column aliases constants
Select columns	column name projection list all columns '*'	subscripting column aliases constants aggregate expressions stored procedures
From Clause	single table	views multiple tables synonyms



From clause – no joins.

Can always restore and then do the work in SQL against restored tables.

Examples:

```
insert into tab1_new select col2, col3, col4 from tab1;

insert into tab2_new (col1, col2, col3, col4, col5) select
    * from tab2;

insert into tab3_new select * from tab3
    where b_char != 'y' or c_integer <> 0 ;
```

Insert into Select From Statement (cont'd)

Feature	Supported	Not Supported
Where Clause	1. Equal, Not Equal (=, ==, <>) 2. Less than, Less than equal to 3. Greater than, Greater than equal to 4. Match, Not Match, Like, Not Like 5. IS NULL, NOT NULL 6. AND 7. OR 8. TODAY/CURRENT	Joins Subqueries Functions Procedures Math Expr.
Having Clause		All Parts
Group By		All Parts
Into Temp		All Parts
Order by		All Parts
Union		All Parts

Set Statement

- **SET COMMIT TO**
  - **Sets the number of records to insert before committing**
  - **SET COMMIT TO <number>**
  - **Default is 1000**
- **SET WORKSPACE TO <dbspace,….>**
  - **Directs working tables and indexes to the dbspaces listed**
  - **Workspaces CANNOT be temp dbspaces**

A lower set value requires more work.

archecker will roll back any insert failure and keep on going.

archecker reports the page number of the error.

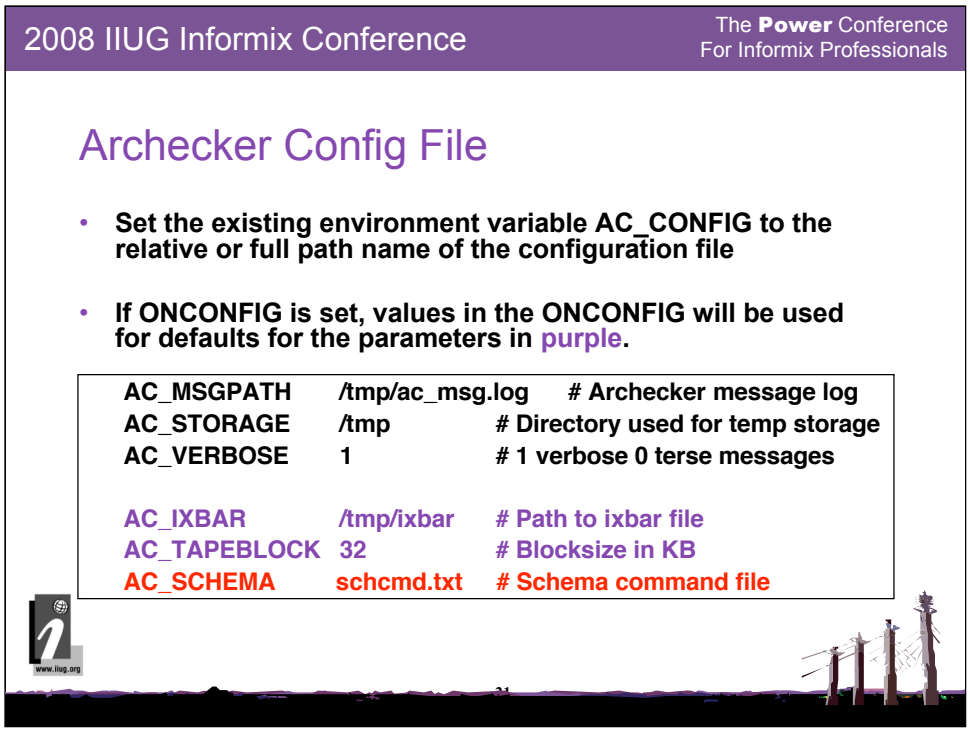
Set higher for better performance.

Restore Command

- **Applies to all the tables being restored in the command file**
- **Defines point in time to restore data to**
- **Defines if logical recovery should occur**
- **Default: RESTORE TO CURRENT**
- **RESTORE**
  [TO <timestamp> | CURRENT] [WITH NO LOG]
- **Physical restore only:**
  - **RESTORE TO CURRENT WITH NO LOG**
- **Logical restore is stopped on**
  - **Drop table log record**
  - **Alter table**
  - **Truncate table**

This is an optional command that is used to identify a single point-in-time that the table(s) specified in the command file should be restored to. In addition, the DBA may instruct the restore to not use logical logs by using the “WITH NO LOG” clause. If more than one “Restore To” statement is present in the command file, an error will be raised. If a “Restore To” statement is not present in the command file, then the system will be restored to the most current point-in-time using the available logical logs.

When replaying the logs, if a drop table or a truncate table logical log record is seen (for the table(s) being restored), the restore of the table is stopped.



AC_IXBAR points to where the onbar IXBAR file.

AC_TAPEBLOCK -- must match the size of block used when backup was made. Default is 32k.

AC_SCHEMA – schema command file -- if not here, it must be specified on the command line.

AC_TAPEBLOCK for onbar --

For onbar, the AC_TAPEBLOCK needs to be PAGESIZE*BAR_XFER_BUF_SIZE.

For Windows, that would be 4*BAR_XFER_BUF_SIZE. If BAR_XFER_BUF_SIZE is 15, then AC_TAPEBLOCK would be 60.

For a 2K pagesize system, the AC_TAPEBLOCK would be 62 if the maximum value of BAR_XFER_BUFSIZE (31) was used during the backup.

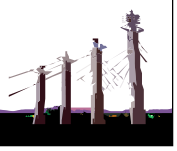
AC_TAPEBLOCK for ontape -- same as ONCONFIG TAPEBLK

If not set in the archecker config file, values like AC_TAPEBLOCK and AC_IXBAR will be read from the ONCONFIG.

Archecker Command Line

- Start archecker in TLR mode w/ schema file
  archecker –[b|t]dvs –f {schema cmd file}
- Start archecker in TLR mode (schema filename is in the AC_CONFIG)
  archecker –[b|t]dvsX

t	Use ontape interface
b	Use onbar interface
d	Do cleanup before running
s	Print information to the screen
v	Verbose
f	Schema command file, implies X
X	Table Level unloads



-f is used to override the AC_CONFIG file setting for the schema command file.

-X is tell archecker to use the AC_SCHEMA in the AC_CONFIG file for the schema command file.

See the manual for more options.

Example Screen Output (onbar TLR restore)

- archecker -bdvs -f schema_cmd.txt

```
AC_STORAGE      /tmp
AC_MSGPATH      /tmp/ac.log
AC_VERBOSE      on
AC_TAPEBLOCK    32
AC_PAGESIZE     4096
AC_IXBAR        /vobs/tristarm/sqldist/etc/ixbar.85
Extracting table dbs:source_table into dbs1:target_table
Scan PASSED
Control page checks PASSED
Table checks PASSED
Table extraction commands 1
Tables/fragments found on archive 1
LOADED: dbs:source_table produced 2008 rows.
```

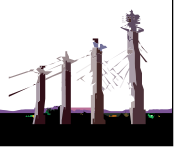
SCAN – when all tapes have been read.

“LOADED:” line printed once per table.

Recovery of Lost Table

- **Extracts a table called testdb:tlr from the most recent backup of dbspace1 and places the data in table testdb:tlr**

```
database testdb;
create table tlr (
       a_serial    serial,
       b_integer   integer,
       c_char      char(20),
       d_decimal   decimal,
       ) in dbspace1;
insert into tlr
  select * from tlr;
```



In this example, the source table and the target table are the same (as used in the insert statement).

Defaults to the current time for the “Restore To” timestamp -- so the log recovery continues to the most recent backed up log.

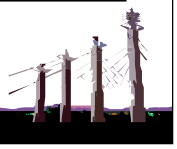
If the table exists, the restored data is appended.

If the table does not exist, the table is created.

Restoring from a Previous Backup

- **Extracts a table called test1:tlr from the level 0 backup of dbspace1 just prior "2008-04-23 16:46:30" and places the data in table test1:tlr**

```
database test1;
create table tlr (
       a_serial    serial,
       b_integer   integer,
       c_char      char(20),
       d_decimal   decimal,
       ) in dbspace1;
insert into tlr
  select * from tlr;
restore to
    '2008-04-23 16:46:30';
```



Restore to a point-in-time.

Example: onbar -P -n 5

```
IBM Informix Dynamic Server  Logical Log display
Software Serial Number AAA#B000000
Copyright IBM Corporation 1996, 2004 All rights reserved.

log uniqid: 5.

addr     len  type     xid      id link
18       44   BEGIN    5        5  0        04/23/2007 16:46:30 15
   informix
44       240  HUPBEF   5        0  18       200004   1301     190
134      240  HUPAFT   5        0  44       200004   1301     190
224      84   HUPBEF   5        0  134      200003   c07      34
278      84   HUPAFT   5        0  224      200003   c07      34
2cc      208  HUPBEF   5        0  278      200004   1302     158
39c      208  HUPAFT   5        0  2cc      200004   1302     158
46c      80   HUPBEF   5        0  39c      200003   c0a      32
4bc      80   HUPAFT   5        0  46c      200003   c0a      32
50c      40   COMMIT   5        0  4bc      04/23/2007 16:46:30
...
```

Extracting a Subset of Columns

- **Extracts a table called test1:source_tab from the most recent backup of dbspace1 and places the data in column c_char into column X_char, d_decimal into Y_decimal in table test1:target_tab**

```
database test1;
create table source_tab (
       a_serial    serial,
       c_char      char(20),
       d_decimal   decimal
       ) in dbspace1;
create table target_tab (
       X_char      char(20),
       Y_decimal   decimal,
       Z_name      char(40)
       ) in dbspace2;
insert into target_tab
  ( X_char, Y_decimal )
  select c_char,d_decimal
  from source_tab;
```

Different schemas, different tables.

Here also using projection in the insert command to pick only certain columns. Here, z_name would be set to NULL.

Using Data Filtering

- **If using filters (where clause in the insert statement), always put the "RESTORE to CURRENT WITH NO LOG;" in the schema command file.**
- **Filters can only be used when doing physical restore only.**
- **Filter will be disabled if TLR believes that it will do a logical restore.**

```
database test1;
create table tlr (
    a_serial    serial,
    b_integer   integer,
    c_char      char(20),
    d_decimal   decimal,
    ) in dbspace1;

insert into tlr
  select * from tlr
  where c_char matches 'a*'
  and d_decimal is NOT NULL
  and b_integer > 100;

RESTORE TO CURRENT WITH NO LOG;
```

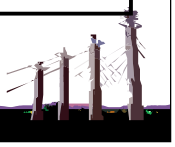
What's new – the where clause.

If you do not have "RESTORE TO CURRENT WITH NO LOG" the filter will be disabled, as TLR thinks that it will be doing a logical restore following the physical restore.

Using External Tables

- **Extracts a table called dl:source_tab from the most recent backup of dbspace1 and sends the data in ASCII format with fields delimited to the file named /TMP/PIPE.**

```
database d1;
create table source_tab
  (cola int)
  in dbspace1;

create external table
  target_tab
  (cola int)
  USING ('/TMP/PIPE',
         'DELIMITED');

insert into target_tab
  select * from source_tab;
```



Data can be sent to a pipe. The filename can be either a named pipe or a file. We send ASCII delimited data.

Two Table Example

- **Extracts a table called test1:tlr_1 and test:tlr_2 from the most recent backup of dbspace1 and places the data in table test1:tlr_1_dest and test1:tlr_2_dest using only one scan of the tape.**

```
database test1;

create table tlr_1
  (  columns  ) in dbspace1;
create table tlr_1_dest
  (  columns  );
create table tlr_2
  (  columns  ) in dbspace1;
create table tlr_2_dest
  (  columns  );

insert into tlr_1_dest
  select * from tlr_1;
insert into tlr_2_dest
  select * from tlr_2;
```

What's new – two tables restored.

Distributed Restore

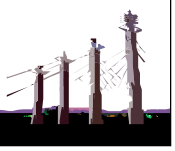
- **Extracts a table called test:source_tab from the most recent backup of dbspace1 and places the data on the database server rem_srv in the table target_dbs:target_tab**

```
database test;
create table source_tab
  (  columns  ) in dbspace1;

database target_dbs@rem_srv;
create table target_tab
   ( columns  );

insert into
  target_dbs@rem_srv.target_tab
  select * from source_tab;
```

What's new -- the target table is on a remote server.

TLR Space Usage and Cleanup

- Creates working tables in sysutils database.
- Creates files in AC_STORAGE directory.
  - AC_STORAGE/SAVE directory -- the partition pages and tape control pages examined from the archive are saved here.
- Does not cleanup working tables after the restore has completed.
- Manually cleanup by running : **archecker -DX**
  - Cleans up working tables in sysutils database and AC_STORAGE.



What are the TLR working tables and files?

- The following directories are created in the specified AC_STORAGE directory (default is /tmp):
  - CHUNK_BM
  - INFO
  - SAVE
- These directories contain working files used when processing the level 0 archive data.

TLR Working Tables in sysutils

Table name	Temp table?	Created during …	Description
acu_restore	No	Physical restore phase	Contains 1 row for each table partition data fragment being restored.
acu_sblob_extent	No	Physical restore phase	Used from 10.00.xC3. Stores smart blob extents info found on the archive.
acu_sblob_request	No	Physical restore phase	Used from 10.00.xC3. Stores smart blob descriptor info for smart blob columns that are being restored from the archive.
acu_tuple	Temp unless ACU_NOTEMP env variable is set.	Physical restore phase	Stores partial rows while waiting to read rest of the row from archived pages. Contents of acu_tuple table are deleted as the tuples are assembled and inserted into the table being restored.
acu_request	Temp unless ACU_NOTEMP env variable is set.	Physical restore phase	Stores info on row pieces/tuples that need to be read from the archive to complete a row. Contents of acu_request table are deleted as the tuples are found in the archived pages.

TLR Working Tables in sysutils (continued)

Table name	Temp table?	Created during …	Description
acu_control	No	Log staging phase	Contains one row for each transaction staged. Unless the environment variable INFX_LEAVE_LOG is set, the contents of acu_control table are deleted as the transactions are applied to the table being restored.
acu_log_stage	No	Log staging phase	Contains log records needed for the transactions stored in acu_control table. Unless the environment variable INFX_LEAVE_LOG is set, the contents of the acu_log_stage table are deleted when the transaction associated with the log records have been applied to the table being restored.

TLR summary

- Modify AC_CONFIG file and set the environment variable if needed.
- Create the schema command file.
- Run archecker (example with schema file called schema.cmd):

  archecker -[b|t]dvs -f schema.cmd

- Cleanup TLR working tables and files:

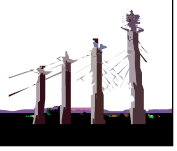
  archecker -DX

TLR from an onbar backup: archecker -bdvs -f schema.cmd

TLR from an ontape backup: archecker -tdvs -f schema.cmd

Usage Tip: Use the -d option

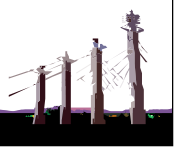
- After a TLR restore, TLR does not automatically clean up the working file and tables.
- So, when you run a new TLR command, it's possible that there are old working files and tables lying around.
- To be safe, always use the -d option when running a TLR restore. The -d option will cause all working files and tables to be cleaned up prior to starting the new restore run.

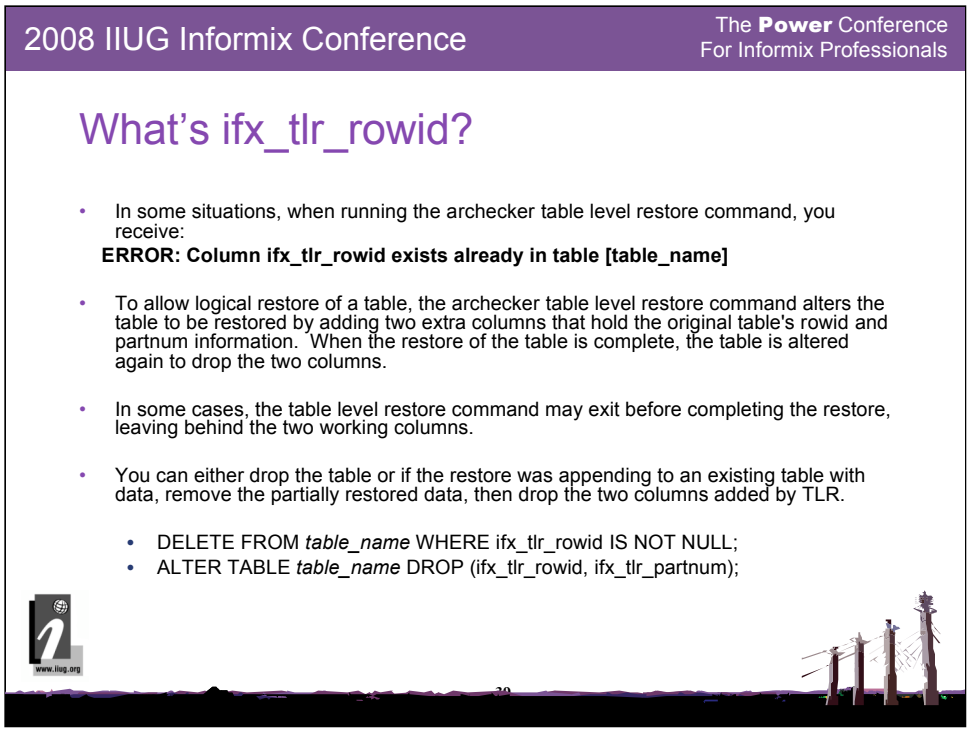


Usage Tip: Look for ERROR's in the log file

- Even in verbose mode, the error messages displayed to the screen is very cryptic. Also, not all error messages are printed to stdout/stderr. The detailed error messages and warnings are only stored in the AC_MSGPATH log. Usually, the AC_MSGPATH log will have enough information to allow you to figure out why the restore failed.
- Always check for WARN and ERROR in the AC_MSGPATH log after a table-level restore.
  - Example:

```
$ egrep "ERROR|WARN" ac_msg.log
```





The extra column names may have a number of different suffixes if any of the tables in the archecker table level restore schema file contained columns with the names ifx_tlr_rowid and ifx_tlr_partnum. For simplicity, this document assumes that the two columns added by the archecker table level restore command are called ifx_tlr_rowid and ifx_tlr_partnum.

In some cases, the table level restore command may exit before completing the restore, leaving behind the two working columns. In that case, the next time the table level restore command is run to restore the same table, you will get the error message. The error message is to prevent restoring to a table that may have only partially restored data (possible if any of the ifx_tlr_rowid or ifx_tlr_partnum column values have non-null data).

Why is TLR doing an ALTER?

- To do the logical restore portion of the table-level restore, TLR adds two extra columns to the target table.
- The two extra columns are dropped at the end of a successful restore with an ALTER command.
- For large tables, this ALTER can take a while, so if you want to look at your restored data as quickly as possible, you can ask TLR to not do the final alter by putting "**SET ALTER off ;** " at the top of the TLR schema command file.

Why didn't TLR create the table correctly?

- TLR will only create the table if the table does not already exist in the target database.
- If the table name already exists, TLR will not create it -- even if the schema specified in the TLR schema file is different than the schema of the already existing table.

TLR assumes DB_LOCALE of en_us.8859-1

- If archecker TLR command is failing to create the target table, it may be because of the DB_LOCALE.
- TLR assumes DB_LOCALE of en_us.8859-1 unless you set it to something else before running the archecker TLR command.
- You should set the environment DB_LOCALE to the DB_LOCALE of the database before running the TLR command.

Performance

- Test showed data being restored to the table at a rate of approximately 1 Meg/sec, which was 25% of the tape's through-put.
- Performance will depend on the speed of the tape and the SQL insert (also update, delete for logical restore) performance.
- If doing logical restore, alter performance to add and drop two tlr working columns will depend on data already existing in destination table and data being restored.

Think of doing a select from an archived table and inserting into a new table.

Session 122 (Alternate)

Using Point-In-Time Table-Level Restore

Hyun-Ju Vega
IBM
vegah@us.ibm.com