Trends in DBS

1. Layer Model of DBS Architecture

Theo Härder
www.haerder.de

Towards a relational DBS architecture

Evolution of the layer model

Optimization issues

Provisions for ACID properties

Architectural extensions for DBS

Towards information systems

Main reference:
Theo Härder:
DBMS Architecture – The Layer Model and its Evolution,
in: Datenbank-Spektrum, dpunkt-Verlag, Heft 13, May 2005, pp. 45-57.

Trends in DBS

The "Religious War" – Which Abstraction Level Wins?

- Network model
  - the more complex the data structure, the better
  - but: very simple operations

⇨ **use of pointers, navigation, record orientation**

- Relational model
  - "data structure of Spartan simplicity" (E. F. Codd)
  - operations with closure property
  - each embellishment needs additional operations
  - therefore: simplicity is the secret for data independence

⇨ **Use of values, declarative queries, set orientation**

Relational Data Model – the Paradigm of the 70's

- *Forbes* called the relational model one of the most important innovations of the past 85 years, placing it squarely in the company of more widely known inventions such as the polio vaccine, automatic transmissions, fast food, disk drives, the mouse, ATMs, CDs, microprocessors, index funds, the Internet, and the World Wide Web.

Structured Programming and Data Independence

- E. W. Dijkstra's ideas
  - "Notes on Structured Programming" (EWD249, 1969)
  - "The Goto Statement Considered Harmful" (Comm. ACM, 1968)

⇨ battle cry translated into the world of data bases:

pointers are the evil!

.

- Since 1971: D. L. Parnas publishes his ideas concerning
  - "Information Hiding" at the IFIP conference
  - hierarchical structuring of software systems, …
- The concept of "information hiding" as a software design principle is widely accepted in academic circles. Many successful designs can be seen as successful applications of abstraction and information hiding. On the other hand, most industrial software developers do not apply the idea and many consider it unrealistic …

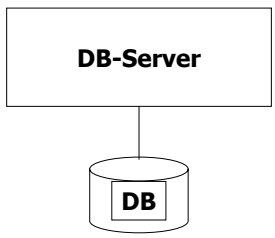
Structured Programming and Data Independence (2)

- The napkin of doom (Parnas' Dutch adventure)

  Compiler and database experts have a lunch.
  They exchange a control block format on a napkin.
  Napkin is punched, copied, and filed.
  Format changes, but napkin does not.
  Components are coupled and don't work.
  They had to do something.
  I did not know what they should have done.

30 Years Ago: Towards a Layered DBMS Architecture

- Relational model
  - declarative set-oriented access and data independence
  - how can the postulated independence at the language and implementation levels be transformed into a system?
- Monolitical approach?
  - **Q1**: Select B.Title, P.Year, A.Name
    From Books B, Authors A
    Where B.Author = A.Author
    And A.Name = „S*" And B.Topic = „DBMS"



  - interface to external memory: read and write of pages (DB is a very long bit string!)

Layered DBMS Architecture – System Evolution!

- Permanent evolution expected
  - long lifetime of a DBMS > 30 years
  - growing information needs: object types, integrity constraints, ...
  - new storage structures and access methods, ...
  - rapid changes in technology, storage media, ...
- But no revolution!
  - space and time, data streams?
  - unstructured and semi-structured documents? …
- Important challenges
  - logical and physical data independence
    - separation of applic. programs and data as strong as possible
    - isolation of the interface from all changes in the DBMS
  - encapsulation and hierarchical structuring in the system

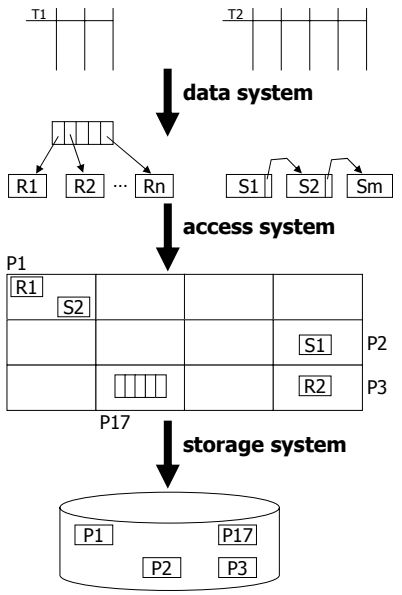
Abstraction is Geared to the Mapping Hierarchy

tables/views with set-oriented operations

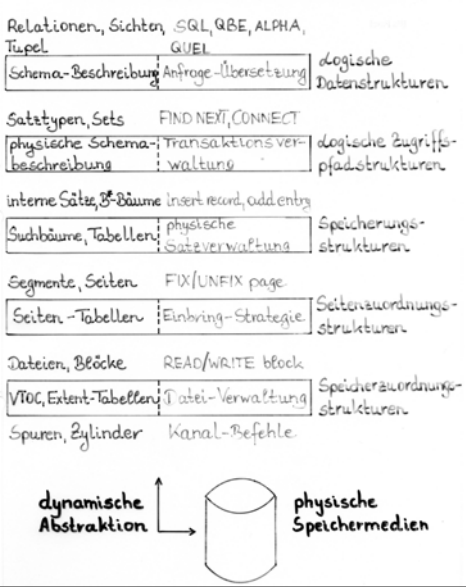
variety of record types and access paths with record-oriented operations

DB buffer in memory providing page access

external storage with files of different types



Database Systems in Business, Technology and Web, March 1985, Karlsruhe



5-Layer Model (1)

A key task of computer science is systematical abstraction. (H. Wedekind)

set-oriented DB interface
SQL, QBE ...

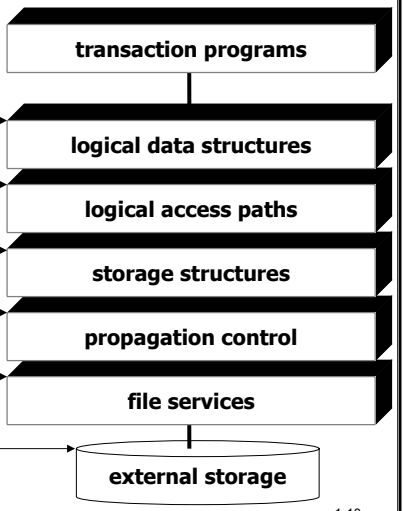
record-oriented DB interface
FIND NEXT/STORE ...

internal record interface
Store record (in B-tree), ...*

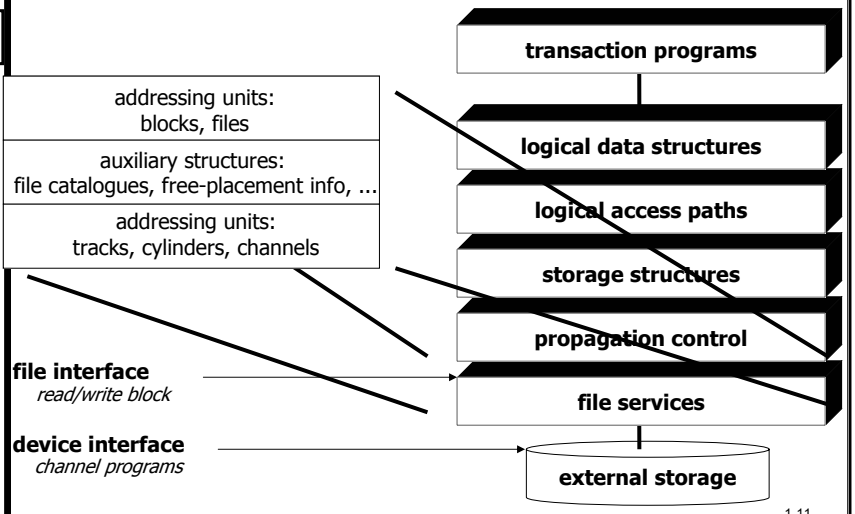
DB buffer interface
fix page / unfix page

file interface
read/write block

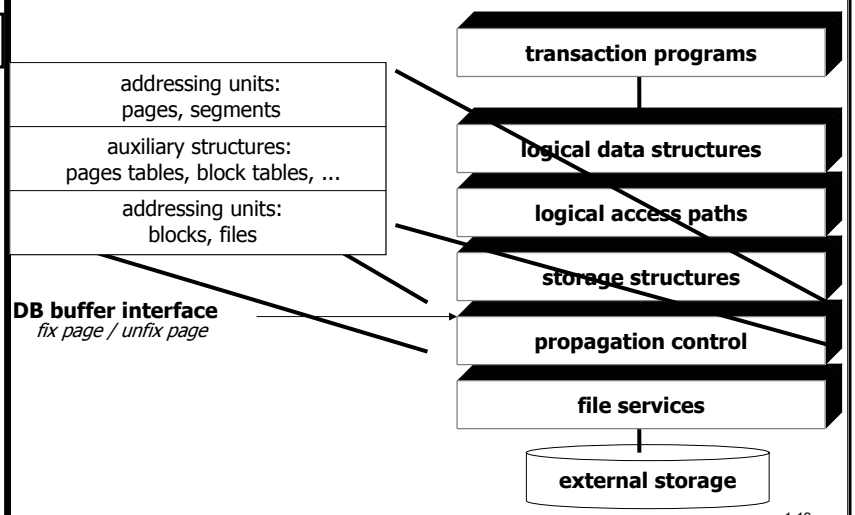
device interface
channel programs



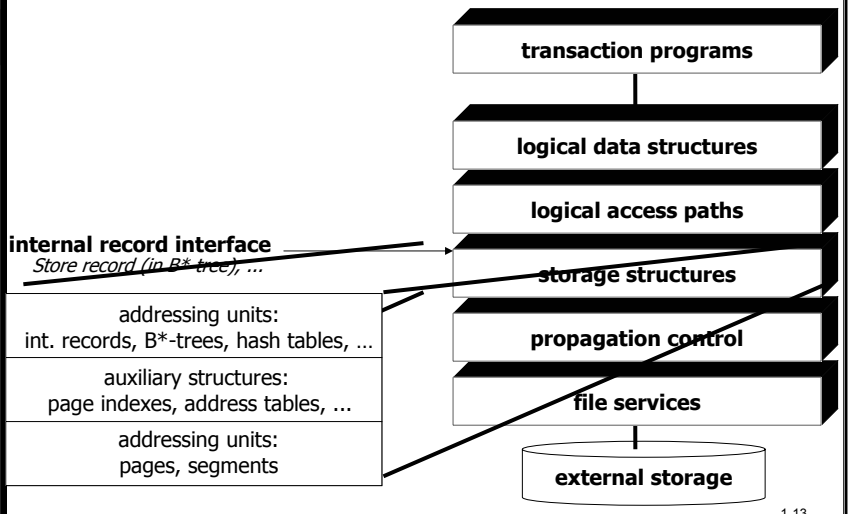
5-Layer Model (2)



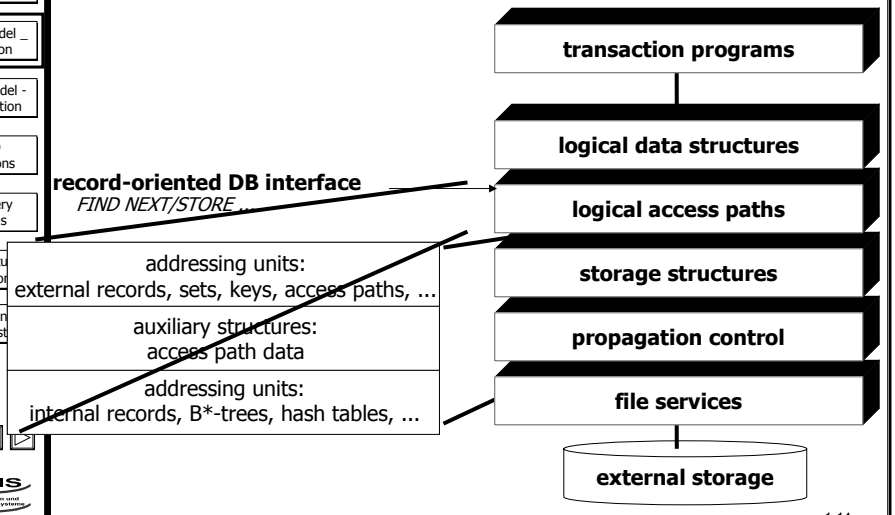
5-Layer Model (3)



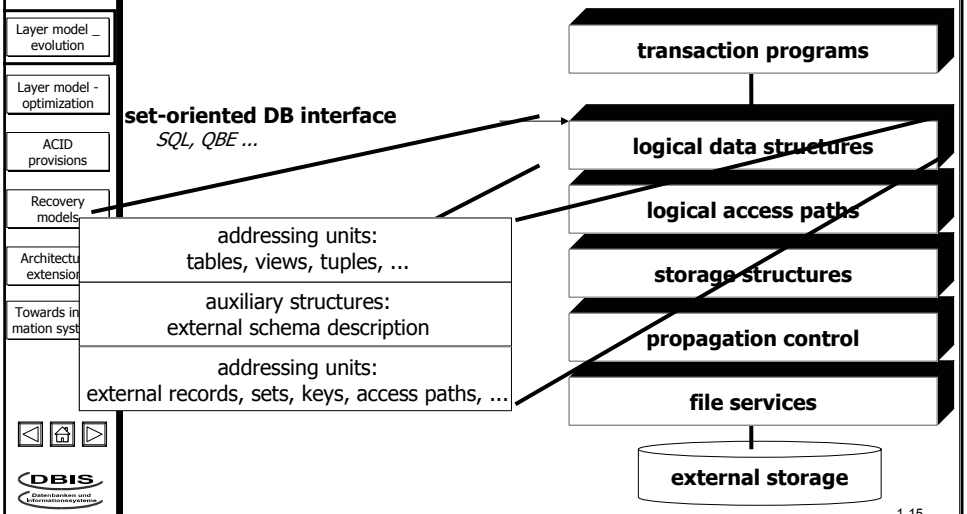
5-Layer Model (4)



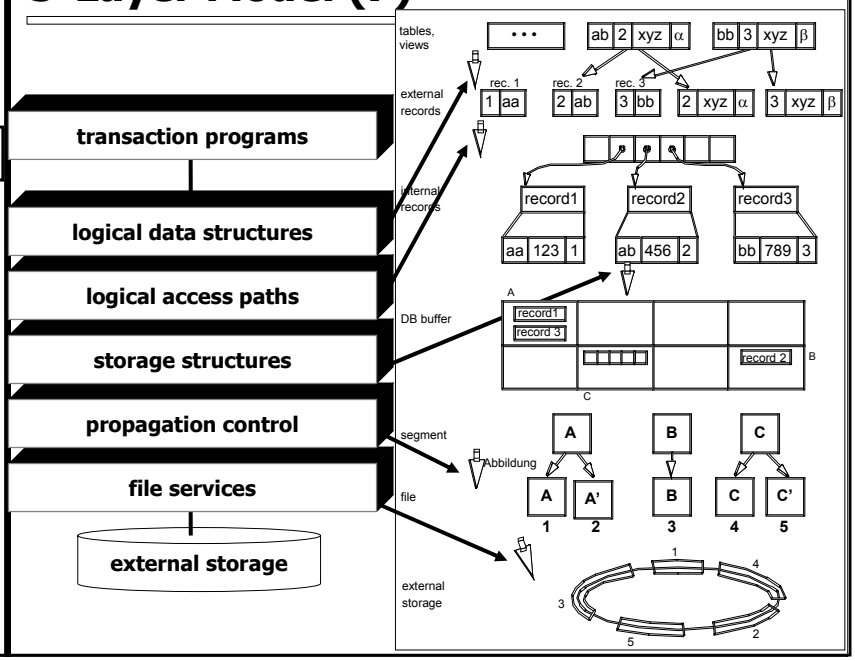
5-Layer Model (5)



5-Layer Model (6)

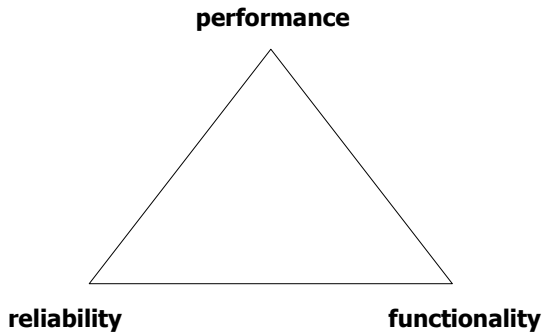


5-Layer Model (7)



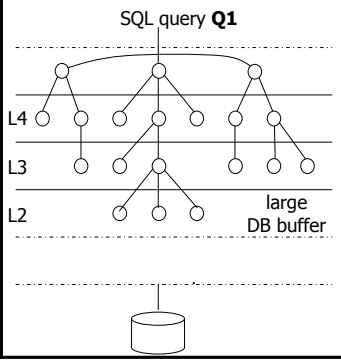
The Magic Triangle

- "Classical" key requirements for a DBMS



Layer Model – Role of n?

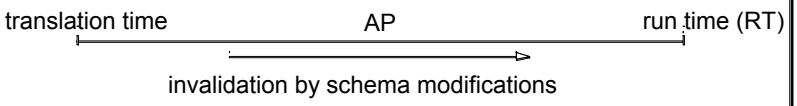
- for DBMSs, it is especially true:
  - "Performance is not everything, but without performance everything is worth nothing!"
  - what is the appropriate number of layers?
- complexity / reliability of the mapping layer vs. optimization potential of the layer
- optimized layer model at run time




```
Open Scan(I_B(Topic),Topic='DBMS',Topic>'DBMS')/*SCB1*/
Sort Access (SCB1) ASC AuthId Into T1 (AuthID, …)
Close Scan (SCB1)
Open Scan (I_A(Name), Name>='S', Name>'S')        /*SCB2*/
Sort Access (SCB2) ASC AuthID Into T2 (AuthID, …)
Close Scan (SCB2)
Open Scan (T1, BOF, EOF)                          /*SCB3*/
Open Scan (T2, BOF, EOF)                          /*SCB4*/
While Not Finished
Do
      Fetch Tuple (SCB3, Next, None)
      Fetch Tuple (SCB4, Next, None)
      . . .
End
```

DBMS – Paradigm of a Generic System

- **Pre-compilation and early binding are good ideas**
  - what – table-, attribute-descriptions, indexes, access rights, assertions, …
  - when – spectrum of points in time



accesses (at RT):
efficient
data dependent!

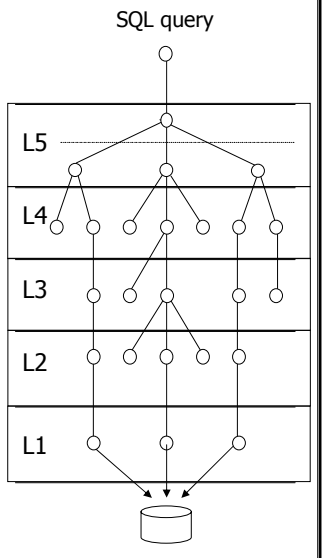
balance required!

accesses (at RT):
expensive
data independent!

- Layer-crossing optimization
  - avoidance of thrashing: L1 – L4/L5
  - reservation of buffer space (Hot Set Model)
  - optimization of lock management (Lock Contention)

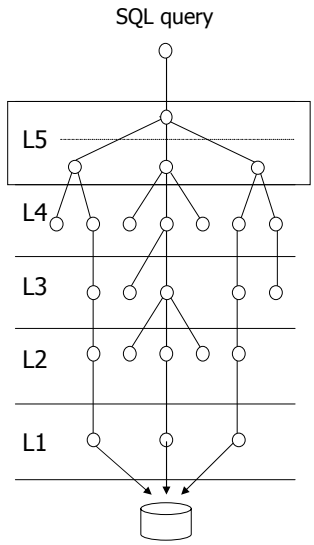
Extensions and Optimizations in the Past 20 Years

- New storage types, LOBs
- Optimization by Moore's law
- New access paths?
  - most important performance improvement by fine-grained locking
- New processing algorithms
- Compilation and optimization (SQL4)



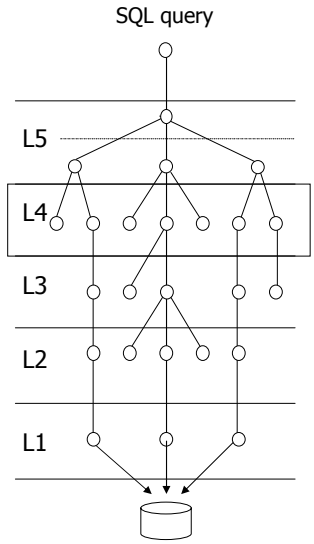
Extensions and Optimizations in L5

- SQL was not standardized in 1985
  - SQL2 and SQL3 essential
  - UDTs
  - type and table hierarchies
  - recursion, constraints, triggers, …
- Translation and optimization
  - "Optimizing the XXX optimizer"
  - cost-based optimizer
    - histograms
    - but: UDTs require their own cost model
  - dynamic QEPs
    - alternative plans dependent on resource availability, …
    - "reduce the braking distance"
  - seduction to gambling



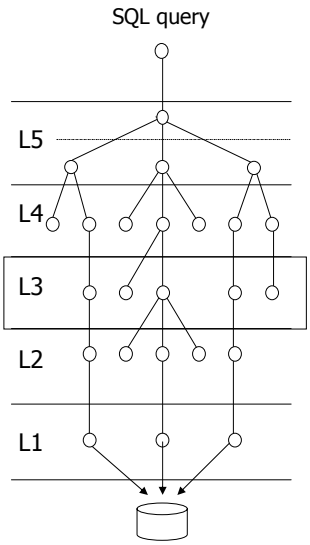
Extensions and Optimizations in L4

- New algorithms
  - hash join
  - "arbitrary" join predicates
  - reuse of results
- Adaptivity
  - Memory adaptive
  - MPL adjustment to the workload
  - shared use of scans
- New functionality
  - for OLAP, Data Warehouses, …
  - "spatial joins", …



Extensions and Optimizations in L3

- New algorithms in L4 need improved support in L3
  - index- vs. heap-organized tables
  - partitioning techniques
  - addressing techniques (Swizzling)
  - adequate and integrated access paths
    - In addition to the "**ubiquitous B-tree**"?
    - R-tree, UB-tree, …
- Low-maintenance access path
  - independent of set of keys and insertion sequence
  - without prerequisites of use such as data preparation and expert knowledge
  - no specialization to a set of operations too narrow
- "PhD-machines" have little success



Trends in DBS

Towards an architecture

Layer model _ evolution

Layer model - optimization

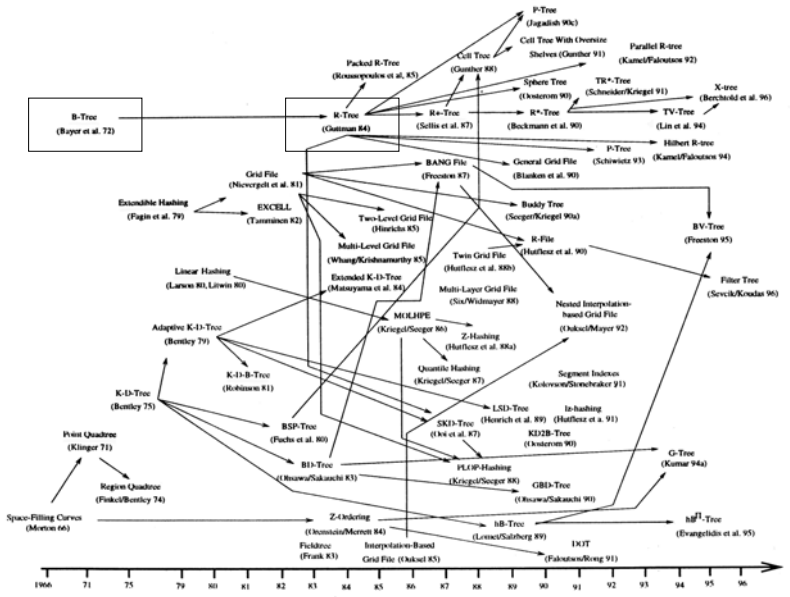
ACID provisions

Recovery models

Architectural extensions

Towards information systems

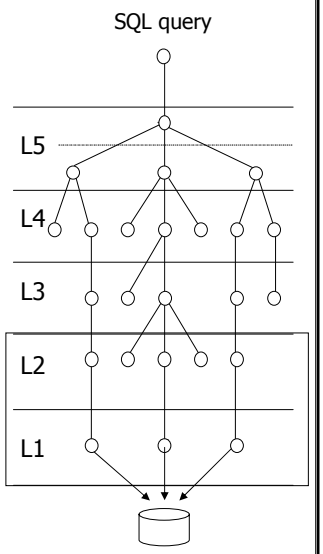
Genealogy of Access Paths



DBIS Datenbanken und Informationssysteme
© 2005 AG DBIS

Extensions and Optimizations in L2/L1

- Optimization by Moore's Law "without a struggle"
  - factor 10^4
  - from 2K up to 8 – 32K
- Buffer management
  - LRU + reference density (LRU-K)
  - prefetching and pipelining using alternate buffers
  - detection of scan-based processing
  - use of many buffers to separate workloads of different types, optimized for specific data types
- Storage and management of LOBs
- Attachment of new storage types
  - solid state disk, WORM, DVD
  - disk array



What can go wrong, will go wrong ...

May all your transactions commit and never leave you in doubt. (J. Gray)

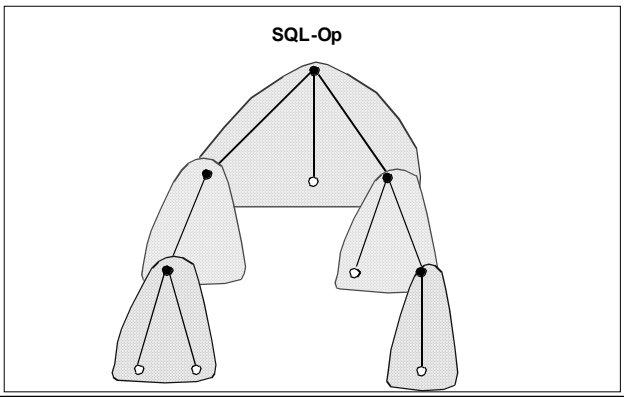
- **Goal of Development**
  Build a system used by millions of people that is always available – out less than 1 second per 100 years = 8 9's of availability! (J. Gray: **1998 Turing Lecture**)
- Availability today (optimistically):
  - for Web sites: 99%
  - for well-administrated systems: 99,99%
  - zSeries OS only: 99,99999%
  - For an entire information system – realistic goal today: 99,999%
- There are some 9' missing (to be achieved until 2010???)
- ACID transactions are introduced to guarantee far-reaching assertions concerning the quality of data

Transaction as Dynamic Control Structure

- **Atomicity**
  "all or nothing" property of any DBMS action
- **Consistency and semantic integrity**
  of the DB is endangered by erroneous data and operations of an application program
- **Isolated execution**
  means "logical single-user mode"
- **Durability**
  requires that modified data of successful transactions must survive any type of failure

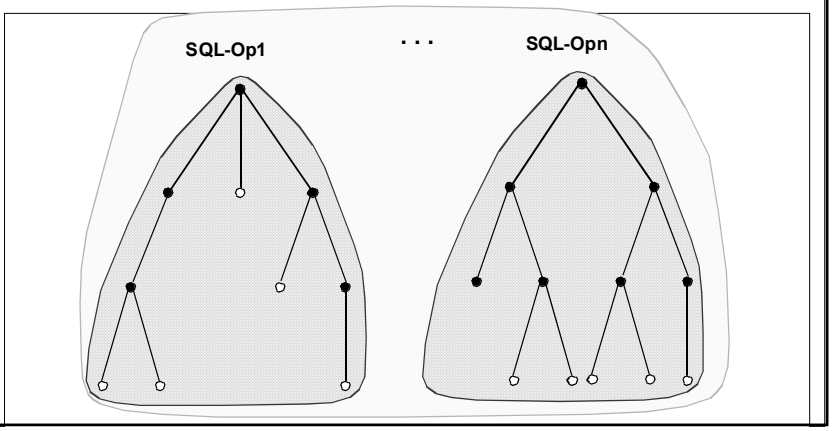
Atomicity is not a Natural Property of Computers

- Transactions are abstractions
  - building blocks: atomic actions (AAs)
  - AAs are abstractions, too
  - even if AAs would be implementable in an atomic way, hierarchy of AAs would not!



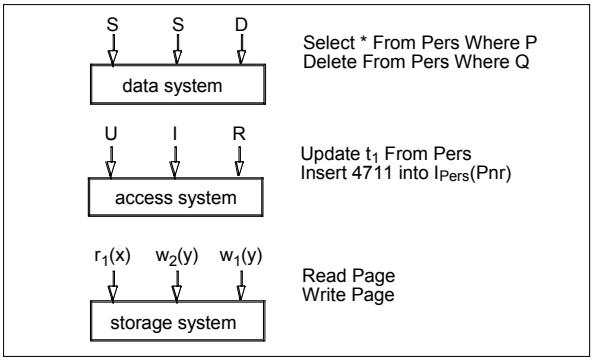
Isolation Requirements of a Transaction

- SQL guarantees "statement atomicity"
  - "Do it twice" or "Do it once"
- And, of course, transaction atomicity
- Key mechanisms:
  - **concurrency control** and **logging & recovery**



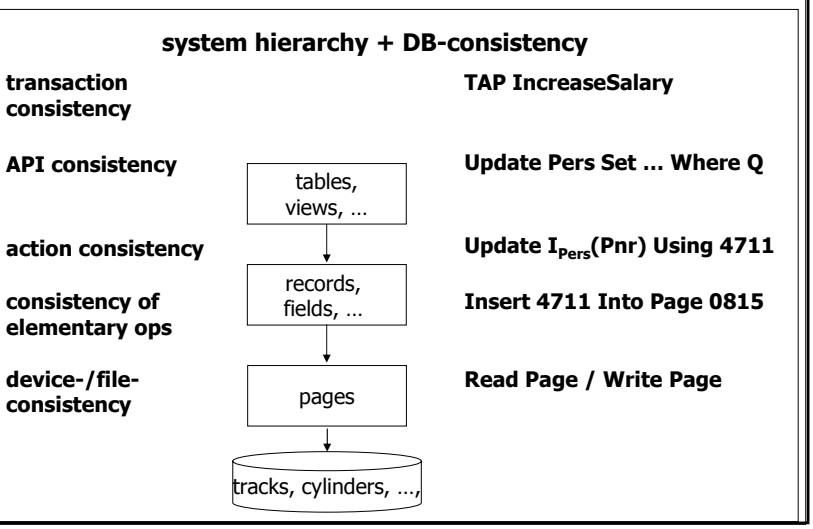
Modeling for Concurrency Control

- Correctness criterion: conflict serializability
- History writer
  - read/write model (Page Model)
  - record operations
  - logical predicates

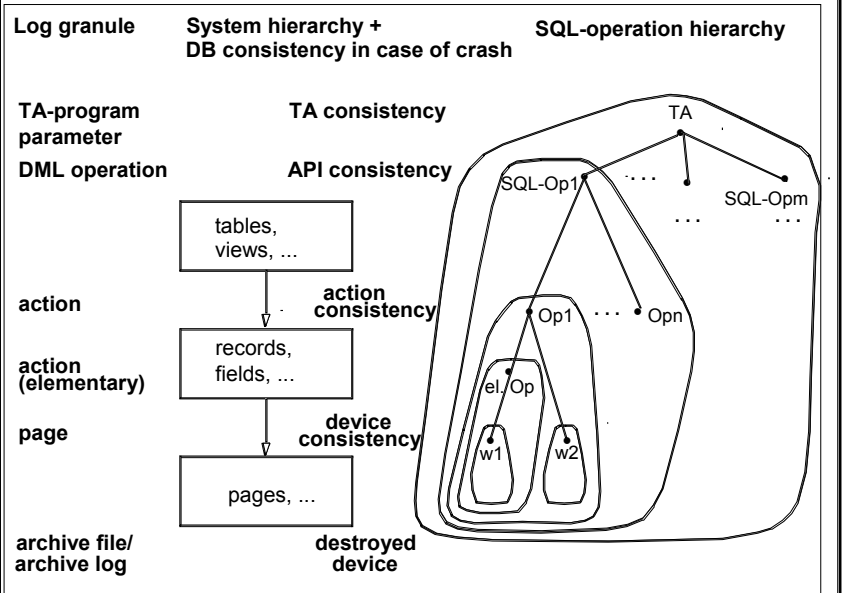


Aspects of DB-Recovery

- Consistency of a layer requires layer-specific consistency of all layers below

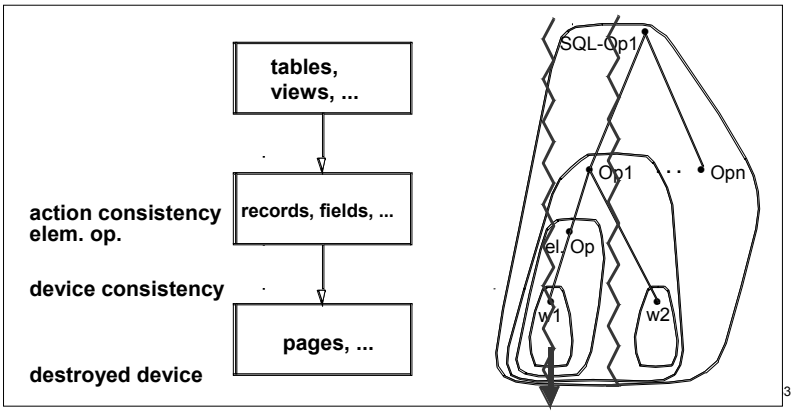


DB Consistency and Logging



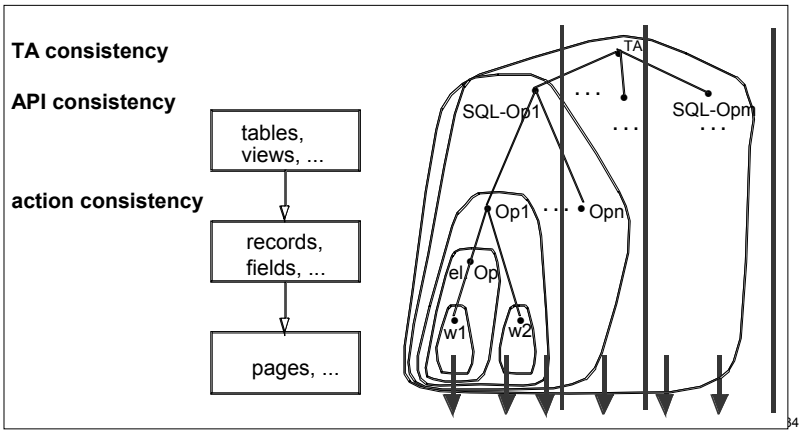
Non-Atomic Propagation Methods

- DB is "chaos consistent" after a crash
- If blocks are destroyed: device recovery
- Otherwise: device consistency as minimal condition, using
  - page logging: entire pages for Redo/Undo (**extremely expensive**!)
  - trick: physiological logging using LSNs



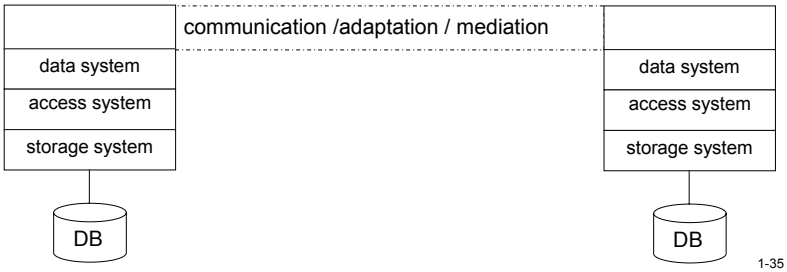
Atomic Propagation Methods

- Operations of higher layers span several pages
  - log unit must perform Undo/Redo in several pages
  - pages must be **completely or not at all** in the DB
- Role of checkpoints (limitation of Redo recovery)

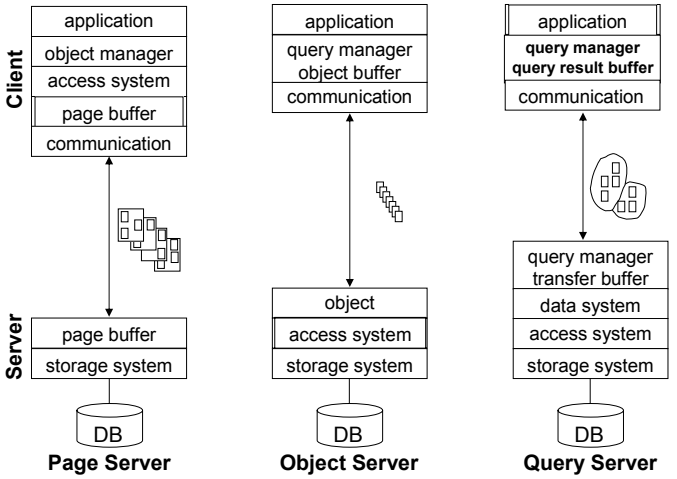


How far does this Architecture Model Reach?

- The **invariants of DBMS processing** determine the mapping steps in the DBMS architecture
  - layer model aims at the navigational or set-oriented processing of record-structured data
  - horizontal distribution of DBMS processing in
    - distributed DBMS (SN/SD)
    - federated DBMS, Multi-DBS
    - ...



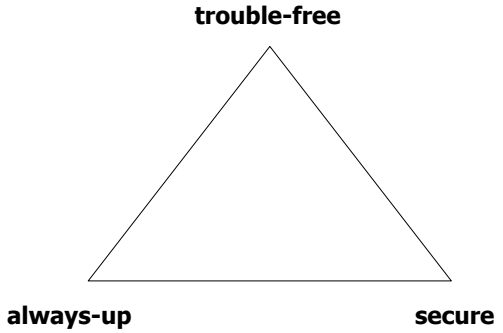
Vertical Distribution of DBMS Services



- vertical distribution of layers: DB functionality in Server and Client
- seamless connection with the application program, but many clients
- caching of predicate-complete DB subsets enabling PSJ operations!

Further Requirements

- Towards dependable adaptive information systems (DAIS)
- Additional properties needed (Jim Gray)

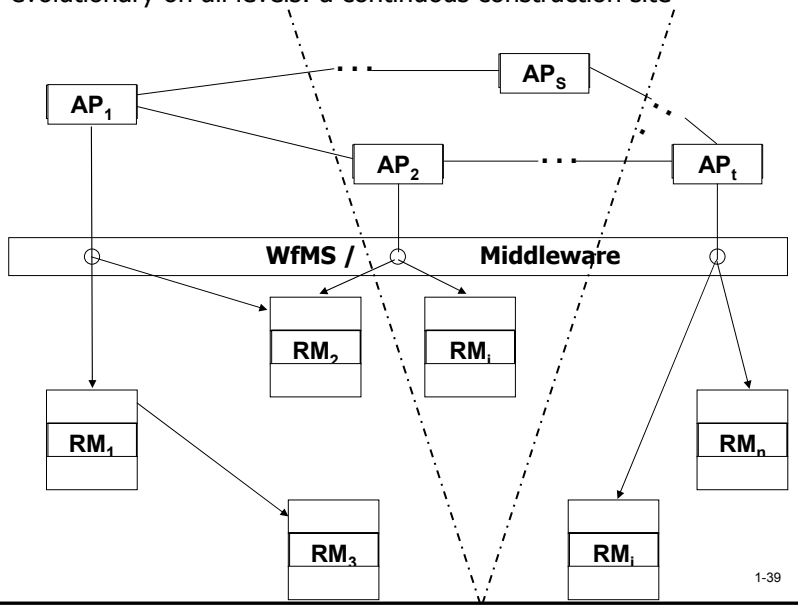


Adaptive Information Systems

- A DBMS is **only an individual (but important) component** in an information system!
- Higher degree of "self-consciousness" required
  - adaptivity concerning
    - users and workloads
    - resources, platforms and environments
    - **information (representation, content, ...)**
- **Behavioral models**
  in the DBMS and in other components needed
  - within the layers
  - crossing layers – additional channels for non-local information
- Adaptivity between the components of the information system
  - heterogeneous and spanning organizations
  - agreement protocols

Adaptive Information Systems (2)

evolutionary on all levels: a continuous construction site



DAIS – The Layer Stack

	Agreement Protocols	Business Logic		Application Layer
	Extended	Control Flow Data Flow		WfMS
	TA-Models	Distribution Caching		Middleware
	Transaction Mgmt	Query Processing		
C	Consistency Control	Compilation Optimization	L5	Data System
I	Transaction Services	Path Processing Algorithms	L4	Access System
I	Concurrency Control	Document Representation Labeling, Indexing	L3	Access System
A D	Logging / Recovery	Buffer Mgmt Propagation Control File Services	L2 L1	Storage System
	Distributed processing platform			**OS, Hardware**

Recovery-Oriented Computing

- Murphy's Law.
  What can happen, will happen (at the worst possible time and in the worst possible way)
- If a problem has no solution, it may not be a problem but a fact, not to be solved but to be coped with over time (Shimon Peres)
- **System availability A**
  - A = MTTF / (MTTF + MTTR)
  - How to achieve A ~ 1,0 ?
    - **MTTF** $\rightarrow \infty$?
    - MTTR << MTTF
  - Well-managed systems: 99.99
  - zSeries (zero downtime) platform: 99.99999
- **Contract per layer needed**
  - Separation of components
  - Simple interfaces
- If each layer of the DAIS is designed to guarantee n 9' of availability, what is the overall availability?

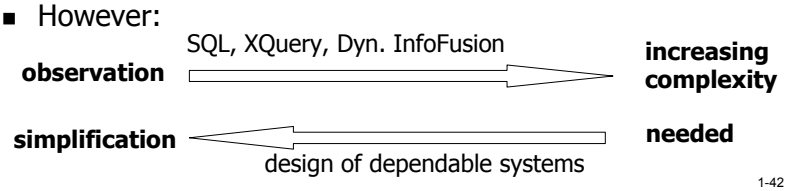
Dependable Adaptive Information Systems

- **Development goals** (J. Gray: **1998 Turing Lecture**)
  Build a system used by millions of people that is always available – out less than
  1 second per 100 years = eight 9s of availability!

  Assure that the system only services authorized users, services cannot be denied by unauthorized users, and information cannot be stolen (and prove it)!

  "trouble-free + secure + always-up"

- However:



Summary

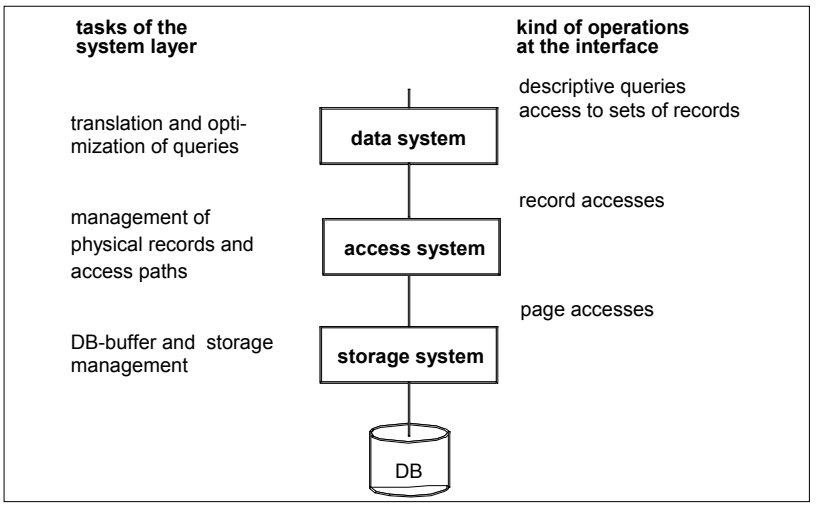
- **All "serious" systems** must manage a large number of voluminous states and keep them consistent
- Layer model has proven itself – 30 years of system evolution and optimization successfully accomplished
- **Invariants of DB management** determine the mapping steps in the DBMS architecture
- Many aspects of adaptivity in the kernel
  - self-tuning of functional properties by feedback control loops is complex and hard to achieve
  - Non-functional properties are even harder
- Adaptivity and dependability are conflicting design goals
- Nevertheless, the vision

"trouble-free + secure + always-up"

Further References

- *Chamberlin, D. D.; Astrahan, M. M.; Blasgen, M. W., Gray, J. et al.:* A History and Evaluation of System R. Commun. ACM 24(10): 632-646 (1981)
- *Härder, T.; Reuter, A.*: Principles of Transaction-Oriented Database Recovery. ACM Comput. Surv. 15(4): 287-317 (1983)
- *Härder, T.; Reuter, A.*: Concepts for Implementing a Centralized Database Management System. Proc. Int. Computing Symp. on Application Systems Development, 1983, Nürnberg, B.G. Teubner-Verlag, 28-60
- *Härder, T.; Reuter, A.*: Database Systems for Non-Standard Applications. Proc. Int. Computing Symp. on Application Systems Development, 1983, Nürnberg, B.G. Teubner-Verlag, 452-466
- *Härder, T.; Reuter, A.*: Architektur von Datenbanksystemen für Non-Standard-Anwendungen. BTW 1985: 253-286
- *Härder, T.:* DBMS Architecture – New Challenges Ahead. Datenbank-Spektrum, dpunkt-Verlag, Heft 14, Aug. 2005, pp. 38–48.
- *Härder, T., Rahm, E.*: Datenbanksysteme: Konzepte und Techniken der Implementierung, 2nd edition, Springer 2001, Kap. 1

Simplified Layer Model – Static Aspects

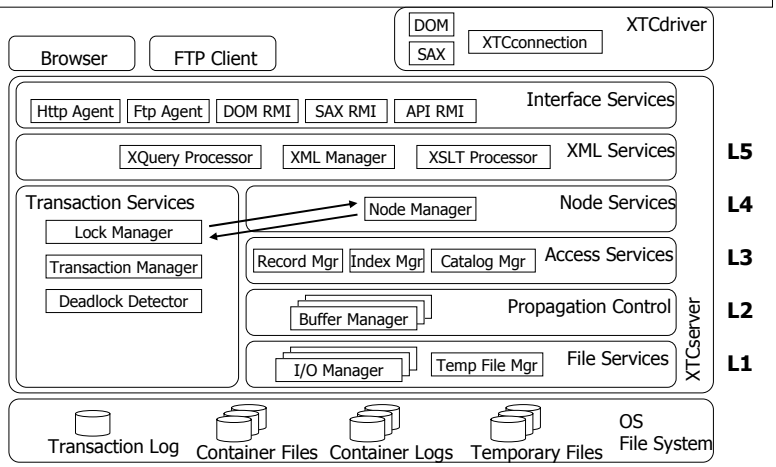


orientation of the abstraction along the data mapping

Logging und Konsistenzebene

- Wenn Log-Daten (**im Fehlerfall**) eingesetzt werden, muss die Konsistenzstufe der materialisierten DB **dies** erlauben!
- **Auswahl eines Logging-Verfahrens**
  Konsistenz der DB im Fehlerfall (Crash) → Log-Information:
  - Gerätekonsistenz → Seiten
  - Aktionskonsistenz (für Elementar-Ops) → Physiologisches Logging
  - Aktionskonsistenz (für interne Operationen) → Aktionen
  - API-Konsistenz → DML-Operationen (SQL-Ops)
  - TA-Konsistenz → TA-Programm-Aufrufe mit Params
  - **Der umgekehrte Schluss ist nicht zwingend!**
- Wenn bei Crash eine bestimmte Konsistenzstufe garantiert wird, können **Logging-Verfahren niedrigerer Konsistenzstufen** gewählt werden (nicht kosteneffektiv!)

A Native XDBMS Architecture



- XTC – architectural overview
  - reuse of layer model is possible, but needs substantial adjustments and, in particular, new functionality in higher layers
  - variations: relational, hybrid, ROX

The Ten Commandments (T. Härder & A. Reuter, 1979)

General rules

- I. Recovery based on logical logging relies on a matching operation-consistent state of the materialized DB at the time of recovery.
- II. The lock granule must be at least as large as the log granule.
- III. Crash recovery under non-atomic propagation schemes requires Redo Winners resp. Redo All (repeatable history) before Undo Losers, whereas the order of Undo and Redo is irrelevant under atomic schemes.

Rules for Undo recovery

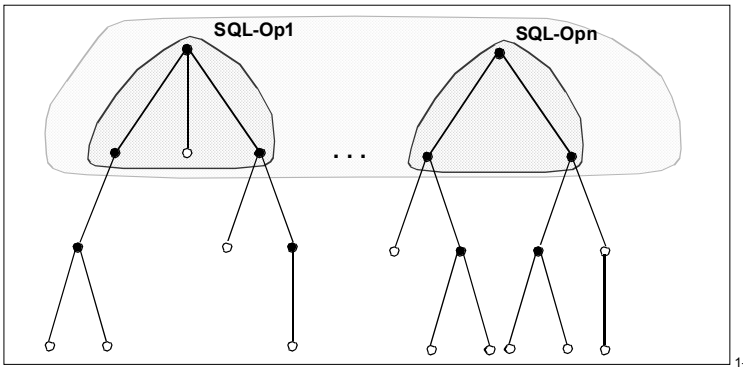
- IV. State logging requires a WAL protocol (if pages are propagated before Commit).
- V. Non-atomic propagation combined with logical logging is generally not applicable (for Redo and Undo recovery).
- VI. If the log granularity is smaller than the transfer unit of the system (block size), a system crash may cause media recovery.
- VII. .Partial rollback within a transaction potentially violates the 2PL protocol (programming discipline necessary).

Rules for Redo recovery

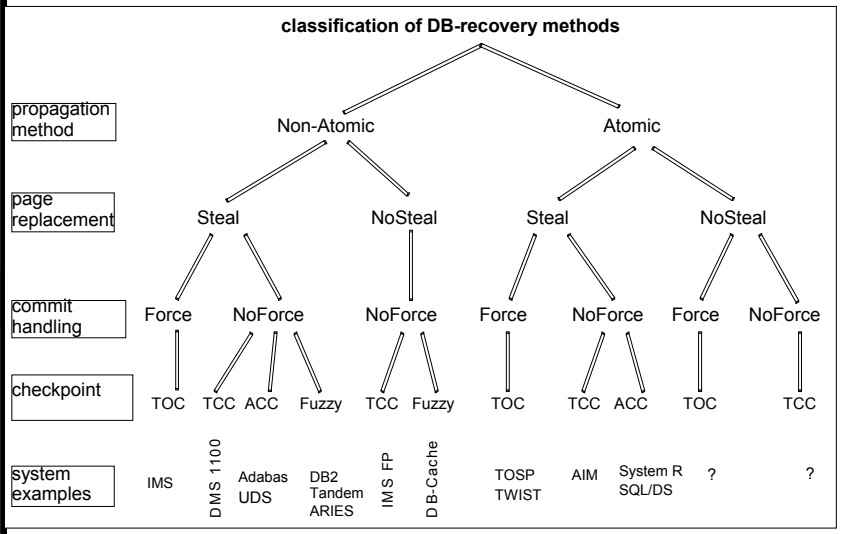
- VIII. Log information for Redo must be collected independently of measures for Undo.
- IX. Log information for Redo must be written at the latest in phase 1 of Commit.
- X. To guarantee repeatability of results of all transactions using Redo recovery based on logical logging, their DB updates must be reproduced on a transaction basis (in single-user mode) in the original Commit sequence.

Optimization – Special Kind of Nesting

- Multi-level transactions
  - early release of resources in lower layers
  - "L2L" serializability
  - compensation instead of rollback in lower layers
- ACID properties for the transaction



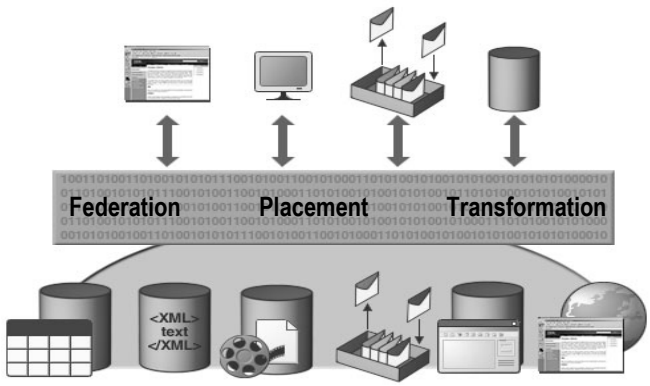
The Full Truth



- Performance optimization and chaos win against esthetic and clear structuring

DB-Middleware and Information Fusion

- Global scenario (N. Mattos)



- Distribution, heterogeneity, autonomy
- Change of paradigm:
  on-demand, scalable coupling and integration of data sources, data streams and data analysis models

Processing Platform – Virtualization of Resources