H2-99-476
Database Replication
Technical Overview

Herb Sutter

Agenda

Scope and Objectives

Key Concepts:

1. Motivation: Why Replication?
2. Major Architectural Alternatives
3. Asynchronous Transaction Model

Major Requirements:

4. Conflict Prevention and Handling
5. Data Subset Management Approaches
6. Transformation and Mapping Overview

Summary

Scope and Objectives

Describe the state of the art:

- what are the major architectural approaches?
- what are the major issues, and how can they be addressed?

Demonstrate that the area is well-understood

Be product-neutral:

- will cover benefits and drawbacks, but no existing product incorporates all "best practices" described in this presentation

1 Agenda

Scope and Objectives

Key Concepts:

1. Motivation: Why Replication?

2. Major Architectural Alternatives
3. Asynchronous Transaction Model

Major Requirements:

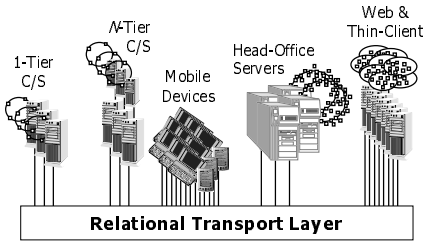
4. Conflict Prevention and Handling
5. Data Subset Management Approaches
6. Transformation and Mapping Overview

Summary

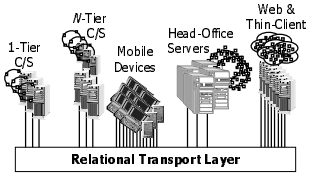
1 Major Distributed Systems Approaches

	Central Server	Traditional Synchronous: N-Phase Commit	Queue-Based Asynchronous Replication	Ideal Asynchronous Replication
Scalability	one site	< 20 sites	< 100 sites (hub/spoke)	1000+ (autodetection, autoconfig, dyn. load bal.)
Autonomy/ Availability	one site	poor	full, local (if update-anywhere)	full, local
Performance	good (if local and one server is enough)	poor for updates	local	local
Points of Failure	one site	many	one or many (for repl.)	any subset live (repl. clustering, failover)
Data Currency	immediate	immediate	propagated	parallelized propagation

1 An "Ideal" Distributed System



1 An "Ideal" Distributed System



- no new relational concepts **(e.g., normalization)**; business-rule driven
- every site a peer; update-anywhere "just as good as head office"
- every site autonomous, all transactions local; no point of failure
- automatic, transparent; just add a database wherever needed
- all user's favorite tools: Java/VB/HTML/X/C++/Perl/Delphi/..., 1-tier/N-tier/Web/Win32/Unix...; OLTP/OLAP/DSS/reporting...

1 Rate of Change: How to measure scalability

	Local Database Size	10% change per month, 4:1 compression	Required Bandwidth for Trickle Replication During Bus. Hours
Mobile Laptop	**50 MB**	60K / day	**Cell, 28.8K modem**
Workgroup Server	**1 GB**	1.1M / day	**28.8K modem**
Regional Server	**20 GB**	23M / day	**56K modem**
Headquarters / Warehouse	**1 TB**	1G / day	**256K ISDN**

2 Agenda

Scope and Objectives

Key Concepts:

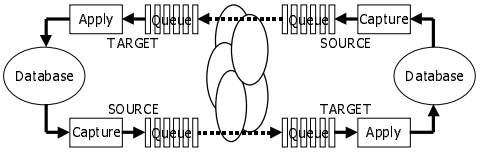
1. Motivation: Why Replication?
2. **Major Architectural Alternatives**
3. Asynchronous Transaction Model

Major Requirements:

4. Conflict Prevention and Handling
5. Data Subset Management Approaches
6. Transformation and Mapping Overview

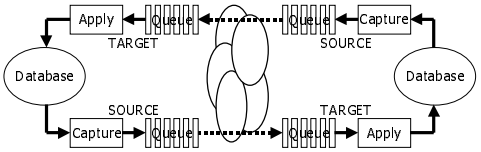
Summary

2 Queue-Based Replication



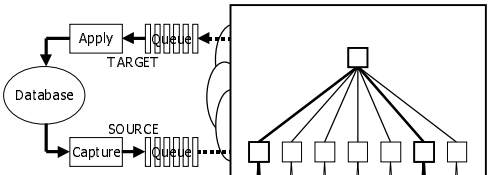
- (a.k.a. Message-Based, Source/Target)
- Has existed in some form since at least mid-1980s
- Major Advantages:
  - Can be built on top of existing e-mail, FTP, and other message-based transports

2 Queue-Based Replication



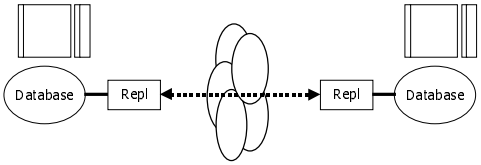
- Major Issues:
  - **Reliability**: Many points of failure; full-queue problems
  - **Integrity**: "Conflict chasing" and resolution problems
  - **Predictability, Footprint**: Dynamic queues are unbounded, and inefficient because data is duplicated
  - **Robustness, Scalability**: "Queue per partner" enforces hub-and-spoke models, prevents dynamic load balancing

2 Queue-Based Replication



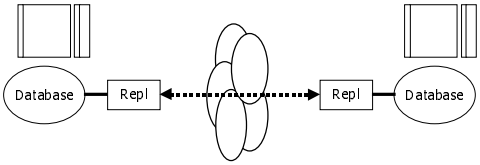
- Major Issues:
  - **Reliability**: Many points
  - **Integrity**: "Conflict chasi
  - **Predictability, Footprin** and inefficient because da
  - **Robustness, Scalability** hub-and-spoke models, p

2 Queueless Replication



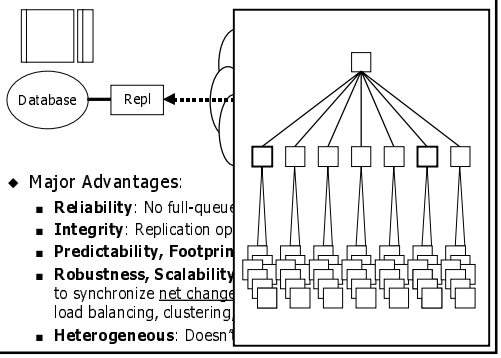
- Major Issues:
  - For a database to be replicated requires that at least one partner replication process must be pingable

2 Queueless Replication



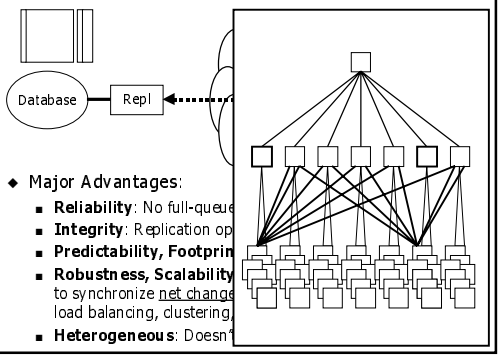
- Major Advantages:
  - **Reliability**: No full-queue problems, no "restart" overhead
  - **Integrity**: Replication operates on live data, not stale data
  - **Predictability, Footprint**: Fixed, data never duplicated
  - **Robustness, Scalability**: Contains all information required to synchronize net changes with any partner; allows dynamic load balancing, clustering, automatic replication failover
  - **Heterogeneous**: Doesn't depend on logs

2 Queueless Replication



- Major Advantages:
  - **Reliability**: No full-queue
  - **Integrity**: Replication op
  - **Predictability, Footprin**
  - **Robustness, Scalability**
    to synchronize net change
    load balancing, clustering,
  - **Heterogeneous**: Doesn't

2 Queueless Replication



- Major Advantages:
  - **Reliability**: No full-queue
  - **Integrity**: Replication op
  - **Predictability, Footprin**
  - **Robustness, Scalability**
    to synchronize net change
    load balancing, clustering,
  - **Heterogeneous**: Doesn't

3 Agenda

Scope and Objectives

Key Concepts:

1. Motivation: Why Replication?
2. Major Architectural Alternatives
3. **Asynchronous Transaction Model**

Major Requirements:

4. Conflict Prevention and Handling
5. Data Subset Management Approaches
6. Transformation and Mapping Overview

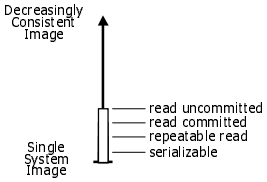
Summary

3 Single-User System

Single System Image —

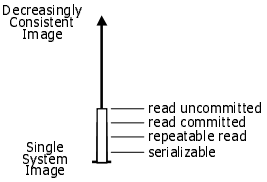
- Full Isolation:
  - **ACID**: Need only the consistent "unit of work" concept + durability; user always runs at full isolation
  - Single system image to all users, because only one user

3 Multi-User System



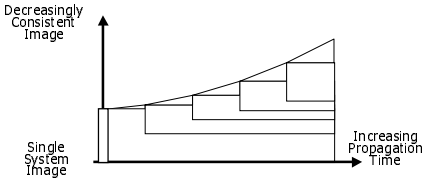
- Introduces "Isolation Levels":
  - **ACID**: Need to select appropriate isolation level
  - **Tradeoff: concurrency vs. single system image**
  - Appropriate level chosen by business requirements
  - Serializable not used in most real-world systems.
    Q: Why not?

3 Multi-User System



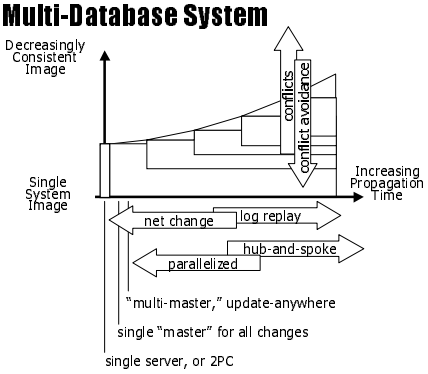
- We already live with inconsistent images:
  - Effects of my transaction now depend, not just on my chosen isolation level, but on <u>other</u> transactions' isolation levels... and even things like their performance / execution speed
  - We live with a real degree of indeterminacy, yet don't lose any sleep... why not? We know how to reason / measure
  - Note: SSI exists even at "read uncommitted" if no conflict

3 Multi-Database System

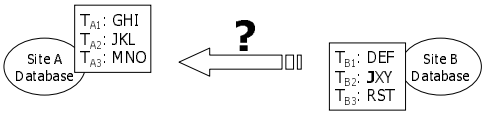


- Introduces "Horizon":
  - **ACID**: Extension of isolation level concept
  - As isolation levels are to multiuser systems, so horizon is to multidatabase systems
  - **Same tradeoff: concurrency vs. single system image**
  - Appropriate level chosen by business requirements

3 Multi-Database System



3 Log Replay vs. Net Change



- Q: How to Replay Site B's Log?
  - T_{B1}: ?
    - Allow? (but could be XD on G = T_{A1})
  - T_{B2}: ?
    - Deny? (not ACID! T_{B2} must be durable)
    - Continue with T_{B3}? (or need to send 'something' back to B?)
    - repl leaves dbs in inconsistent state... actively <u>diverges</u> dbs, instead of converging!
  - T_{B3}: ?
    - Allow? (but could be XD on T_{B2})

3 Replication Transactions

- Major Cases:
  - need to specify transactions replication is to use, using business rules expressed i.t.o. tables/keys
  - records within same table:
    - canonical problem: Debit/Credit (inserts)
    - canonical problem: Account Balance (updates)
  - related records in multiple tables:
    - canonical problem: Invoice/LineItem
    - more generally: object-relational representations
  - inventory level problem (a.k.a. ABM), ticket sales problem (a.k.a. unique-items):
    1. allow update-anywhere, manage exposure with horizons (practical, same reasoning as non-serializable iso. levels)
    2a. activity at only one site (inelegant)
    2b. enter 'requests,' fulfilled by single site (* prop. speed)

4 Agenda

Scope and Objectives

Key Concepts:

1. Motivation: Why Replication?
2. Major Architectural Alternatives
3. Asynchronous Transaction Model

Major Requirements:

4. Conflict Prevention and Handling

5. Data Subset Management Approaches
6. Transformation and Mapping Overview

Summary

4 Motivating Example: The Canonical Customer Table

Name	Addr	City	Zip	Ctry	Pymt	Rating	Limit

- Major Advantages:
  - **Efficiency**: Low (in fact, optimal) network overhead
  - **Ease of Use**: No false collisions
  - **Integrity**: No missed/unreported collisions; data that should change together will change together (func. dependencies)

4 Major Conflict Issues

- Insert Conflicts:
  - key uniqueness, GUIDs
  - fat GUIDs vs. "LUID-like" GUIDs
- Other Single-Record Conflicts:
  - record- vs. field- vs. fragment-level replication
  - aggregated/calculated values
- Multi-Record Conflicts:
  - persistent object-like structures
  - ability to select one competing image, even if no individual records/fragments are in conflict

5 Agenda

Scope and Objectives

Key Concepts:

1. Motivation: Why Replication?
2. Major Architectural Alternatives
3. Asynchronous Transaction Model

Major Requirements:

4. Conflict Prevention and Handling

5. Data Subset Management Approaches

6. Transformation and Mapping Overview

Summary

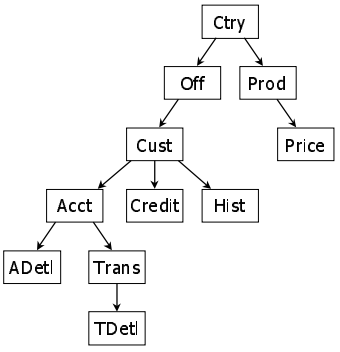
5 Common Approaches

Writing Custom SQL (WHERE clauses):

- per table, per site (NxM)
- not all products support joins

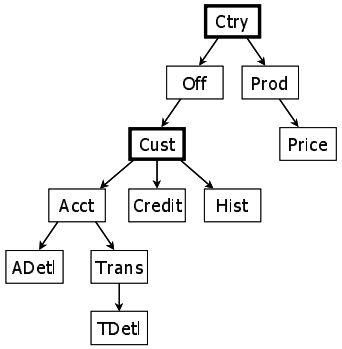
Object-Based Approaches

5 Object-Based Subsetting



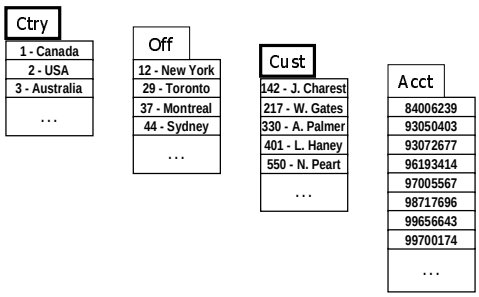
5

Object-Based Subsetting



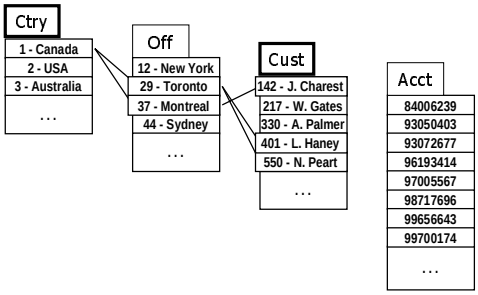
5

Start: No subscriptions



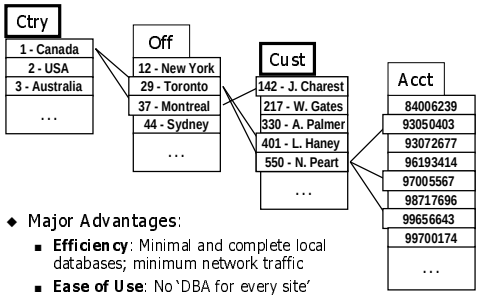
5

Subscribe: Canada



5

Subscribe: Customer #550



- Major Advantages:
  - **Efficiency**: Minimal and complete local databases; minimum network traffic
  - **Ease of Use**: No 'DBA for every site'
  - **Integrity**: Relationally intact slices of information; business rule-driven

6

Agenda

Scope and Objectives

Key Concepts:

1. Motivation: Why Replication?
2. Major Architectural Alternatives
3. Asynchronous Transaction Model

Major Requirements:

4. Conflict Prevention and Handling
5. Data Subset Management Approaches
6. **Transformation and Mapping Overview**

Summary

6

Common Requirements

Basic Field Type Mapping:

- type mapping: Double ⟷ Numeric(12,6)
- type conversion: Varchar(11) ⟷ Integer

Intra-Record Transformation:

- (FirstName,LastName) ⟷ Name

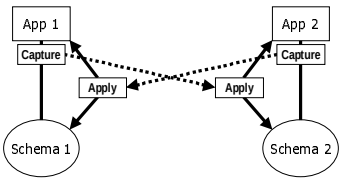
Multiple-Record Transformation:

- rollups, aggregations
- denormalization, star schemas (warehousing)

Note:

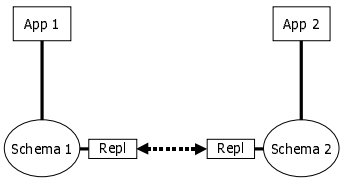
- No inherent reason why all of the above cannot be done bidirectionally

6 Application Integration: Via Middleware (e.g., CORBA, COM+)



- Major Issues:
  - **Cost, Effort**: Intrusive, typically requires significant application changes; highly customized, not easily reusable
  - **Reliability, Integrity**: Data lag; multiple points of failure; the source/target model in another guise

6 Application Integration: Via Replication



- Major Advantages:
  - **Cost, Effort**: Nonintrusive, no application customization; reusable; solves "the important 80%" of integration
    (e.g., have you tried changing flight seating through a "partner" lately?)
  - **Reliability, Integrity**: No data lag; inherently bidirectional; business rule-driven

Agenda

Scope and Objectives

Key Concepts:

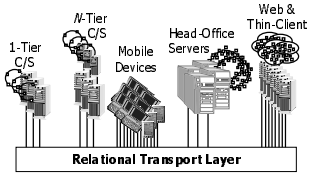
1. Motivation: Why Replication?
2. Major Architectural Alternatives
3. Asynchronous Transaction Model

Major Requirements:

4. Conflict Prevention and Handling
5. Data Subset Management Approaches
6. Transformation and Mapping Overview

Summary

An "Ideal" Distributed System



- no new relational concepts **(e.g., normalization)**; business-rule driven
- every site a peer; update-anywhere "just as good as head office"
- every site autonomous, all transactions local; no point of failure
- automatic, transparent; just add a database wherever needed
- all user's favorite tools: Java/VB/HTML/X/C++/Perl/Delphi/..., 1-tier/N-tier/Web/Win32/Unix...; OLTP/OLAP/DSS/reporting...