The DB2 log in practice
Disclaimer

- I do not work for IBM
- I did not develop any part of DB2
- All information in this presentation is based on publicly available API documentation, examination of the behavior of DB2 for z/OS, and long hours of trial and error
- This presentation has been made to the best of my knowledge but I cannot guarantee correctness
- Some aspects are simplified because it either makes things easier to understand or I just don’t know any better
The basics
Purpose of the DB2 log

- Essential for maintaining the consistency of the database
- Also essential for recovering objects
- DB2 considers the log so important that it has the option to keep two identical copies
- Log records are only added, existing log records are never change or removed
- Bottom line: The log is a protocol of every event that modified data
Role of the log in ACID properties

- Allows DB2 to roll back a transaction
  - Allows DB2 to undo all changes made by a transaction
  - Either after an explicit ROLLBACK or if the transaction fails for any other reason
  - Key element in guaranteeing **atomicity** of transactions

- Allows DB2 to achieve consistency after a crash
  - Write-ahead-log: Changes are written to the log first, then to the table space
  - Key element in guaranteeing **durability** of transactions
How does DB2 write to the log?

- Log records are written into the log output buffer (fixed in real storage), flushed to DASD when full

- At COMMIT time, the log buffer is synchronously written to the active log (on DASD)
  - Unlike the modified pages, which stay in the BP
  - COMMIT is not done until log records are on DASD

- Current active log data set is copied into an archive log (on DASD or tape) when it is full, or when the ARCHIVE LOG command is invoked
Log data sets
Active log data sets

- **Active log**
  - VSAM clusters, always on DASD
  - Fixed number, used in cycles
  - DSNJU003 can add or remove active log data sets
  - DB2 may have exclusive access (SHROPTIONS)
  - DB2 keeps track of the active log data sets in its BSDS
Active log data sets

- Prefix (up to 30 characters)
- Data set number

<table>
<thead>
<tr>
<th>Copy 1 Prefix</th>
<th>Data set number</th>
<th>Cluster</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS01</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS01.DATA</td>
<td>Cluster</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS02</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS02.DATA</td>
<td>Cluster</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS03</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS03.DATA</td>
<td>Cluster</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS04</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>DSNA10.DBAG.LOGCOPY1</td>
<td>DS04.DATA</td>
<td>Cluster</td>
<td></td>
</tr>
</tbody>
</table>
Active log data sets

DB2

LOGCOPY1.DS01
NOTREUSABLE

LOGCOPY1.DS04
NOTREUSABLE

LOGCOPY1.DS02
REUSABLE

LOGCOPY1.DS03
REUSABLE

DSNA10.DBAG.ARCLOG1.D16042.T0950411.A0001807
DSNA10.DBAG.ARCLOG1.D16042.T1001539.A0001808
DSNA10.DBAG.ARCLOG1.D16042.T1001539.B0001808
Active log data sets

DB2

LOGCOPY1.DS01
NOTREUSABLE

LOGCOPY1.DS04
NOTREUSABLE

LOGCOPY1.DS03
REUSABLE

LOGCOPY1.DS02
REUSABLE

DSNA10.DBAG.ARCLOG1.D16042.T0950411.A0001807
DSNA10.DBAG.ARCLOG1.D16042.T1001539.A0001808
DSNA10.DBAG.ARCLOG1.D16042.T1001539.B0001808
DSNA10.DBAG.ARCLOG1.D16042.T1001608.A0001809
Active log data sets

DB2

LOGCOPY1.DS01
NOTREUSABLE

LOGCOPY1.DS02
REUSABLE

LOGCOPY1.DS03
REUSABLE

LOGCOPY1.DS04
REUSABLE

DSNA10.DBAG.ARCLOG1.D16042.T0950411.A0001807
DSNA10.DBAG.ARCLOG1.D16042.T1001539.A0001808
DSNA10.DBAG.ARCLOG1.D16042.T1001539.B0001808
DSNA10.DBAG.ARCLOG1.D16042.T1001608.A0001809
Active log data sets

As long as DS04 is not overwritten, these two data sets contain the same information.

- LOGCOPY1.DS01
  - NOTREUSABLE

- LOGCOPY1.DS02
  - REUSABLE

- LOGCOPY1.DS03
  - REUSABLE

- LOGCOPY1.DS04
  - REUSABLE

- DSNA10.DBAG.ARCLOG1.D16042.T1001539.A0001808
- DSNA10.DBAG.ARCLOG1.D16042.T1001608.A0001809
Archive log data sets

- Archive logs
  - Sequential data sets on DASD or tape
  - New archive log data sets are created when DB2 switches to the next active log data set
    - This initiates the “log offload” process
    - Log offload is asynchronous
  - Old data sets are not deleted (can use HSM)
  - DB2 keeps track of the archive log data sets in its BSDS (10,000 slots)
Archive log data sets

- **Prefix (up to 39 / 19 / 17 characters)**
- **Timestamp**
  - optional, format is `Dyyddmm.Thhmmss` or `Dyyyyyddd.Thhmmss`
- **Data set number**

<table>
<thead>
<tr>
<th>Copy 1 Prefix</th>
<th>Timestamp</th>
<th>Data set number</th>
<th>Log records</th>
<th>BSDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T0937534</td>
<td>A0001805</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T0937534</td>
<td>B0001805</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T0945003</td>
<td>A0001806</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T0945003</td>
<td>B0001806</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T0950411</td>
<td>A0001807</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T0950411</td>
<td>B0001807</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T1001539</td>
<td>A0001808</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
<tr>
<td>DSNA10.DBAG.ARCLOG1</td>
<td>D16042.T1001539</td>
<td>B0001808</td>
<td>Log records</td>
<td>BSDS</td>
</tr>
</tbody>
</table>
RBA and LRSN
Format of the log

- Binary format
- Sequence of varying length log records
  - Physically: Thousands of data sets
  - Logically: One continuous stream of data
- Every log record has an RBA
  - RBA = Relative Byte Address
  - The RBA is unique
- Every log record also has an LRSN
The RBA

- Position of the log record in the “imaginary eternal log” that DB2 has been writing since it was first started
- Increases faster when there is a lot of activity in DB2
- Up to DB2 V10: 6 bytes
  - $2^{48}$ Bytes or 256 TB
- Starting with DB2 V11: 10 bytes
  - $2^{80}$ Bytes or 1,073,741,824 TB or 1 YB
- Conversion: Add X’00 00 00 00’ to the left
The RBA

- Find out current RBA: -DISPLAY LOG

```
DSNJ370I  !DBAG DSNJC00A LOG DISPLAY
CURRENT COPY1 LOG = DSNA10.DBAG.LOGCOPY1.DS04 IS 59% FULL
CURRENT COPY2 LOG = DSNA10.DBAG.LOGCOPY2.DS04 IS 59% FULL
    H/W RBA = 003336712D26
    H/O RBA = 003335349FFF
FULL LOGS TO OFFLOAD = 0 OF 8
OFFLOAD TASK IS (AVAILABLE)
SOFTWARE ACCELERATION IS ENABLED
DSNJ371I  !DBAG DB2 RESTARTED 12:38:34 MAY 11, 2016
    RESTART RBA 0026C63B9000
    CHECKPOINT FREQUENCY 5 MINUTES
    LAST SYSTEM CHECKPOINT TAKEN 18:49:56 JUL 6, 2016
DSN9022I  !DBAG DSNJC001 '-DIS LOG' NORMAL COMPLETION
```

H/O RBA: The highest RBA of all log records that are in the archive log

H/W RBA: The RBA of the last log record that DB2 has written
The LRSN

- Log record sequence number
- Really a timestamp – increases independently from rate of activity in DB2
- Up to DB2 V10: 6 bytes
  - Will overflow on 2042-09-17-23.53.47.370480
- Starting with DB2 V11: 10 bytes
  - 1 byte added for range, 3 bytes added for granularity
  - Will not overflow for the next 36,000 years or so
- Conversion: Add X’00’ to the left and X’00 00 00’ to the right
The LRSN

- Find out current LRSN:
  - Based on current time (STCK)
  - Therefore increases continuously even when DB2 is inactive
  - No SELECT statement returns current LRSN. Need to use a small application (e.g., a REXX)
  - Some data sharing groups also have an additional offset (can be seen by using DSNJU004) that needs to be added
Convert LRSN to timestamp

```
SELECT TIMESTAMP(X'\xD0\x38\xD1\x32\xE5\x3E' || X'\x00\x00')
FROM SYSIBM.SYSDUMMY1
```

This is your 6 byte LRSN

Two padding bytes because the TIMESTAMP function needs an 8-byte argument

Result: `COL0001`

```
2016-01-29-17.59.04.313824
```

For 10 byte LRSNs, cut off one byte on each side.
The 10 byte format

- For RBAs: increases the maximum size of the log by factor 4 billion
- For LRSNs, fixes two problems:
  - Increases the highest possible timestamp that can be represented by roughly 36,000 years
  - Avoids duplicate LRSNs by providing 16.7 million times the granularity (from 16 µs to 1 ps)

Old: D038D132E53E D038D132E53F
     D038D132E53F

New: 00D038D132E53E000000
      00D038D132E53E000001
      00D038D132E53E000002
      [...] (16.7 million records)
      00D038D132E53EFFFFFF
      00D038D132E53F000000
That was the theory.

Now let’s look at the gory details
Structure of the log

(not to any scale)

All log records have a fixed length log record header (LRH) that always contains the same fields. Data after the log record header depends on what the log record represents.
Log record types

- Indicated by type and subtype in the LRH
  - Unit of recovery log records
    - Begin of UR, Commit, Rollback
  - Data change log records
    - Describe physical changes to a page
    - Can represent insert, update, delete in a table space / index
    - Can also represent space map changes or other changes
  - Checkpoint log records
    - Created whenever DB2 creates a checkpoint
    - Contain list of open transactions, modified page sets
Data change log records

- Written whenever something on a page changes
- Always contains:
  - LRH
  - LGDBHEAD (fields for DBID, PSID, page#)
- Mostly insert / update / delete, in which case it also contains:
  - LGBENTRY (fields for OBID, slot# in page)
Making log records visible

- **DSN1LOGP**
  - Specify start and end RBA/LRSN
  - Optionally specify filters (DBID, PSID, URID, log record type)
  - Output is a hex dump of all matching log records
  - Some header fields are formatted, but the rest is hard to read
Let’s dissect a log record
Example

- DB2 V10 Log record as printed by DSN1LOGP
- INSERT of a row into a tablespace
- DSN1LOGP formats parts of LRH, LGDBHEAD
- Log record structure: SDSNMACS(DSNDQJ00)
The new row exactly as it appears in the table space including 6 byte row header

Indicates that this is a data change record with redo and undo information

Indicates that this log record represents a basic data page change

URID of the log record. Identical for all log records that belong to this transaction.

RBA of previous log record within this transaction

LRSN of the log record

DBID and PSID of the modified table space (DBID X'0142', PSID X'0002')

Page number of the page that was modified (Page X'00000002')

If compensation record: RBA of the log record that is compensated by this log record

Slot number (“ID map entry”) inside the page

OBID of the table to which the new row belongs (OBID X'0003')

Note that the log record does not contain its own RBA. The RBA is the position of the log record inside the imaginary eternal log. The BSDS keeps track of the first and last RBA of each log data set. DSN1LOGP prints the RBA before the formatted header fields.
INSERT, UPDATE, DELETE

- INSERT and DELETE are simple
  - Contain the entire row

- UPDATE is more complex
  - Contains before image and after image
  - Can be split to more than one log record
  - Roughly a dozen different update variations
  - With or without data capture changes
  - In-place or non in-place
  - Can change normal records to pointer records
UPDATE

- Non-DCC UPDATE records log a partial row
  - Only the bytes that changed
  - That’s all DB2 needs to apply the log record
- Tricky to restore the full row:
  - Identify page and slot number
  - Find an older full image of the row (where?)
  - Look for additional updates since the identified full image
  - Possible, but can take a long time
Redoing and undoing changes

- **REDO**: This log record contains information required to **apply** the change
  - In this case: DB2 inserts the row found in the log record into the table space
  - Example: Recovery process
    - Restore an image copy
    - Then apply all log records up to the desired point in time
Redoing and undoing changes

- **UNDO**: This log record contains information required to **reverse** the change
  - In this case: DB2 removes the row found in the log record from the table space
  - Example: Canceling a transaction
    - Reverse the effects of all changes that were made in the transaction that is being canceled
    - While DB2 reverses the changes, it writes log records to protocol what it is doing (compensation records)
Transaction / COMMIT

- Transaction start log record:
  - BEGIN UR (the RBA of this log record becomes the URID)
- Log records describing data changes
- Transaction end log records:
  - BEGIN COMMIT PHASE1
  - SWITCH PHASE 1 TO 2
  - END COMMIT PHASE 2
### Transaction / COMMIT

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000700</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000600</td>
<td></td>
</tr>
<tr>
<td>000800</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000700</td>
<td></td>
</tr>
<tr>
<td>000900</td>
<td>BEGIN COMMIT1</td>
<td>000500</td>
<td></td>
<td>000800</td>
<td></td>
</tr>
<tr>
<td>000A00</td>
<td>PHASE 1 TO 2</td>
<td>000500</td>
<td></td>
<td>000900</td>
<td></td>
</tr>
<tr>
<td>000B00</td>
<td>END COMMIT2</td>
<td>000500</td>
<td></td>
<td>000A00</td>
<td></td>
</tr>
</tbody>
</table>
# Multiple Parallel Transactions

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td>000500</td>
</tr>
<tr>
<td>000700</td>
<td>BEGIN UR</td>
<td>000700</td>
<td></td>
<td>000700</td>
<td></td>
</tr>
<tr>
<td>000800</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td>000600</td>
</tr>
<tr>
<td>000900</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td>000800</td>
</tr>
<tr>
<td>000A00</td>
<td>INSERT</td>
<td>000700</td>
<td>N</td>
<td>000700</td>
<td>000700</td>
</tr>
<tr>
<td>000B00</td>
<td>BEGIN COMMIT1</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td>000900</td>
</tr>
<tr>
<td>000C00</td>
<td>PHASE 1 TO 2</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td>000B00</td>
</tr>
<tr>
<td>000D00</td>
<td>INSERT</td>
<td>000700</td>
<td>N</td>
<td>000700</td>
<td>000A00</td>
</tr>
<tr>
<td>000E00</td>
<td>END COMMIT2</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td>000C00</td>
</tr>
<tr>
<td>000F00</td>
<td>BEGIN COMMIT1</td>
<td>000700</td>
<td></td>
<td>000700</td>
<td>000D00</td>
</tr>
<tr>
<td>001000</td>
<td>PHASE 1 TO 2</td>
<td>000700</td>
<td></td>
<td>000700</td>
<td>000F00</td>
</tr>
<tr>
<td>001100</td>
<td>END COMMIT2</td>
<td>000700</td>
<td></td>
<td>000700</td>
<td>001000</td>
</tr>
</tbody>
</table>
Transaction / ROLLBACK

- Transaction start log record:
  - BEGIN UR

- Log records describing data changes

- Transaction end log records (for commit):
  - BEGIN ABORT
  - Log records describing how all changes are undone
  - END ABORT
### Transaction / ROLLBACK

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td>000500</td>
</tr>
<tr>
<td>000700</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td>000600</td>
</tr>
<tr>
<td>000800</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td>000700</td>
</tr>
<tr>
<td>000900</td>
<td>BEGIN ABORT</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td>000800</td>
</tr>
</tbody>
</table>

- DB2 must undo all changes from this transaction
- It follows the “undo next” chain
- For each log record that carries UNDO information:
  - The change that this log record describes is reverted
  - A new log record is written, documenting what was done
# Transaction / ROLLBACK

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000700</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000600</td>
<td></td>
</tr>
<tr>
<td>000800</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000700</td>
<td></td>
</tr>
<tr>
<td>000900</td>
<td>BEGIN ABORT</td>
<td>000500</td>
<td></td>
<td>000800</td>
<td></td>
</tr>
<tr>
<td>000A00</td>
<td>UPDATE</td>
<td>000500</td>
<td>Y</td>
<td>000800</td>
<td>000900</td>
</tr>
</tbody>
</table>
## Transaction / ROLLBACK

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000700</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000600</td>
<td></td>
</tr>
<tr>
<td>000800</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000700</td>
<td></td>
</tr>
<tr>
<td>000900</td>
<td>BEGIN ABORT</td>
<td>000500</td>
<td></td>
<td>000800</td>
<td></td>
</tr>
<tr>
<td>000A00</td>
<td>UPDATE</td>
<td>000500</td>
<td>Y</td>
<td>000800</td>
<td>000900</td>
</tr>
<tr>
<td>000B00</td>
<td>DELETE</td>
<td>000500</td>
<td>Y</td>
<td>000700</td>
<td>000A00</td>
</tr>
</tbody>
</table>
# Transaction / ROLLBACK

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td></td>
<td>000500</td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td></td>
</tr>
<tr>
<td>000700</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000600</td>
<td></td>
</tr>
<tr>
<td>000800</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000700</td>
<td></td>
</tr>
<tr>
<td>000900</td>
<td>BEGIN ABORT</td>
<td>000500</td>
<td></td>
<td>000800</td>
<td></td>
</tr>
<tr>
<td>000A00</td>
<td>UPDATE</td>
<td>000500</td>
<td>Y</td>
<td>000800</td>
<td>000900</td>
</tr>
<tr>
<td>000B00</td>
<td>DELETE</td>
<td>000500</td>
<td>Y</td>
<td>000700</td>
<td>000A00</td>
</tr>
<tr>
<td>000C00</td>
<td>DELETE</td>
<td>000500</td>
<td>Y</td>
<td>000600</td>
<td>000B00</td>
</tr>
</tbody>
</table>
### Transaction / ROLLBACK

<table>
<thead>
<tr>
<th>RBA</th>
<th>Type</th>
<th>URID</th>
<th>Compens.</th>
<th>Comp.RBA</th>
<th>Undo</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>000500</td>
<td>BEGIN UR</td>
<td>000500</td>
<td></td>
<td>000500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000600</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000700</td>
<td>INSERT</td>
<td>000500</td>
<td>N</td>
<td>000600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000800</td>
<td>UPDATE</td>
<td>000500</td>
<td>N</td>
<td>000700</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000900</td>
<td>BEGIN ABORT</td>
<td>000500</td>
<td></td>
<td>000800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000A00</td>
<td>UPDATE</td>
<td>000500</td>
<td>Y</td>
<td>000800</td>
<td>000900</td>
<td></td>
</tr>
<tr>
<td>000B00</td>
<td>DELETE</td>
<td>000500</td>
<td>Y</td>
<td>000700</td>
<td>000A00</td>
<td></td>
</tr>
<tr>
<td>000C00</td>
<td>DELETE</td>
<td>000500</td>
<td>Y</td>
<td>000600</td>
<td>000B00</td>
<td></td>
</tr>
<tr>
<td>000D00</td>
<td>END ABORT</td>
<td>000500</td>
<td></td>
<td>000C00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- This is the result
Applying the log (part 1)

- When DB2 applies log records (for example, when running RECOVER to do a point-in-time recovery), it will:
  - Start at a “baseline” point in time, such as a full copy
  - Identify the latest checkpoint before the baseline
  - Apply log records in forward direction using “REDO” information.
  - Including records from aborted transactions (both the regular and the compensation records) – also using the “REDO” information
Applying the log (part 2)

- When DB2 applies log records (for example, when running RECOVER to do a point-in-time recovery), it will:
  - Keep track of when transactions open and close
  - Use information from checkpoint records to learn about transactions that may be idle, but still open
  - After reaching the target PIT, undo changes from all records that belong to transactions that are still open, using “UNDO” information
  - Write log records that describe how changes from open transactions were undone
Indexes

- DB2 also writes log records for all indexes (including COPY NO indexes)
- Index log records describe:
  - Addition / Deletion of Keys
  - Addition / Deletion of RIDs
  - Index structure changes (e.g., page splits)
  - And more
LOBs

- LOBs with LOG YES
  - Log records for space map changes
  - Log records for data

- LOBs with LOG NO
  - Only log records for space map changes

- LOB updates are never in-place
  - Therefore, DB2 can always rollback a transaction, even if the LOB is LOG NO
Table spaces with LOG NO

- DB2 does not write any log records about data changes
- Improves performance
- No ROLLBACK possible
  - ROLLBACK results in RECP state
  - Programs may cancel a transaction when a SQL error occurs: Also results in RECP state
  - Need to recover to an image copy
Checkpoint records

- Written whenever a checkpoint is created. Contain information about:
  - all transactions in progress at the time of checkpoint
  - all objects that were modified by these transactions
  - and more information about the current status
- Essentially all the information about the state of all transactions, collected in one place
- Which is why DB2 looks for the last checkpoint on the log when it is restarted
How to find out who changed something
The problem

- In example table from DB2: DSN81010.EMP, the salary of one of the employees looks fishy
- DSN1LOGP cannot filter by column contents
- Data change records do not tell us who is responsible for the change
- Everything is binary data, not human readable (EBCDIC text is readable, though)
Some assumptions

- We are looking for a row that still exists
- The table space is not compressed
- The row has not moved since the change (e.g. because of a REORG)
<table>
<thead>
<tr>
<th>EMPNO</th>
<th>FIRSTNAME</th>
<th>MIDINIT</th>
<th>LASTNAME</th>
<th>WORKDEPT</th>
<th>PHONENO</th>
<th>HIREDATE</th>
<th>JOB</th>
<th>Top of Data</th>
<th>ELEVEL</th>
<th>SEX</th>
<th>BIRTHDATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>000010</td>
<td>CHRISTINE</td>
<td>I</td>
<td>HAASE</td>
<td>A00</td>
<td>3978</td>
<td>01.01.1965</td>
<td>PRES</td>
<td>18</td>
<td>F</td>
<td>14.08.1933</td>
<td>52750.00</td>
<td></td>
</tr>
<tr>
<td>000020</td>
<td>MICHAEL</td>
<td>L</td>
<td>THOMPSON</td>
<td>B01</td>
<td>3476</td>
<td>10.10.1973</td>
<td>MANAGER</td>
<td>18</td>
<td>M</td>
<td>02.02.1948</td>
<td>41250.00</td>
<td></td>
</tr>
<tr>
<td>000030</td>
<td>SALLY</td>
<td>A</td>
<td>KUAN</td>
<td>C01</td>
<td>4738</td>
<td>05.04.1975</td>
<td>MANAGER</td>
<td>20</td>
<td>F</td>
<td>11.05.1941</td>
<td>36250.00</td>
<td></td>
</tr>
<tr>
<td>000050</td>
<td>JOHN</td>
<td>B</td>
<td>GEYER</td>
<td>E01</td>
<td>6709</td>
<td>17.06.1949</td>
<td>MANAGER</td>
<td>16</td>
<td>M</td>
<td>15.09.1925</td>
<td>40175.00</td>
<td></td>
</tr>
<tr>
<td>000060</td>
<td>IRVING</td>
<td>F</td>
<td>STERN</td>
<td>D11</td>
<td>6423</td>
<td>14.09.1973</td>
<td>MANAGER</td>
<td>16</td>
<td>M</td>
<td>07.07.1945</td>
<td>32250.00</td>
<td></td>
</tr>
<tr>
<td>000070</td>
<td>EVA</td>
<td>D</td>
<td>PULASKI</td>
<td>D21</td>
<td>7831</td>
<td>30.09.1980</td>
<td>MANAGER</td>
<td>16</td>
<td>M</td>
<td>26.05.1953</td>
<td>36170.00</td>
<td></td>
</tr>
<tr>
<td>000090</td>
<td>EILEEN</td>
<td>W</td>
<td>HENDERSON</td>
<td>E11</td>
<td>5498</td>
<td>15.08.1970</td>
<td>MANAGER</td>
<td>16</td>
<td>M</td>
<td>15.05.1941</td>
<td>29750.00</td>
<td></td>
</tr>
<tr>
<td>001000</td>
<td>THEODORE</td>
<td>D</td>
<td>SPENSER</td>
<td>E21</td>
<td>0972</td>
<td>19.06.1960</td>
<td>MANAGER</td>
<td>14</td>
<td>M</td>
<td>18.12.1956</td>
<td>26150.00</td>
<td></td>
</tr>
<tr>
<td>001110</td>
<td>VINCENZO</td>
<td>G</td>
<td>LUCCHESI</td>
<td>A00</td>
<td>3490</td>
<td>16.05.1958</td>
<td>SALESPREP</td>
<td>19</td>
<td>M</td>
<td>05.11.1929</td>
<td>46500.00</td>
<td></td>
</tr>
<tr>
<td>001120</td>
<td>SEAN</td>
<td>O</td>
<td>O'CONNELL</td>
<td>A00</td>
<td>2167</td>
<td>05.12.1963</td>
<td>CLERK</td>
<td>14</td>
<td>M</td>
<td>18.10.1942</td>
<td>29250.00</td>
<td></td>
</tr>
<tr>
<td>001330</td>
<td>DOLORES</td>
<td>M</td>
<td>QUINTANA</td>
<td>C01</td>
<td>4578</td>
<td>28.07.1971</td>
<td>ANALYST</td>
<td>16</td>
<td>F</td>
<td>15.09.1925</td>
<td>23800.00</td>
<td></td>
</tr>
<tr>
<td>001400</td>
<td>HEATHER</td>
<td>A</td>
<td>NICHOLLS</td>
<td>C01</td>
<td>1793</td>
<td>15.12.1976</td>
<td>ANALYST</td>
<td>18</td>
<td>F</td>
<td>19.01.1946</td>
<td>28420.00</td>
<td></td>
</tr>
<tr>
<td>001500</td>
<td>BRUCE</td>
<td>A</td>
<td>ADAMSON</td>
<td>C11</td>
<td>4510</td>
<td>12.02.1972</td>
<td>DESIGNER</td>
<td>18</td>
<td>M</td>
<td>17.05.1947</td>
<td>25280.00</td>
<td></td>
</tr>
<tr>
<td>001600</td>
<td>ELIZABETH</td>
<td>R</td>
<td>PIANKA</td>
<td>D11</td>
<td>3782</td>
<td>11.10.1977</td>
<td>DESIGNER</td>
<td>17</td>
<td>F</td>
<td>12.04.1955</td>
<td>22250.00</td>
<td></td>
</tr>
<tr>
<td>001700</td>
<td>MASATOSHI</td>
<td>J</td>
<td>YOSHIMURA</td>
<td>D11</td>
<td>2990</td>
<td>15.09.1978</td>
<td>DESIGNER</td>
<td>16</td>
<td>M</td>
<td>05.01.1951</td>
<td>24600.00</td>
<td></td>
</tr>
<tr>
<td>001800</td>
<td>MARILYN</td>
<td>S</td>
<td>SCOUTTEN</td>
<td>D11</td>
<td>1682</td>
<td>07.07.1973</td>
<td>DESIGNER</td>
<td>17</td>
<td>F</td>
<td>21.02.1949</td>
<td>21340.00</td>
<td></td>
</tr>
<tr>
<td>001900</td>
<td>JAMES</td>
<td>H</td>
<td>WALKER</td>
<td>D11</td>
<td>2986</td>
<td>26.07.1974</td>
<td>DESIGNER</td>
<td>16</td>
<td>M</td>
<td>25.06.1952</td>
<td>40900.00</td>
<td></td>
</tr>
<tr>
<td>002000</td>
<td>DAVID</td>
<td>D</td>
<td>BROWN</td>
<td>D11</td>
<td>4501</td>
<td>03.03.1966</td>
<td>DESIGNER</td>
<td>16</td>
<td>M</td>
<td>29.05.1941</td>
<td>27740.00</td>
<td></td>
</tr>
<tr>
<td>002100</td>
<td>WILLIAM</td>
<td>T</td>
<td>JONES</td>
<td>D11</td>
<td>0942</td>
<td>11.04.1979</td>
<td>DESIGNER</td>
<td>17</td>
<td>M</td>
<td>23.02.1953</td>
<td>18270.00</td>
<td></td>
</tr>
<tr>
<td>002200</td>
<td>JENNIFER</td>
<td>K</td>
<td>LUTZ</td>
<td>D11</td>
<td>0672</td>
<td>29.08.1968</td>
<td>DESIGNER</td>
<td>18</td>
<td>F</td>
<td>19.03.1948</td>
<td>29340.00</td>
<td></td>
</tr>
<tr>
<td>002300</td>
<td>JAMES</td>
<td>J</td>
<td>JEFFERSON</td>
<td>D21</td>
<td>4265</td>
<td>21.11.1966</td>
<td>CLERK</td>
<td>14</td>
<td>M</td>
<td>30.05.1935</td>
<td>22180.00</td>
<td></td>
</tr>
<tr>
<td>002400</td>
<td>SALVATORE</td>
<td>M</td>
<td>MARINO</td>
<td>D21</td>
<td>3700</td>
<td>05.12.1979</td>
<td>CLERK</td>
<td>17</td>
<td>M</td>
<td>31.03.1954</td>
<td>20760.00</td>
<td></td>
</tr>
<tr>
<td>002500</td>
<td>DANIEL</td>
<td>S</td>
<td>SMITH</td>
<td>D21</td>
<td>0961</td>
<td>30.10.1969</td>
<td>CLERK</td>
<td>15</td>
<td>M</td>
<td>12.11.1939</td>
<td>19180.00</td>
<td></td>
</tr>
<tr>
<td>002600</td>
<td>SYBILL</td>
<td>V</td>
<td>JOHNSON</td>
<td>D21</td>
<td>8593</td>
<td>11.09.1975</td>
<td>CLERK</td>
<td>16</td>
<td>F</td>
<td>05.10.1936</td>
<td>17250.00</td>
<td></td>
</tr>
<tr>
<td>002700</td>
<td>MARTA</td>
<td>L</td>
<td>PEREZ</td>
<td>D21</td>
<td>9001</td>
<td>30.09.1980</td>
<td>CLERK</td>
<td>15</td>
<td>F</td>
<td>26.05.1953</td>
<td>27380.00</td>
<td></td>
</tr>
<tr>
<td>002800</td>
<td>ETHEL</td>
<td>R</td>
<td>SCHNEIDER</td>
<td>E11</td>
<td>8997</td>
<td>24.03.1967</td>
<td>OPERATOR</td>
<td>17</td>
<td>F</td>
<td>28.03.1936</td>
<td>26250.00</td>
<td></td>
</tr>
<tr>
<td>002900</td>
<td>JOHN</td>
<td>R</td>
<td>PARKER</td>
<td>E11</td>
<td>4502</td>
<td>30.05.1980</td>
<td>OPERATOR</td>
<td>12</td>
<td>M</td>
<td>09.07.1946</td>
<td>15340.00</td>
<td></td>
</tr>
<tr>
<td>003000</td>
<td>PHILIP</td>
<td>X</td>
<td>SMITH</td>
<td>E11</td>
<td>2095</td>
<td>19.06.1972</td>
<td>OPERATOR</td>
<td>14</td>
<td>M</td>
<td>27.10.1936</td>
<td>17750.00</td>
<td></td>
</tr>
<tr>
<td>003100</td>
<td>MAUDE</td>
<td>F</td>
<td>SETRIGHT</td>
<td>E11</td>
<td>3332</td>
<td>12.09.1964</td>
<td>OPERATOR</td>
<td>12</td>
<td>F</td>
<td>21.04.1931</td>
<td>15900.00</td>
<td></td>
</tr>
<tr>
<td>003200</td>
<td>RAMLAL</td>
<td>V</td>
<td>MEHTA</td>
<td>E21</td>
<td>9990</td>
<td>07.07.1965</td>
<td>FIELDREP</td>
<td>16</td>
<td>M</td>
<td>11.08.1932</td>
<td>19950.00</td>
<td></td>
</tr>
<tr>
<td>003300</td>
<td>WING</td>
<td>L</td>
<td>LEE</td>
<td>E21</td>
<td>2103</td>
<td>23.02.1976</td>
<td>FIELDREP</td>
<td>14</td>
<td>M</td>
<td>18.07.1941</td>
<td>25370.00</td>
<td></td>
</tr>
<tr>
<td>003400</td>
<td>JASON</td>
<td>R</td>
<td>GOUNOT</td>
<td>E21</td>
<td>5698</td>
<td>05.05.1947</td>
<td>FIELDREP</td>
<td>16</td>
<td>M</td>
<td>17.05.1926</td>
<td>23840.00</td>
<td></td>
</tr>
<tr>
<td>003500</td>
<td>DIAN</td>
<td>J</td>
<td>HEMMINGER</td>
<td>A00</td>
<td>3970</td>
<td>01.01.1965</td>
<td>SALESPREP</td>
<td>18</td>
<td>F</td>
<td>14.08.1933</td>
<td>46500.00</td>
<td></td>
</tr>
<tr>
<td>003600</td>
<td>GREG</td>
<td>O</td>
<td>ORLANDO</td>
<td>A00</td>
<td>2167</td>
<td>05.05.1972</td>
<td>CLERK</td>
<td>14</td>
<td>M</td>
<td>18.10.1942</td>
<td>29250.00</td>
<td></td>
</tr>
<tr>
<td>003700</td>
<td>KIM</td>
<td>N</td>
<td>NATZ</td>
<td>C01</td>
<td>1793</td>
<td>15.12.1976</td>
<td>ANALYST</td>
<td>18</td>
<td>F</td>
<td>19.01.1946</td>
<td>28420.00</td>
<td></td>
</tr>
</tbody>
</table>

- Employee James Walker is making twice as much as the average designer.
Step 1

- Determine internal IDs of the affected object
  - In our case: DBID 0x0106, PSID 0x0004
- Determine which row is affected
  - `SELECT HEX(RID(DSN81010.EMP))`  
    `FROM DSN81010.EMP`  
    `WHERE EMPNO = '000190'`
  - Result: 00000000000000211
  - Red = page number 0x00000002, blue = slot number 0x11
- Determine approximate time of change (smaller time frame = better)
Step 2

- Run DSN1LOGP

```//DSN1LOGP EXEC PGM=DSN1LOGP
//STEPLIB DD DISP=SHR,DSN=DSNA10.SDSNLOAD
//BSDS DD DISP=SHR,DSN=DSNA10.DBAG.BSDS01
//SYSPRINT DD SYSOUT=* 
//SYSSUMRY DD SYSOUT=* 
//SYSIN DD * 
RBASTART (00ECD81ECF8F) 
RBAEND (00ECD81FAD0B) 
DATAONLY (YES) 
DBID (0106) 
OBID (0004) 
RID (0000000211) 
SUBTYPE (1) 
/*
```

Subtype 1 means Undo/Redo records, basic data page change
- DSN1LOGP converts the LRSN into a timestamp
- The EBCDIC text confirms it is the correct row
- The URID tells us in which transaction the change was made
- A field was changed from 20450.00 to 40900.00
- Standard representation would be 0xF002045000 and 0xF004090000
- DB2 starts logging after ...F0 because everything up to and including this byte has not changed
- Also, this column has an editproc...
Step 3

- Run DSN1LOGP again

```plaintext
//DSN1LOGP EXEC PGM=DSN1LOGP
//STEPLIB DD DISP=SHR, DSN=DSNA10.SDSNLOAD
//BSDS DD DISP=SHR, DSN=DSNA10.DBAG.BSDS01
//SYSPRINT DD SYSOUT=*  
//SYSSUMRY DD SYSOUT=*  
//SYSIN DD *
RBASTART (00ECD81ECF8F)
RBAEND (00ECD81FAD0B)
DATAONLY (YES)
URID (00ECD81FAAB9)  
/* Limit output to the transaction we have identified */
```
The BEGIN UR record for this URID shows, among other things, the user name and the plan name.
Revisit our assumptions

- We are looking for a row that still exists
  - So we can use the RID function
  - If the row has been deleted: no practical way to determine old row position
Revisit our assumptions

- The table space is not compressed
  - Log records contain raw binary data, therefore they are also compressed / encrypted
  - Compression: Need to decompress row using the correct decompression dictionary
  - Decompression dictionary from VSAM may not be the correct one
Revisit our assumptions

- The row has not moved since the change (e.g. because of a REORG)
  - If it has moved, the RID function will return the new position
  - But the log record refers to the old position
Thank you for listening

- Have a piece of code:

```rexx
/* REXX */
NUMERIC DIGITS 64
RBAFORMAT = 'E'         /* Output format (E = extended, B = basic) */
OFFSET = '000000000000' /* Set to STCK offset of data sharing group */
CVT = C2D(STORAGE(10, 4))
CHKBYTE = STORAGE(D2X(CVT + 304), 1)
CVTTZ = STORAGE(D2X(CVT + 304), 4)
IF BITAND(CHKBYTE, '80'X) = '80'X THEN CVTTZ = C2D(CVTTZ, 4)
ELSE CVTTZ = C2D(CVTTZ)
CVTTZ = CVTTZ * 1.048576 / 3600
CVTTZ = FORMAT(CVTTZ, 2, 0)
MS = (DATE('BASE') - 693595) * 24 * 60 * 60
MS = MS + TIME('SECONDS')
MS = MS - (CVTTZ * 3600)
MS = MS * 1000000
LRSNABIN = X2D(SUBSTR(D2X(MS), 1, 12))
IF LENGTH(OFFSET) = 20 THEN OFFSET = SUBSTR(OFFSET, 3, 12)
LRSN = D2X(LRSNABIN + X2D(OFFSET))
IF RBAFORMAT = 'E' THEN LRSN = '00'LRSN'000000'
SAY 'Current LRSN (with STCK offset 'OFFSET') = 'LRSN
```