What's hidden in the ARCHIVES - DB2's answer to American Pickers?

Steen Rasmussen, Sr. Engineering Services Architect
CA technologies
All live scenarios have been performed on a DB2 11 NFM system with maintenance applied up to June 2015.

Once we get to the LIMITATIONS / RESTRICTIONS section, the IBM documentation does describe limitations I did not find to be true, but maintenance can have changed this.
ABSTRACT

The attendee will get a detailed overview of what’s needed in terms of DDL to enable data archiving automatically as well as the changes needed from the SQL/application view.

There's no perfect world (Utopia doesn't exist) so we will also look into the limitations and restrictions when Archive tables are exploited in DB2 11 NFM.

Next topic is to go over what needs to be considered in terms of backup and recovery as well as other DB2 utilities which are impacted.

The entire presentation will be done based on a real DB2 11 subsystem to see how all the moving parts work together.

BIO:

Steen Rasmussen is a Sr. Engineering Services Architect currently instrumental in the ongoing development and support of the CA Technologies DB2 tools. In 1985 Steen started as an IMS/DB2 DBA at a major insurance company in Denmark working with all aspects of DB2 - like tuning, application design and implementation, education of developers, backup and recovery planning and automation of housekeeping processes. During this job, Steen also served as a member of the planning committee for DB2 GUIDE SHARE Nordic Region. In 1995 Steen became a technical manager at PLATINUM Technology managing technical support and presales for the DB2 products.

Steen has been working with DB2 for z/OS Release 1.0 since 1985 and is always looking for new opportunities in the CA DB2 solutions which can help customers manage DB2 more efficiently. Besides from providing support to the teams in the field as well as internal groups at CA technologies working with DB2, Steen is also a frequent speaker at IDUG in North America, EMEA and Australia as well as local DB2 User Groups around the world. Since 2014 Steen has been an IBM Information Management Champion. Since 2013 Steen has been the CA liaison for IDUG NA.
First we will cover why archiving can be of interest and what we have done so far since this has been an issue since the early age of computing.
Next we will dive into the DB2 schema changes needed to enable archiving before moving into the application / SQL changes introduced.
We will spend quite some time dealing with the limitations introduced once archive tables are in place – talking about what to be aware of, how to deal with these challenges and some interesting gotcha’s I discovered while “playing” with archive tables.
Next is what impacts DB2 utilities and what you need to be aware of.

Throughout the presentation, a live DB2 11 NFM system has been used to illustrate the power of archive tables and how all the pieces really work.
The TV channel in the US named HISTORY CHANNEL has a program called “American Pickers”. Two guys named Mike and Frank tour around the US “picking” from what people have been collecting for decades in order to resell the items for a profit.
The TV channel in the US named HISTORY CHANNEL has a program called “American Pickers”. Two guys named Mike and Frank tour around the US “picking” from what people have been collecting for decades in order to resell the items for a profit.

The tv program can be quite educational since the history behind old artifacts are covered.
Just because DB2 11 introduced Archive Tables doesn’t really mean we haven’t had this need in the past – it has existed since the first day of IT.

What has been introduced in DB2 11 to handle archival of data looks very similar to the DB2 10 feature named TEMPORAL TABLES – especially the System Time Temporal table design.
Both have a HISTORY table associated, but the archival design only saves the image of the base table when a delete is executed unlike System time temporal tables where also UPDATES are saved in the history table.

As already mentioned, the need to archive data has existed since forever. The current implementation used range from pure Application logic to using triggers and using the DB2 log to capture transactions and apply these to history tables via one of the log tools available in the market.
So why is the DB2 11 Archive Table feature more attractive than the current implementations?

It takes time to code, maintain, test application code to insert rows into a history table prior to deleting. Archive-enabled tables handle this automatically. One of the most expensive pieces in today’s DB2 environment is TRIPS TO DB2. Every trip saved is resources preserved, and the INSERT followed by a DELETE from the application is TWO trips – not to count the retrieval.

Triggers can reduce the previous mentioned pieces, but then again – triggers add complexity in a DB2 environment.

Despite all the GREAT stuff introduced, as we will see later, there are some schema changes we need to be aware of and consider how we will cope with these.
There are many reasons why IT organizations want to archive data:

1) DASD savings despite the fact that DASD has dropped dramatically over the years.
2) One of my favorites is less INDEX LEVELS – every level decreased is one less I/O for EVERY statement using the index.
3) GETPAGE activity – another favorite similar to the previous one – if you have 25% less rows there’s a good change you will save 25% GETP from DASD into the bufferpool unless doing single-row random access.
4) UTILITY COSTS can be a major factor since every LOAD, REORG, RUNSTATS, RECOVER and potentially other utilities will need to access fewer pages.
5) If we really look back – when new applications are implemented, NOBODY think about cleaning up or archiving – and the fact that regulations might change over time so you MIGHT have to retain deleted rows.
6) Once the application is implemented – what does it cost to implement a DATA ARCHIVE DATA solution ????? In fact it never happens – well almost.
Let’s start with the official IBM DB2 terminology:

The table where your BASE/Production data is (the current table having your transactional data is named the ARCHIVAL-ENABLED table – I like to name it the BASE TABLE – and the ARCHIVE TABLE which is where the DELETED rows end up is named the ARCHIVE table – I like to name it the HISTORY table.
DB2 Archive Tables DDL – how does it all work

- Tablespaces created for BASE and HISTORY
  - Tablespace attributes can be different

```sql
CREATE TABLESPACE ARCHBASE
  IN IDUG2015
  USING STOGROUP SYSDEFLT
  ERASE NO
  FREEPAGE 0
  PCTFREE 0
  MAXPARTITIONS 1
  BUFFERPOOL BP1
  LOCKSIZE ANY
  CLOSE YES
  SEGSIZE 04
  LOCKMAX 0

;  
```

```sql
CREATE TABLESPACE ARCHHIST
  IN IDUG2015
  USING STOGROUP SYSDEFLT
  ERASE NO
  FREEPAGE 0
  PCTFREE 0
  NUMPARTS 9
  BUFFERPOOL BP1
  LOCKSIZE ANY
  CLOSE YES
  SEGSIZE 04
  LOCKMAX 0
  CCSID EBCDIC
  DSSIZE 4G

;  
```
DB2 Archive Tables DDL – how does it all work

Tables for BASE and HISTORY (have to be identical – NO differences – well, sort of !!!)

CREATE TABLE IDUG15.ARCHBASE
( EMPNO CHAR ( 6 ) NOT NULL,
  FIRSTNME VARCHAR(12) NOT NULL,
  MIDINIT CHAR ( 1 ) NOT NULL,
  LASTNAME VARCHAR ( 15 ) NOT NULL,
  WORKDEPT CHAR ( 3 ),
  PHONENO CHAR ( 4 ),
  HIREDATE DATE,
  JOBCHAR ( 8 ),
  FOR SBCS DATA,
  EDELEVEL SMALLINT,
  SEXCHAR ( 1 ),
  BIRTHDATE DATE,
  SALARY DECIMAL ( 9, 2 ),
  BONUS DECIMAL ( 9, 2 ),
  COMM DECIMAL ( 9, 2 )
) IN IDUG2015.ARCHBASE
DATA CAPTURE CHANGES CCSID EBCDIC ;

CREATE TABLE IDUG15.ARCHHIST
( EMPNO CHAR ( 6 ) NOT NULL,
  FIRSTNME VARCHAR(12) NOT NULL,
  MIDINIT CHAR ( 1 ) NOT NULL,
  LASTNAME VARCHAR ( 15 ) NOT NULL,
  WORKDEPT CHAR ( 3 ),
  PHONENO CHAR ( 4 ),
  HIREDATE DATE,
  JOBCHAR ( 8 ),
  FOR SBCS DATA,
  EDELEVEL SMALLINT,
  SEXCHAR ( 1 ),
  BIRTHDATE DATE,
  SALARY DECIMAL ( 9, 2 ),
  BONUS DECIMAL ( 9, 2 ),
  COMM DECIMAL ( 9, 2 )
) IN IDUG2015.ARCHHIST
PARTITION BY ( HIREDATE )
( PARTITION 1 ENDING AT ( '2010-12-31' ),
  PARTITION 2 ENDING AT ( '2011-12-31' ),
  PARTITION 3 ENDING AT ( '2012-12-31' ),
  PARTITION 4 ENDING AT ( '2013-12-31' ),
  PARTITION 5 ENDING AT ( '2014-12-31' ),
  PARTITION 6 ENDING AT ( '2015-12-31' ),
  PARTITION 7 ENDING AT ( '2016-12-31' ),
  PARTITION 8 ENDING AT ( '2017-12-31' ),
  PARTITION 9 ENDING AT ( '2018-12-31' ) )
DATA CAPTURE CHANGES CCSID EBCDIC ;
DB2 Archive Tables DDL – how does it all work

- Indexes created for both base and history

```
RQTXC 18.0  -------- RC/Q Table Index Column list --------  2015/08/04 12:36
COMMAND ===>> SCROLL ===> PAGE

DB2 Object ===> T
Table Name ===> ARC%
Qualifier ===> *

Loc: LOCAL --------- SSID: D11A --------- RASST02 -
CMND     TABLE NAME  INDEX NAME  INDEXED COLUMN  COLSEQ  ORD  CLS  UNQ
--------- ------ -------------------- ------------- ---- ---- ---- ----
        ARCHBASE      ARCHBASEIX      EMPNO         1    A    N    D
        ARCHIST      ARCHHISTIX      EMPNO         1    A    N    D
        ARCHHISTIX  ARCHHISTIX2      EMPNO         1    A    N    U
        HIREDATE

******************************************************************************

13
DB2 Archive Tables DDL – how does it all work

- TIE the BASE and HISTORY to ENABLE (or disable) archiving is very simple.

```
ALTER TABLE IDUG15.ARCHBASE
  ENABLE (DISABLE) ARCHIVE
USE IDUG15.ARCHHIST;
```

- Now it’s time to start INSERT/DELETE/UPDATE processing and see how data flows.

<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>CREATOR</th>
<th>DATABASE</th>
<th>TBLSPACE</th>
<th>NUMBER OF ROWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHBASE</td>
<td>IDUG15</td>
<td>IDUG2015</td>
<td>ARCHBASE</td>
<td>45</td>
</tr>
<tr>
<td>ARCHHIST</td>
<td>IDUG15</td>
<td>IDUG2015</td>
<td>ARCHHIST</td>
<td>0</td>
</tr>
</tbody>
</table>
DB2 Archive Tables DML – how does it all work

- Insert / Update BASE doesn’t impact the ARCHIVE table

<table>
<thead>
<tr>
<th>COLUMN NAME</th>
<th>NULL</th>
<th>DATA FOR ROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>EMENO</td>
<td>Y</td>
<td>900000</td>
</tr>
<tr>
<td>FIRSTNAME</td>
<td>N</td>
<td>STEEN</td>
</tr>
<tr>
<td>MIDINIT</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>LASTNAME</td>
<td>N</td>
<td>RASMUSSEN</td>
</tr>
<tr>
<td>WORKDEPT</td>
<td>N</td>
<td>E21</td>
</tr>
<tr>
<td>PHONE</td>
<td>N</td>
<td>4667</td>
</tr>
<tr>
<td>HIREDATE</td>
<td>N</td>
<td>1995-04-01</td>
</tr>
<tr>
<td>JOB</td>
<td>N</td>
<td>FIELDF</td>
</tr>
<tr>
<td>EDLEVEL</td>
<td>N</td>
<td>16</td>
</tr>
<tr>
<td>SEX</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>N</td>
<td>1961-04-13</td>
</tr>
<tr>
<td>SALARY</td>
<td>N</td>
<td>850000.00</td>
</tr>
<tr>
<td>BONUS</td>
<td>N</td>
<td>25000.00</td>
</tr>
<tr>
<td>COMN</td>
<td>N</td>
<td>320000.00</td>
</tr>
</tbody>
</table>

UPDATE IDU015.ARCHBASE
SET SEX='F';
HIREDATE = '2000-01-15';
WHERE EMENO='000015';

SELECT * FROM IDU015.ARCHIST;
DSN404I SQLCODE = 100, NOT FOUND: ROW
NOT FOUND FOR FETCH, UPDATE, OR DELETE,
OR THE RESULT OF A QUERY IS AN EMPTY
TABLE
Special registers / Global variables:

- Two new GLOBAL VARIABLEs will help manage DML:

- SYSIBMADM.GET_ARCHIVE = Y / N
  - Should SQL against BASE include rows from ARCHIVE table?

- SYSIBMADM.MOVE_TO_ARCHIVE
  - E: Delete from BASE results in INSERT into ARCHIVE table
  - Y: SQL INSERT/UPDATE on ARCHIVE ENABLED / HISTORY table will result in SQL-ERROR
  - N: Delete from BASE will not insert into ARCHIVE
DB2 Archive Tables DML – how does it all work

- DELETE processing without special registers enabled.
  - Lets “FIRE” Steen Rasmussen
  - Without setting the special REGISTER – ARCHIVE table not maintained !!!

<table>
<thead>
<tr>
<th>COLUMN NAME</th>
<th>NULL</th>
<th>DATA FOR ROW #</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPNO</td>
<td></td>
<td>900000</td>
</tr>
<tr>
<td>FIRSTNAME</td>
<td></td>
<td>STEEN</td>
</tr>
<tr>
<td>MIDINIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LASTNAME</td>
<td></td>
<td>RASMUSSEN</td>
</tr>
<tr>
<td>WORKDEPT</td>
<td>N</td>
<td>E21</td>
</tr>
<tr>
<td>PRIMEENO</td>
<td>N</td>
<td>4667</td>
</tr>
<tr>
<td>HIREDATE</td>
<td>N</td>
<td>1995-04-01</td>
</tr>
<tr>
<td>JOB</td>
<td>N</td>
<td>FJELDKR</td>
</tr>
<tr>
<td>EDLEVEL</td>
<td>N</td>
<td>16</td>
</tr>
<tr>
<td>SEX</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>N</td>
<td>1961-04-13</td>
</tr>
<tr>
<td>SALARY</td>
<td>N</td>
<td>85000.00</td>
</tr>
<tr>
<td>BONUS</td>
<td>N</td>
<td>25000.00</td>
</tr>
<tr>
<td>COMM</td>
<td>N</td>
<td>320000.00</td>
</tr>
</tbody>
</table>

DELETE FROM IDUG15.ARCHBASE WHERE EMPNO='900000' ;
DSN74001 SQLCODE = 000, SUCCESSFUL EXECUTION
COMMIT;
DSN74001 SQLCODE = 000, SUCCESSFUL EXECUTION
SELECT * FROM IDUG15.ARCHHIST ;
Nothing displays !
DB2 Archive Tables DML – how does it all work

- Let’s ACTIVATE the special REGISTERS – and DELETE !!
  - What happened ?
  - The PARTITION LIMITKEY for the HISTORY table did NOT accept.

```
SET SYSPROC.WRITE_TO_ARCHIVE = E' ;
DB2400I SQLCODE = 005, SUCCESSFUL EXECUTION
DELETE FROM IDG010.ARCHIVE WHERE EMPNO < '000030' ;
DB2400I SQLCODE = '237', ERROR: THE ROW CANNOT BE INSERTED BECAUSE IT
IS OUTSIDE THE BOUND OF THE PARTITION RANGE FOR THE LAST
PARTITION

<table>
<thead>
<tr>
<th>COLUMN NAME</th>
<th>NULL DATA FOR ROW # 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPNO</td>
<td>000015</td>
</tr>
<tr>
<td>FIRSTNAME</td>
<td>Bob</td>
</tr>
<tr>
<td>MIDDLEINIT</td>
<td>I</td>
</tr>
<tr>
<td>LASTNAME</td>
<td>Smith</td>
</tr>
<tr>
<td>WORKDEPT</td>
<td>M 550</td>
</tr>
<tr>
<td>PHONE</td>
<td>M 5590</td>
</tr>
<tr>
<td>HIREDATE</td>
<td>Y</td>
</tr>
<tr>
<td>JOB</td>
<td>M  PRES</td>
</tr>
<tr>
<td>ELEVEL</td>
<td>M 18</td>
</tr>
<tr>
<td>SEX</td>
<td>M  T</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>Y</td>
</tr>
<tr>
<td>SALARY</td>
<td>N 80200.00</td>
</tr>
<tr>
<td>BONUS</td>
<td>N 20000.00</td>
</tr>
<tr>
<td>COMM</td>
<td>N 0220.00</td>
</tr>
</tbody>
</table>
```

------------------------------- BOTTOM OF DATA -----------------------------
DB2 Archive Tables DML – how does it all work

- Let’s ACTIVATE the special REGISTERS – and DELETE !!
  - In the mean time – HIREDATE updated to be NOT NULL
  - TWO rows qualified for the DELETE
  - And these have been INSERT’ed into the ARCHIVE table.

```
SET SYSPRINT.NEW TO_ARCHIVE = 'E' ;
DBR7400I SQLCODE = 000, SUCCESSFUL EXECUTION
DELETE FROM IDUG15.ARCHBASE WHERE EMPNO < '000030' ;
DBR7400I SQLCODE = 000, SUCCESSFUL EXECUTION
COMMIT WORK ;
DBR7400I SQLCODE = 000, SUCCESSFUL EXECUTION

SELECT * FROM IDUG15.ARCHBASE ;
EMPNO FIRSTNAME MIDDLENAME LASTNAME N.WORKDEPT N.PHONENO N
000018 Bob Smith     N A00     N 3978     N
000020 Michael L Thompson N B01    N 3476     N
```
DB2 Archive Tables – RETRIEVE data

- You can select from BASE or HISTORY as you like

```sql
SELECT * FROM IDGO15.ARCHHIST;
DSNT4001 SQLCODE = 000, SUCCESSFUL EXECUTION
SELECT * FROM IDGO15.ARCHBASE;
DSNT4001 SQLCODE = 000, SUCCESSFUL EXECUTION
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>FIRSTNAME</th>
<th>MIDINIT</th>
<th>LASTNAME</th>
<th>DEPT</th>
<th>PHONE</th>
<th>HIREDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000015</td>
<td>Bob</td>
<td>I</td>
<td>Smith</td>
<td>A00</td>
<td>3978</td>
<td>2000-01-15</td>
</tr>
<tr>
<td>000020</td>
<td>Michael</td>
<td>L</td>
<td>Thompson</td>
<td>B01</td>
<td>3476</td>
<td>1973-10-10</td>
</tr>
</tbody>
</table>

43 ROWS RETRIEVED

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>FIRSTNAME</th>
<th>MIDINIT</th>
<th>LASTNAME</th>
<th>DEPT</th>
<th>PHONE</th>
<th>HIREDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000340</td>
<td>Jason</td>
<td>R</td>
<td>Gounot</td>
<td>E21</td>
<td>5698</td>
<td>1947-05-05</td>
</tr>
<tr>
<td>000050</td>
<td>Sally</td>
<td>B</td>
<td>Khan</td>
<td>C01</td>
<td>4738</td>
<td>1975-04-05</td>
</tr>
<tr>
<td>000060</td>
<td>John</td>
<td>B</td>
<td>Geyer</td>
<td>E01</td>
<td>6789</td>
<td>1949-09-17</td>
</tr>
<tr>
<td>000060</td>
<td>Irving</td>
<td>F</td>
<td>Stern</td>
<td>D11</td>
<td>6423</td>
<td>1973-09-14</td>
</tr>
<tr>
<td>000070</td>
<td>Eva</td>
<td>D</td>
<td>Fulinski</td>
<td>D21</td>
<td>7831</td>
<td>1980-09-30</td>
</tr>
<tr>
<td>000090</td>
<td>Kilmen</td>
<td>W</td>
<td>Henderson</td>
<td>E11</td>
<td>5498</td>
<td>1970-08-15</td>
</tr>
<tr>
<td>000100</td>
<td>Theodore</td>
<td>Q</td>
<td>Spencer</td>
<td>E21</td>
<td>0972</td>
<td>2005-06-19</td>
</tr>
</tbody>
</table>
DB2 Archive Tables – RETRIEVE data

- Implement the ARCHIVE feature into your applications without rewriting SQL in the “good old programs”
- The special REGISTERS come to help
- Modifications to application code NOT needed – just one small addition (below the result without special register).

```
SELECT * FROM IDUG15.ARCHBASE WHERE EMPNO < '000040';
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
EMPNO FIRSTNAME MIDINIT LASTNAME
000030 SALLY B RIAN

SELECT * FROM IDUG15.ARCHHist WHERE EMPNO < '000040';
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
EMPNO FIRSTNAME MIDINIT LASTNAME
000015 Bob I Smith
000020 Michael L THOMPSON
```
DB2 Archive Tables – RETRIEVE data

- Use GLOBAL VARIABLE to describe if rows retrieved from BASE and HISTORY

```sql
SET SYSIBMADM.GET_ARCHIVE = 'Y';

SELECT * FROM IDUG15.ARCHBASE WHERE EMPNO < '000040';

EMPNO  FIRSTNAME   MIDINIT  LASTNAME
000030  SALLY       B        KWAN
```

- Didn’t you expect to see three rows in total?
- One additional parameter is the BIG BOSS.
  - Bind parameter ARCHIVESENSITIVE(YES) was missing.
DB2 Archive Tables – RETRIEVE data

- Let’s try again.

```sql
SET SYSIBMADM.GET_ARCHIVE = 'Y';

DSHE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

SELECT * FROM IDUG15.ARCHBASE WHERE EMPNO < '000040';

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>FIRSTNAME</th>
<th>MIDINIT</th>
<th>LASTNAME</th>
<th>WORKDEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>000030</td>
<td>SALLY</td>
<td>B</td>
<td>WILSON</td>
<td>001</td>
</tr>
<tr>
<td>000015</td>
<td>Bob</td>
<td>I</td>
<td>Smith</td>
<td>000</td>
</tr>
<tr>
<td>000020</td>
<td>Michael</td>
<td>L</td>
<td>Thompson</td>
<td>001</td>
</tr>
</tbody>
</table>

DSHE616I NUMBER OF ROWS DISPLAYED IS 3
DSHE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 1
```
DB2 Archive Tables – RETRIEVE data

- What does DB2 do when GET_ARCHIVE and ARCHIVESENSITIVE in place?
- EXPLAIN uncovers the truth.

```sql
SET SYSIBMADM.GET_ARCHIVE = 'Y';
EXPLAIN PLAN SET QUERYNO = 5002 FOR
SELECT * FROM IDUGIS.ARCHBASE WHERE EMPNO < '000040';
COMMIT;
SELECT * FROM PLAN_TABLE WHERE QUERYNO = 5002 ;
```

- DB2 rewrites the single select to become a UNION ALL

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PLANNO</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>TABNO</th>
<th>ACCESTYPE</th>
<th>MATCHCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5002</td>
<td>1</td>
<td>1</td>
<td>IDUGIS</td>
<td>ARCHHIST</td>
<td>2</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td>5002</td>
<td>5</td>
<td>1</td>
<td>IDUGIS</td>
<td>ARCHBASE</td>
<td>1</td>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>5002</td>
<td>2</td>
<td>1</td>
<td>IDUGIS</td>
<td>ARCHBASE</td>
<td>0</td>
<td>O</td>
<td>0</td>
</tr>
</tbody>
</table>
DB2 Archive Tables – manipulate data

- It does matter how MOVE_TO_ARCHIVE is set

```sql
SET SYSCMD.MOVE_TO_ARCHIVE='Y' ;
UPDATE IDUG15.ARCHBASE SET SEX = 'X' WHERE SEX = 'M' ;

DSNT408I SQLCODE = -20555, ERROR: AN ARCHIVE-ENABLED TABLE IS NOT ALLOWED IN
THE SPECIFIED CONTEXT. REASON CODE 2
DSNE618I ROLLBACK PERFORMED, SQLCODE IS 0

SET SYSCMD.MOVE_TO_ARCHIVE='E' ;
UPDATE IDUG15.ARCHBASE SET SEX = 'X' WHERE SEX = 'M' ;

DSNE615I NUMBER OF ROWS AFFECTED IS 24
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

- Both “E” and “Y” will insert DELETEs into the associated history table.
- Only UPDATE / INSERT are impacted
Limitations, Restrictions and Challenges and . . . . 

- If the SYSIBMADM.GET_ARCHIVE global variable is set to 'Y' and the ARCHIVESENSITIVE bind option is set to 'Y', an archive-enabled table cannot be referenced in an inline SQL table function or in the definition of a row permission or column mask that is activated by a data change statement or query.

- A SELECT statement can't reference both BASE and HISTORY (I am confused !!)

```
SELECT * FROM IDDG15.ARCHBASE UNION ALL SELECT * FROM IDDG15.ARCHHIST
ORDER BY EMPNO

EMPNO FIRSTNAME MIDINIT LASTNAME WORKDEPT PHONE HIREDATE
----- ------------ ------ --------------------- ------ ------- --------
0000015 Bob        I      Smith         A00    3978  2000-01-15
000020 Michael    L      Thompson     B01    3478  1973-10-10
000030 Sally      B      Evan          D01    4738  1975-04-05
000050 John       B      Geyer         E01    6798  1949-08-17
000060 Irving     F      Stern         D11    6423  1973-09-14
000070 Eva        D      Polaski       D21    7831  1980-09-30

SELECT A.EMPNO FROM IDDG15.ARCHBASE A, IDDG15.ARCHHIST B WHERE A.EMPNO=B.EMPNO
This one also works
```
Limitations, Restrictions and Challenges and . . .

- Let’s do the UNION with GET_ARCHIVE=‘Y’

```
SET SYSIBMADM.GET_ARCHIVE = 'Y';
SELECT * FROM IUD15.ARCHBASE UNION ALL SELECT * FROM IUD15.ARCHHIST
ORDER BY EMPNO
```

- Be prepared to get duplicates
- Due to GET_ARCHIVE = ‘Y’ the HISTORY table is accessed twice
**Limitations, Restrictions and Challenges and . . . .**

- ** Lets do the JOIN with GET_ARCHIVE=‘Y’  

```
SET SYSPRIVADM.GET_ARCHIVE = 'Y' ;
SELECT * FROM IDU015.ARCERBASE A , IDU015.ARCHRIST B
WHERE A.EMPHO=B.EMPHO
ORDER BY A.EMPHO
```

<table>
<thead>
<tr>
<th>EMPHO</th>
<th>FIRSTNAME</th>
<th>MIDINIT</th>
<th>LASTNAME</th>
<th>WORKSHEET</th>
<th>PHONE_NO</th>
<th>HIREDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000015</td>
<td>Bob</td>
<td>I</td>
<td>Smith</td>
<td>A00</td>
<td>3978</td>
<td>2000-01-15</td>
</tr>
<tr>
<td>000020</td>
<td>MICHAEL</td>
<td>L</td>
<td>THOMPSON</td>
<td>B01</td>
<td>3476</td>
<td>1973-10-10</td>
</tr>
</tbody>
</table>

— Now you can see which rows have been DELETED from the BASE and RE-INSERTED
Limitations, Restrictions and Challenges and . . . .

- Let’s INSERT into BASE from HISTORY – and RE-DELETE

```
SET SYSIMADM.MOVE TO_ARCHIVE = 'E';
-----------------------------
INSERT INTO IDUG15.ARCHBASE SELECT * FROM IDUG15.ARCHHIST WHERE EMPNO = '000015';
-----------------------------
DSN615I NUMBER OF ROWS AFFECTED IS 1

COMMIT;

DELETE FROM IDUG15.ARCHBASE WHERE EMPNO = '000015';
-----------------------------
DSN7408I SQLCODE = -403, ERROR: AN INSERTED OR UPDATED VALUE IS INVALID
BECAUSE INDEX IN INDEX SPACE ARCH164 CONSTRAINTS COLUMNS OF THE TABLE
SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THESE COLUMNS.
RID OF EXISTING ROW IS X'0000000201'.
```

- Think about UNIQUE indexes for the HISTORY table!!
Limitations, Restrictions and Challenges

- You can’t:
  - ADD column to history ➔ ADD column to BASE, DB2 will handle history
  - SECURITY LABEL column or ROW ACCESS CONTROL
  - Column attribute alterations for any of them
  - RENAME column or RENAME table
  - DROP column
  - ADD PERIOD (temporal) for any of them
  - ADD versioning (temporal)
  - PK / FK for history
  - ROTATE partition
Limitations, Restrictions and Challenges

- You can’t:
  - ADD column LONG VARCHAR / VARGRAPHIC
  - ALTER CCSID
  - MQTT involved
  - ADD CLONE
  - AUX objects
  - IDENTITY column
  - PENDING CHANGES ?!
  - VIEW and DGT
  - Only one table in tablespace ➔ has to be UTS

```
ALTER TABLESPACE IDUG2015.ARCHBASE
SEGSIZE 08;

DSNT404I SQLCODE = 610, WARNING:
A CREATE/ALTER ON OBJECT
IDUG2015.ARCHBASE HAS PLACED OBJECT
IN ADVISORY REORG PENDING
```
How to cope with Schema Changes

- Schema changes can be a challenge as noted on the previous slides

```
ALTER TABLE IDUG15.ARCHHIST ADD COLUMN NEWCOL CHAR(4) NOT NULL WITH DEFAULT;

DSNT4081 SQLCODE = -20525, ERROR: THE REQUESTED ACTION IS NOT VALID FOR TABLE IDUG15.ARCHHIST BECAUSE THE TABLE IS THE WRONG TYPE OF TABLE. REASON CODE = 13

ALTER TABLE IDUG15.ARCHBASE ADD COLUMN NEWCOL CHAR(4) NOT NULL WITH DEFAULT;

DSNT4081 SQLCODE = -20385, ERROR: THE STATEMENT CANNOT BE PROCESSED BECAUSE THERE ARE PENDING DEFINITION CHANGES FOR OBJECT IDUG2015.ARCHBASE OF TYPE TABLESPACE (REASON 2)
```
How to cope with Schema Changes

- Once the PENDING REORG is fixed:

```sql
ALTER TABLE IDUG15.ARCHBASE
ADD COLUMN NEWCOL CHAR(4)
NOT NULL WITH DEFAULT;

DSNE616I STATEMENT EXECUTION
WAS SUCCESSFUL, SQLCODE IS 0
```

- DB2 does really add the new column to the history automatically

```
ARCHBASE  EMPNO  FIRSTNAME  MIDINIT
          BONUS
          COMM
          NEWCOL

ARCHIST   EMPNO  FIRSTNAME  MIDINIT
          BONUS
          COMM
          NEWCOL
```
How to cope with Schema Changes

- The method to handle schema changes which are allowed for archive-enabled and history tables is very identical to those related to temporal tables:
  - Remove the tie: ALTER TABLE ARCHBASE DISABLE ARCHIVE
  - Potentially unload data from both base and history
  - Perform schema changes – and assure the restrictions aren’t violated.
  - Potentially load the data back (no need to think about PERIODOVERRIDE and TRANSIDOVERRIDE)
  - Alter table ENABLE ARCHIVE.
  - Copy, Runstats, Rebind etc.
Utilities and LISTDEF processing

- LISTDEF has a new parameter – it is a great feature – but.....

```sql
OPTIONS PREVIEW
LISTDEF TEMPOR1
  INCLUDE TABLESPACE IDUG2015.ARCHBASE
LISTDEF TEMPOR2
  INCLUDE TABLESPACE IDUG2015.ARCHBASE ALL
LISTDEF TEMPOR3
  INCLUDE TABLESPACE IDUG2015.ARCHBASE ARCHIVE
LISTDEF TEMPOR4
  INCLUDE TABLESPACE IDUG2015.ARCHBASE ALL ARCHIVE
LISTDEF TEMPOR5
  INCLUDE TABLESPACE IDUG2015.ARCHBASE
LISTDEF TEMPOR6
  INCLUDE TABLESPACE IDUG2015.ARCHHIST
LISTDEF TEMPOR7
  INCLUDE TABLESPACE IDUG2015.ARCHHIST ALL
LISTDEF TEMPOR8
  INCLUDE TABLESPACE IDUG2015.ARCHHIST ARCHIVE
```
Utilities and LISTDEF processing

INCLUDE base : only BASE selected
INCLUDE base ALL : only BASE selected
INCLUDE base ARCHIVE : NONE selected
INCLUDE base ALL ARCHIVE : only HISTORY selected

INCLUDE hist : only HISTORY selected
INCLUDE hist ALL : only BASE selected
INCLUDE hist ARCHIVE : only HISTORY selected
INCLUDE hist ALL ARCHIVE : only HISTORY selected
Utilities and LISTDEF processing

- Always include “special object” keywords.
- Best practices – no need to memorize.
- Might be overkill . . . But better safe than sorry

```
OPTIONS PREVIEW
LISTDEF TEMPOR0

INCLUDE TABLESPACE DAP02DB.DAP02TS

INCLUDE TABLESPACE DAP02DB.DAP02TS
   ALL ARCHIVE

INCLUDE TABLESPACE DAP02DB.DAP02TS
   ALL HISTORY
```
Utilities and LISTDEF processing

- Some of the newer object types have special considerations regarding utilities.
  - RUNSTATS for CLONE related objects
  - LOAD for Temporal tables
  - RECOVER for Temporal tables

- Archive tables seem a lot easier to deal with
  - No restrictions identified when loading data – including REPLACE of HISTORY
  - Archive-enabled and archive table can be recovered independently
WRAP UP

- Implementing ARCHIVE tables is very straight forward
- Minimal application changes to adopt this technology
- No need to be scared implementing archive tables

- Hoping this session was useful and that you can benefit right away.

THANK YOU !!