Usage Scenarios of DBMS

Rudolf Munz

SAP AG

Agenda

1. Who are we?
2. Where do we come from?
3. Where do we want to go?

SAP Today

- **SAP AG in 1998 revenues: 8.47 Bill. DEM (5.05 Bill. USD)**
  - 4th largest independent software vendor in the world
  - Market Leader in Enterprise Applications Software Licenses (36% market share amongst TOP TEN; IDC, 1999)
- **22,000+ customers in 100+ countries team with us to**
  - Extend their competitive capabilities
  - Integrate their business processes
  - Get a better return on information at a lower total cost of ownership
- **Focused on users in all enterprises regardless of size**
  - Increased customer satisfaction and strong customer loyalty
  - Heavy investment into SAP's worldwide business community
  - 21,000+ SAP employees

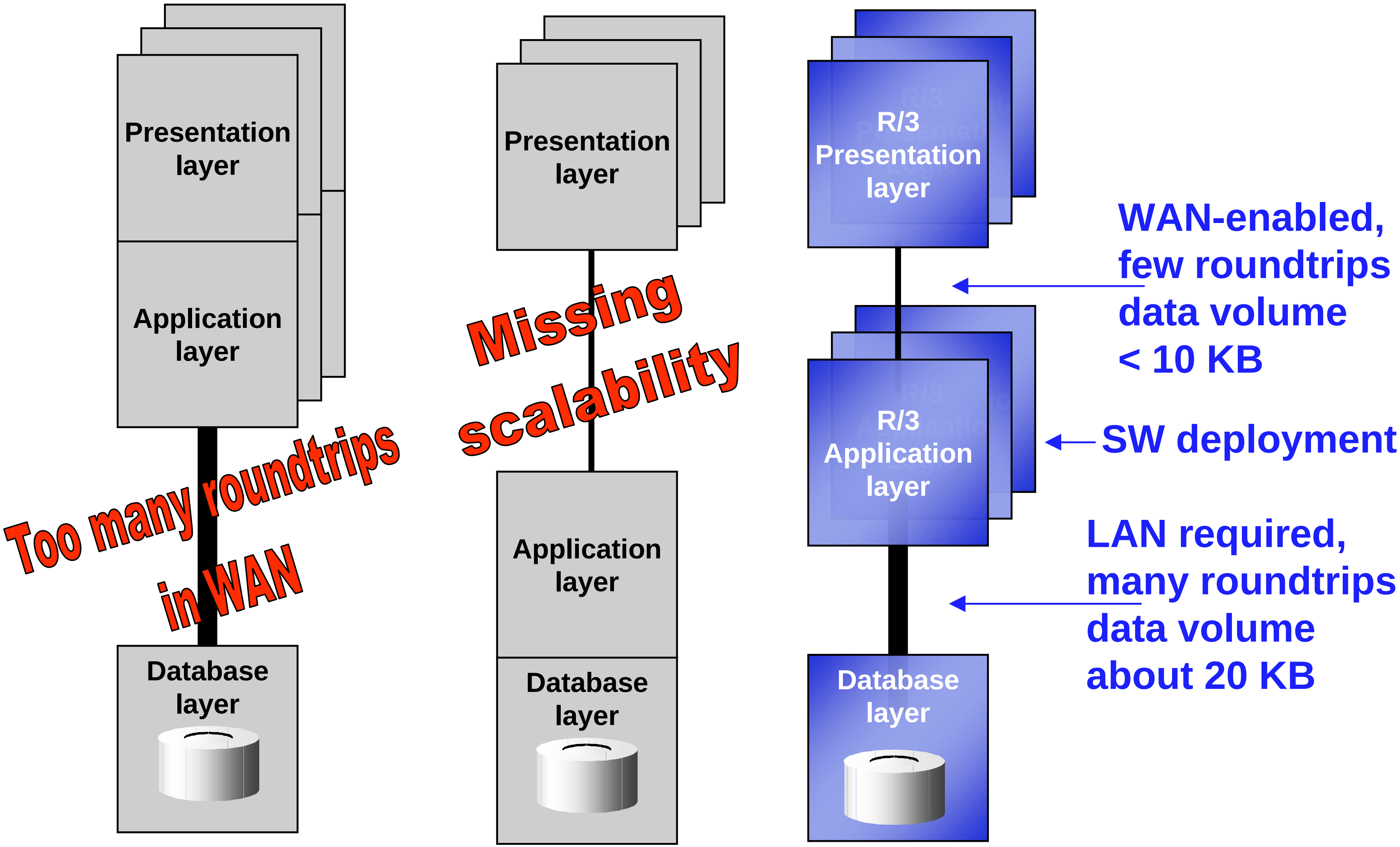
Why Do Customers Buy SAP Software?

- **Outsourcing of enterprise application development including its maintenance and enhancement**
- **SAP is integrating software**
- **Outsourcing of system technology/platform decisions**

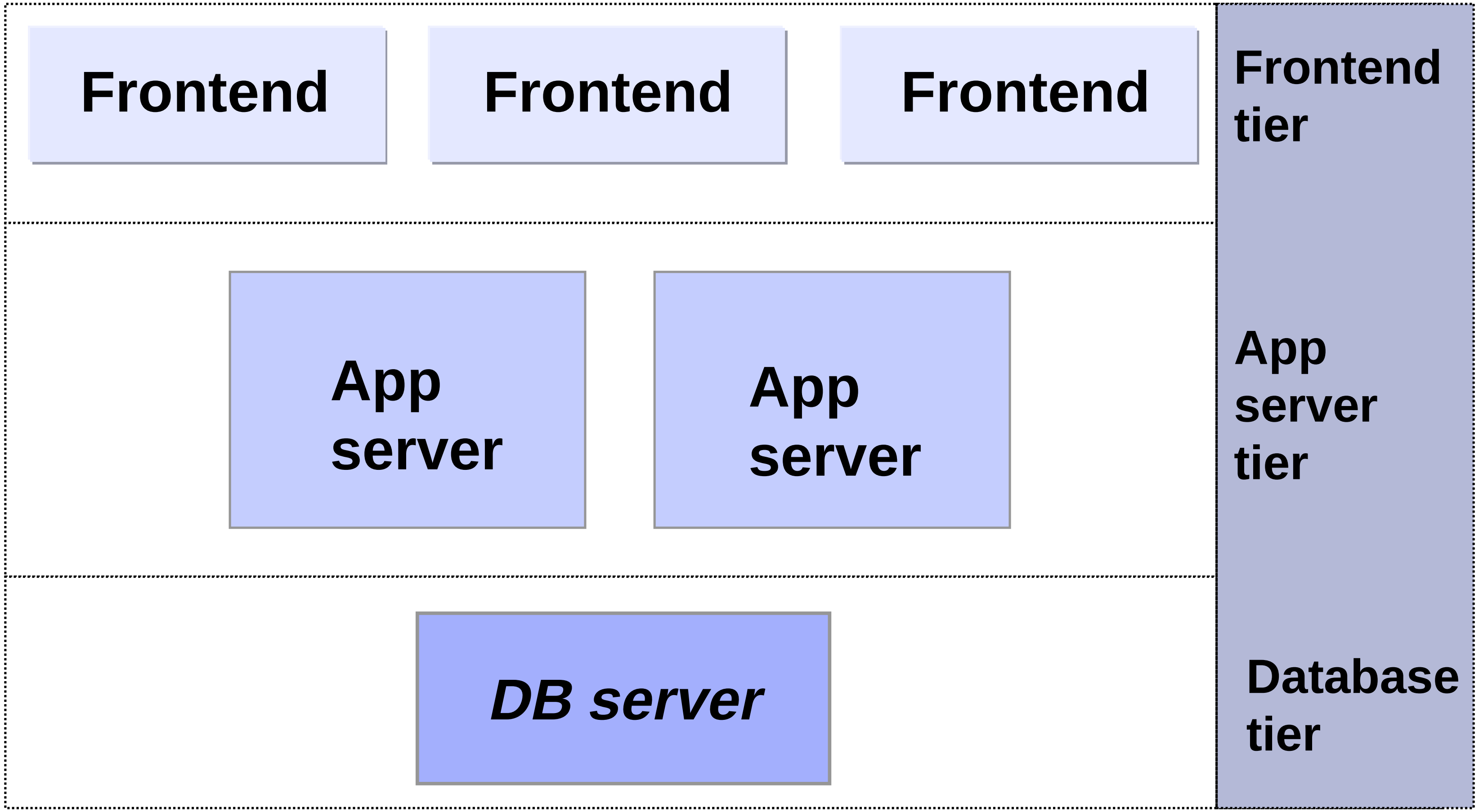
Agenda

1. Who are we?
2. **Where do we come from?**
3. Where do we want to go?

1992: SAP Introduces the 3-Tier Architecture



R/3 Ideology

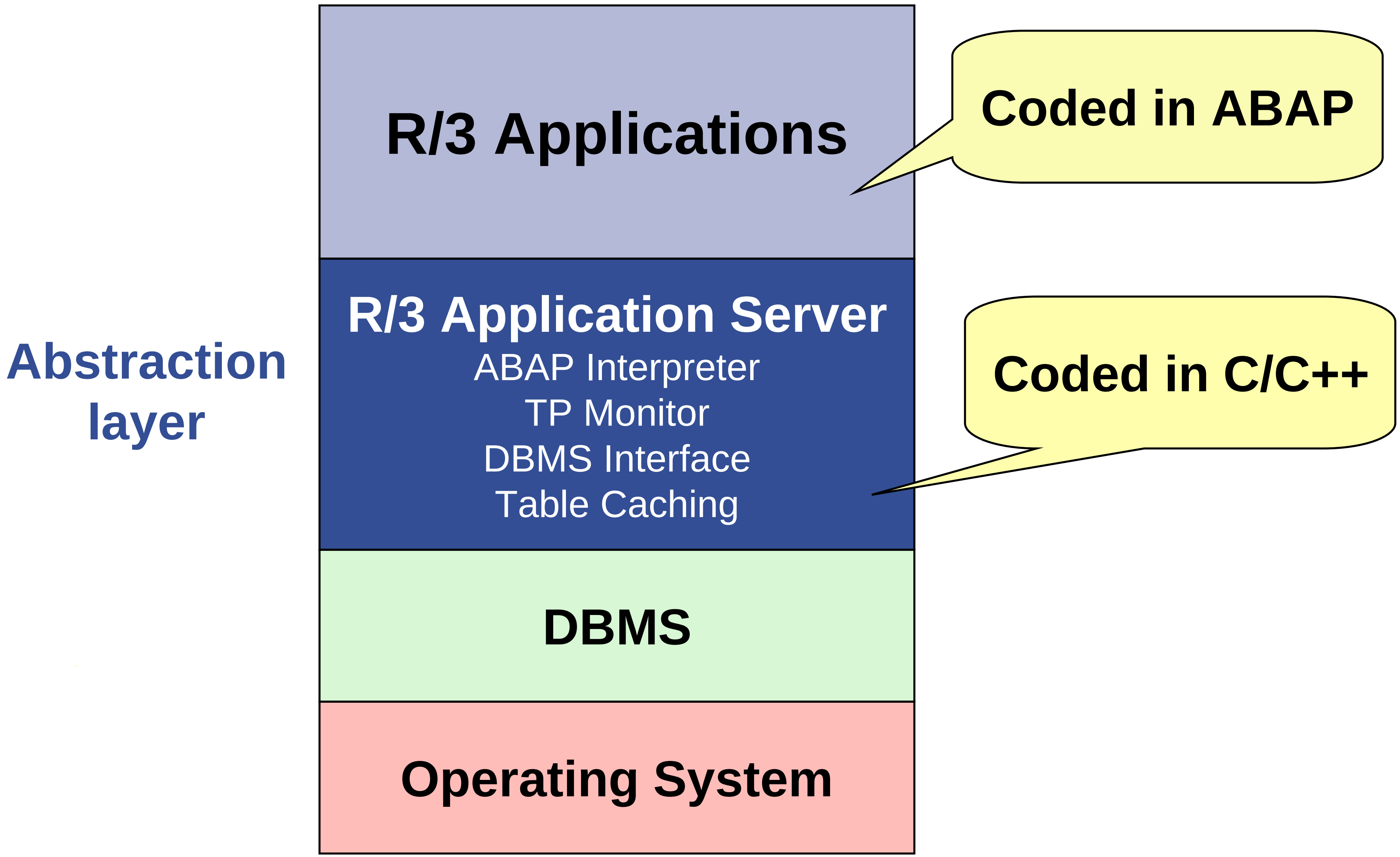


Enterprise integration via one single database (no borders between enterprise units)

R/3 Statistics (as of Rel. 4.5)

- 15 000+ tables
- 200 000+ columns
- Growth rate per major release: + 30%
- 35 000 000 lines of application coding
- 8 GB footprint on disk
- more than 20 languages supported
- 650+ software developers in SAP‘s system technology
- 2500+ software developers in SAP‘s applications

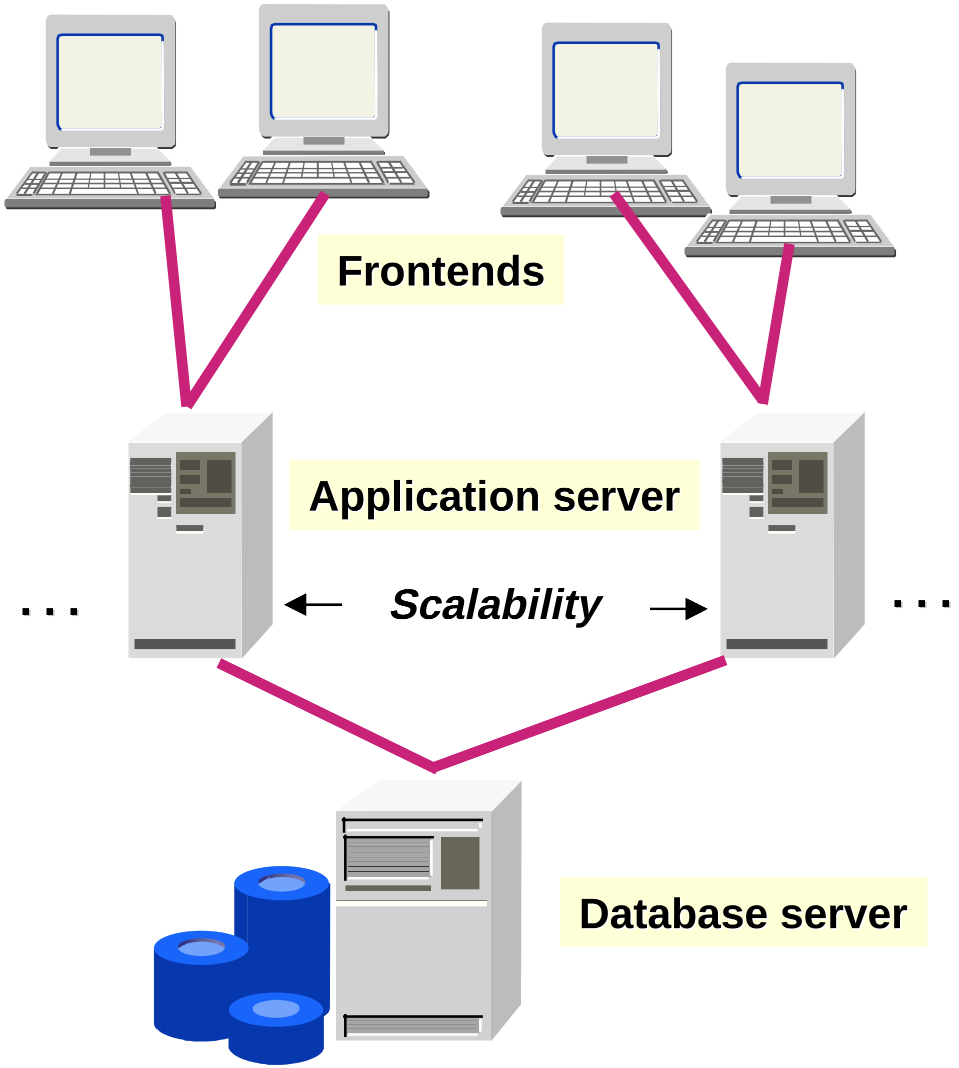
R/3 Architecture



R/3 Platforms

Hardware	UNIX Systems: Intel, IBM, Digital, SNI, HP, SUN	Intel Systems	IBM AS/400	IBM S/390
Operating systems	AIX, Reliant, Digital UNIX, Linux, HP-UX, SOLARIS	Windows NT	OS/400	OS/390
Data bases	DB2 UDB, SAP DB, INFORMIX, ORACLE	DB2 UDB, INFORMIX, ORACLE, MS SQL Server	DB2 for AS/400	DB2 for OS/390
Frontend SAPGUI	Windows 95, Windows 98, Windows NT, Java Desktops and Browsers, HTML Browsers			

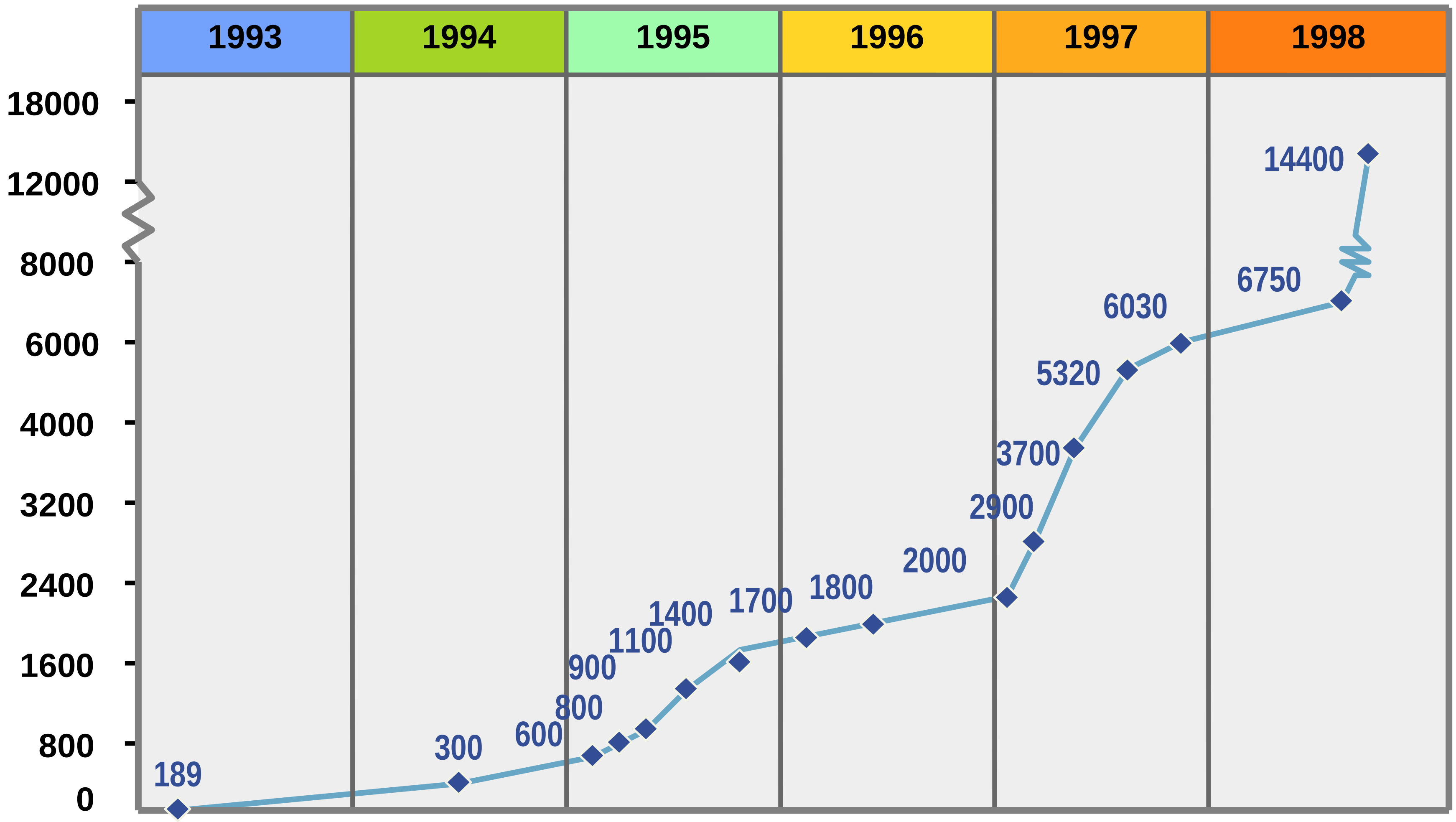
R/3 Scalability



SAP's Scalability Benchmarks

- Presentation Layer
  - More than 14,000 very active users connected to one database
- Application Layer
  - Up to 143 application servers
  - The highest number of application servers at customer sites is less than 30
- Database Layer
  - Scalability through SMP architecture of the database server
  - More than 60 CPUs
  - More than 700 GB database size

Published Results for SD Benchmarks



Highest Number of Sales & Distribution (SD) Benchmark Users

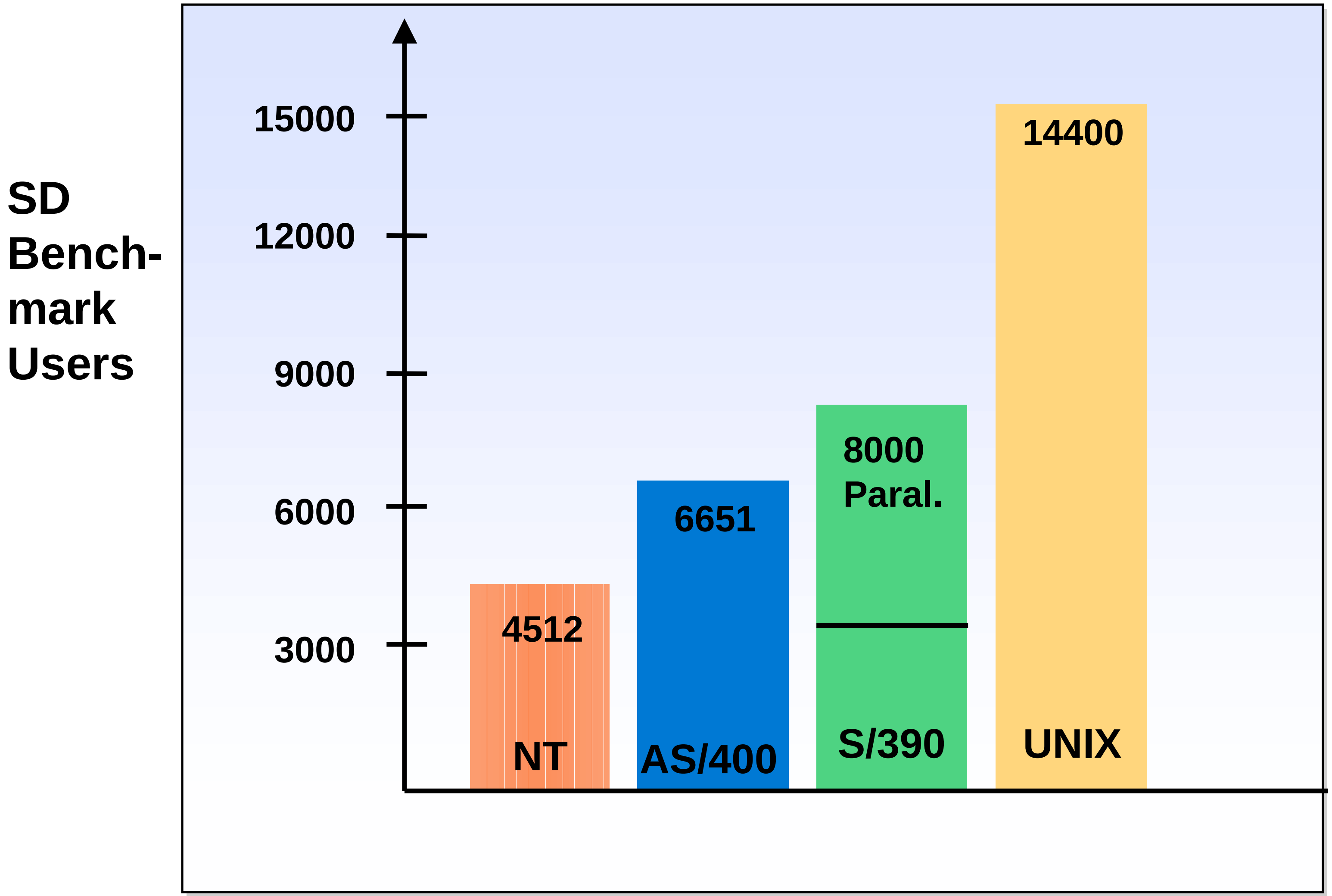
Key figures for the 14,400 SD User Benchmark

- **593 DB transactions / second**
- **65,200 DB calls / second**
- **14,900 DB changes / second**
- **167 Mbit / second network traffic to the database server**
- **1.9 MB average disk read / second**
- **17 MB average disk write / second (peak: 50 MB/sec)**
- **1.1 TB total disk space**

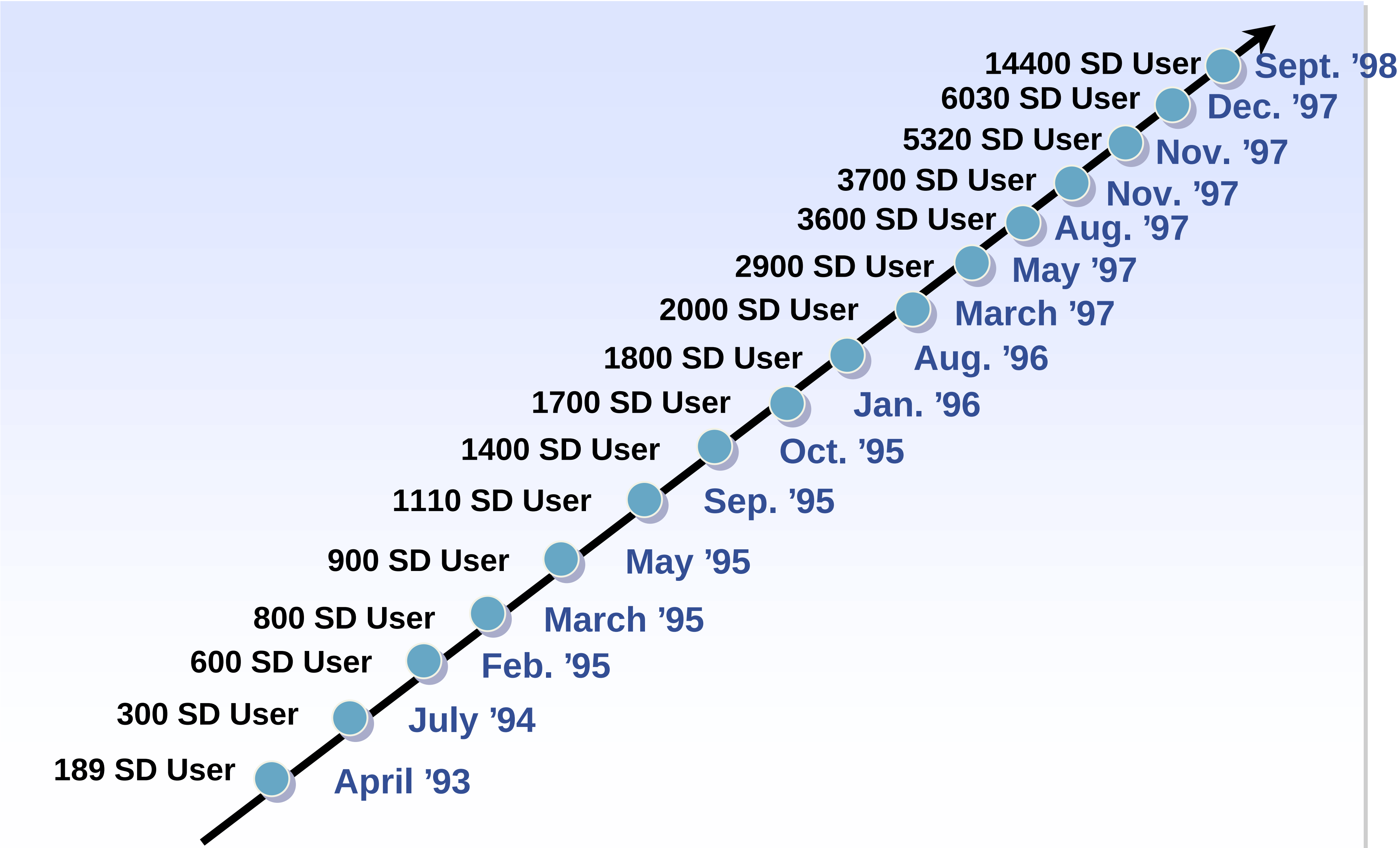
Key figures for the 14,400 SD User Benchmark

- **Database was running on a 64-processor SMP server**
- **R/3 application servers used 391 processors**
- **10.4 GB of dirty data pages written in 25 minutes**
- **1.21 GB of data pages read in 25 minutes**

R/3's Scalability on Different Platforms



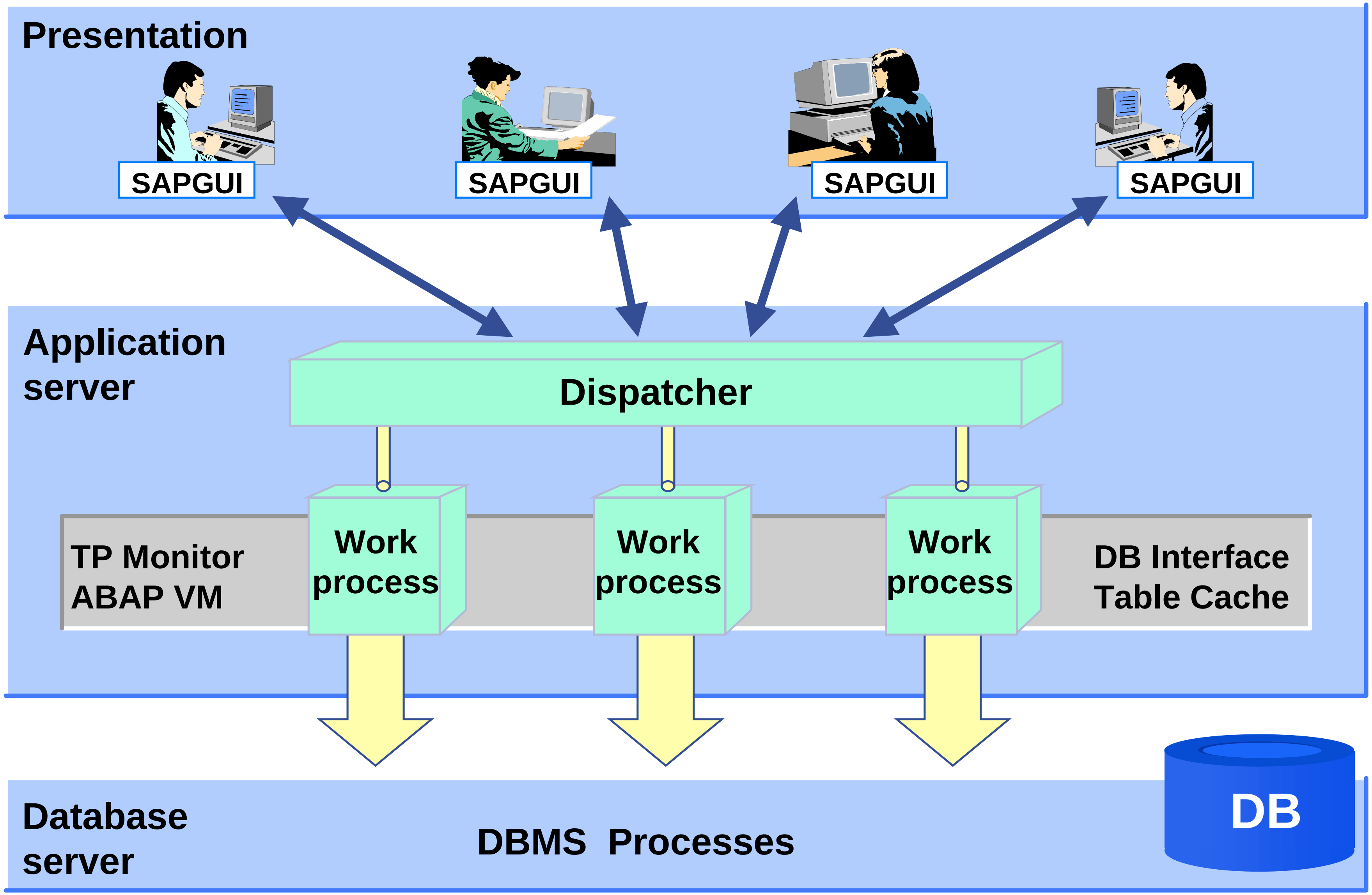
R/3 Standard SD Benchmark Figures



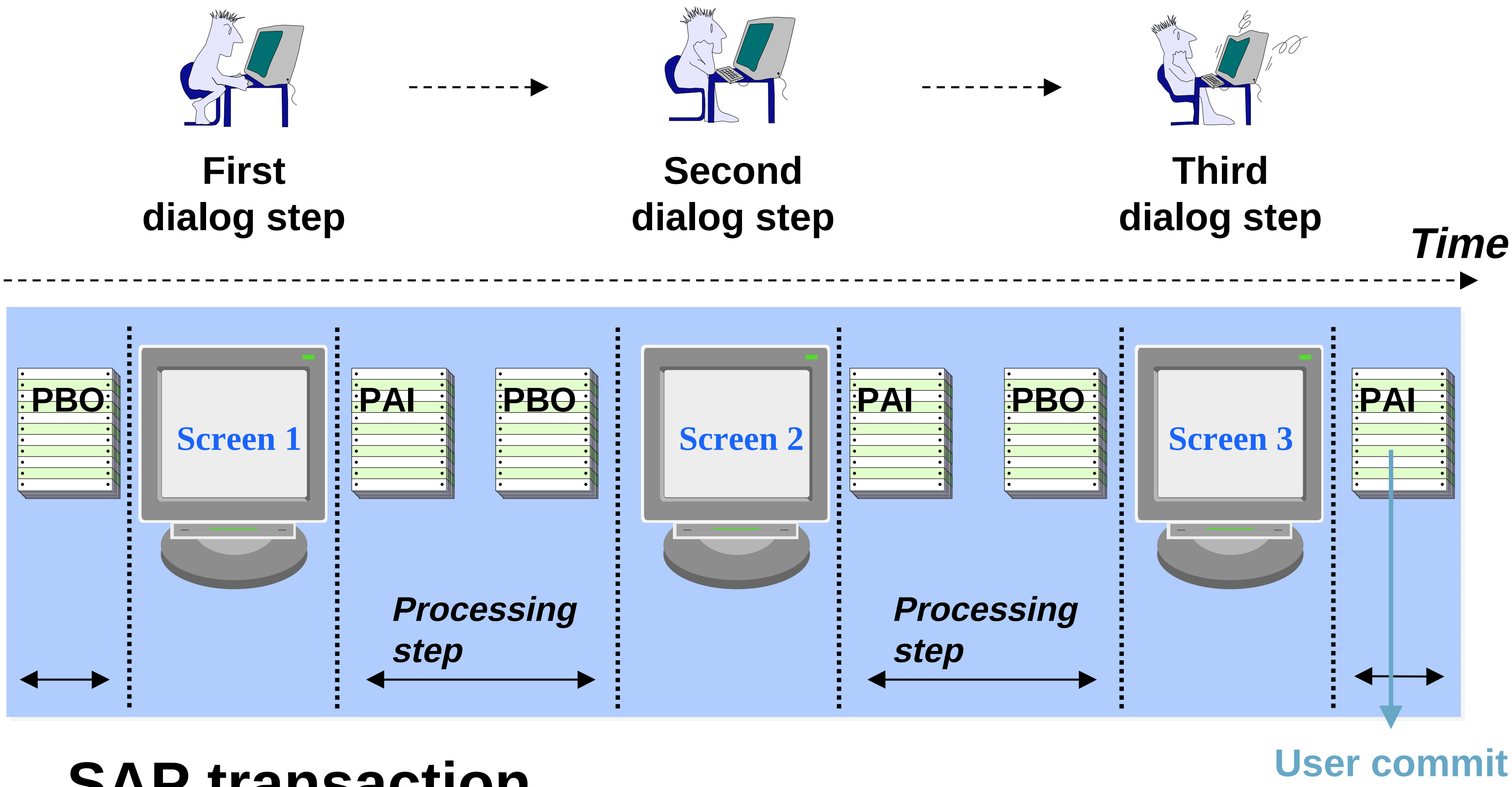
Reasons for this Progress

- More RAM in application and database servers
- Increased CPU power, especially in SMP configurations
- Table caching in the R/3 application server
- Asynchronous database update (via update task)
- Application level locking

R/3 Architecture

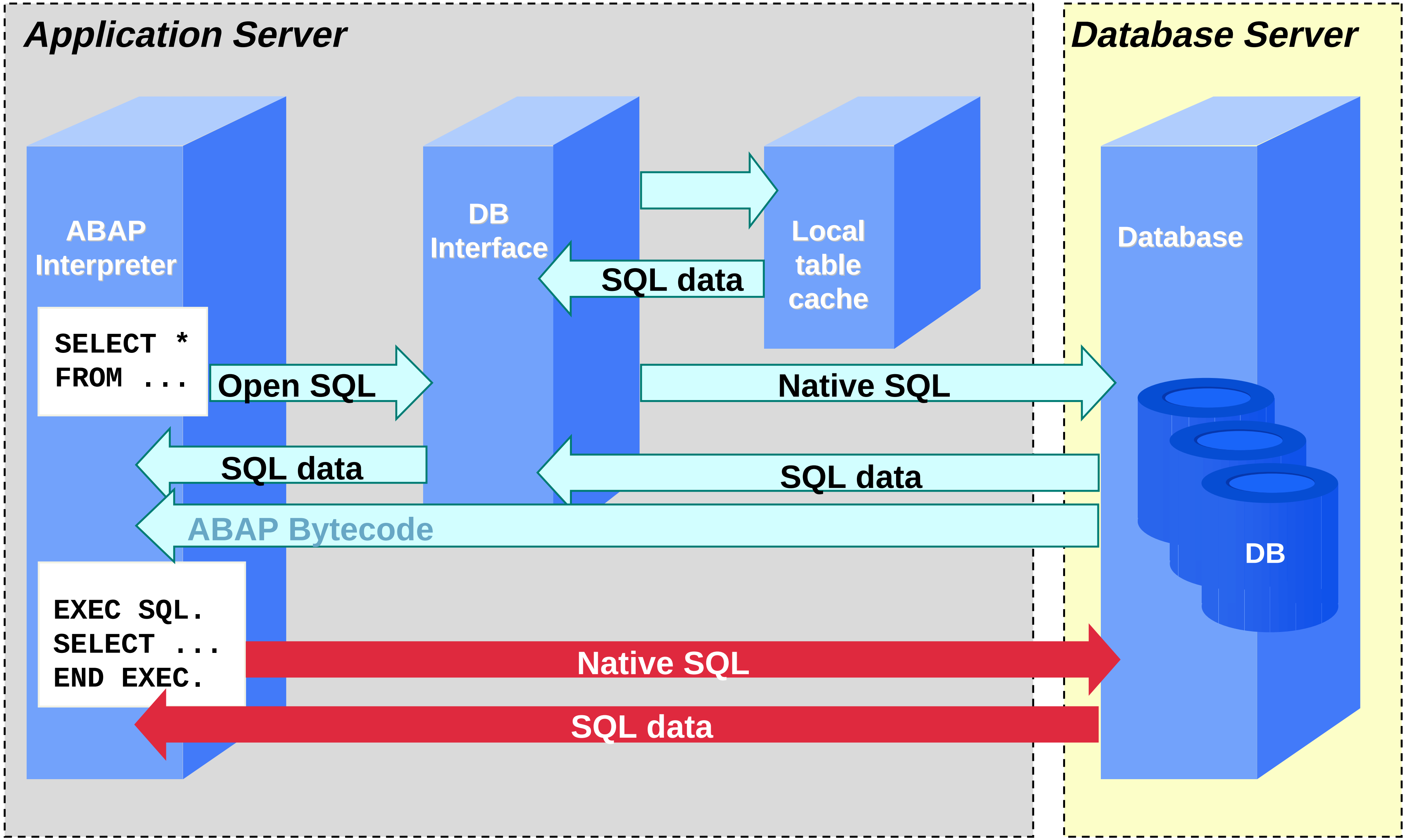


SAP Transactions and Dialog Steps

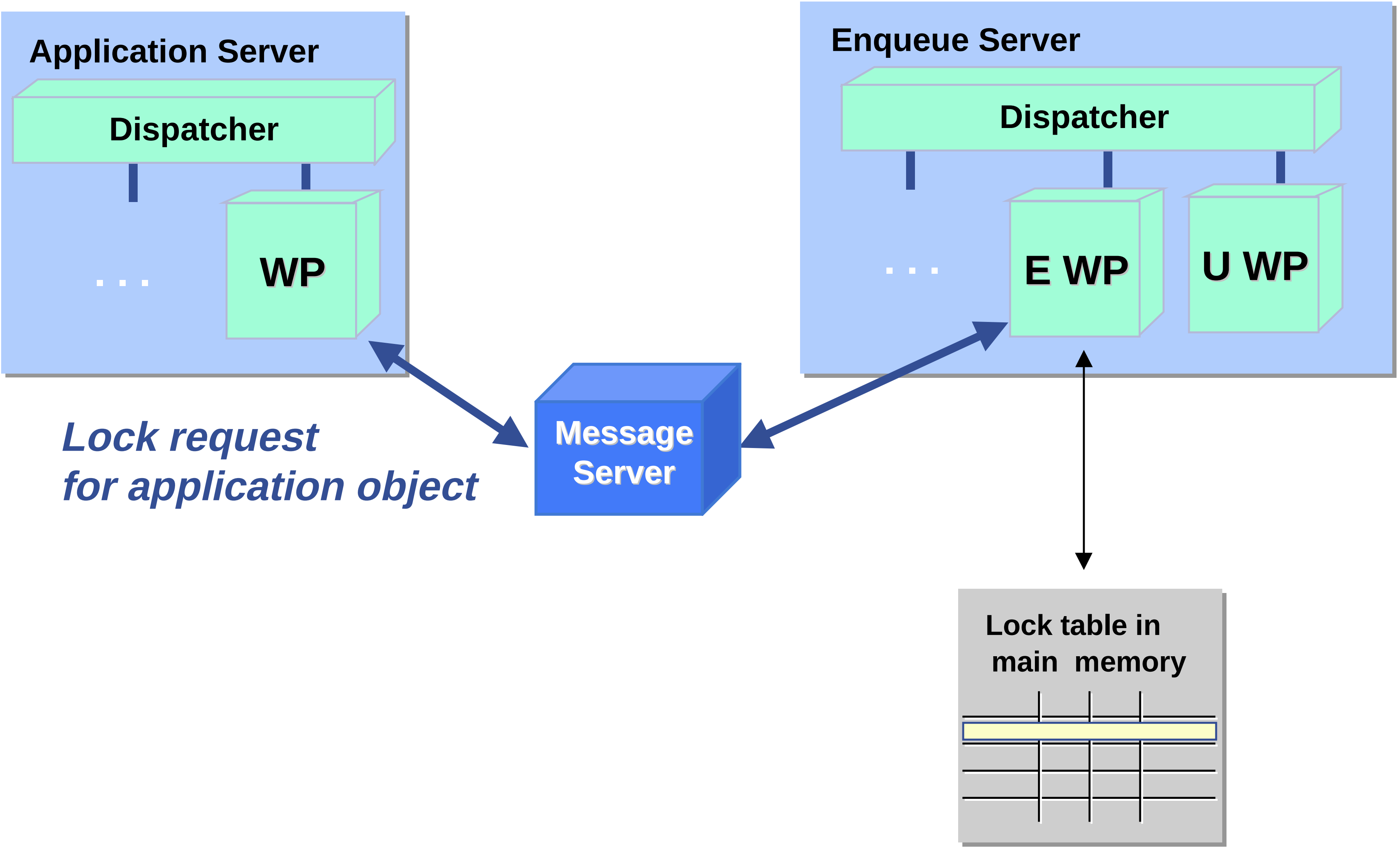


SAP transaction

Table Caching in R/3 App Server

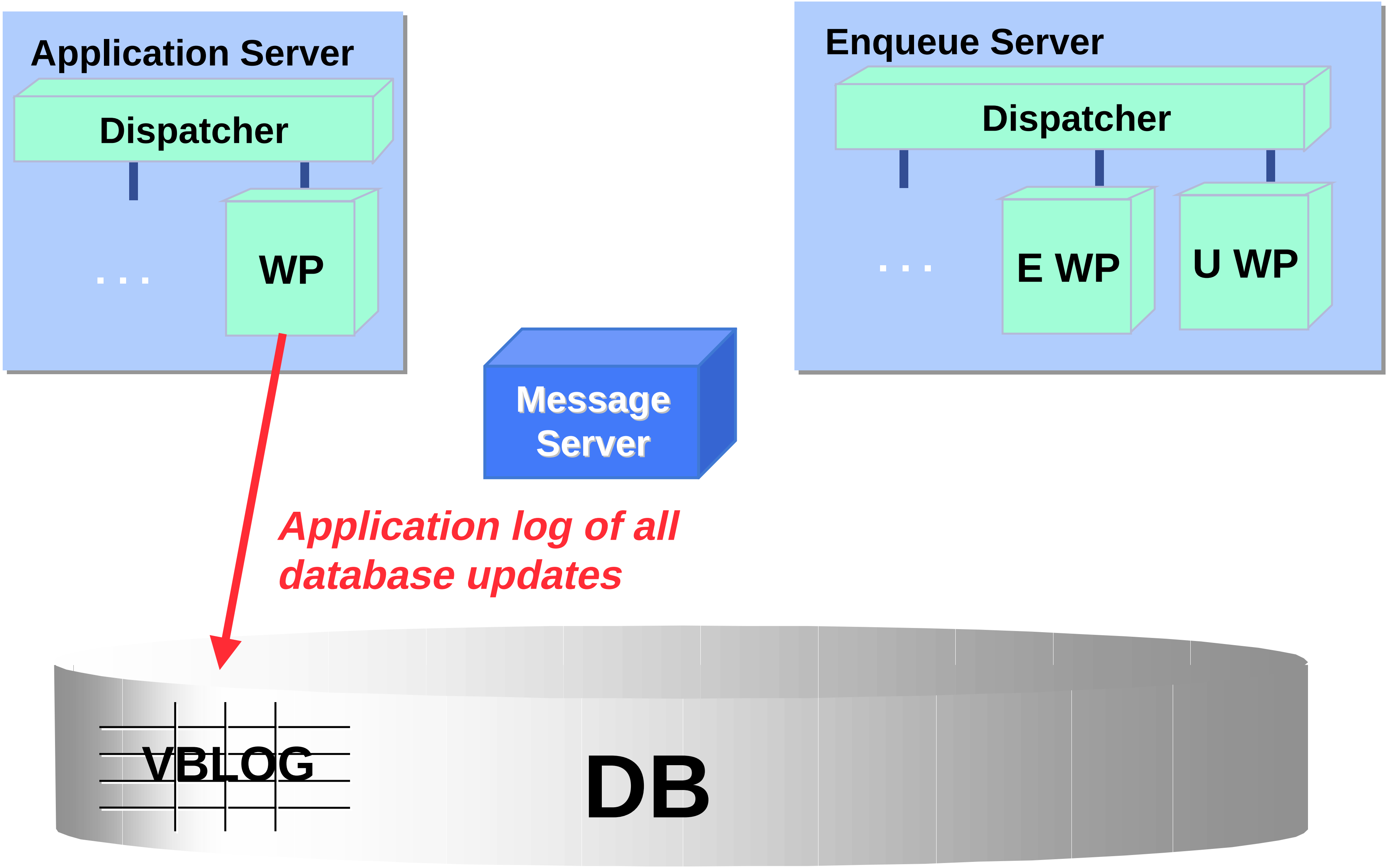


Application Level Locking

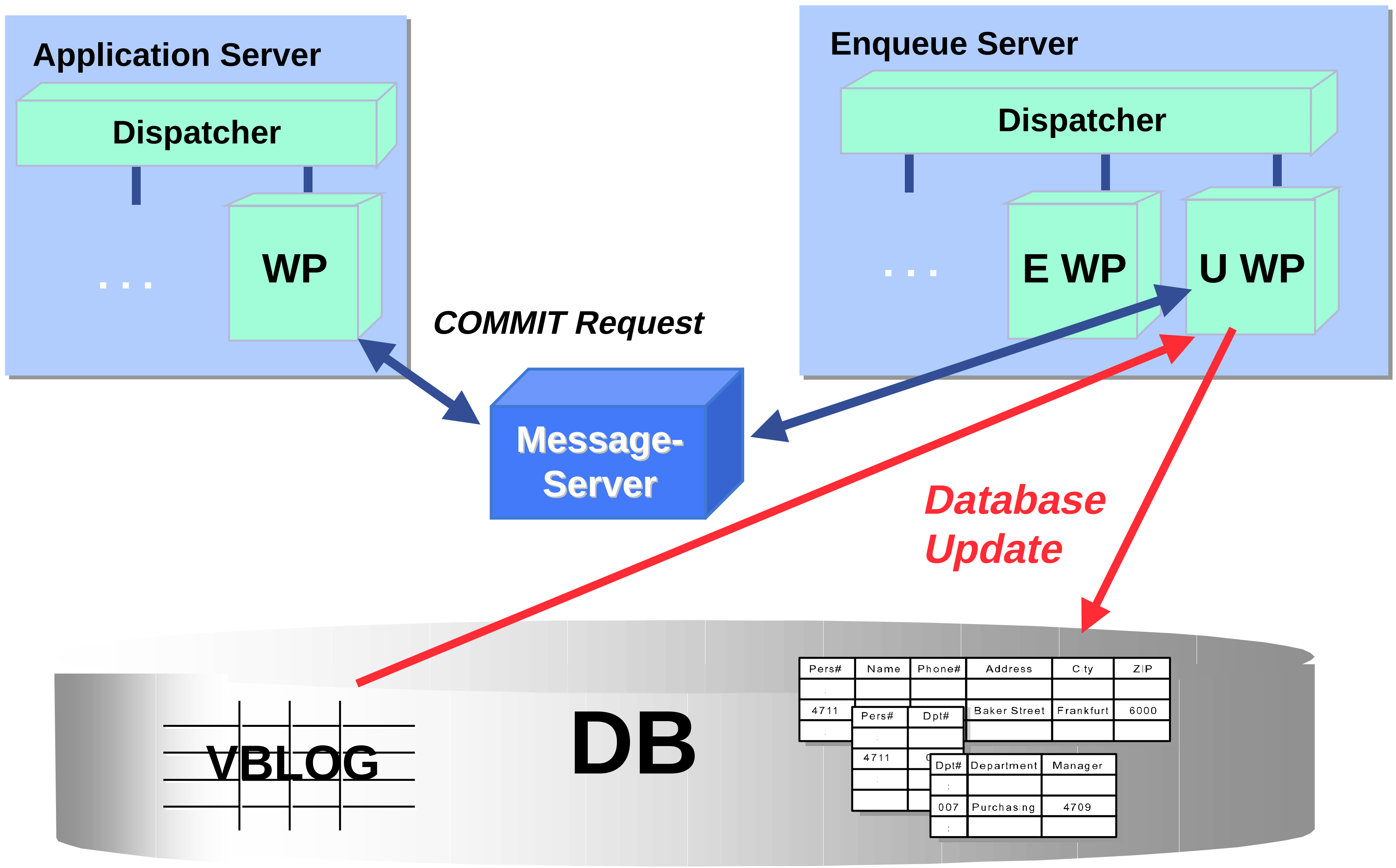


Lock request for application object

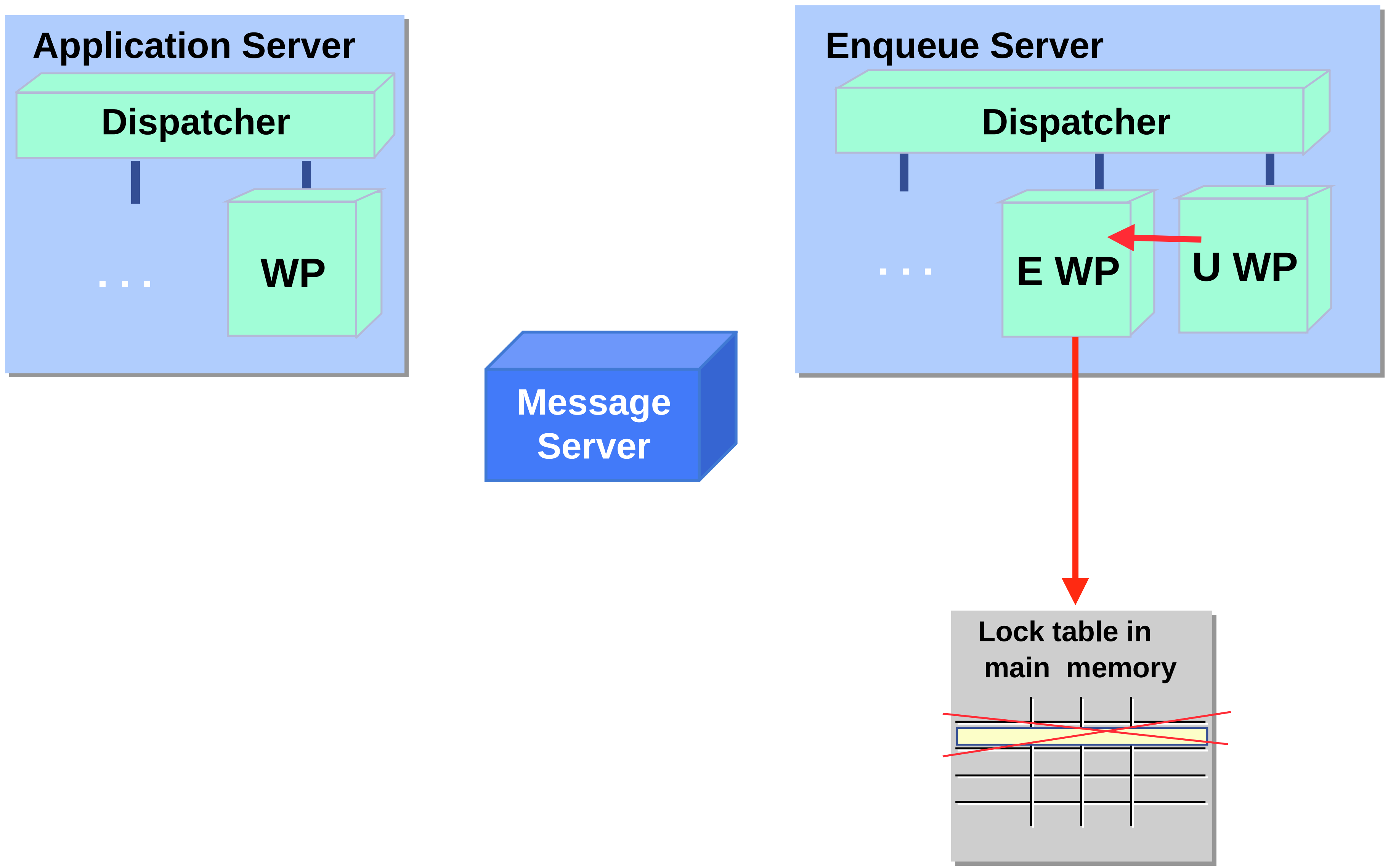
Asynchronous Database Update: Phase 1



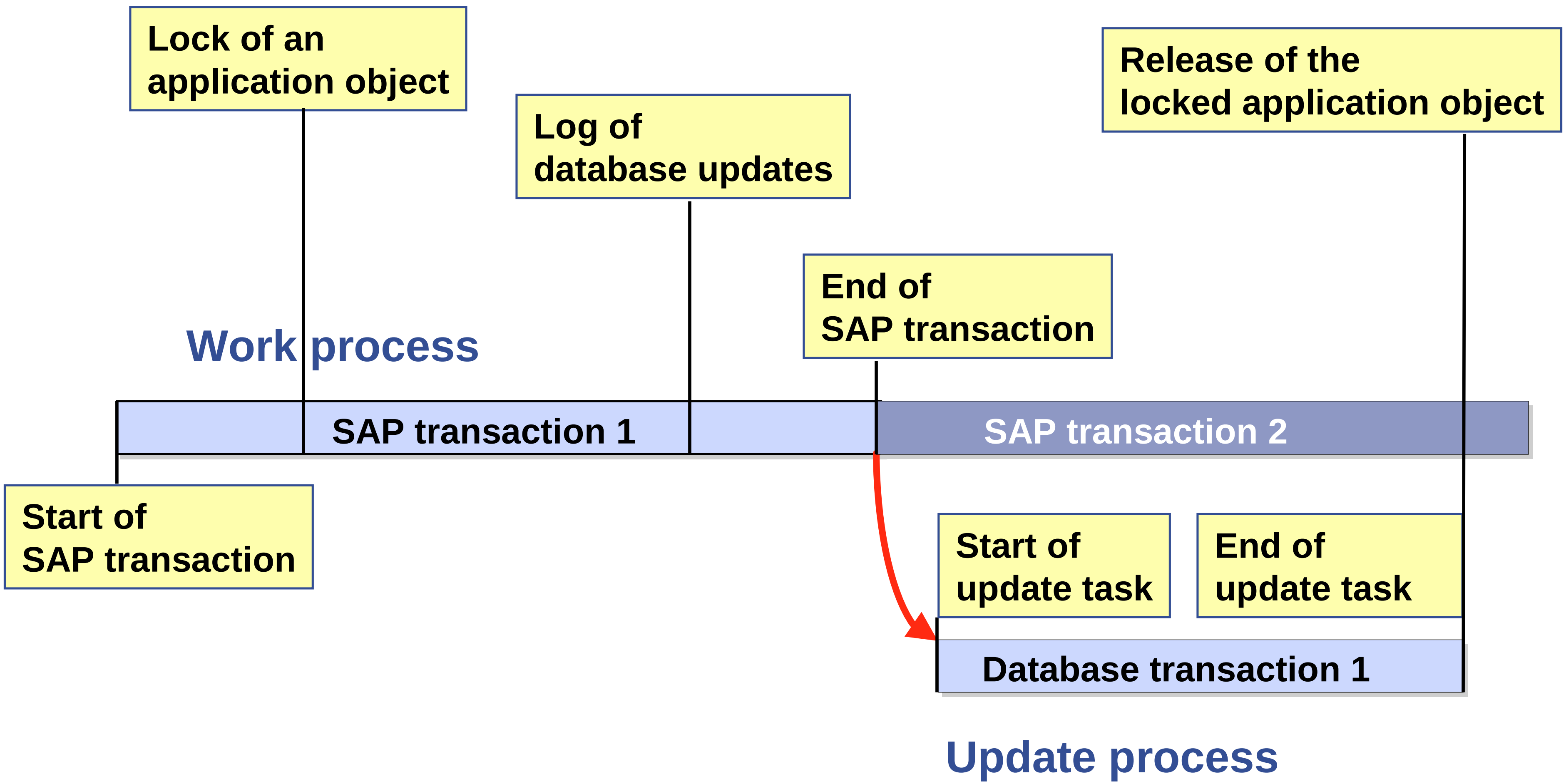
Asynchronous Database Update: Phase 2



Asynchronous Database Update: Phase 3



Asynchronous Update Protocol



Agenda

1 Who are we?

2 Where do we come from?

3 **Where do we want to go?**

DBMS Application Areas

- **Laptop / Palmtop / Handheld**
  (e. g. SAP Customer Relationship Management)
- **OLTP (Online Transaction Programming)**
  (e. g. SAP R/3)
- **Data Warehouse**
  (e. g. SAP Business Information Warehouse)
- **Document Server**
  (e. g. SAP Content Server)
- **Message Server**
  (e. g. mySAP.com)
- **Object-Oriented Database Management**
  (e. g. SAP *live*Cache, persistent ABAP Objects)

Laptop / Palmtop / Handheld

- Low footprint to (nearly) no footprint (ultra-lite DBMS)
- Fast start-up/shut-down
- Simple or no administration
- Unattended installation and upgrade

Application Profile

OLTP

- Many users
- Short transactions
- High transaction frequencies
- High number of updates
- Simple SQL commands
- Known transaction profile
- Medium to large databases (10 - 500 GB)
- Snapshot of the organization

Database Requirements

OLTP

- **Performance**
- **Performance**
- **Performance**

- **Availability**
- **Ease of use**

Data Warehouse

- Many users (report producers vs. report consumers)
- Queries (read-only)
- No transactions, no updates, no locking
- Complex, long-running SELECT commands
- Unknown workload
- High command frequency
- Very large databases (500 GB - 2 TB)
- History of the organization

Data Warehouse

- Consistent data across the enterprise
- Better performance for ad-hoc queries, decoupling of OLTP workload from queries/reports
- Easier data access for end-users

Data Warehouse

- **For the OLTP database:**
  - **Mass-data extraction**
  - **Parallel extracts**
  - **Consistent extracts without locking**

Data Warehouse

- **For the data warehouse database:**
  - **Fast loading of mass-data**
  - **Fast incremental indexing**
  - **Fast deletion of mass-data (partitions)**
  - **Very large database support (parallel backup/restore, no reorgs, no down-times)**

Data Warehouse

- **For the data warehouse database:**
  - **Transactions are irrelevant**
  - **Updates/logging/locking are irrelevant**
  - **But: staging area for data translation/cleaning**
  - **But: forcasting applications need updateable versions**
  - **Checkpoint recovery (= previous state) is sufficient**
  - **Star join support**
  - **Compact indexing**
  - **Aggregate support (maintenance, navigation, statistics/wizard)**

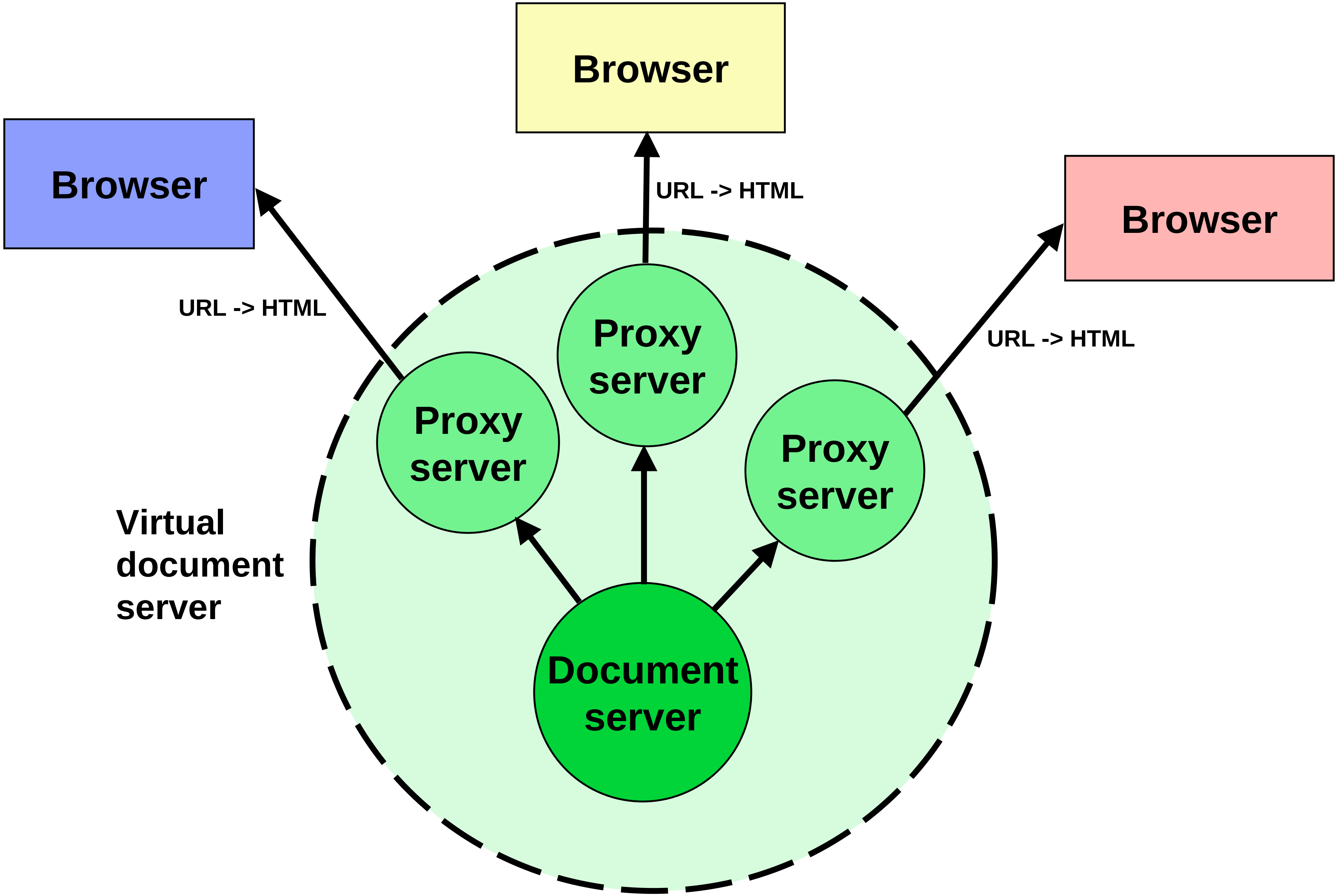
Document Server (1)

- DBMS-based storing of documents (text, image, audio, video)
- Documents are stored in BLOB container fields
- Documents are produced by editors and presented by viewers
- Internal document format and coding is unknown to the DBMS
- Document attributes are stored in descriptive fields
- Documents are organized via nested folders (similar to hierarchical file systems)
- Content is static, life span varies
- Read-only scenarios (user documentation, training)
- Read/write scenarios with shared folders

Document Server (2)

- The browser is the general document viewer
- Many (remote) users
- Workload is dominated by read accesses
- Client-side caching by proxy servers required to save roundtrips and bandwidth
- Very large databases (500 GB to 2 TB)
- Knowledge of an organization

Document Server Proxies



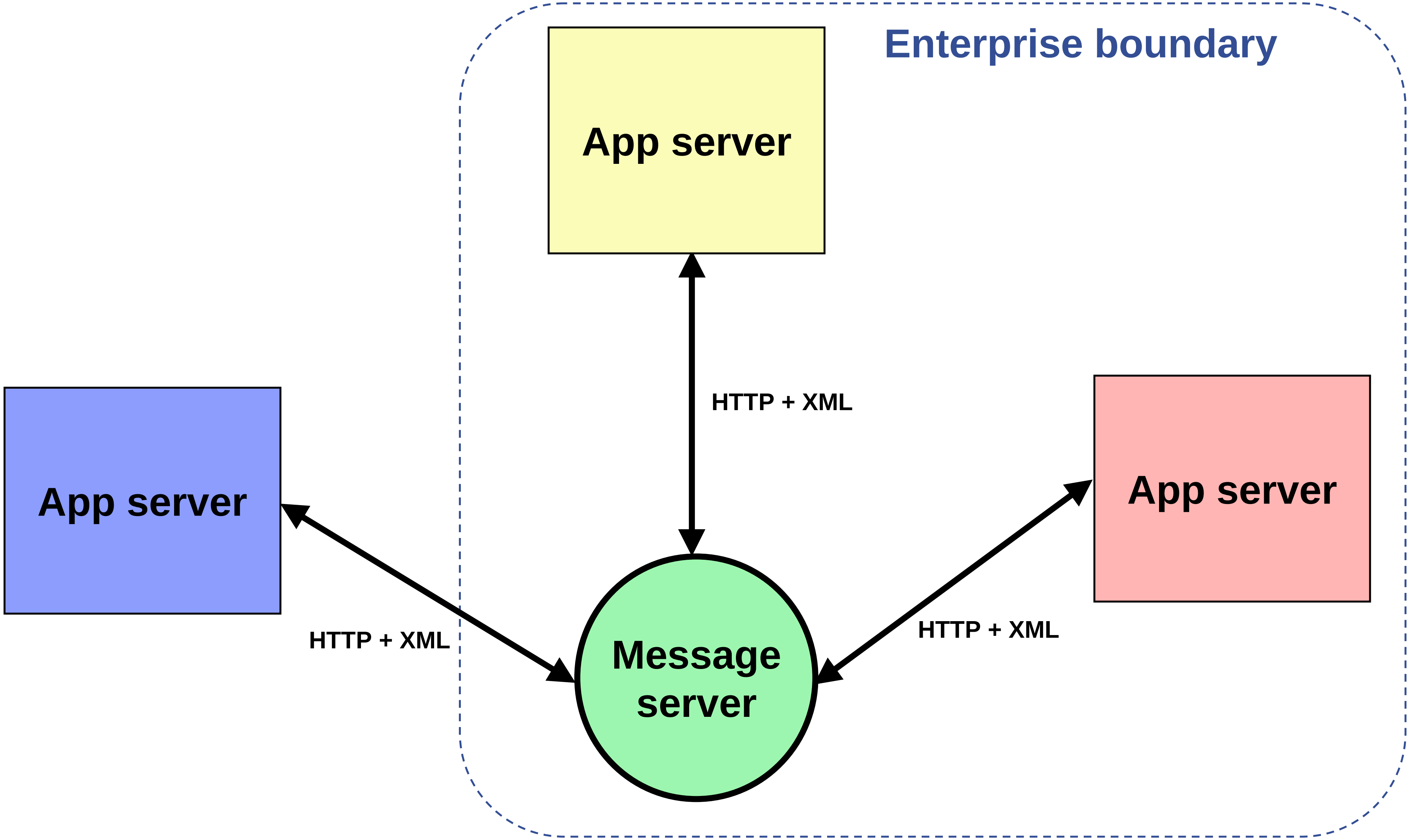
Document Server

- **Good BLOB implementation (storage overhead, read/write performance, logging overhead)**
- **Good support for complex selects, especially recursive selects for implementing folder hierarchies**
- **Support of browser-based accesses (URL -> SQL -> HTML) including session pooling**
- **Search engine (full-text retrieval) integration, but search engine should run on separate server**
- **Good handling of extremely large databases (parallel backup/restore, no reorgs, no down-times)**

Application Collaboration in the Internet

- Asynchronous flow of XML messages (documents) based on HTTP
- Collaboration of loosely coupled applications

Message Server



Collaboration of different applications

Message Server Concepts (1)

- **Availability**
- **Queuing**
- **Routing & delivery**
- **Mapping & translation**

Message Server Concepts (2)

- Store and forward of XML messages (documents) via HTTP
- Service requestors do not need to know service providers, they just address services
- Message servers are message communication hubs
- High availability independent of the service provider availability
- Message translation from XML format A to XML format B needed, due to lack of XML message standardization

XML Messages

- Messages are short-lived pieces of information
- Message delivery (one-to-one, one-to-many) follows the send/receive paradigm (send/receive/delete, messages are delivered exactly once)
- Message translation is based on XML content, otherwise the message content is not interpretable
- XML messages are typically data used for server/server interaction
- XML messages can be forms (data + presentation) suitable for user/server or user/user interaction
- XML messages can be mail as a special type of a form

Message Server

- DBMS-based storing of XML messages
- Messages are stored in BLOB container fields
- Message translation is programmed as stored procedures operating on XML data
- Message attributes are stored in descriptive fields
- Messages are kept in a single message table
- Content is very dynamic and short lived
- Reliable message delivery required
- High transaction workload (Insert, Select, Delete)
- Small (active) database size but message tracking will be necessary (message history)

Message Server

- Good BLOB implementation
  (storage overhead, read/write performance, logging overhead)
- Support for HTTP-based services in the database kernel
- Stored procedure concept with XML parser and rendering support, programming environment

Object-oriented Applications

- Object relationships expressed by OID references
- Explicit navigation and access paths modeled via OIDs
- Uni-directional access paths mirror intended usage
- Low abstraction, programmer-based optimization
- Main-memory biased representation of object relationships
- Fast object navigation (in main-memory)
- No performance degeneration for complex objects
- No query support
- Persistence does not blend well with object-orientation
- Object-oriented concepts are main-memory minded

Relational Applications

- Object relations expressed by data (foreign keys)
- Implicit and bi-directional access paths via data
- Access paths determined by SQL optimizer
- High abstraction, system-based optimization
- Disk biased represention of object relationships
- Object navigation via key accesses
  (slow compared to main-memory references)
- Performance degeneration for complex objects
  (too many table accesses)
- Good query support
- Relational concepts are disk minded

Object-oriented DBMS Trends

OO-DBMS

- **Disk-based implementations of OIDs and object navigation**

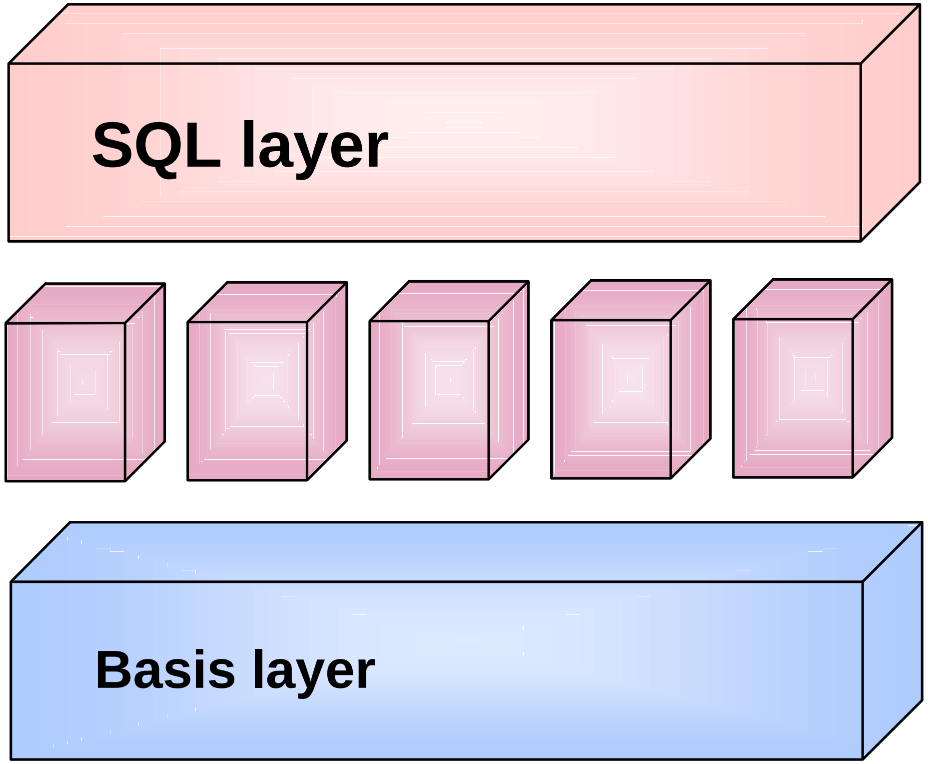
Object-relational DBMS

- **Extend SQL with object-orientation**

Hybrid DBMS

- **Put SQL and object-oriented technology side-by-side**

ORDBMS Approach

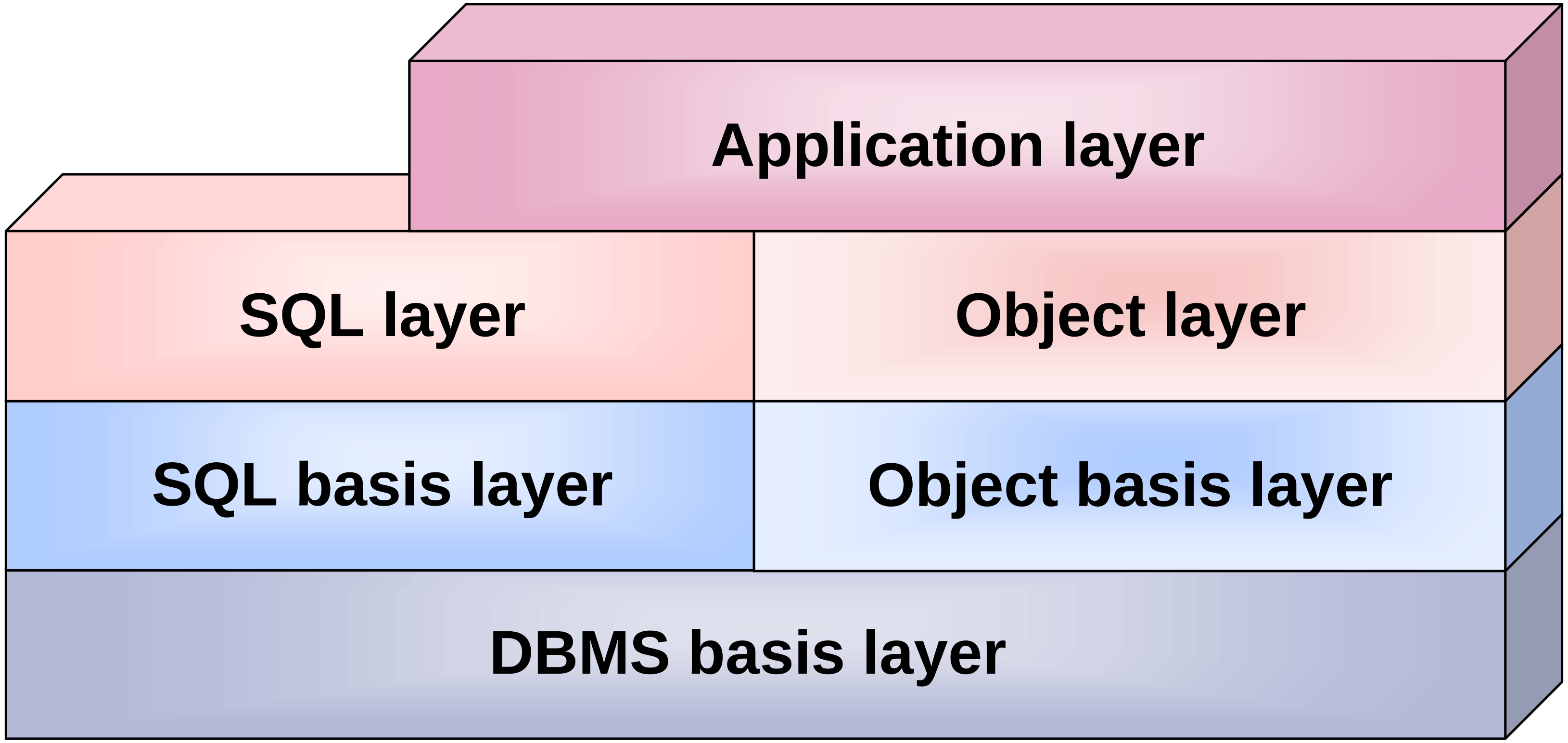


Data Blades
Data Cartridges **(third party)**
Type Extenders

Interaction with:
- *** Indexing**
- *** Optimizer & execution**
- *** Administration**

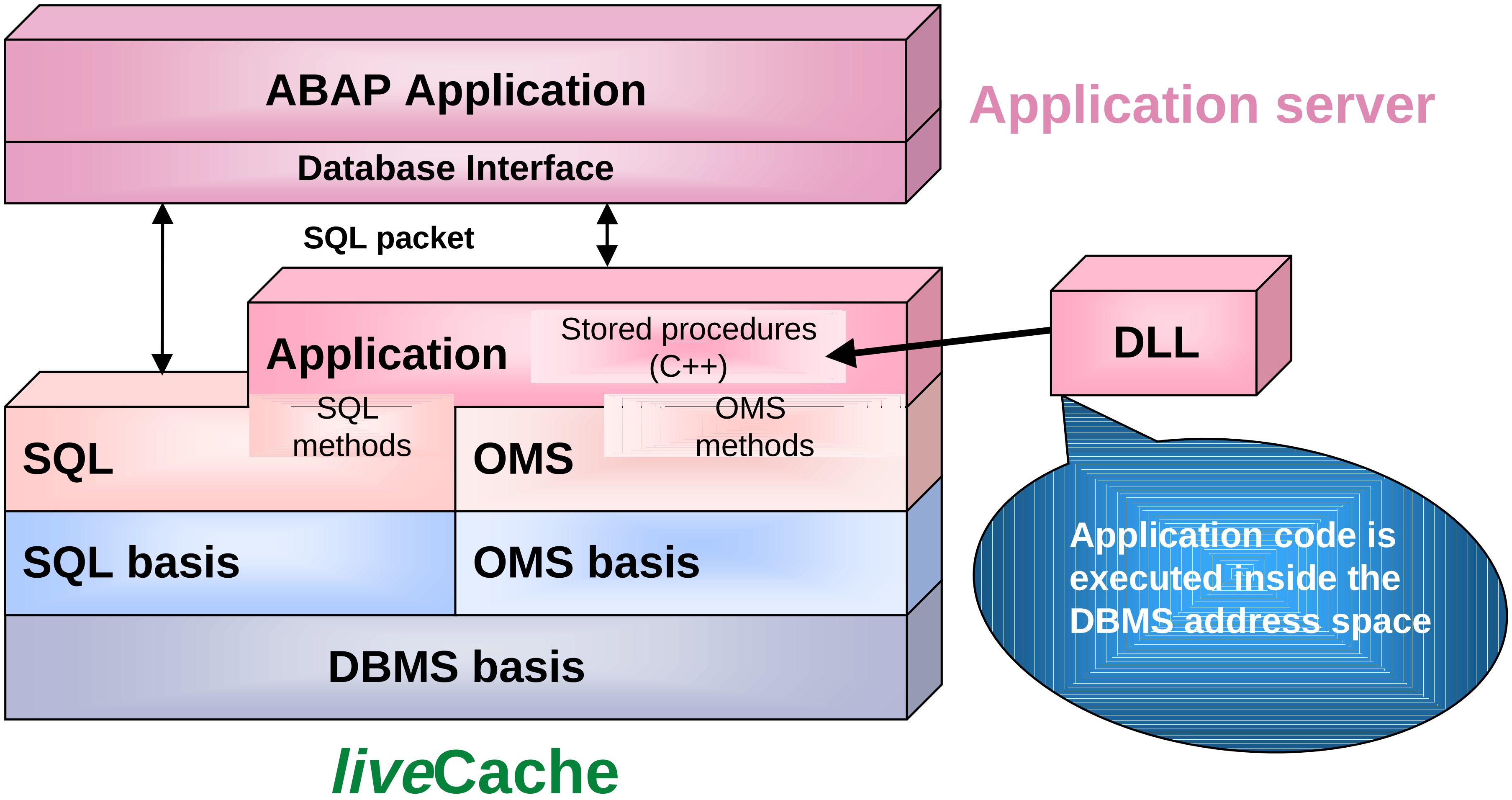
SQL wrapping of object-oriented extensions

Hybrid Approach: *live*Cache

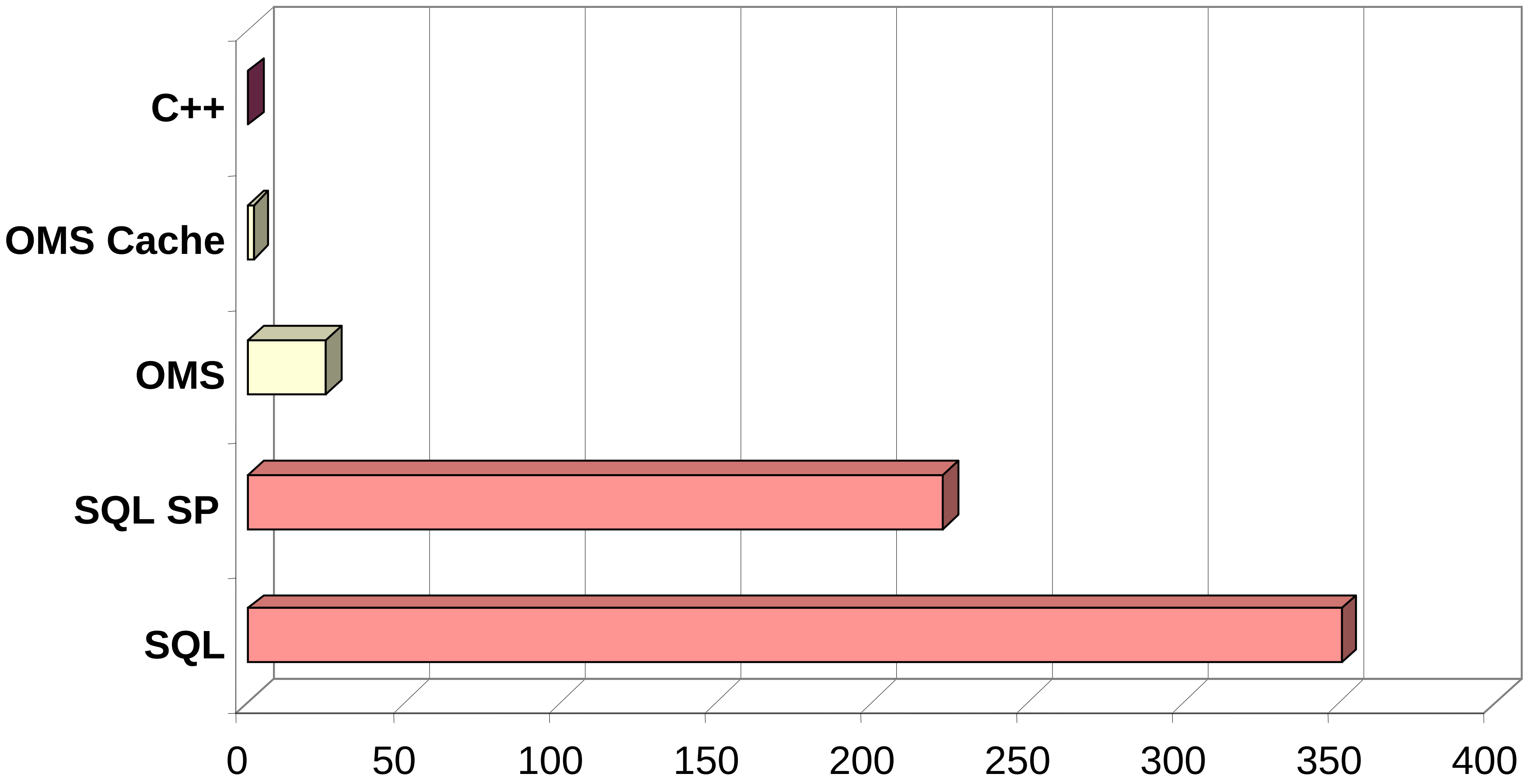


Two separate worlds of SQL data on disk and object instances in main-memory, OIDs supported as SQL attributes (anchors), SQL UDFs allow access to object instances

Architecture



Navigational Access Time (µs)



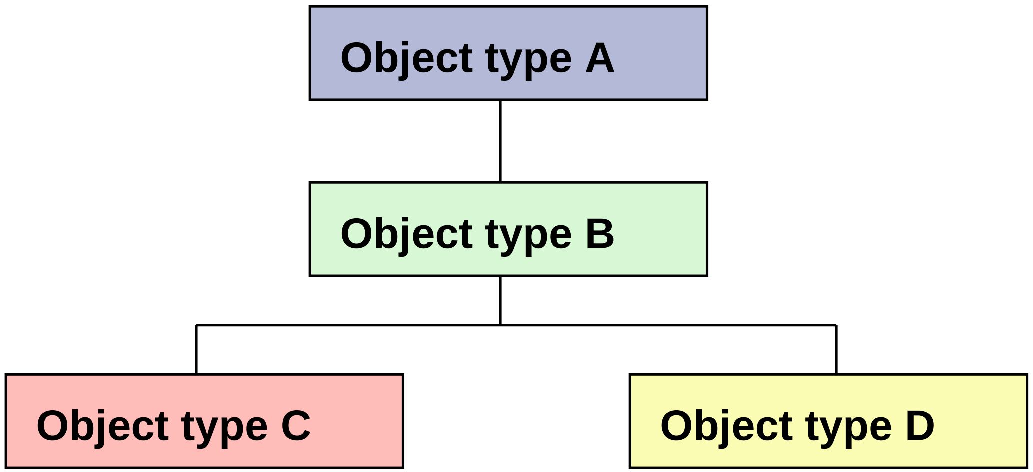
Object Persistence

- *live*Cache supports only transient shared objects
- Support for persistent shared objects is needed
- Simple persistent objects are directly mapped to rows of a corresponding SQL table (class ↔ table)
- Persistence concepts for complex objects?

Complex Objects (1)

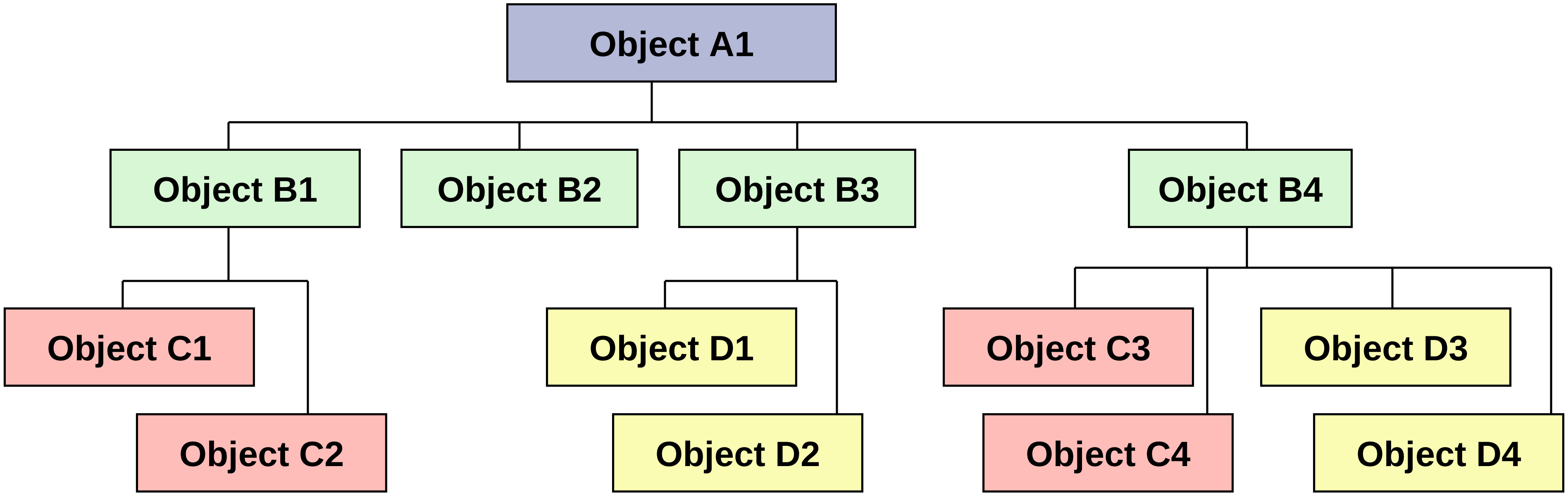
Complex objects = hierarchically nested objects

Nested object example (type level):



Complex Objects (2)

Instance Level



Persistence Support for Complex Objects

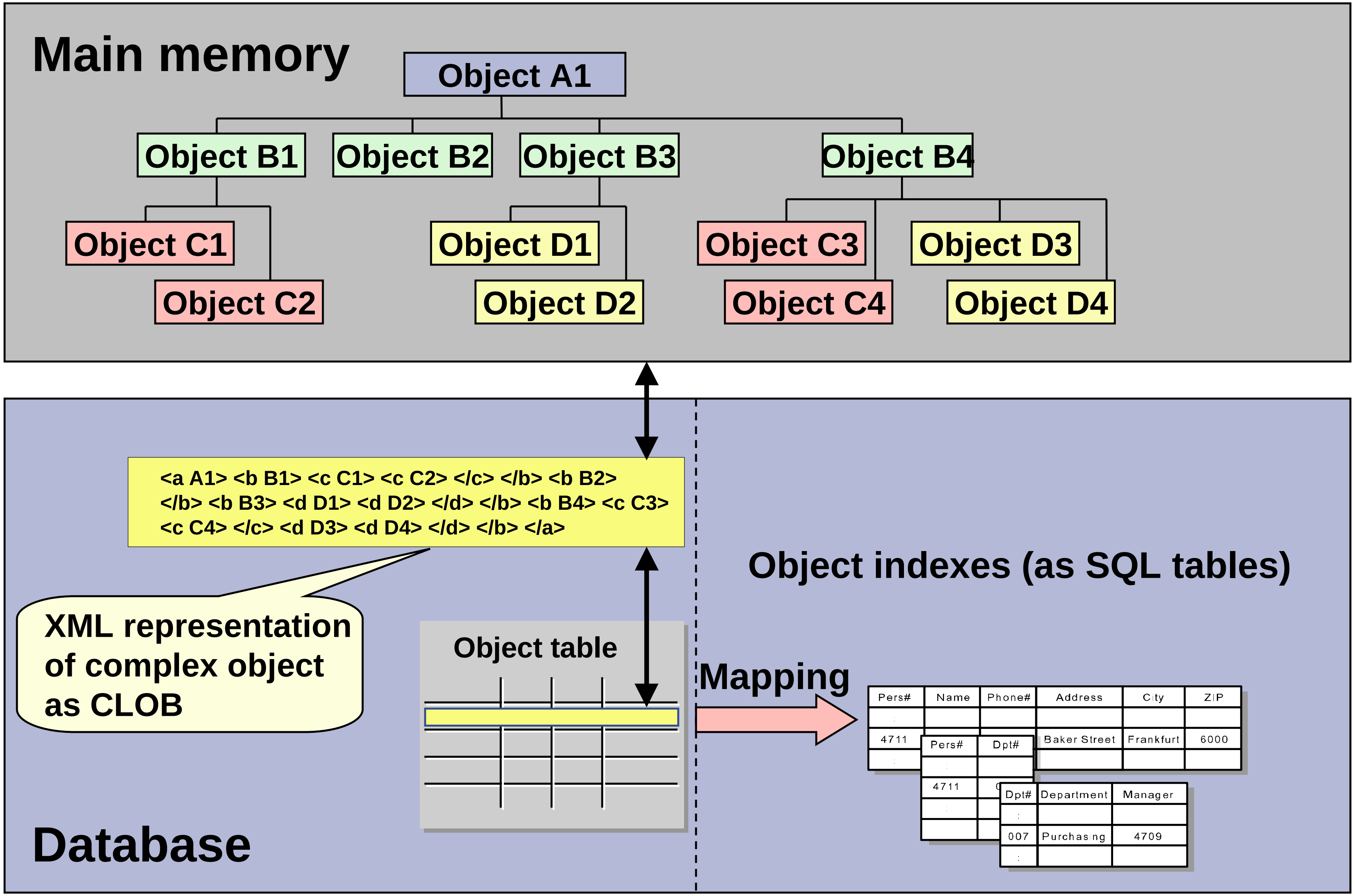
- Mapping of complex objects to nested tables (multi-level master-detail structures):
  - Relational modelling with OO extensions
  - Too many tables, too many rows involved
  - Too many DML operations needed to assemble the object
- Sequentialize complex object using XML
  - Document-type modelling
  - Store XML presentation in CLOB column
  - Storing and retrieving is very fast
  - Queries on complex objects not (directly) supported

Querying XML Documents (= Complex Objects)

- **Search engines for full-text retrieval**
  - **Too general approach to be efficient**
  - **XML documents are structured, not unstructured**
  - **XQL?**

- **Index XML documents using SQL tables**
  - **Define a mapping from XML components (tags) to a set of SQL tables and SQL columns**
  - **These tables and their columns act as indexing structure for complex objects**
  - **SELECTs an these tables provide object filters which can be used in iterators**
  - **XML documents are the original data,**
    SQL-based indexing structure can be redefined and rebuilt

Persistence Model for Complex Objects



Lessons Learned

- DBMS instances should know about their usage (config param)
- There is an application layer on top of a DBMS
- Do not try to push all data semantics down to the DBMS, a scalable system architecture has a well-defined split of work between the application layer and the database layer
- SAP‘s systems technology is driven by application needs not vice versa

Questions ?