Database Systems II
Architecture and Implementation of DBS
Prof. R. Bayer, SS 99

Chapt. 1 Introduction
Chapt. 1.1 Tasks of a DBS

1. Provide the DBS-model: e.g.

- relational model : ralations, views
  SQL, System-Schema
- oo – model : class hierarchies
  methods
- Geo – DBS : geometric objects
  spatial indexes
  Geo – SQL
- digital library systems: documents
  archiving
  retrieval
- IMS . . .
- Codasyl . . .

Goals: high level of abstraction
efficiency
} goal conflict

Example:

SQL - - - -
Interpreter
rel. Algebra - - - -
Operators
tuple, attributes - - - -
access structures
files and records - - - -
File System
Operating System - - - -

HW - - - -

Alltogether 3 – 10 Interfaces

Subtasks: storage of relations
translation of SQL
identification of tuples
indexes, optimization

2. Multiuser Mode of Operation and Transactions

Goal: common access to large shared datasets

begintrans
  {consistency}
    A_1 ;
    A_2 ;
    .
    .
    A_n ;
  {consistency}
endtrans

at user – IF
as atomic
operation w.r. to

DB – state
i.e.
commit or
abort

⇒ state – transition of the DB
must be atomic

achievable by combination of techniques e.g. synchronization, durability (logging), recovery (including, storage failures), integrity conditions, etc.

ACID – Properties:

A :	Atomicity	:	for author of the transaction
I :	Isolation	:	atomicity for others
C :	Consistency	:	correctness of program or admissibility of the DB-state
D:	Durability	:	store data securely and permanently, survive transactions

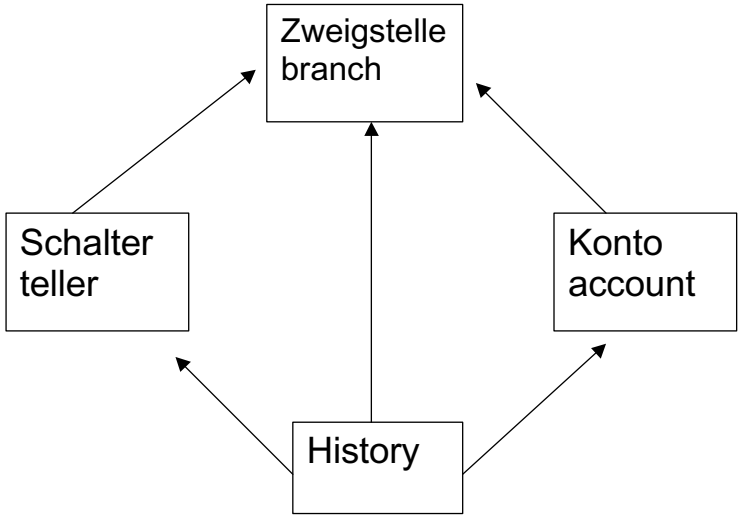
Distinguish: transaction – code
transaction – execution

Question: What is an action?

Question: ACID and single user mode?

Debit-Credit transaction between Accounts

TPC – Benchmark : TPC-A (1989)



Example for key of history?

⇒ long composite key

typical queries:

teller activity?

balance statement?

complete activity Do 16 – 20 h?

acitvity of branch?

⇒ Data – Mining, Data Warehousing

⇒ Integrity

Debit-Credit TAP:

read terminal input (acct#, branch#, teller#, delta);
begintrans
update ACCOUNT **set** balance = balance +:delta
  where acct_no = :acct # ;
select balance **into** :abalance **from** ACCOUNT
  where acct_no = :acct #;
update TELLER **set** balance = balance +:delta
  where teller_no = : teller # ;
update BRANCH **set** balance = balance +:delta
  where branch_no = :branch # ;
insert into HISTORY (Tid, Bid, Aid, delta, time)
  values (:teller#, :branch#, :acct#, :delta, CURRENT);
endtrans ;
write message to terminal (abalance, ...)

Questions about Debit-Credit Transaction:

- What is an „Action"?
- Integrity constrains and cecking thereof?
- Why atomic?
- Where are indexes needed?
- Keys for relations and performance?
- Keys for history, reason?
- Optimization across several SQL-statements ?
- Compensation for them?
- Which integrity constrains cannot be checked by DBS?
- Remedy for that?
- Partitioning of DB, criteria?
- Modification of credit line?
- Modification for ATM-Payments? (ATM = automtic teller machine)
- Consequences for performance?
- Collateral execution: Advantages and disadvantages?

3. Parallel Execution of Transactions

- pseudo parallel: i.e. interleaving of several transactions, action?
- truely parallel: on multiprocessor
  see MIDAS-Project, SFB 342

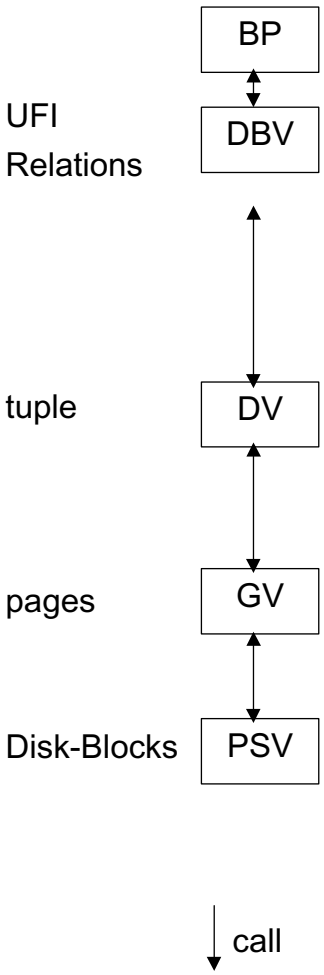
Problems: Synchronization: locks optimistic

deadlocks: discovery,
breaking of deadlocks

Cross connections to logging, recovery
e.g. **abort** to break deadlocks

Granularity of parallelization
e.g. - of sequential transactions (Inter-Trans-Parallelism)
-single SQL-statements
-single rel. operations
-parallelization of operator

4. General Achitecture of a DBS



C-program with sequence of embedded SQL – statements or

DB – Manager
DDL with catalogue management
DML with compiler, optimizer
interpreter

file – management: files of records, B*-Trees, temp. files, utilities, Sorting-Alg., ...

Area (Realm): pages, DB-Buffer, DB-Cache/Safe

Peripheral-Storage Management
Operating Syst., e.g. UNIX-Kernel

↓ call ↑ reply

Implenentation – Variants:

- set of procedures, modules bound to applicaion
- process: ↕ Pipe on the same computer
- Subsystem on other ↕ computer: communication protocol via LAN; GAN

Special problem with DBS:

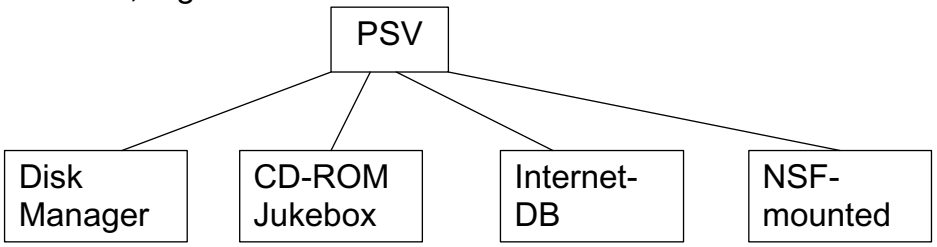
exchange of large datasets or common acces from several transactions

⇒ Simulation of a common virtual store

⇒ common virtual cache with special coherency – control

(see MIDAS and Diss. Listl)

let  be a functional unit, which offers certain services, may consist itself of several processes, e.g.



to increase parallelism and performance
potential for parallelization on many levels, e.g.

- 1 manager per area
- 1 manager per B-Tree
- several managers per B-Tree
  - e.g. Bayer, Schkolnick, 1976

5. User Requirements

Bequem: high level of abstraction,
i.e. declarative language,
natural,
set oriented,
objectoriented

Effizent: fast algorithms,
little storage,
optimization
indexes

Sicher:

against own mistake:	C,A
against mistakes of others:	I, A
against HW failures:	D, A
against SW failures:	A,C,I,D

Transactionoriented: encapsulation of complex process in 1 TAC,
e.g. booking a flight,
retrieval of literature,
ordering and checking out a book

$\Rightarrow$ BEST - Principles