Chapter 1

Transactions and transactional properties
3-Tier Reference Architecture

Users
Clients
Application Server
Data Server

Application Program 1
Application Program 2
...

Request
Reply

Objects

Request
Reply

encapsulated data

exposed data

Stored Data (Pages)

© Weikum, Vossen, 2002
3-Tier Reference Architecture

• **Clients:**
  presentation (GUI, Internet browser)

• **Application server:** business processes
  • application programs (business objects, servlets)
  • request brokering (TP monitor, ORB, Web server)
    based on **middleware** (CORBA, DCOM, EJB, SOAP, etc.)

• **Data server:**
  database / (ADT) object / document / mail / etc. servers

Specialization to 2-Tier Client-Server Architecture:
• Client-server with “fat” clients (app on client + ODBC)
• Client-server with “thin” clients (app on server, e.g., stored proc)
Process abstraction

Stored Data
(Pages)

© Weikum, Vossen, 2002
Process abstraction

- Process step 1
- Process step 2
- Process step 3
- Process step 4
- Process step 5

- Resource manager 1
- Resource manager 2
- Resource manager 3
- Resource manager 4

- Business process 1
- Computing process 1
- Computing process 2
- Business process 2
Transactions (1)

Consistent Database
Transactions (2)

Application transaction

Database transaction

Process step 1

Resource manager 1

Process step 2

Resource manager 2

Process step 3

Resource manager 3

Process step 4

Resource manager 4

Process step 5

Computing process 1

Process step 1

Process step 2

Process step 3

Process step 4

Process step 5

Computing process 2
ACID properties

A atomicity
C consistency
I isolation
D durability
**Consistency (1)**

- **Ideal:** full and up-to-date agreement between database and miniworld
- **Semantic consistency**
- **Consistency constraints**
  - for further narrowing of the state space
- **Transactional procedures**
  - Sequence of operations for procedural control of consistency

---

Legitimate states as a result of executing operators while interpreting the database schema

**Formal consistency**

---

Ideal: full and up-to-date agreement between database and miniworld

**Semantic consistency**
Consistency (2)

- **Database transaction** (for short: transaction): Execution of a transactional procedure on a single database.
  - A consequence: Consistency is only defined after completing the transaction.

- **Application transaction**: Consistent performance of a business process.
  - A consequence: Coordination of several database transactions needed.

A **transaction** is **consistent** if upon termination it produced a consistent database state.
ACID properties

A atomicity
C consistency ✓
I isolation
D durability
Atomicity (1)

Persistency: The effect of a completed transaction cannot be lost again.

Failure resilience: A transaction reaches a well-defined consistent state even in the presence of disturbances.

Computing process 1

Process step 1

Resource manager 1

Process step 2

Resource manager 2

Process step 3

Resource manager 3

Process step 4

Resource manager 4

Process step 5

application transaction

Persistency: The effect of a completed transaction cannot be lost again.

Failure resilience: A transaction reaches a well-defined consistent state even in the presence of disturbances.
Atomicity (2)

- Old consistent database state: persistent
- Successful transaction
- New consistent database state: persistent
- Failed transaction
- Inconsistent database state: volatile
- ??
- Consistent database state: persistent
Atomicity (3)

Examples of errors and failures:
- Incorrect input
- Programming errors
- Incorrect database management software
- Crashes of processor or memory hardware
- Crashes of operating systems

Persistency event: design a transactional procedure such that it will terminate once its results must no longer become obsolete.

Without failures the database is continuously updated.
A transaction is **atomic** if it has the all-or-nothing property.

Standard solution in case of failure.

**Atomicity (4)**

- Old consistent database state: **persistent**
- Successful transaction
- New consistent database state: **persistent**
- Failed transaction
- Inconsistent database state: **volatile**
- All-or-nothing

**Atomicity (4)**

Atomicity (4) is a property that ensures a transaction either succeeds in updating the database state or leaves it unchanged. This means that if a transaction is atomic, it either completes successfully, updating the database to a new consistent state, or it fails, leaving the database in the same state as before the transaction began. The diagram illustrates this concept with arrows representing the possible outcomes of a transaction: successful execution leading to a new consistent state, or failure leading to an inconsistent state. The all-or-nothing property is a fundamental requirement for reliable database transactions, ensuring that the database state is either restored to its previous state or updated to a new consistent state, without any intermediate inconsistent states.
ACID properties

A atomicity ✓
C consistency ✓
I isolation
D durability
Threat to consistency: If several client transactions attempt to access the same resource concurrently (compete for the same resource) they may interact in undesirable ways: they are in conflict.
Isolation (2)

- **Needed**: Synchronization protocol that avoids conflicts between concurrent and competing client transactions within a database management system (DBMS): conflict resilience.

- **Correctness**: Concurrent database transactions are correctly synchronized if each one proceeds as if there was no competition, that is, if the only inconsistencies visible to a client are those caused by its own transaction, and each transaction again reaches a persistent database state.
ACID properties

A atomicity ✓
C consistency ✓
I isolation ✓
D durability
**Durability (1)**

**Persistency:** Make sure on completion of a transaction that its effect cannot be lost.

**Durability:** Survival in case of loss non-volatile data: Make sure that the effect of a transaction *never* gets lost.
**Durability (2)**

- **Problem**: Storing data for an unlimited time makes consistency an even harder problem because of the increased probability of disturbances or failures occurring, with a concomitant corruption or complete loss of the data.
  - Sample threats: Failures of peripheral storage, storage media or processors; external events such as fire, water, climate; aging of media with ensuing destruction of the database.
- **Durability**: Overcome loss by somehow restoring an earlier, still useful database state.
  - Persistency is a prerequisite for durability.
Responsibilities (1)

- **A** atomicity
- **C** consistency
- **I** isolation
- **D** durability

**Transaction management (TM):** control of a set of potentially overlapping transactions with guarantees for consistency and robustness for all of them, taking into account further factors such as performance.

- **Transaction**: resilient execution of a consistent state transition (single operator or transactional procedure) with persistency of the final state.
  - Design time: **Transactional procedure**: Sequence of primitive operations considered as a unit of consistency and resilience.
  - Runtime: **Transaction (TA)**: Execution of a transactional procedure, guaranteeing consistency and resilience.
Responsibilities (2)

A atomicity

C consistency

I isolation

D durability

- **Transaction**: resilient execution of a consistent state transition (single operator or transactional procedure) with persistency of the final state.
  
  - Design time: **Transactional procedure**: Sequence of primitive operations considered as a unit of consistency and resilience.
  
  - Runtime: **Transaction (TA)**: Execution of a transactional procedure, guaranteeing consistency and resilience.
Responsibilities (3)

Basic assumption:

The transaction itself is responsible for **local consistency**: If undisturbed a transaction effects a consistent transition, i.e., results in a consistent database state provided it started from a consistent database state.
Responsibilities (4)

- **A** atomicity
- **C** consistency
- **I** isolation
- **D** durability

- **Transaction**: resilient execution of a consistent state transition (single operator or transactional procedure) with persistency of the final state.
  - Design time: **Transactional procedure**: Sequence of primitive operations considered as a unit of consistency and resilience.
  - Runtime: **Transaction (TA)**: Execution of a transactional procedure, guaranteeing consistency and resilience.
Responsibilities (5)

- **Recovery**: Enforcement of persistency and failure resilience.

- **Atomicity**: The transaction shows an external effect only as a whole. Prior to successful completion there is no observable effect (transiency), after successful completion the effect is generally visible (persistency).

- **Responsibilities**:
  - Persistency: Transaction has already completed.
  - Failure resilience: Transaction has lost control.
  - Prime responsibility lies with the recovery manager as part of database management.
Responsibilities (6)

- Atomicity
- Consistency
- Isolation
- Durability

**Transaction**: resilient execution of a consistent state transition (single operator or transactional procedure) with persistency of the final state.

- **Design time**: **Transactional procedure**: Sequence of primitive operations considered as a unit of consistency and resilience.
- **Runtime**: **Transaction** (**TA**): Execution of a transactional procedure, guaranteeing consistency and resilience.
Responsibilities (7)

- Local consistency is not sufficient to guarantee database consistency in a set of concurrent and competing transactions. We must enforce **global consistency** by conflict resilience:
  
  ◆ **Isolation**: Concurrent transactions execute as if each would have its resources all by itself (no „mixing“ of state transitions).
  
  ➢ Other concurrently executing transactions remain invisible to a given transaction.

- Responsibilities:
  
  ◆ Conflict resilience: Transaction does not know other transactions.
  
  ➢ The responsibility rests with the Scheduler as part of database management.
Responsibilities (8)

- **Atomicity** (A)
- **Consistency** (C)
- **Isolation** (I)
- **Durability** (D)

**Transaction**: resilient execution of a consistent state transition (single operator or transactional procedure) with persistency of the final state.

- **Design time**: Transactional procedure: Sequence of primitive operations considered as a unit of consistency and resilience.
- **Runtime**: Transaction (TA): Execution of a transactional procedure, guaranteeing consistency and resilience.
Responsibilities (9)

- **Durability**: Long-duration persistency:
  - The effect of a successfully completed transaction is forever preserved unless it is explicitly renounced by a further transaction.

- Since the transaction does no longer exist after completion and may have occurred a long time ago, the responsibility can only rest with a separate system component (archive management).
System architecture

Transaction 1  Transaction 2  ...  Transaction n

Scheduler

Consistency  Isolation

Database Manager

Backup/Recovery Manager

Atomicity

Database

Archive Manager

Durability

restart
restore
ACID good for all seasons?

- Useful for standard commercial applications where:
  - transactions are independent and of short duration (e.g., debit/credit transactions),
  - correctness requirements are high.

- Less useful for:
  - cooperating applications (e.g., CAD): Strict isolation makes no sense in cooperative development (e.g., collaborative design of an automobile engine), because intermediate results should be shared by other engineers.
  - long-lasting sessions (e.g., Internet reservations): For long-duration, resource-intensive transactions a complete undo of all work so far according to atomicity seems unacceptable.