Classification of Logging Approaches
Redo Logging - Example

Transaction T42: INSERT INTO WORLD_POPULATION VALUES(“Max”, “Meyer”, “male”, “Germany”, “Potsdam”, “1986/05/25”)

Operation order:

1. Write log entry <START T42>
2. Write log entry <T42, new_entry, WP, (“Max”, “Meyer”, …)> All tuple values must be logged.
3. Write log entry <COMMIT T42>
4. Flush log
5. Write changed element

Classification of Logging Approaches
Redo Logging - Example

Transaction T42: INSERT INTO WORLD_POPULATION VALUES(“Max”, “Meyer”, “male”, “Germany”, “Potsdam”, “1986/05/25”)

Operation order:

1. Write log entry <START T42>
2. Write log entry <T42, new_entry, WP, (“Max”, “Meyer”, …)>
3. Write log entry <COMMIT T42>
4. Flush log
5. Write changed element

1.) Identify committed transactions

Logging Details

Classification of Logging Approaches
Redo Logging - Example

Transaction T42: INSERT INTO WORLD_POPULATION VALUES(“Max”, “Meyer”, “male”, “Germany”, “Potsdam”, “1986/05/25”)

Operation order:

1. Write log entry <START T42>
2. Write log entry <T42, new_entry, WP, (“Max”, “Meyer”, …)>
3. Write log entry <COMMIT T42>
4. Flush log

---

5. Write changed element

1.) Identify committed transactions

T42 is committed:
Redo all changes <T42, …>

Logging Details

Classification of Logging Approaches
Redo Logging - Example

Transaction T42: INSERT INTO WORLD_POPULATION VALUES(“Max”, “Meyer”, “male”, “Germany”, “Potsdam”, “1986/05/25”)

Operation order:

1. Write log entry <START T42>
2. Write log entry <T42, new_entr…)>
3. Write log entry <COMMIT T42>
4. Flush log
5. Write changed element

T42 is not committed:
No changes to redo

1.) Identify committed transactions

Recovery

Logging Details

Logging in IMDBs
Redo Logging for Dictionary-Encoded Columns

- Three different change log types <T, X, v> to record primary data changes:
  - **Dictionary logs** for new dictionary entries
    - Required information: table ID, column ID, value, value ID
  - **Value logs** for inserts and updates
    - Required information: transaction ID, table ID, row ID, value ID vector
    - Optimization for Updates: The value ID vector can be compressed by only storing updated attributes using a bitmask and a shortened value ID vector (+ old row ID for an insert-only approach)
  - **Invalidation logs** for deletes and updates (insert-only)
    - Required information: transaction ID, table ID, row ID

Logging in IMDBs
Redo Logging for Dictionary-Encoded Columns - Example

Column first_name

Value ID vector		Dictionary
1	1	Martin
1	2	

Column last_name

Value ID vector		Dictionary
1	1	Schulze
2	2	Meyer

MVCC Columns

	Begin CID	End CID
1	1	∞
2	1	∞

Logging Details

Logging in IMDBs
Redo Logging for Dictionary-Encoded Columns - Example

Column first_name

	Value ID vector		Dictionary
1	1	1	Martin
2	1	2	Michael
3	2	3	

Column last_name

	Value ID vector		Dictionary
1	1	1	Schulze
2	2	2	Meyer
3	3	3	Berg

MVCC Columns

	Begin CID	End CID
1	1	∞
2	1	∞
3	4	∞

- **Transaction T42, CID 4**: INSERT INTO WORLD_POPULATION VALUES("Michael", "Berg")
  - <Dictionary log entry, WORLD_POPULATION, first_name, "Michael", 2>
  - <Dictionary log entry, WORLD_POPULATION, last_name, "Berg", 3>
  - <Value log entry, T42, WORLD_POPULATION, 3, {2, 3}>

Logging in IMDBs
Redo Logging for Dictionary-Encoded Columns - Example

Column first_name

	Value ID vector		Dictionary
1	1	1	Martin
2	1	2	Michael
3	2	3	
4	2	4	

Column last_name

	Value ID vector		Dictionary
1	1	1	Schulze
2	2	2	Meyer
3	3	3	Berg
4	1	4	

MVCC Columns

	Begin CID	End CID
1	1	∞
2	1	∞
3	4	5
4	5	∞

- **Transaction T42, CID 4**: INSERT INTO WORLD_POPULATION VALUES("Michael", "Berg")
  - <Dictionary log entry, WORLD_POPULATION, first_name, "Michael", 2>
  - <Dictionary log entry, WORLD_POPULATION, last_name, "Berg", 3>
  - <Value log entry, T42, WORLD_POPULATION, 3, {2, 3}>
- **Transaction T43, CID 5**: UPDATE WORLD_POPULATION SET last_name="Schulze" WHERE first_name="Michael"
  - <Invalidation log entry, T43, WORLD_POPULATION, 3>
  - <Value log entry, T43, WORLD_POPULATION, 4, {2, 1}>

Logging Details

Logging in IMDBs
Redo Logging for Dictionary-Encoded Columns - Example

Column first_name

	Value ID vector		Dictionary
1	1	1	Martin
2	1	2	Michael
3	2	3	
4	2	4	

Column last_name

	Value ID vector		Dictionary
1	1	1	Schulze
2	2	2	Meyer
3	3	3	Berg
4	1	4	

MVCC Columns

	Begin CID	End CID
1	1	∞
2	1	6
3	4	5
4	5	∞

- **Transaction T42, CID 4**: INSERT INTO WORLD_POPULATION VALUES("Michael", "Berg")
  - <Dictionary log entry, WORLD_POPULATION, first_name, "Michael", 2>
  - <Dictionary log entry, WORLD_POPULATION, last_name, "Berg", 3>
  - <Value log entry, T42, WORLD_POPULATION, 3, {2, 3}>
- **Transaction T43, CID 5**: UPDATE WORLD_POPULATION SET last_name="Schulze" WHERE first_name="Michael"
  - <Invalidation log entry, T43, WORLD_POPULATION, 3>
  - <Value log entry, T43, WORLD_POPULATION, 4, {2, 1}>
- **Transaction T44, CID 6**: DELETE FROM WORLD_POPULATION WHERE last_name="Meyer"
  - <Invalidation log entry, T44, WORLD_POPULATION, 2>

Logging Details

Checkpointing/Snapshotting

- Logging allows the DBMS to recover the database after a crash/restart
- But entire log has to be read each time
- **Checkpointing** allows the DBMS to ignore large segments of the log to reduce recovery time
- A snapshot writes the table data of the in-memory database to persistent memory

Checkpointing/Snapshotting
Main-Delta and Chunk based Architecture

- Snapshots for dictionary compressed main-delta stores:
  - Ideally written after each merge process
    - Snapshot contains new read-optimized main store
    - Invalidations in the main store and changes in the delta store are logged in log files
- Snapshots for chunk based architecture:
  - Ideally written for compressed chunks

Summary

- Databases need to provide durability guarantees to recover from failures
- There are different logging approaches to persist transaction changes
  - Redo logging is suitable for IMDBs
- Checkpointing allows the DBMS to ignore large segments of the log to reduce recovery time