

# Desperate Table Designs

Susan Lawson



Yevich, Lawson & Assoc. Inc.  
2743 S. Veterans Pkwy PMB 226  
Springfield, IL 62704

[www.ylassoc.com](http://www.ylassoc.com)  
[www.db2expert.com](http://www.db2expert.com)

---

IBM is a registered trademark of International Business Machines Corporation.  
DB2 is a trademark of IBM Corp.

© Copyright 1998-2008, YL&A, All rights reserved.



© YL&A 1999-2008

***Disclaimer PLEASE READ THE FOLLOWING NOTICE***

- The information contained in this presentation is based on techniques, algorithms, and documentation published by the several authors and companies, and in addition is the result of research. It is therefore subject to change at any time without notice or warning.
- The information contained in this presentation has not been submitted to any formal tests or review and is distributed on an "As is" basis without any warranty, either expressed or implied.
- The use of this information or the implementation of any of these techniques is a client responsibility and depends on the client's ability to evaluate and integrate them into the client's operational environment.
- While each item may have been reviewed for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere.
- Clients attempting to adapt these techniques to their own environments do so at their own risks.
- Foils, handouts, and additional materials distributed as part of this presentation or seminar should be reviewed in their entirety.



© YL&amp;A 1999-2008

***Abstract***

This presentation covers new, bold, creative solutions to achieve high availability and high performance. New challenges mean thinking outside the old rules. We also look at how to synergize creative table designs with applications to achieve our goals.

**Objectives:**

- Discuss some new innovative ways to create tables
- Discuss new challenges and opportunities for index design
- Discuss how to integrate designs with applications for best performance and availability
- Discuss how to use new features of DB2 to solve problems
- Discuss ways to think differently about designs and see examples from real implementations

This presentation was developed by Susan Lawson and Dan Luksetich of YLA. They can be reached at [Susan\\_Lawson@ylassoc.com](mailto:Susan_Lawson@ylassoc.com) and [Dan\\_Luksetich@ylassoc.com](mailto:Dan_Luksetich@ylassoc.com) respectively.



© YL&amp;A 1999-2008

### *Outline – Stories about Desperate Table Designs*

- **24X7 Does Not Mean 24X7**
  - It Means Hiding Your Outages
- **Providing for Logical Recovery from Data Corruption**
- **Partial Denormalization for Performance**
  - Indicator Columns for “Denormalization Light”
  - “Full Partial Denormalization”???



© YL&A 1999-2008

## ***24X7 Does Not Mean 24X7***

**Hiding Your Outages are Key to  
Apparent High Availability**

### ***High Availability Requirement for Summary Table***

- **High Profile Retailer**
  - Online Sales
  - 24X7 Television Show
- **Management Gets a Summary Every 15 Minutes**
  - How Many Orders are being Taken
  - How Many Credit Cards have been Processed
  - How Many Products placed in Boxes
  - How Many Boxes placed on Trucks
  - How Many Trucks in Shipment
  - How Many Products Delivered to Customers
- **The “Big Guy” Has a Screen in His Office**
  - Information Constantly Displayed
  - Screen Cannot Go Blank!



© YL&A 1999-2008

### ***High Availability Summary Table Challenge***

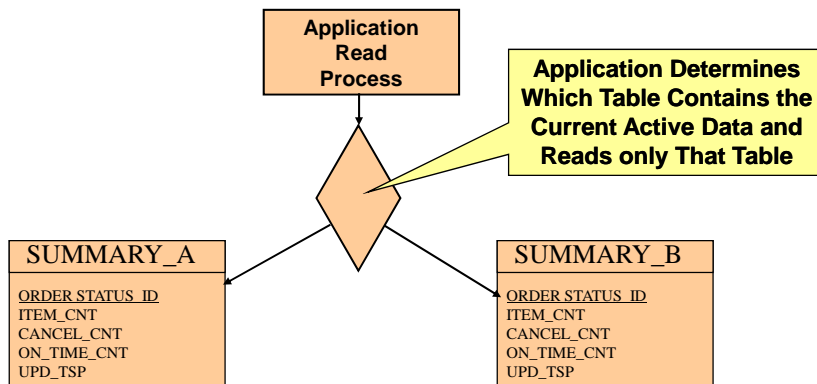
- **Problem**
  - Summary Table Cannot be Populated in 15 Minutes using SQL
  - Data can be Gathered in Less than 15 Minutes via Summary Query Unload
  - Complicated Summary SQL Joins make Triggers Undesirable
  - LOAD REPLACE of Summary Table Data Results in an Outage
- **Solution**
  - Two Summary Tables and Switching Logic
    - Application Reads the Active Table
    - LOAD Replaces Inactive Table



© YL&A 1999-2008

### Table Switching Example

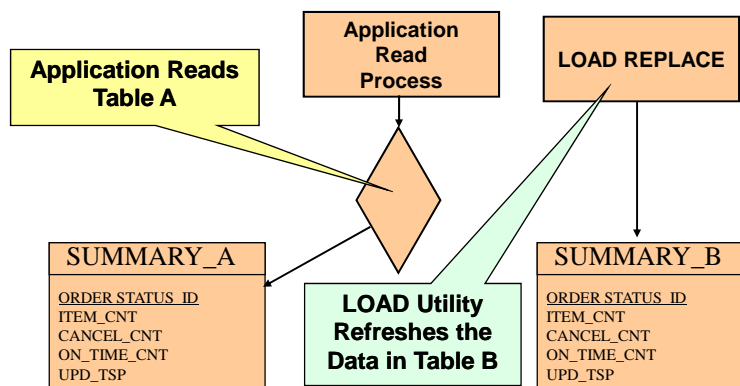
- **Table Switching is an Application Based High Availability Solution**
  - Very Flexible
  - Has to be Coded in Application
  - Can Use DB2 Database Features to Simplify Application Design



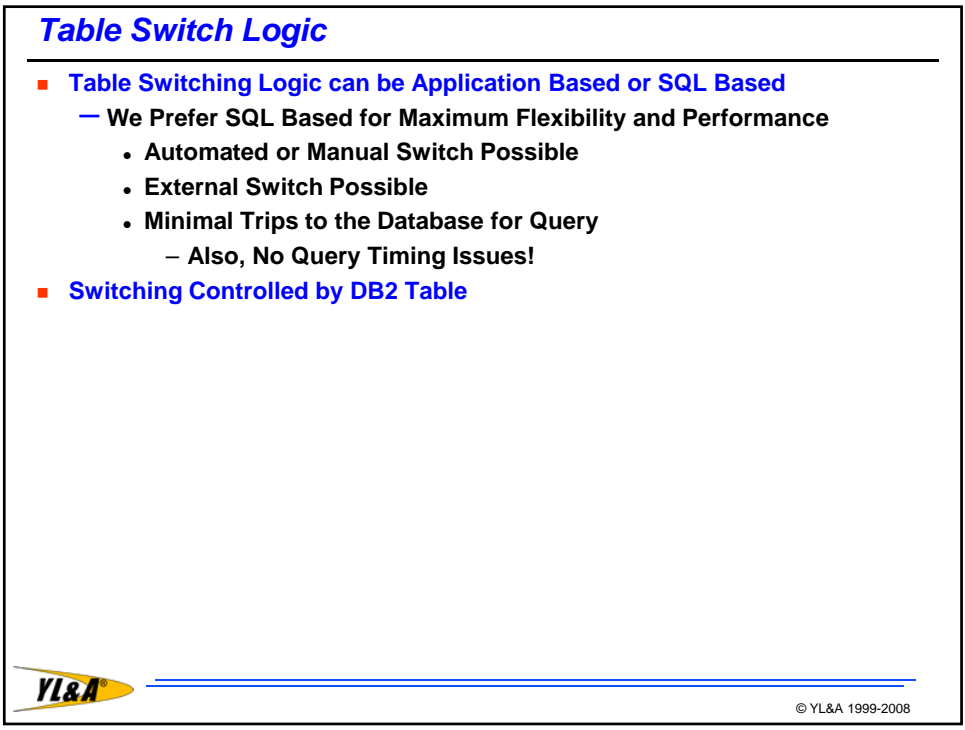
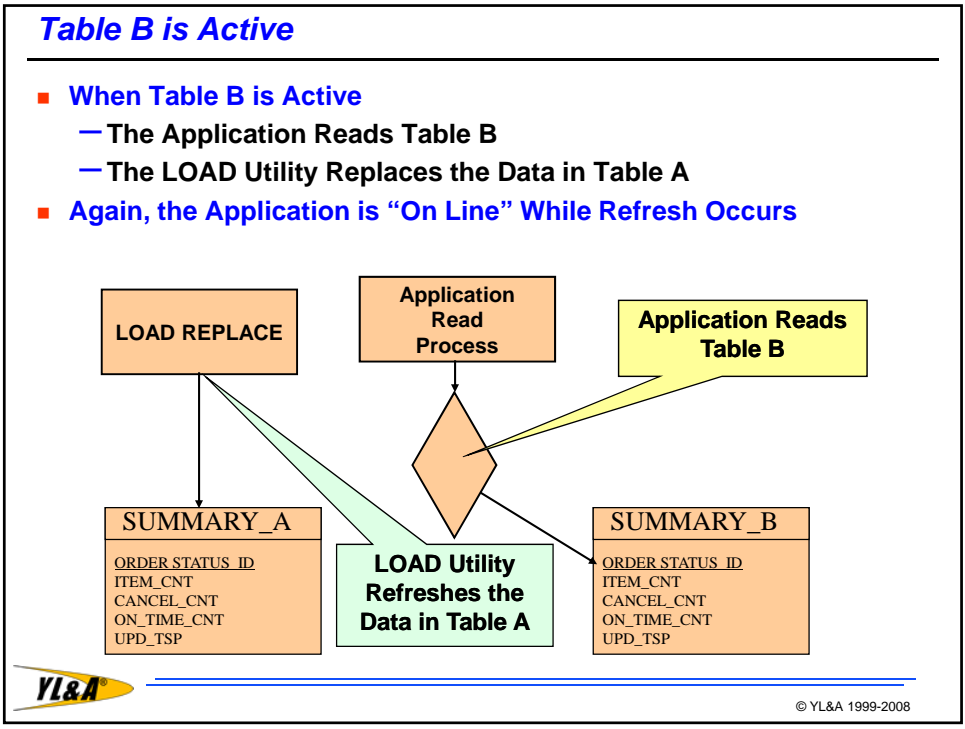
© YL&A 1999-2008

### Table A is Active

- **When Table A is Active**
  - The Application Reads Table A
  - The LOAD Utility Replaces the Data in Table B
- **The Application is "On Line" While Refresh Occurs**



© YL&A 1999-2008





### Switching Inside a Query

■ **Switching Inside a Query**

- Utilizes a During-Join Predicate
- Reduces Application Logic
- Reduces Potential Outages
  - Read of Switch and Read of Active Table in One Statement Instead of Two

```
SELECT COALESCE(A.ORDER_STATUS_ID, B.ORDER_STATUS_ID)
      ,COALESCE(A.ITEM_CNT, B.ITEM_CNT)
      ,COALESCE(A.CANCEL_CNT, B.CANCEL_CNT)
      ,COALESCE(A.ON_TIME_CNT, B.ON_TIME_CNT)
FROM   SWITCH_TABLE AS SW
LEFT JOIN
      SUMMARY_A AS A
ON SW.ACTIVE_TABLE = 'A'
LEFT JOIN
      SUMMARY_B AS B
ON SW.ACTIVE_TABLE = 'B';
```

The first non-null value is returned. The outer join will supply nulls for the table that is not "active"

SUMMARY\_A will only return the summary data when this join predicate is true

SUMMARY\_B will only return the summary data when this join predicate is true



© YL&A 1999-2008

### Table Switching Using Clone Support in DB2 9

- **DB2 9 simplifies this type of high availability applications**
  - A clone can be created for a table
  - The EXCHANGE statement does the switch
- **The refresh process is simplified**
  - The clone table is truncated
  - Then fresh data is inserted
  - Then the exchange – online LOAD REPLACE!

```
ALTER TABLE SUMMARY_A
ADD CLONE SUMMARY_B;
```

```
SELECT A.ORDER_STATUS_ID,
A.ITEM_CNT, A.CANCEL_CNT,
A.ON_TIME_CNT
FROM SUMMARY_A AS A;

TRUNCATE SUMMARY_B;
INSERT INTO SUMMARY_B
SELECT ....;
EXCHANGE DATA BETWEEN
SUMMARY_A AND SUMMARY_B;
COMMIT;
```

Returns the data from the base table. No complex SQL required!

This script refreshes the data in the clone table, and then exchanges the clone and base tables. Now the base has become the clone and the clone the base! No outage! No switch table required.



© YL&A 1999-2008

## *Providing for Logical Recovery from Data Corruption*

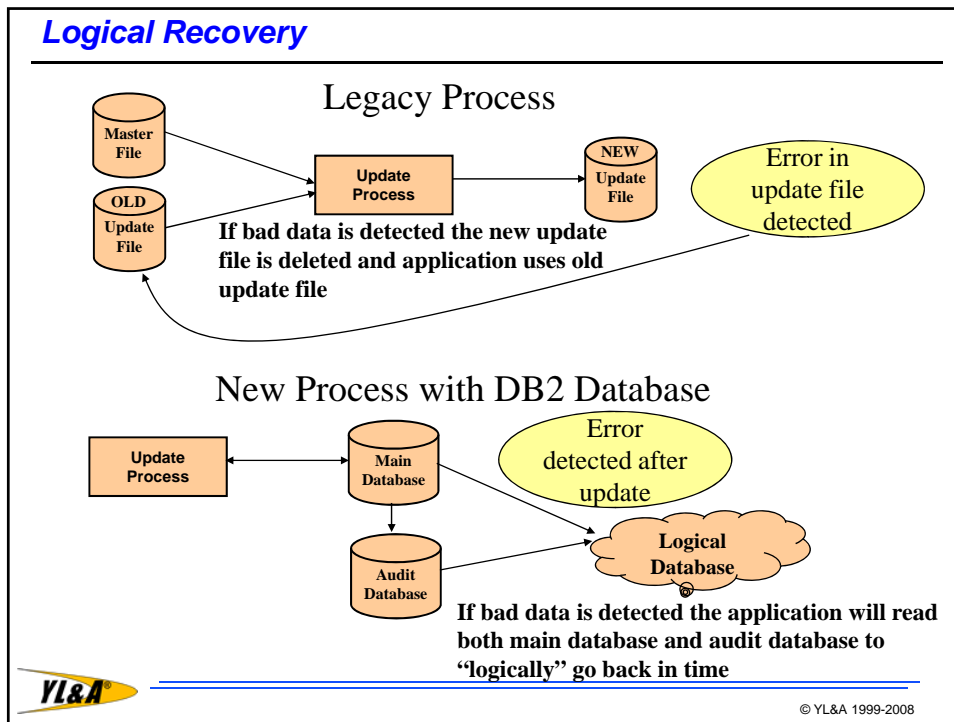
**Remove Bad Data Instantly  
Without an Outage**

### *The Situation*

- **Migrating a Database From a Flat File Technology to DB2**
- **Current Application Update Process Creates Updates in an Update File**
  - Update Files Are Concatenated Before the Main File
  - If Application Corrupts the Update File
    - It is Deleted
    - Updates are Instantly Removed
- **DB2 Will Have to Provide This Exact Functionality**
  - However, We Update DB2 Directly
  - We Do Not Want to Create an Update Database
  - We Cannot Use RECOVER to Remove Application Errors
    - Will Create an Outage
  - We Cannot Use SQL to Remove Application Errors
    - Will Result in “Dirty” Updates During Removal Processing
  - We Cannot Use an Application Process to Remove Errors
    - Each Error Can be Different
    - We Must Remove the Errors Quickly
- **The Solution is an Audit Database and “Logical Recovery”!!!**



© YL&A 1999-2008



### Main Database and Audit Database

■ **The Audit Database Contains Before Change Images of All Tables**

— When Main Table Changes (DELETE or UPDATE) Audit Table Gets and Old Data

- For Updates
  - STRT\_TSP in Audit Table is UPD\_TSP from Main Table Before the Update
  - END\_TSP in Audit Table is UPD\_TSP from Main Table After the Update
- For Deletes
  - STRT\_TSP in Audit Table is UPD\_TSP from Main Table Before the Delete
  - END\_TSP in Audit Table is the CURRENT TIMESTAMP When the Delete is Executed

MAIN_TABLE	
CUST_ID	INTEGER
DATA_COL	CHAR(10)
UPD_TSP	TIMESTAMP

AUDIT_TABLE	
CUST_ID	INTEGER
STRT_TSP	TIMESTAMP
DATA_COL	CHAR(10)
END_TSP	TIMESTAMP

### Triggers to Replicate Updates and Deletes

- **These Triggers Replicate Before Images to the Audit Tables**
  - Appropriate Timestamps are Replicated or Generated
    - For Updates the Start and end Timestamps Come from the Before and After Images of the Main Table
    - For Deletes the Start Timestamp Comes from the Main Table and End Timestamp Comes from the CURRENT TIMESTAMP

```
CREATE TRIGGER UPDTRG2
AFTER UPDATE ON MAIN_TABLE
REFERENCING OLD AS OLDROW
NEW AS NEWROW
FOR EACH ROW MODE DB2SQL
INSERT INTO AUDIT_TABLE
VALUES (OLDROW.CUST_ID, OLDROW.UPD_TSP, OLDROW.DATA_COL , NEWROW.UPD_TSP);
END!
```

```
CREATE TRIGGER PPOCSSR.DELTRG1
AFTER DELETE ON MAIN_TABLE
REFERENCING OLD AS OLDROW
FOR EACH ROW MODE DB2SQL
INSERT INTO AUDIT_TABLE
VALUES (OLDROW.CUST_ID, OLDROW.UPD_TSP , OLDROW.DATA_COL , CURRENT TIMESTAMP);
END!
```



© YL&amp;A 1999-2008

### Result of Trigger Usage

- **Pros**
  - Instant Replication of the Data to the Audit Tables
    - Allows for Instant Logical Recovery
  - No Application Programming Required for Replication
- **Cons**
  - Additional Complexity to Database
    - Especially for Migrations, Changes, and Tests
  - More Overhead for the Update Processes
    - Trigger Invocation
    - Inserts into Audit Tables
  - Potential Availability Impact
    - Application Dependent Upon Audit Tables for Normal Operations



© YL&amp;A 1999-2008

### Logical Recovery Table

- The Logical Recovery Table is the Heart of the Logical Recovery**
  - Initiates a Logical Recovery
    - Normally Empty
    - Timestamp Inserted to Indicate Logical Recovery
      - The Timestamp is the "As Of" Time
  - Objective of Logical Recovery
    - All Table Access Data as it Appeared as of the "As Of" Time
    - Keep Availability 100% During a Backout Process
- Database Interface Program Always Read Logical Recovery Table**
  - It a Row Exists then Logical Recovery Mode is Activated
    - Application Interface then Read Logical Recovery Views
    - All Other Process Remains the Same
    - Updates are Prohibited

```

INSERT INTO LOGRCVRY
VALUES ('2008-05-21-01.00.00.000000')
    
```

```

SELECT RCVRY_TSP
FROM LOGRCVRY
    
```

© YL&A 1999-2008

### Logical Recovery

- Each Table in the Database Has a Corresponding View**
  - The View UNIONS each Main Table and its Audit Table
  - The Tables are Joined to the Logical Recovery Table
- During Logical Recovery Data Comes from Either the Main Table or Audit Table Dependent upon the Logical Recovery Timestamp**
  - If Last Update to the Base Table was Prior to the Recovery Timestamp then it is Used
  - If the Logical Recovery Timestamp is During the Active Period of the Audit Data then that Audit Table Data is Used

```


CREATE VIEW RCVRY_TBL
( CUST_ID, DATA_COL, UPD_TS )
AS
SELECT CUST_ID, DATA_COL, UPD_TS
FROM MAIN_TABLE BASETB
INNER JOIN LOGRCVRY AS RCVRY1
ON BASETB.UPD_TS <= RCVRY1.RCVRY_TS
UNION ALL
SELECT CUST_ID, DATA_COL , STRT_TS
FROM AUDIT_TABLE AUDITB
INNER JOIN LOGRCVRY AS RCVRY2
ON AUDITB.STRT_TS <= RCVRY2.RCVRY_TS
AND AUDITB.END_TS > RCVRY2.RCVRY_TS;
    
```

© YL&A 1999-2008

### Logical Recovery Backout Process


- **Once the Database is in Logical Recovery Mode the Bad Data Must be Backed Out and Old Data Restored**
  - This is Done While the Database is Still Online
  - Backout Scripts are Employed
  - The Logical Recovery Table Drives the Process

Step1: QUIESCE	Ultimate Recovery Point if Backout Mistakes Happen
Step2: Lock Affected Partitions in Exclusive	Database API Reads WITH UR so No Lock Contention
Step 3: Delete Data from Base Table if Updated after Logical Recovery Timestamp	The Logical Recovery Timestamp was the Point in Time Where the Data was Last Good (Before the Corruption)
Step 4: Repair Base Table Data by Copying Data from Audit Table that was Effective "As Of" the Logical Recovery Timestamp	Normal Operations can now Resume, and the Update can be Rerun Following any Program or Input Fixes
Step 5: Delete Data from Audit Table if Updated after Logical Recovery Timestamp	
Step 6: Remove Logical Recovery Timestamp	

 © YL&A 1999-2008

### Impacts of Logical Recovery Mode

- **Read Performance is Seriously Degraded During a Logical Recovery**
- **Additional Complexity is Introduced**
  - For Update Processing
  - For any Readers
  - When Going in or Coming Out of Logical Recovery Mode
- **Benefits**
  - Ability to Surgically Recover
    - Works Like Magic!
  - Full Availability During a Backout
  - Very Little Additional Application Programming

 © YL&A 1999-2008

## ***Partial Denormalization for Performance***

### **Indicator Columns for “Denormalization Light”**

#### ***The Situation***

---

- **Migration from Legacy Application to DB2 Database**
  - Single Record Design to Several Tables
  - Performance Expected to be “Same or Better than Legacy”
- **Normalized Design Important**
  - Great for Future Applications
  - Must be Retained for Business Flexibility
- **Performance is Unacceptable**
  - Yes, it Does Take Longer to Read Many DB2 Tables Versus one VSAM File



© YL&A 1999-2008

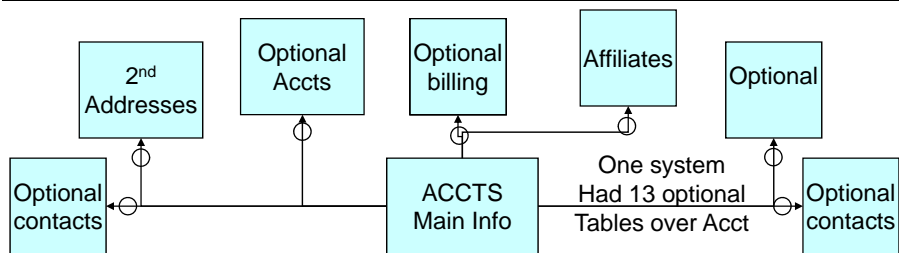
**The Solution**

- **Do Not Denormalize**
  - Normalization is One Reason for the Move to a Relational Database
  - Future Development will be More Expensive
- **Try Other Solutions**
  - Increase Buffers
  - Minimize Readers
    - Add Special Codes from Legacy to Request Limited Information
  - Add or Modify Indexes
    - Make Frequent Queries Index Only
  - Modify Clustering to Match Major Processes
  - And Many Others!
- **Normal Solutions Don't Succeed?**
  - Try Indicator Columns
  - AKA "Denormalization Light"
  - This Once Again Takes Advantage of During Join Predicates



© YL&A 1999-2008

**Optional Tables: Original Design**



Original table design

Account no	names	addr	details
------------	-------	------	---------

```

SELECT columns
FROM ACCTS A LEFT JOIN OPT1 O1
ON A.ACCT_NO = O1.ACCT_NO
LEFT JOIN OPT2 O2
ON A.ACCT_NO = O2.ACCT_NO
.....
WHERE A.ACCT_NO = 1
    
```

**DB2 must join to the optional tables. In many cases the join will not result in a match.**



© YL&A 1999-2008

### Optional Tables: New Design

New table design      Optional relationship indicators

Account no	names	addr	details	X	X		X	X
------------	-------	------	---------	---	---	--	---	---

```

SELECT columns
FROM ACCTS A LEFT JOIN OPT1 O1
ON A.ACCT_NO = O1.ACCT_NO
AND A.FLAG1 = 'Y'
LEFT JOIN OPT2 O2
ON A.ACCT_NO = O2.ACCT_NO
AND A.FLAG2 = 'Y'
.....
WHERE A.ACCT_NO = 1
    
```

**DB2 will only join to optional tables when ON clause is true. Avoids unnecessary joins when optional data does not exist.**

YL&A © YL&A 1999-2008

### Impacts of Denormalization Light

- **Readers Must Use Indicator Columns for Performance**
  - In During Join Predicates
    - DB2 Join Will almost always Outperform a Program Join
  - In IF-THEN Logic After Initial Read
- **Updaters Must Care for Indicator Columns**
  - Broken Relationships Between Indicators and Tables can Result in Anomalies
  - Additional Performance Impact to Updaters to Maintain Indicators

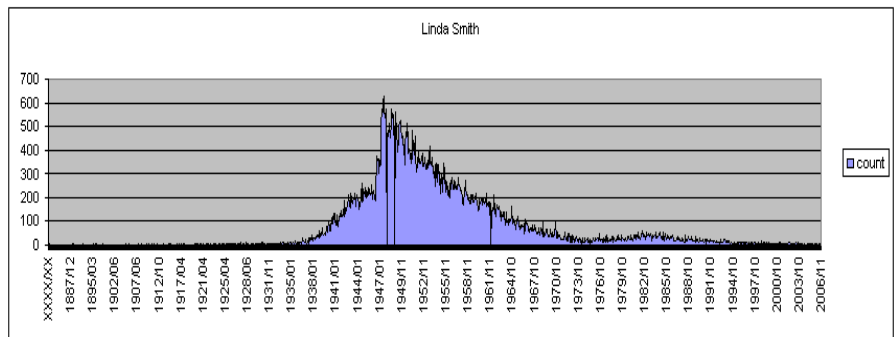
YL&A © YL&A 1999-2008

# *Partial Denormalization for Performance*

## Full Partial Denormalization AKA “The Pineapple Martini Design”

### *The Situation*

- Migration from Legacy Data Store to DB2
- No Application Rewrite Allowed for Name Search
- Current Application Name Search Process Very Fast
  - Single Record Read
- Name Search Using DB2 is Fast for Only Some Names
  - Two Tables Searched
- Application Will Eventually be Rewritten for Better Name Search in DB2
  - However, We Need Performance Today!!!



© YL&A 1999-2008

**The Situation**

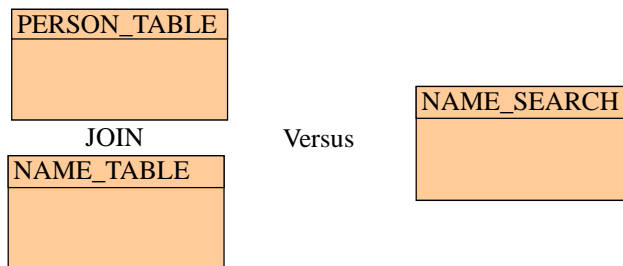
- **Name Search is Driven by a Simple Key**
  - Soundex
  - First Four Characters of First Name
  - Year and Month of Birth
- **Name Search is Fast for Uncommon Names**
  - L232, DANI, 1963, 10 is less than 100ms
    - There is Only 1 of me!
  - L250,SUSA, <content edited>, 04 is less than 100ms
    - Only 1 Susan Lawson Born during that Month!
  - S530,LIND, 1947, 07 is about 3.5 seconds
    - Over 600 Linda Smiths Born during that Month!
- **Name Searches over Multiple Months for Common Names Took up to 10 Minutes!**
  - This Completely Blew Our SLA
  - We Could Not Change the Key!



© YL&A 1999-2008

**Attempts to Make Name Search Faster for 1 Billion Names**

- **Increase Buffers**
  - Not Enough Memory for 1 Billion Names
- **Put All Data Needed into Indexes**
  - Don't Want to add an Extra Index
- **Denormalize**
  - Not Good for the Future
  - Still Not Fast Enough
    - Denormalized Table is Extremely Large
  - Huge Table Size Increases Maintenance Problems



© YL&A 1999-2008

**The Solution (Invented over a Pineapple Martini)**

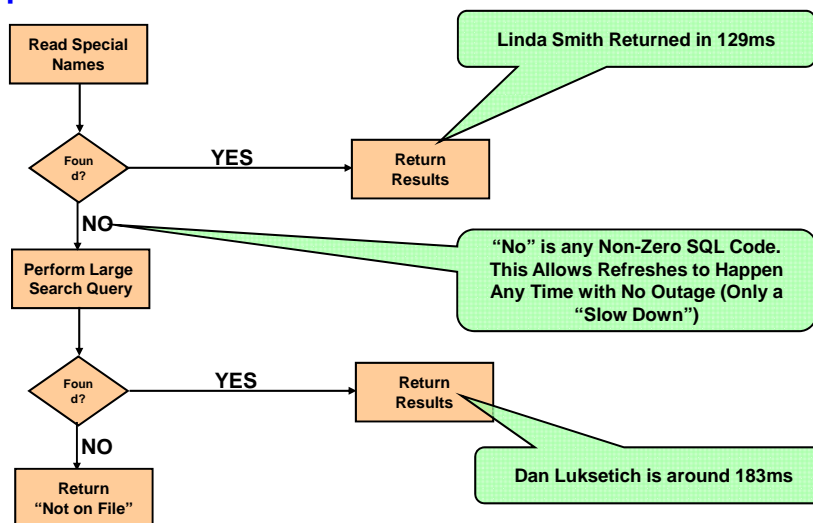
- **Partial Denormalization of Two Main Tables for a Special Name Search Table**
  - This is Done for Only the Common Names
  - Analysis of the Data
    - Regular Query Would Perform Well with Under 50 Keys per Month
      - Soundex
      - First 4 Letters of First Name
      - Year and Month of Birth
    - So We Only Need Keys with 50 or More Occurrences per Month
      - 600,000 Key Values
      - 30,000,000 Rows of Data
- **A Test Was Run**
  - The Special Names Search Table Was Accessed First
  - If Nothing is Found in the Special Names Table Then the Regular Name Search Query is Run
  - Result: All Queries are Subsecond!
- **PROBLEM!**
  - How Do We Maintain the Special Names Search Table?



© YL&A 1999-2008

**New API Access for Name Search**

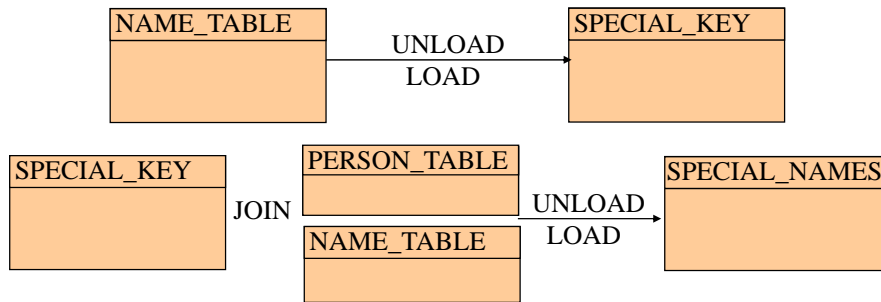
- **The Database API Program was Modified to Check “Special Names” First**



© YL&A 1999-2008

**Populating Special Names via Refresh**

- **A Special Keys Table is Created**
  - This Table Contains the 600,000 Keys that are Common
  - The Keys can be Unloaded from the PERSON and PERSON\_NAME Tables Anytime for Keys That Occur More than 50 Times per Month
- **The Special Keys Table Can Then be Used to Repopulate The Special Names Table via UNLOAD and LOAD**
  - The Application will Revert to the Normal Query When the Special Names Query Returns a SQLCODE -904
- **This Process will Run Once per Year or On Demand**

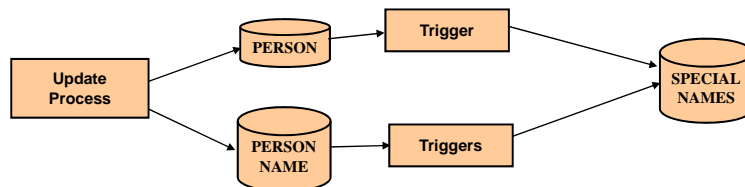


© YL&A 1999-2008

**Maintaining Special Names Daily with Triggers**

- **This is a Temporary Solution**
  - We Don't Want Programming Changes
  - We Can't Run the Refresh Process Every Night
    - 6 Hours Elapsed
  - Update Process Runs Every Night in Batch
- **DB2 Triggers are the Solution to Nightly Maintenance Issue**
- **We Need 5 Triggers**
  - PERSON Table UPDATE
  - PERSON\_NAME Table INSERT
  - PERSON\_NAME Table DELETE
  - 2 PERSON\_NAME Table UPDATE Triggers

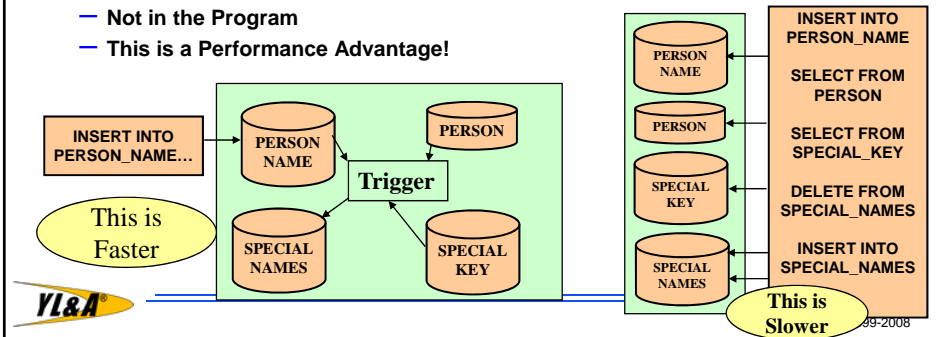
We don't need a PERSON insert or delete trigger because that is handled by the PERSON\_NAME triggers and DB2 RI



© YL&A 1999-2008

### Result of Trigger Usage

- **Special Names Table Allows Us to Meet our Service Level Agreement**
- **Application Does Not have to Update Special Names Table**
  - Triggers Take Care of That
  - No Extra Programming
- **When Name Search Process is Eventually Rewritten**
  - We DROP the Name Search Table
  - We DROP the Triggers
  - Not One Line of a Program Has to Change
  - Months of Programming Changes Avoided!
- **Also, Denormalization Process is Handled within the Database**
  - Not in the Program
  - This is a Performance Advantage!



### Impacts of Full Partial Denormalization

- **Triggers for Updates**
  - Firing Triggers When Names Change Impacts Performance
  - Additional Complexity to Database Maintenance
  - Trigger Logic is Complicated
- **Need Refresh Processes**
  - Need Scripts/Processes for Periodic Refresh
- **Additional Tables Mean More Database Maintenance**
- **Benefits**
  - Meet SLA
  - Minimum Application Changes
    - Implementation Took Days
  - Can Drop Everything with No Application Impact

## Conclusion

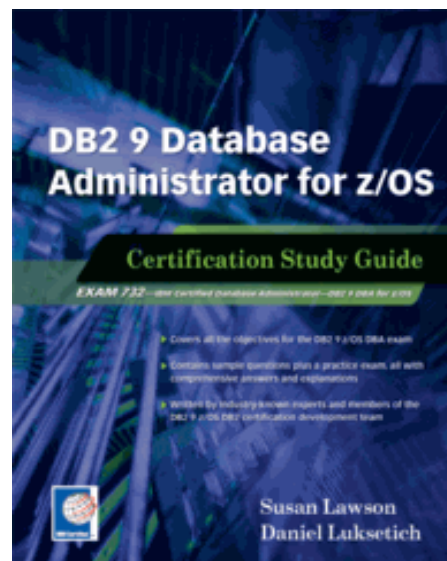
- **There are Always Alternatives**
  - Creative Solutions
  - Maximize Performance
  - Minimize Application Impact
  - Maintain Integrity and Ease of Future Development
- **The Easy Way Out Always Means More Work Later**
  - Don't Denormalize
  - Don't "Work Around" DB2
- **Take Advantage of DB2 Features**
  - During Join Predicates
  - Triggers, UDFs, Stored Procedures
  - UNION
  - Clone Table Support
  - And Many, Many More



© YL&amp;A 1999-2008

## DB2 9 for z/OS DBA Certification Guide

- "DB2 9 for z/OS DBA Certification Guide"
- McPress
- October 2007
- Authored By Susan Lawson and Dan Luksetich



© YL&amp;A 1999-2008

***DB2 Information on the Web***

---

- **DB2 9 for z/OS**
  - [ibm.com/software/db2zos/db2zosv91.html](http://ibm.com/software/db2zos/db2zosv91.html)
- **IBM Software**
  - [ibm.com/software](http://ibm.com/software)
- **DB2 Family**
  - [ibm.com/software/db2](http://ibm.com/software/db2)
- **DB2 Solutions Directory Applications**
  - [ibm.com/developerworks/db2](http://ibm.com/developerworks/db2)
- **"Red Books"**
  - [ibm.com/redbooks](http://ibm.com/redbooks)
- **DB2 for z/OS**
  - [ibm.com/software/db2zos](http://ibm.com/software/db2zos)
- **DB2 Support**
  - [ibm.com/software/db2zos/support.html](http://ibm.com/software/db2zos/support.html)
- **DB2 for z/OS Papers**
  - <ftp://ftp.software.ibm.com/software/data/db2zos>
- **DB2 Magazine**
  - <http://www.db2mag.com>
- **DB2 Certification**
  - <http://www.ibm.com/certify>
- **DB2 Experts**
  - [www.db2expert.com](http://www.db2expert.com)



© YL&A 1999-2008


***Courses by YL&A - Taught by skilled instructors world-wide!***

---

- **DB2 9 for z/OS Transition**
  - Application or DBA or both
- **SQL (z/OS and LUW)**
  - Basic SQL
  - Advanced and Complex SQL
  - SQL Performance Tuning and Optimization
- **Application Development**
  - Stored Procedure Development and Implementation
  - UDFs and Triggers Development
- **Database Design**
  - Physical Design, Logical Design
- **Data Sharing**
  - Implementation, Performance, Recovery
- **High Performance Design and Tuning**
  - Application, Database, Systems
- **DB2 9 for z/OS Certification Crammer Course**

**We customize all classes based upon customer requirements**

**ONLY YL&A offers you the DB2 9 DBA Certification Crammer Course to help you become certified!!**



© YL&A 1999-2008

### **CPU Reduction Through Performance Audits**

- **DB2 Performance Audits**
  - Existing or new database designs and applications
  - Certification of design and implementation acceptance
  - Evaluation of all the performance 'points' in a DB2 environment
    - Physical Design
    - Subsystem
    - Application Code and SQL
  - Help with bringing legacy application to an e-business environment – the rules have changed!
    - What was acceptable performance in the past is NOT acceptable in an e-business environment
  - Experienced in 'fighting fires' – many performance problems do not become reality until production
  - **Results:** problems identified, solutions proposed (many implemented immediately), continual knowledge transfer during the process

**Cost Avoidance Through Performance Tuning!!!!**



© YL&A 1999-2008