

Lars Frank:

- **1971** Master of computer science (Datalog) and mathematics
- **1975** HD in organization
- **1975-93** Database consultant (primært i banksektoren)
- **1994-** Associate professor at CBS
- **2007** Dr. Merc. Thesis accepted by evaluators

Overordnet designregel for distribuerede databaser:

De rette data skal være tilgængelige
til rette tid,
til rette person/system,
på rette sted,
i rette kvalitet,
til de mindste omkostninger.

Der er konflikter mellem de undestregede krav, hvorfor afvejningen i sidste ende er politisk.

Det er derfor vigtigt at have vælgmuligheder!

Architectures for integrating e-health records	Evaluation criteria			
	Local autonomy	Consistency/anomaly problems	Read performance	Software costs
1. Integration by using SOA health services	Best	Worst	Worst	Worst
2. Central database solution	Worst	Best	Best	Best
3. Central database solution mixed with SAO integration	Average	Above worst	Above worst	Average
4. Distributed subscriber solution	Best	Average	Best	Worst
5. Central subscriber that offer SOA services to others	Best	Average	Average	Worst
6. Central database solution with central subscription and SOA services to others (5+2).	Average	Average	Average	Average
7. Central database solution mixed with distributed subscription on top of central subscription (6+4).	Average	Average	Best	Average

Architectures for integrating e-health records	Evaluation criteria			
	Local autonomy	Consistency/ anomaly problems	Read performance	Software costs
1. Integration by using SOA health services	Best	Worst	Worst	Worst
2. Central database solution	Worst	Best	Best	Best
3. Central database solution mixed with SAO integration	Average	Above worst	Above worst	Average

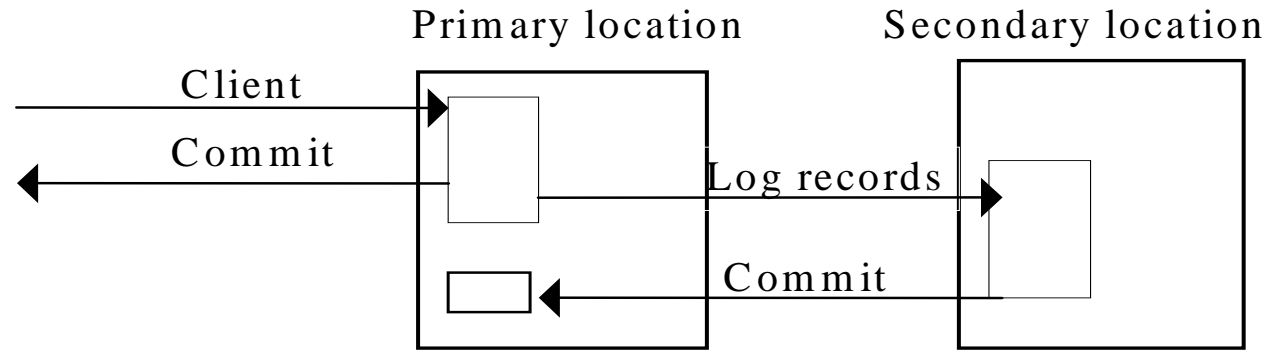
Architectures for integrating e-health records	Evaluation criteria			
	Local autonomy	Consistency/anomaly problems	Read performance	Software costs
1. Integration by using SOA health services	Best	Worst	Worst	Worst
2. Central database solution	Worst	Best	Best	Best
3. Central database solution mixed with SAO integration	Average	Above worst	Above worst	Average
4. Distributed subscriber solution	Best	Average	Best	Worst
5. Central subscriber that offer SOA services to others	Best	Average	Average	Worst
6. Central database solution with central subscription and SOA services to others (5+2).	Average	Average	Average	Average
7. Central database solution mixed with distributed subscription on top of central subscription (6+4).	Average	Average	Best	Average

Architectures for integrating e-health records	Evaluation criteria			
	Local autonomy	Consistency/ anomaly problems	Read performance	Software costs
1. Integration by using SOA health services	Best	Worst	Worst	Worst
2. Central database solution	Worst	Best	Best	Best
3. Central database solution mixed with SAO integration	Average	Above worst	Above worst	Average
4. Distributed subscriber solution	Best	Average	Best	Worst
5. Central subscriber that offer SOA services to others	Best	Average	Average	Worst
6. Central database solution with central subscription and SOA services to others (5+2).	Average	Average	Average	Average
7. Central database solution mixed with distributed subscription on top of central subscription (6+4).	Average	Average	Best	Average

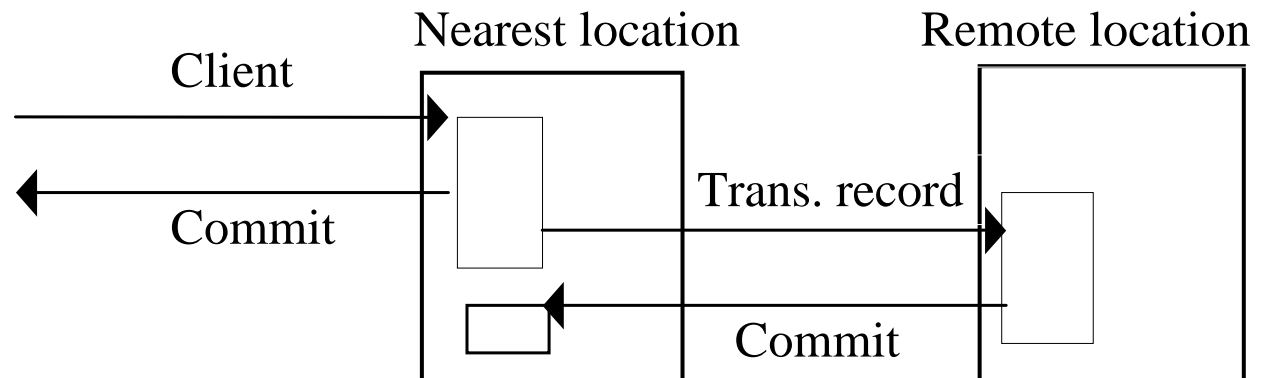
Architectures for integrating e-health records	Evaluation criteria			
	Local autonomy	Consistency/anomaly problems	Read performance	Software costs
1. Integration by using SOA health services	Best	Worst	Worst	Worst
2. Central database solution	Worst	Best	Best	Best
3. Central database solution mixed with SAO integration	Average	Above worst	Above worst	Average
4. Distributed subscriber solution	Best	Average	Best	Worst
5. Central subscriber that offer SOA services to others	Best	Average	Average	Worst
6. Central database solution with central subscription and SOA services to others (5+2).	Average	Average	Average	Average
7. Central database solution mixed with distributed subscription on top of central subscription (6+4).	Average	Average	Best	Average

**In *x-safe design*,
x out of n replicated tables are consistent and up to date**

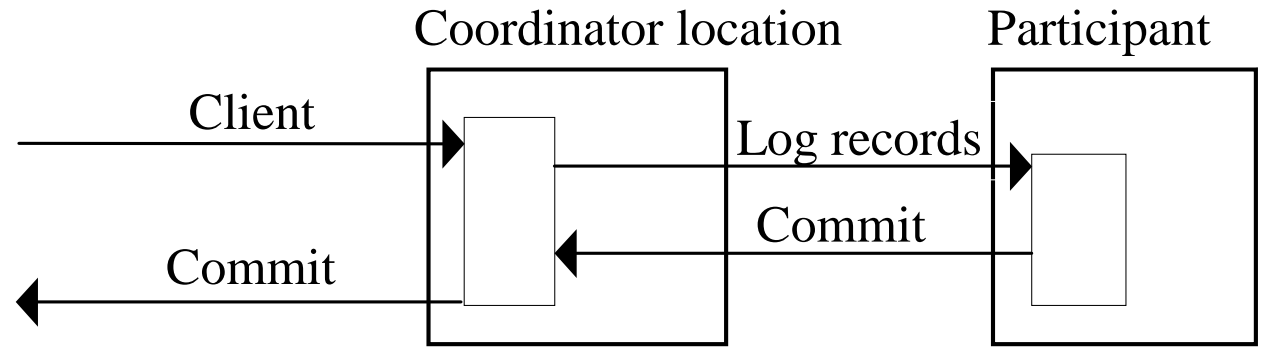
**The Basic
1-safe Design**



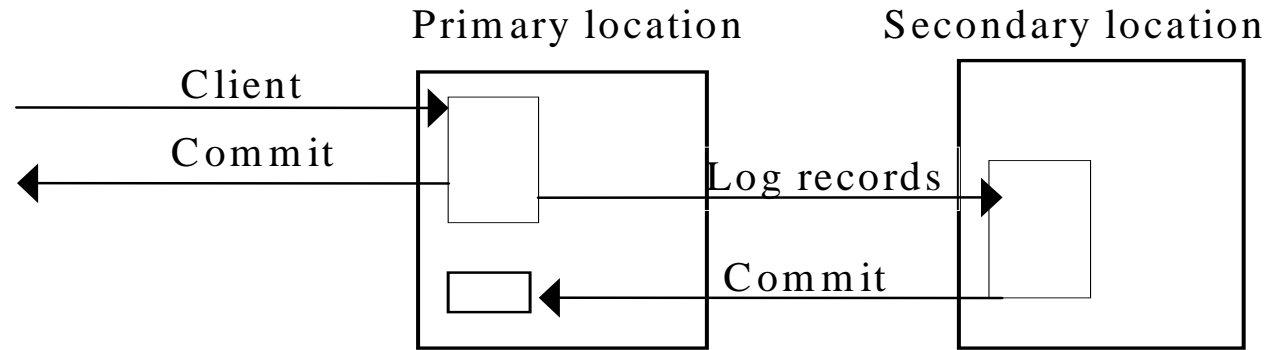
**The Basic
0-safe Design**



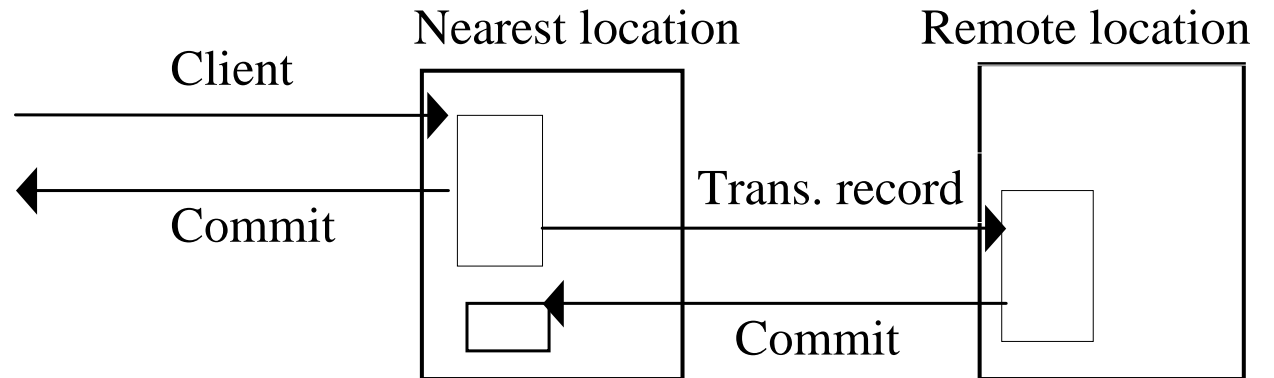
The Basic n-safe Design



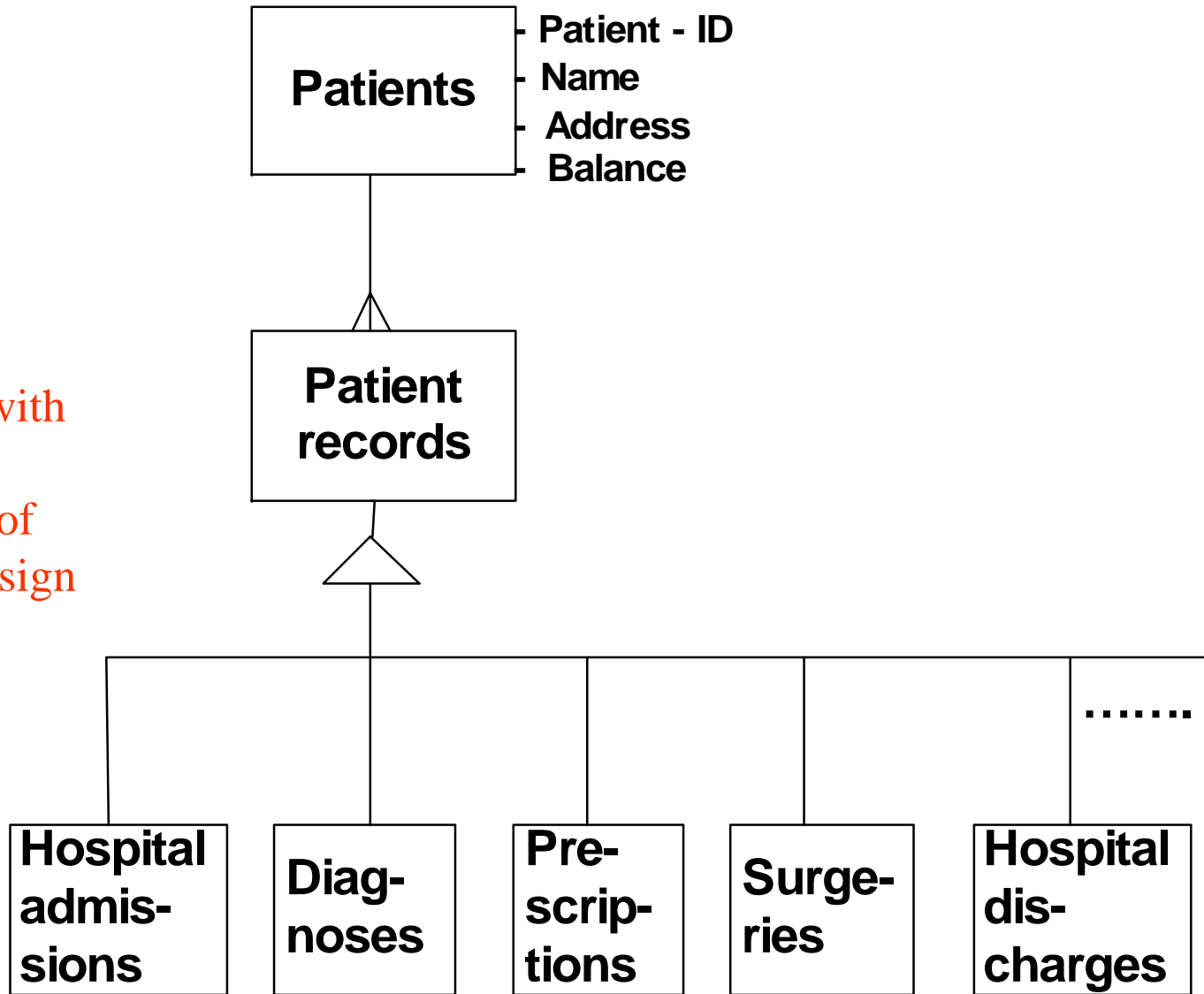
The Basic 1-safe Design



The Basic 0-safe Design



Health Records Database

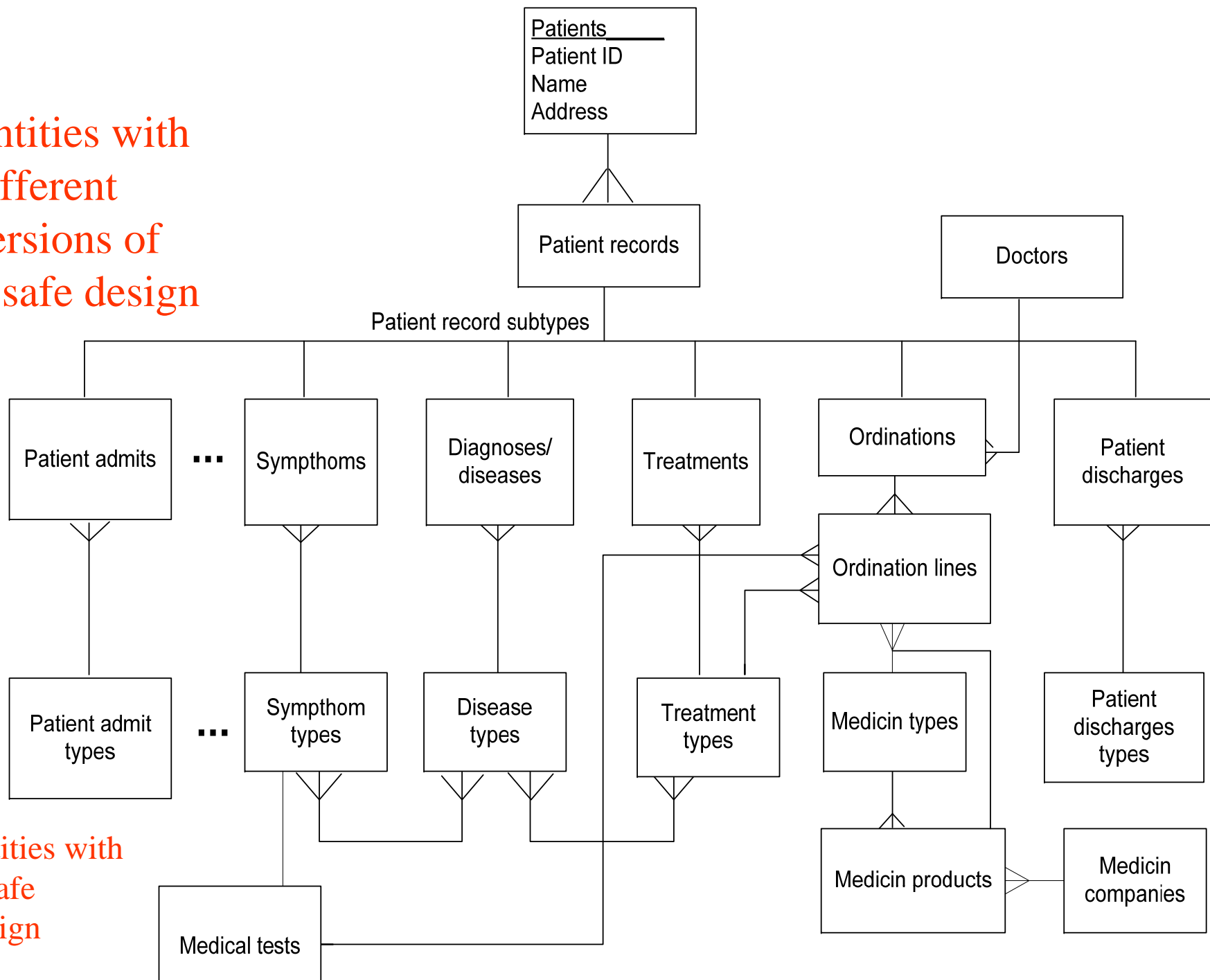


Entities with different versions of 0-safe design

Deffered commit

Local commit

Entities with different versions of 0-safe design



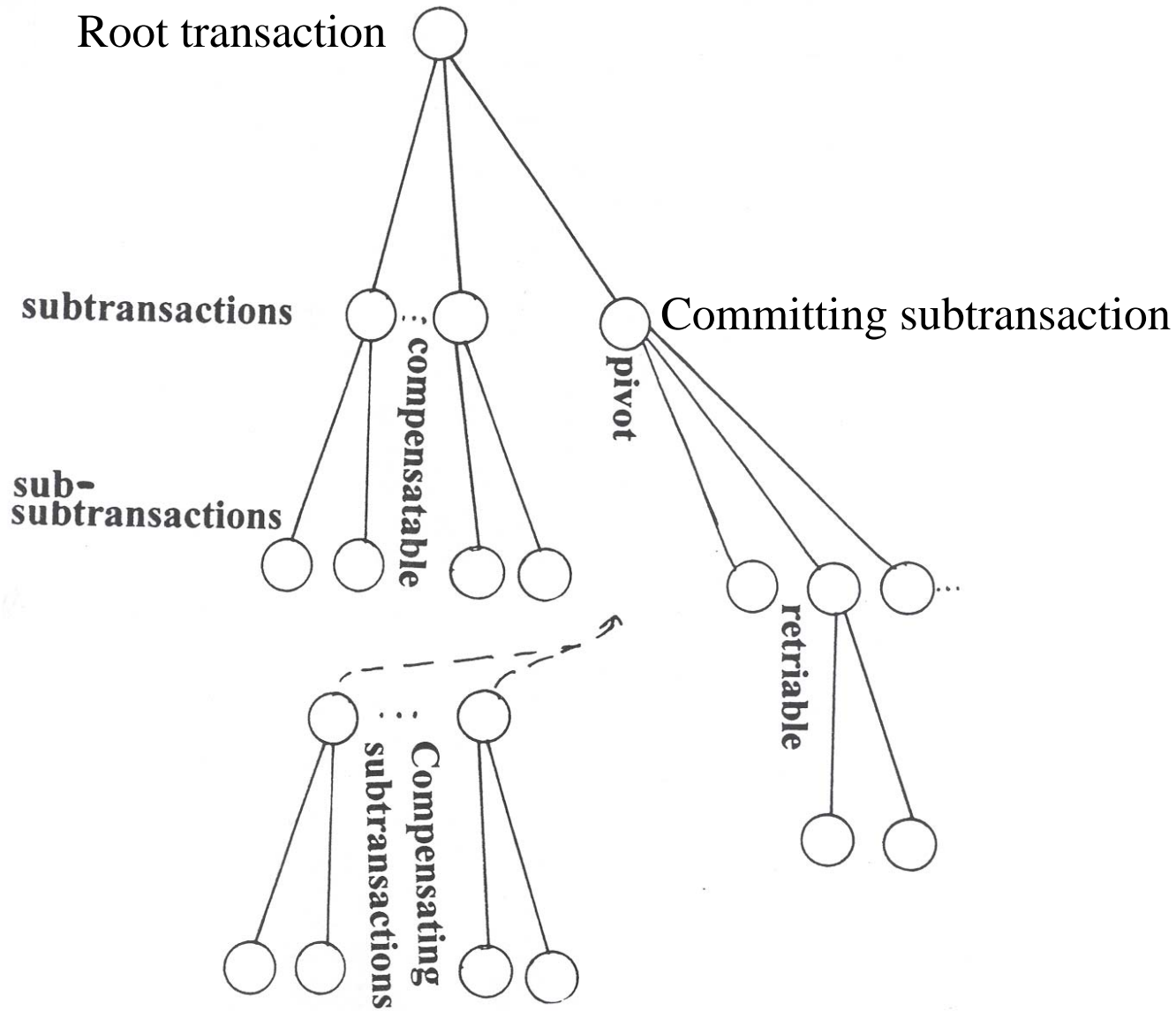
Entities with 1-safe design

Design Rules for Replicating Data in Distributed Databases with Relaxed ACID Properties

1. Use only replicated data when it is necessary autonomy or convenient for economical reasons.
2. The basic 1-safe design is recommended when it does not make major problems if an update is delayed in case of disconnection.
3. The 0-safe design with local commit is recommended in situations where it is possible to implement sufficient local countermeasures against the isolation anomalies and if it is important to operate in disconnected mode.
4. The 0-safe design with deferred commit is recommended in situations where it is not possible to implement sufficient local countermeasures against the isolation anomalies and if it is important to operate in disconnected mode.

Properties	DBMS supported replication methods						
	n-safe design with the ROWA protocol	Quorum-safe design	1-safe design. Basic solution	1-safe design with commutative updates	0-safe design with local commit	0-safe designs with deferred commit	No-replication design
Read performance/capacity	Best	Worst	Average	Average	Best	Best	Average
Write performance	Worst	Above worst	Average	Average	Best	Below best	Average
Ease of failure recovery	Average	Average	Worst	Average	Best	Best	Average
Ease of disaster recovery	Best	Below best	Above worst	Average	Average	Average	Worst
The probability of lost data	Best. p^n	Below best $p^{\lceil n/2 \rceil}$	Worst p	Average	Average	Average	Worst p
Logging of the update transaction	Not supported	Not supported	Not supported	Recommended	Recommended	Recommended	Not supported
Availability	$1-q^n$	$1-q^n$	$1-q^n$	$1-q^n$	$1-q^n$	$1-q^n$	$1-q$
Atomicity	Best	Best	Worst	Best	Best	Best	Best
Consistency	Best	Best	Average	Average	Worst	Worst	Best
Isolation	Best	Best	Average	Average	Worst	Worst	Best
Durability	Best	Best	Worst	Best	Best	Best	Best
Development costs	Best	Best	Best	Average	Worst	Worst	Best

Atomicity by using flexible transactions:



Isolation anomalies

The lost update anomaly
(Dirty write anomaly)

← Write does not
exclude write.

The dirty read anomaly

← Write does not
exclude read.

The non-repeatable read anomaly
(or fuzzy read)

← Read does not
exclude write.

The phantom anomaly

Countermeasures against isolation anomalies:

- The Reread Counter-measure
- The Commutative Updates Counter-measure
- The Version File Counter-measure
- The Version File and Commutative Updates Counter-measure
- The Pessimistic View Counter-measure
- The Semantic Lock Counter-measure
- The End of Day Transaction Countermeasure
- Counter-measures by Value
- The Pivot Lock Counter-measure
- The Enforcement Method Counter-measure

Countermeasures against missing isolation property

