ACID implementation in RDBMS

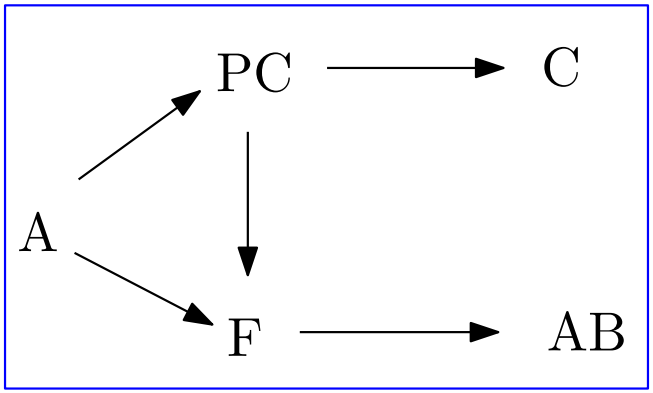
Michal Valenta

Department of Software Engineering
Faculty of Information Technology
Czech Technical University in Prague
©Michal Valenta, 2011

DB2, 2011

https://users.fit.cvut.cz/ valenta/
(odkaz **“Výuka na BIVŠ”**)

State Diagram of Transaction



- Active - starting state, data processing is in progress
- Partially Commited - after the last operation of transaction, waiting for journal to be written on disk
- Commited - finished, content of journal buffer is in persistent storage
- Failed - it is not possible to continue with transaction processing, ROLLBACK is initialized
- ABorted - ROLLBACK finished, database is in the same state as it was before the transaction started

ACID Properties

ACID Properties:

- Atomicity - Transaction is either completely done or it is completely rolled back.
- Consistency - Transaction transforms database from one consistent state to another consistent state.
  I.e. transaction may be inconsistent during the processing.
  Consistency is related to integrity constraints (IC)
- Independence - Data changes done in one transaction are not visible to the others unless it is commited (see levels of isolation),
- Durability - Changes made by finished transaction are stored in a persistent storage (for the case of loss of operational memory).

Begin and End of Transaction

End of transaction

- Explicit
  - COMMIT
  - ROLLBACK
- Implicit
  end of session (depends on client – either commit or rollback).

Begin of transaction

End of previous transaction or starting a new session.

Careful for client, it may be in autocommit mode.

Database recovery (after a crash)

Uses transaction journal (log file).
There are “change vectors” in journal.

Operations used for recovery

- UNDO
- REDO

Information from journal are used only for crash recovery.
There are another data structures to provide ROLLBACK and read consistency.

Database crashes – classification

Global errors - affect the whole instance

- Instance failure (power outage for example). Loss of operational memory.
- System errors; affect some transactions, but not the whole database (for example. deadlock, loss of connection between client and server).
- Media failure

Local errors (inside one transaction).

- Logical errors, which can be repair by rollback operation on a transaction (IC violated, divide by zero for example)

Recovery after system restart (Instance Failure)

synchronization marks

timestamps (in journal and database files headers); they are used to locate the point (in journal) where to start with database recovery

Requirements:

- Transactions which were not fully finished (commited) in time of system failure, must be rolled back.
- Transactions which were commited before system failure, but their data still remained in database buffer cache (only redo log buffer had already been stored on a persistent media (journal)) must be played again and their data must be written into data files.

Recovery after system restart (Instance Failure)

Technically recover consists of two parts (stages)

1. Roll Forward – Redo (replay) all change vectors stored in journal (records are stored in time order); Redo starts from chosen synchronization mark and runs until the last written record in journal. In fact, lost content of database buffer cache is reconstructed during this stage.
2. Roll Back – Roll back of transactions which were not yet commited in the time of system failure. Necessary rollback data (before images) are included in change vectors in journal.

Media Failure Recovery

Suppose one or more database files are missing / damaged.
Recovery depends on database mode (archive/noarchive):

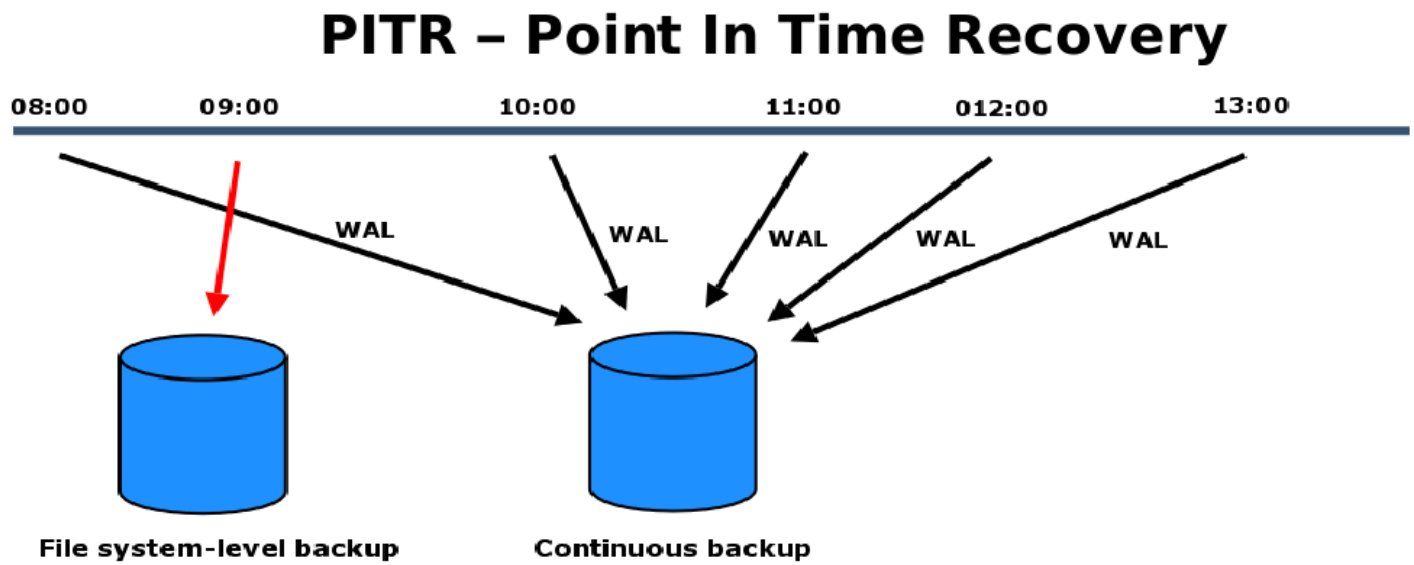
archive mode:

- Restore missing file(s) from backup.
- In archived logs locate timestamp where to start with recovery. All range of archived logs is necessary.
- Reply data changes until the end of logs (including online logs). Same as Instance recovery.
- The same method is used for PITR (Point In Time Recovery) only recovery process stops at required point in history.

nonarchive mode:

- We have not archived logs (nonarchive mode).
- Either push the whole database back in time (time of the last full backup).
- Or forgot damaged data.

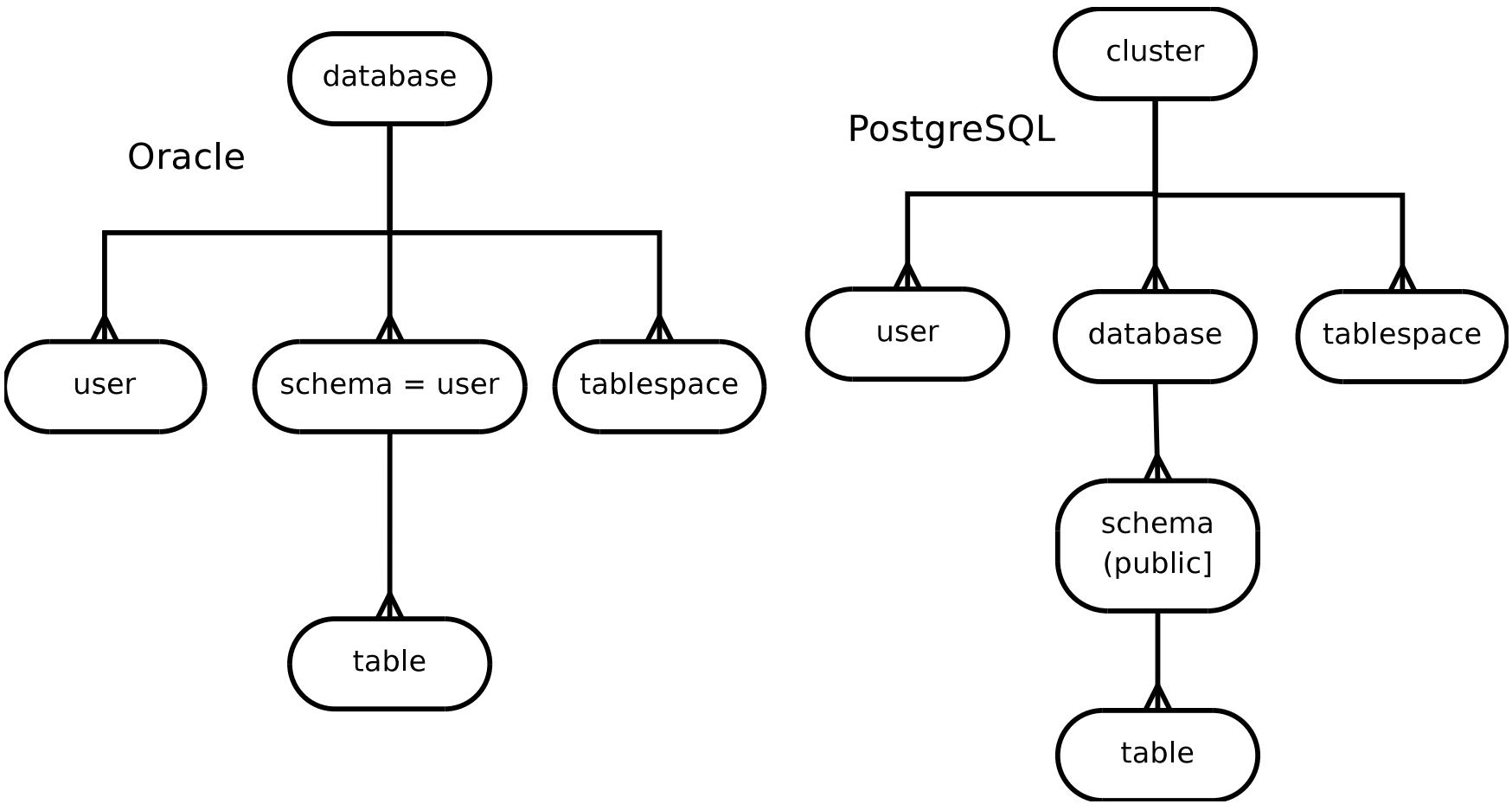
PostgreSQL - PITR Concept



- Hot backup
- Combines a file-system-level backup with backup of WAL files
- The file-system-level backup can be inconsistent
- Only restoration of an entire database cluster can be done
- Enables recover to the time of crash or an arbitrary chosen point in time
  since last file-system-level backup
- More difficult to administrate

27

Oracle and PostgreSQL



Remark: MySQL structure similar to PostgreSQL (without Schemes).

Oracle and PostgreSQL – key points

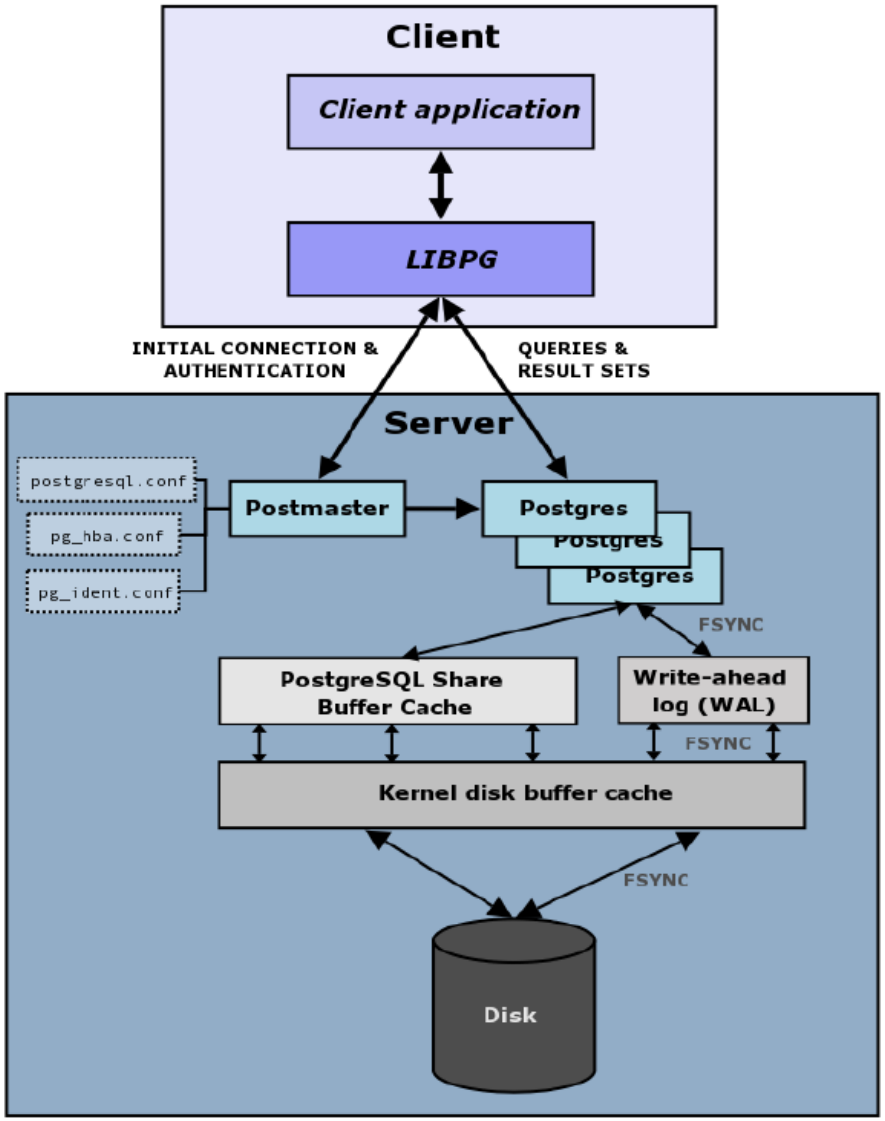
Oracle

- database creation has a huge overhead, only few instances on server
- new application typically uses a new scheme and tablespace (data separation)

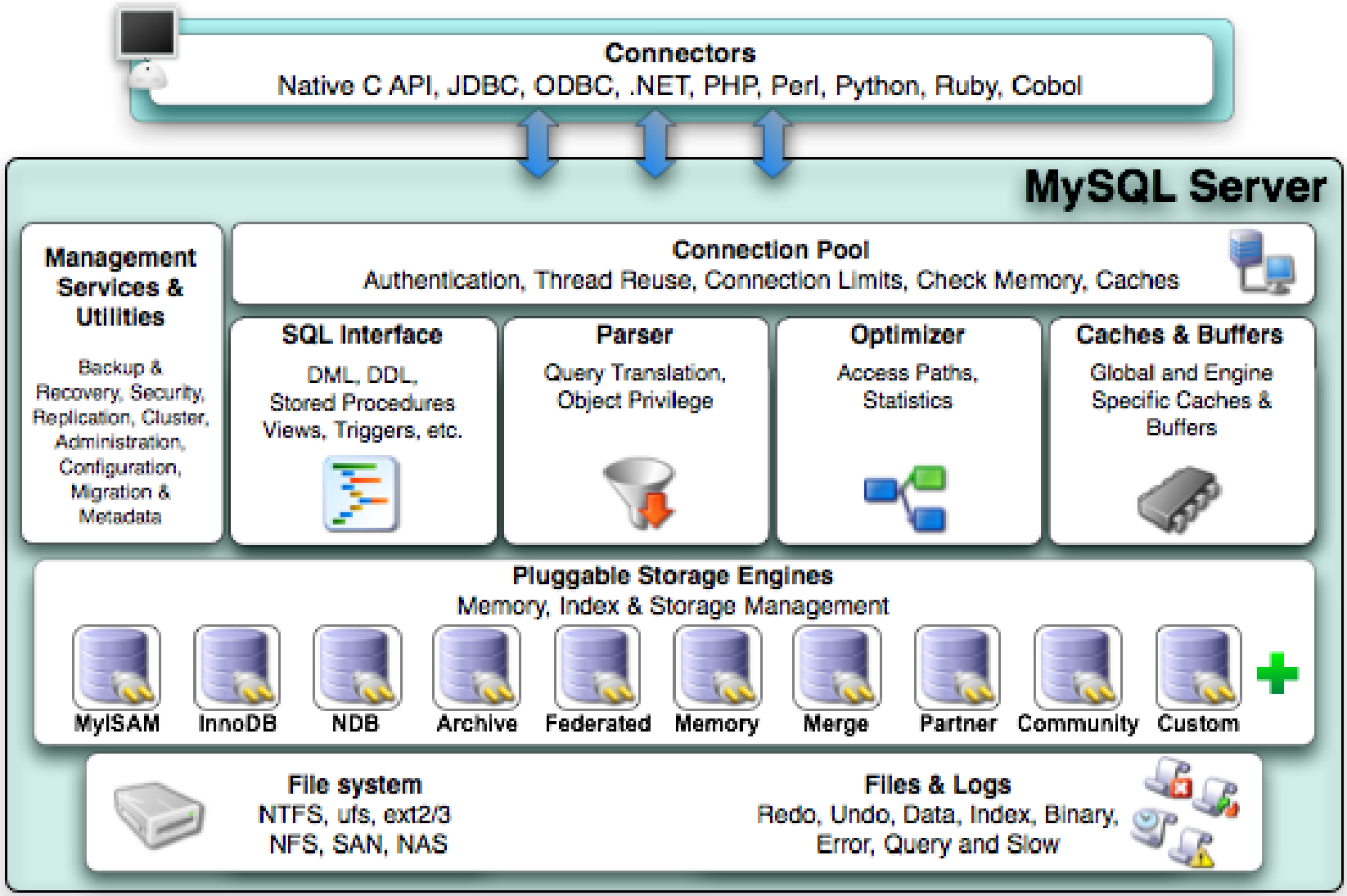
PostgreSQL, MySQL

- running cluster contains one cluster (operative) database
- overhead for creation a new database in cluster is small
- new database for new application

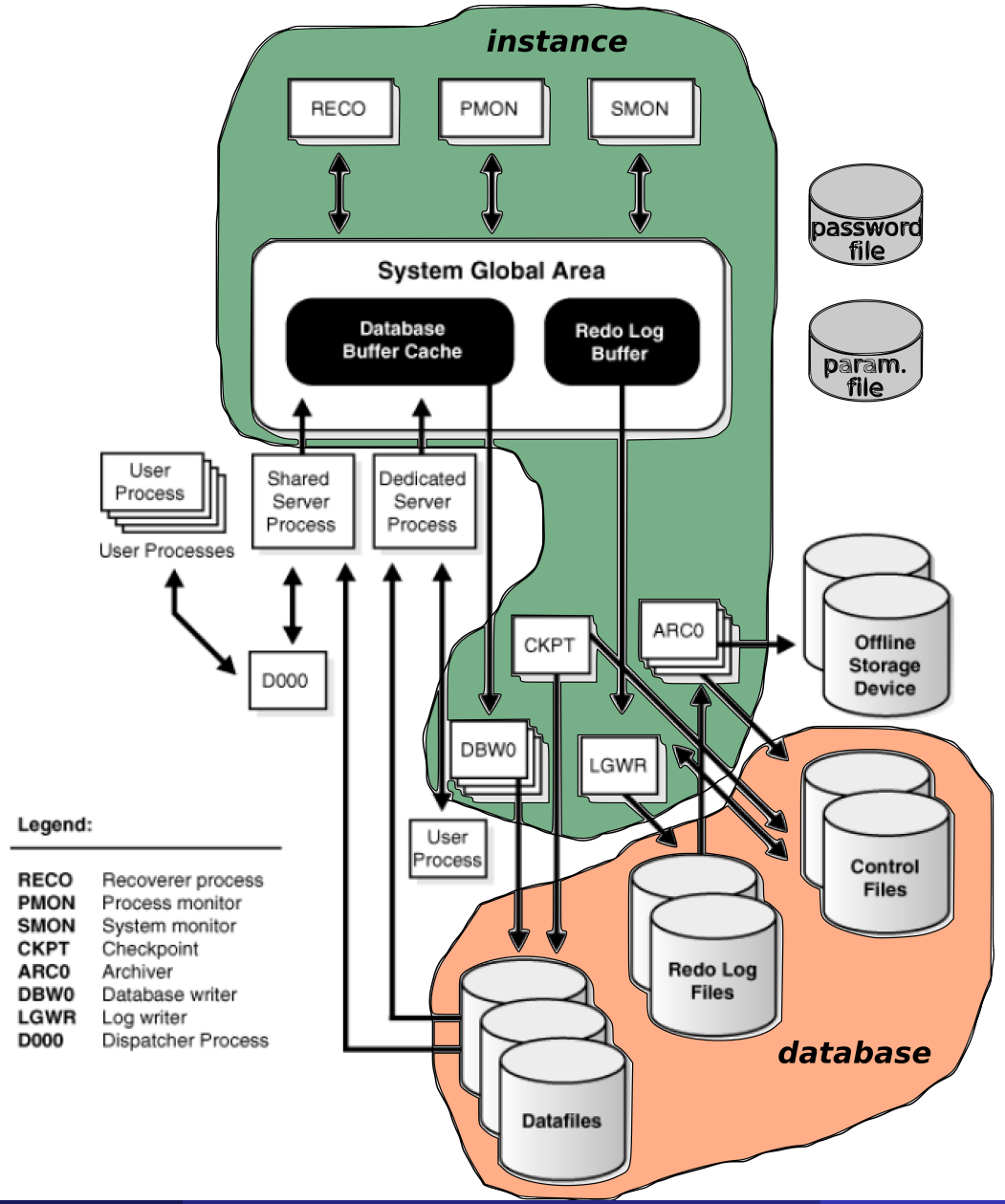
PostgreSQL Architecture



MySQL Architecture



Oracle Architecture



Oracle – DML processing

client send a DML statement (suppose UPDATE employees set salary=salary*1.1 where department_id = 10;)

1. data to be changed are loaded into buffer cache (server process); may be cache hit or start DBWR
2. apply exclusive locks on data to be changed (row-level locking)
3. alocate UNDO segment, copy old images of changing data into, mark them by transaction ID
4. set new values for locked data in buffer cache
5. in redo log buffer create change vectors (containing new and old values and transaction ID) for all affected rows

after finishing of DML:

changed data (and undo segments) still remains locked and are in buffer cache (eventually they can be moved into appropriate data files if buffer cache is needed for others transactions)

Oracle – COMMIT processing

1. create record in redo log buffer that transaction (identified by transaction ID) did commit
2. LGWR process writes the whole content of redolog buffer into journal (log file), typically using FSYNC or similar service; redo log buffer is written sequentially and is shared by all concurrently running sessions; it contains change vectors from both commited and yet uncommited transactions
3. when content of redo log buffer is in disk, user obtains information “transaction complete”; durability is now guaranteed
4. appropriate blocks in UNDO segments are allowed to be rewritten
5. exclusive locks on changing data are released

when instance crash appears in this time

Instance recovery from online logs (journals) is automatically done on system restart. Already commited transactions are recovered from logs, not commited transactions are rolled back. Atomicity and Durability properties are implemented.

PostgreSQL DML processing

conceptually similar to Oracle, differences:

- there are no UNDO segments, changing row is copied, original is marked not to change (locked), new copy is also exclusively locked
- physical structure: one table - one file unlike to Oracle
- after commit or rollback either original or new row is marked as free for rewriting
- implication for data maintenance: necessity of VACUUM statement

MVCC - remarks

- Both implementations (Oracle and PostgreSQL) can be regarded as MVCC (Multi Version Concurrency Control) architecture.
- In both cases it is possible to utilize systems to query data changes of one row across several transactions.
- Oracle already implemented this feature (in limited scope) in 10g server and laters.

Isolation Levels

Independence property is affected by chosen level of isolation:

- read uncommited
- read commited
- repeatable read
- serializable

It is important to understand, that level of isolation affects concurrency.

Isolation levels in practice

- Oracle and PostgreSQL did not support read uncommited
- MySQL supports read uncommited in engine MyISAM
- read commited is practically useful and widely used compromise between data consistency and concurrency processing (usually it is default)
- repeatable read and serializable are possible for DBMS, but usually are regarded as too blocking

Conclusion

- importance of (transaction) journal (log files) for crash recovery, atomicity and durability ACID properties
- MVCC and appropriate locking management for implementation of consistency and independence ACID properties