Inconsistency and Outconsistency

SYSTEMATIC THOUGHT LEADERSHIP FOR INNOVATIVE BUSINESS

Shel Finkelstein, SAP Labs
Pat Helland, Independent

What's the Test of a First-Rate Data Management System?

The test of a first-rate intelligence is the ability to hold two opposed ideas in the mind at the same time, and still retain the ability to function.

F. Scott Fitzgerald "The Crack-Up" (1936)

Outconsistency (outward consistency) describes which inconsistencies each app can see, and how those inconsistencies are addressed

Outline

- What is Consistency/Inconsistency?
- What is Outconsistency?
  - How do applications deal with Outconsistency?
- Summary

Are These Events Consistent?

- Temperature is 42°C
- Temperature is 55°C
- Temperature is 72°C
- Temperature is 90°C

How About These Events?

- John reports temperature is 42°C in Palo Alto at 7am on September 2, 2011.
- John reports temperature is 55°C in Palo Alto at 11am on September 2, 2011.
- Mary's thermostat reports temperature is 72°C at 11am on September 2, 2011, in Los Angeles in her home with air conditioner running.
- Mary's car temperature gauge reports temperature is 90°C outside at 11am on September 2, 2011 in Los Angeles when she leaves her home.

Business examples are often much less obvious than this one.

Event Details and Provenance

- Measurement (or sighting or other occurrence)
  - Who/what measured/saw? What was measured/seen?
  - How? Where? When? Why? Units?
- Report of measurement
  - Who/what reported? What exactly was reported?
  - How? Where? When? Why?
- Other aspects
  - Confidence, reliability, completeness

"Transaction Processing: Concepts and Techniques", Gray and Reuter

- "A transaction is a collection of operations on the physical and abstract application state." *Includes external actions.*
- ACID properties: Atomicity, **Consistency**, Isolation, Durability
  - "Consistency: A transaction is a correct transformation of the state. The actions taken as a group do not violate any of the integrity constraints associated with the state."
  - *If each transaction preserves constraints and execution is serializable, any transaction sequence preserves constraints.*
- "A transaction processing system provides tools to ease or automate application programming execution and administration."
  - *Hmmm, applications.*

Inconsistencies

- Integrity constraints violations
- Logical impossibilities
- Replication issues
- Limited transactional isolation

Inconsistencies: Integrity Constraint Violations

- Violations
  - Within a column, such as domain constraint
  - Within a row, such as impossible address/state/zipcode
  - Across rows, such as referential integrity, unique keys or “master data” constraints
  - Business rules, such as HR requirements about people transfers
  - Over time, such as strictly increasing serial numbers
  - With external actions, such as cash drawer paying wrong $

Inconsistencies: Logical Impossibilities

- Contradictory data in database
  - In current state, e.g., unrealizable design
  - Over time, e.g., location change that can't occur
  - With real world, e.g., inventory data vs actual inventory
  - Across multiple data sources, e.g., multiple business statuses
- Erroneous, unsound, incomplete, e.g. trip itinerary with missing leg
- Unanticipated or extremely unlikely, e.g., Antarctic ozone hole readings (may be apochryphal)
- Violating governance rules, e.g., unaudited data accesses
- Violating business rules, e.g., contract accepted with inappropriate terms

Inconsistencies: Replication Issues

- Asynchronously feeding/streaming data between different data management systems or data caches
  - Streaming database
  - ETL-based data replication
  - Disaster protection protocols based on log shipping
- Replication protocols within a distributed data management system without single-copy semantics
  - Loose consistency; CAP issues
  - Eventual convergence
- Process flows transmitting data between different management systems
  - Order-entry systems capturing orders, which are sent to fulfillment systems handling scheduling/delivery
  - Eventual consistency and apologies

Inconsistencies: Limited Transactional Isolation

- Databases run without Serializability as default isolation
  - Read committed/cursor stability
  - Snapshot isolation
- Well-know inconsistencies are possible
  - Constraint: A<=B, A is 0, B is 100; 0<=100 ✓
  - T1 reads A and B, and changes A to 60; 60<=100 ✓
  - T2 reads A and B, and changes B to 20; 0<=20 ✓
  - But now A is 60, B is 20; A>B, so constraint is violated ✗
  - Why isn't this a problem in practice?
- See "A Theoretical Study of Snapshot Isolation", EDBT 2010, for a surprisingly example

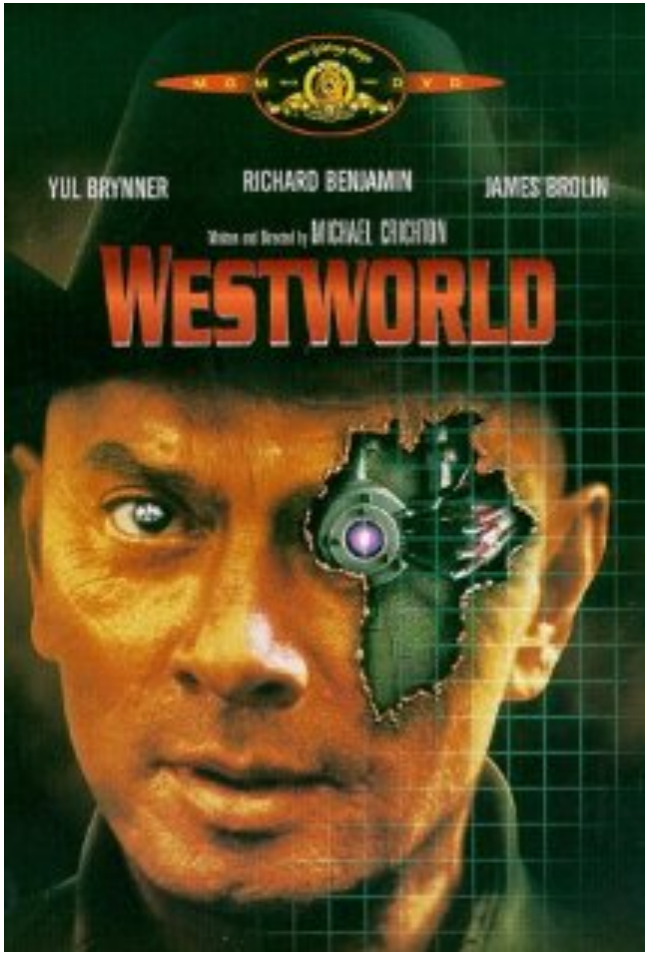
Outline

- What is Consistency/Inconsistency?
- What is Outconsistency?
  - How do applications deal with Outconsistency?
- Summary

In An Ideal World …

- There are no data entry errors
  - … or at least error are detected and corrected immediately
- Applications share a common view of consistency
  - … and no transaction violates that common view
  - … and apps/transactions can co-exist intelligibly
- Inconsistencies within and across data sources never happen
  - … or they don't matter
  - … or they're not noticed
  - … or they're automatically fixed without lasting harm
  - … or they're manually fixed, including side effects
- The Closed World Assumption is valid
  - Data reflects the Real World accurately and completely

But We're Not In A Ideal World in which Nothing Can Go Wrong



©Metro-Goldwyn-Meyer Company

Consistency Assumptions

- Every app makes consistency assumptions about its data sources and the world
  - It may be able to tolerate certain inconsistencies but not others.
  - Example: Inventory application can handle possibility that inventory in the bin is less than inventory in the database.
- Different apps may have different consistency assumptions, even if their models are defined on the same data
  - Example: Sales app allows entry of sales opportunity for a company that's not an approved partner, but Shipping app won't see that data and ship until company is an approved partner.

Outconsistency

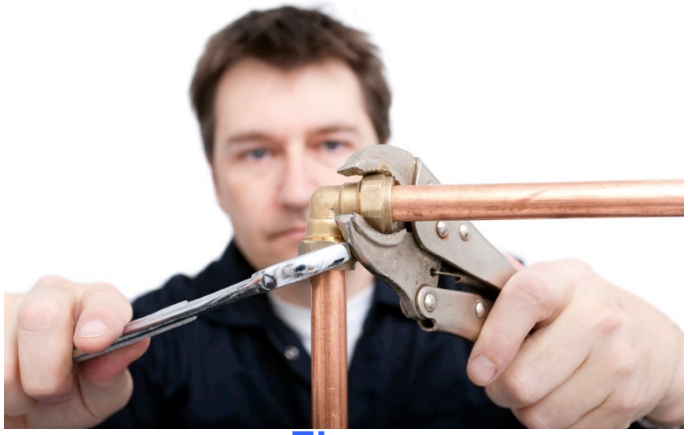
- Outconsistency refers to the outwardly consistent view of the data and the world that an application expects and handles
  - Like other views, outconsistent views might be materialized, but need not be
- Outconsistency is more than just a view; it's also a regimen for addressing inconsistencies
- Outconsistency helps applications operating on the same data play nicely with each other
  - Helps isolate applications from each other's effects
  - It also helps address application lifecycle issues, including customization, composition, integration and app/schema upgrade

How Do Database Apps Provide Outconsistency?

Prevent

Tolerate

Ignore



Fix

How Do Database Apps Address Outconsistency?

- Prevent inconsistent data
- Tolerate inconsistent data
- Ignore inconsistent data
- Fix inconsistent data

In practice, a combination of techniques is used

How Apps Address Outconsistency: Prevent Inconsistent Data

- Outconsistent view is identical to the data, and there are no anticipated inconsistencies
- Some approaches
  - Include strong integrity constraints in schema/metadata
  - Check business rules at commit
  - Determine Transaction Intent using read-only data with loose isolation
    - Execute separate one-message transaction corresponding to Intent
    - Examples in "Transactional Intent", CIDR 2011
- Challenges
  - Factoring models for constraints/rules across different apps
    - … while allowing apps flexibility in what they see and do
  - Executing efficiently
  - Handling unanticipated inconsistencies
  - Supporting app/schema lifecycle

How Apps Address Outconsistency: Tolerate Inconsistent Data

- Outconsistent view is identical to the data, but there are anticipated exceptions
- Some approaches
  - Null values in columns (unknown, not applicable, …)
  - Partial configuration designs where may be incomplete or unsound
  - Decisions based on collective information, confidence, learning
  - Incremental schema evolution
  - Case statements with robust exception handling
- Challenges
  - Programming tolerantly is complex and imperfect
  - Governing apps to ensure that they are all tolerant
  - Supporting app/schema lifecycle

How Apps Address Outconsistency: Ignore inconsistent Data

- Outconsistent view only includes data consistent for the app
- Some approaches--Restrict app's data view to:
  - Limited domains, e.g., critical columns are non-null
  - Business objects correlating to master data, e.g., with valid suppliers for purchase orders
  - Integrated schemas between different data sources only show data with matches
- Challenges
  - Avoiding unpleasant surprises for human or programmatic users
  - Ignoring exception cases doesn't address them
    - Kicking the can down the road
  - Interaction with security authorization requirements

How Apps Address Outconsistency: Fix Inconsistent Data

- Applications detect and fix inconsistencies internally so that they have an outconsistent view
  - May save fixes for other instances of same or different apps
- Some approaches
  - Interpolation and Extrapolation
  - Data cleansing applications and services, with alerts
  - Renewal processes, such as SAP's APO
  - Reconciliation and Apologies; see CIDR 2007, Life beyond Distributed Transactions: an Apostate's Opinion
- Challenges
  - Doing the fixes
  - Coping with the inconsistencies until they're fixed
  - Supporting app/schema lifecycle

Events and Reports and Decisions

Claim: Data/transaction processing consists only of:

- Events
  - From users, sensors, other systems
- Reports
  - Including Business Intelligence, OLAP and predictions
- Decisions
  - Actions taken based on those decisions

What else is there?

If this factoring is correct, should we write apps based on it?

- Use outconsistent views and regimens so that apps co-exist intelligibly
- … not just “now”, but as apps and schemas evolve

Summary

- **Inconsistency** comes from many sources
  - That's well-understood in practice
  - Should be better described, and handled more systematically
- **Outconsistency** describes which inconsistencies each app can see, and how those inconsistencies are addressed
  - Prevent, Tolerate, Ignore, Fix inconsistent data
  - App/schema lifecycle is challenging, and needs more attention
- **"Inconsistency and Outconsistency"** is just a start
  - More work needed on apps and schema/data together
  - Boundary between apps and schema/data is surprisingly fluid

Inconsistency and Outconsistency

Shel Finkelstein, SAP Labs

Pat Helland, Independent

With thanks to Thomas Heinzel, Rainer Brendle and Heinz Roggenkemper at SAP