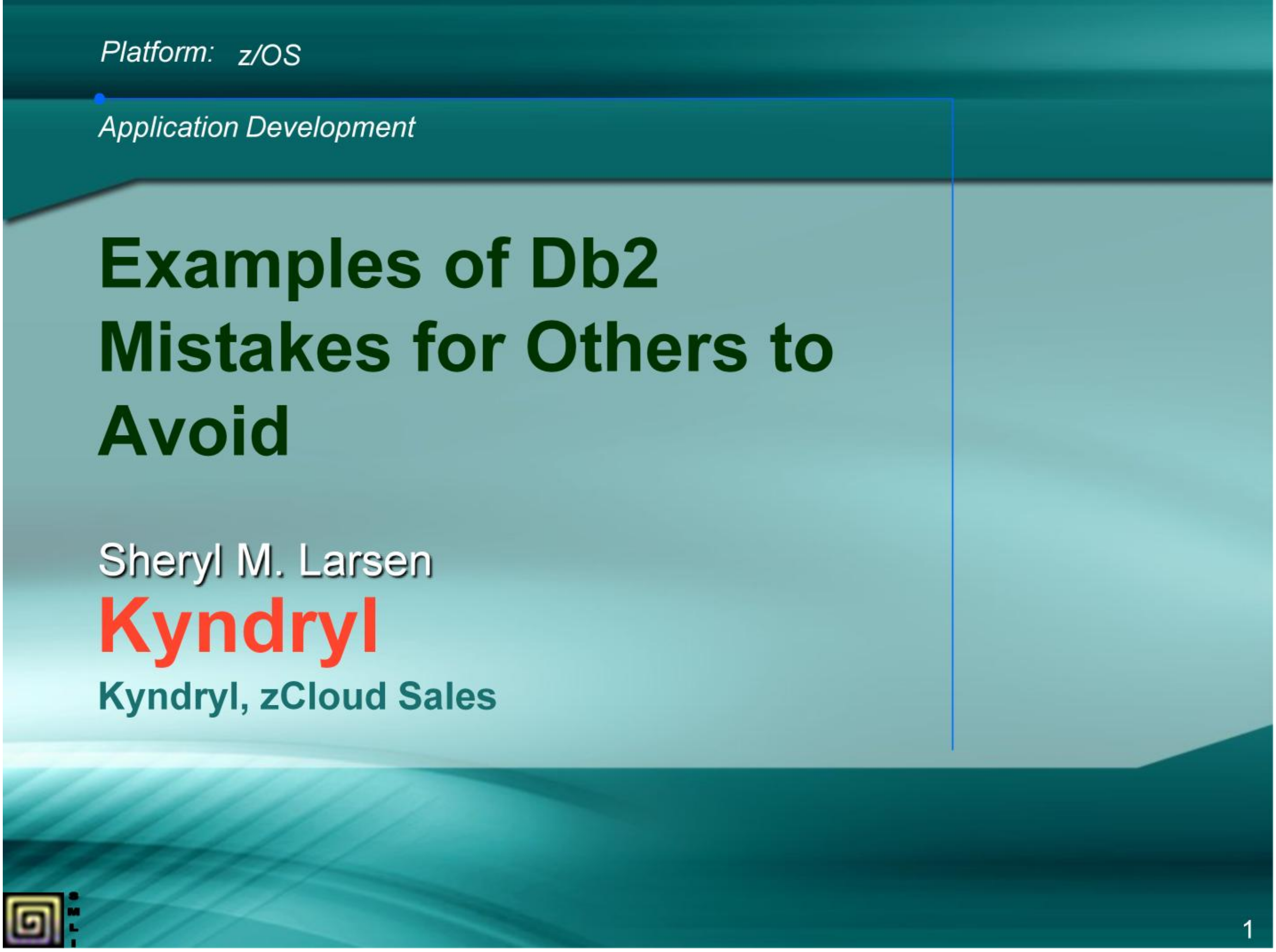


Notes: This presentation divulges discoveries and recommendations from various SQL performance review assignments. See if you have similar SQL performance issues and get the instructions on how to fix them. Issues discussed include non-optimal index design, access paths gone wild, Stage 2 predicates, misuse of SQL, excessive sorting, delayed filtering, lack of implementation of powerful new SQL features.

Skills Taught:

•Be able to determine *where* to use *what* SQL features, for example, can identify when to use joins versus subqueries and vice versa.

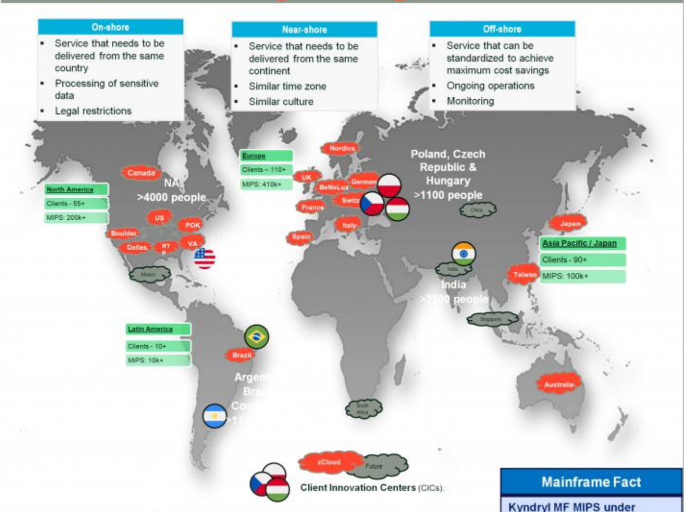
We deploy and manage a multi-tiered service delivery model allowing us to balance client needs and economics for hundreds of Mainframe and zCloud clients worldwide

Kyndryl Mainframe Facts

400+ Mainframe Clients 200+ zCloud Clients
>600 Machines
~6.1M MIPS - Globally ~680K MIPS – zCloud
>112 PB Storage
>5,000 LPARs
>28,200 Subsystems Mgd.
>100M batch jobs / day
>8,000 Kyndryl z Experts in 51 countries

kyndryl
Kyndryl Confidential

Kyndryl



Mainframe Fact

Kyndryl MF MIPS under management = 6.1M MIPS is more than all of our competitors combined

All Top Competitors Combined = 6M MIPS

Notes:

First SQL Class DB2 V1 beta

```
SELECT          O.ORDERID, C.CUSTOMERID,
                B.BILL, SUM(B.AMOUNT) AS TOTAL
FROM            ORDER O, CUSTOMER C, BILL B
WHERE           B.DATE > ‘01-01-2017’
          AND   O.ODERID = C.ORDERID
          AND   C.CUSTOMERID = B.CUSTOMERID
GROUP BY        O.ORDERID, C. CUSTOMERID, B.BILL
HAVING          TOTAL > 100000
ORDER BY        TOTAL DESC
```

Notes:

1984

- No internet
- No cell phone
- No laptop
- No ear buds
- No email
- No …….

- rldicl (Rotate Left Double Word Immediate then Clear Left) instruction
- rlmi (Rotate Left Then Mask Insert) instruction
- rrib (Rotate Right and Insert Bit) instruction
- sld (Shift Left Double Word) instruction
- sle (Shift Left Extended) instruction
- sleq (Shift Left Extended with MQ) instruction
- sliq (Shift Left Immediate with MQ) instruction
- slliq (Shift Left Long Immediate with MQ) instruction

Notes:

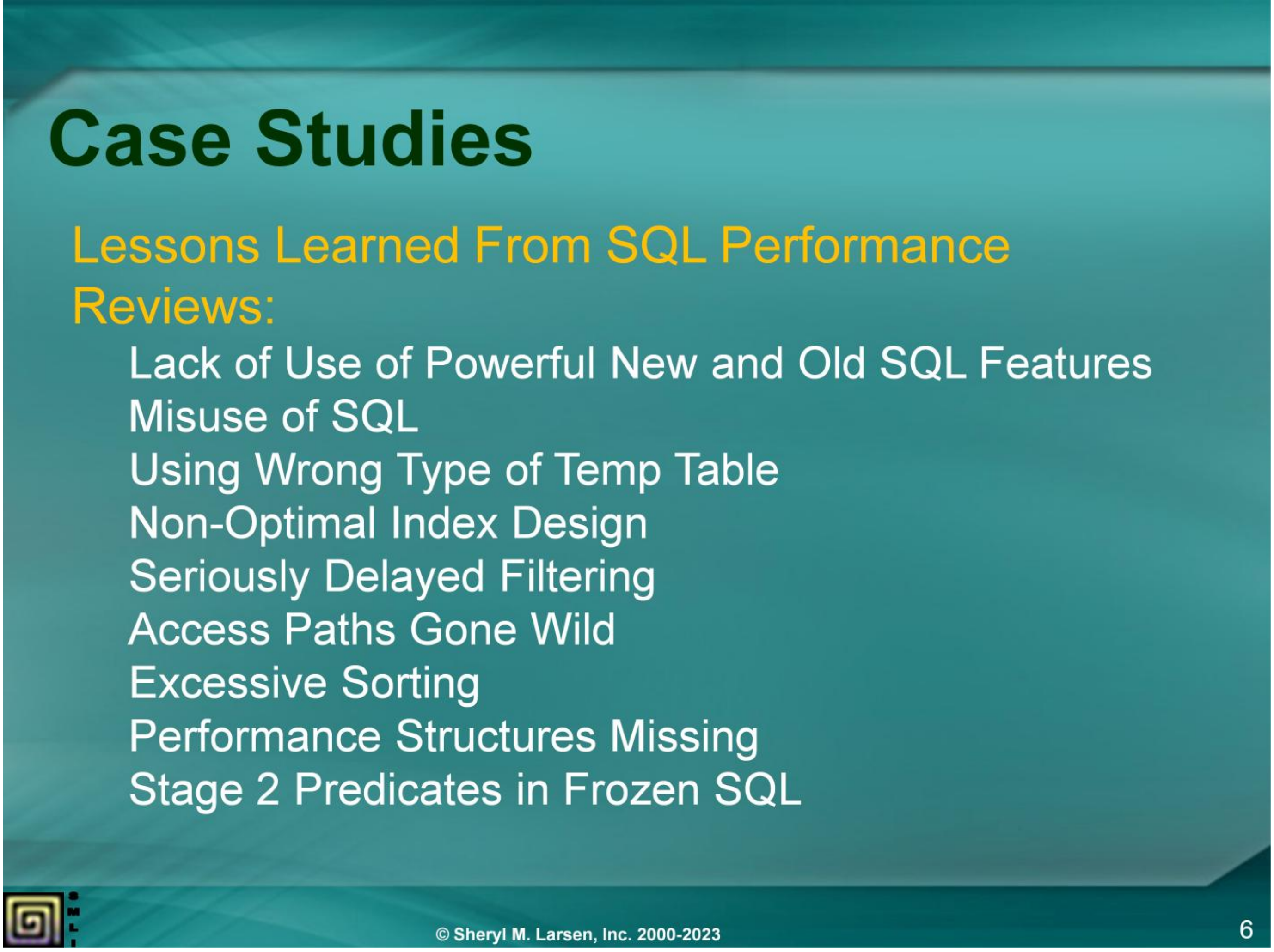
Fast Forward

35 YEARS

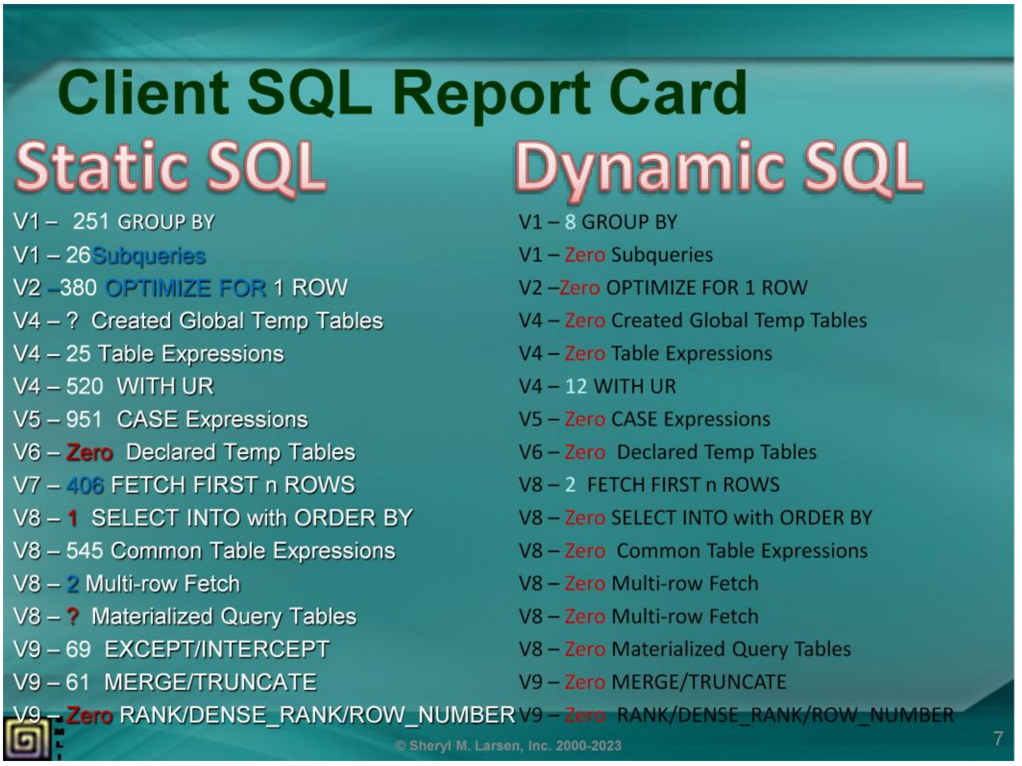
1984
1989
2003
2010
2015
2019
2020
2021

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, ORDER BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join, Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, SQL PL in routines, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, MERGE not logged tables, OmniFind, spatial, range partitions, data compression, DECFLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, implicit cast, increased timestamp precision, currently committed, moving sum & average, index include columns, row and column access controls, time travel query, GROUPING SETS, ROLLUP, CUBE, global variables, Text Search functions, accelerated tables, DROP COLUMN, array data type, XML enhancements, moving SUM/AVG, Array variables, ARRAY_EXISTS,COUNTBIG,
SELECT INTO statements with UNION or UNION ALL allowed
SELECT INTO statements with UNION or UNION ALL disallowed, OFFSET, 11 global variables, LISTAGG + 33 more BIFs, LIMIT fetch row count1

Notes:



Notes: The **Lessons Learned from SQL Performance Review** presentation divulges discoveries and recommendations from various SQL performance review assignments. Come see if you have similar SQL performance issues and get the instructions on how to fix them. Issues discussed include non-optimal index design, access paths gone wild, Stage 2 predicates, misuse of SQL, excessive sorting, delayed filtering, lack of implementation of powerful new SQL features.



Notes: Some use of SQL technology but still used CURSORs for fetching 1 ROW with ORDER BY.

Name: ____________________

SQL Coding Skill Self Assessment

Before:________

Range	Level	Assessment = YOU CAN FULLY UNDERSTAND THE FEATURE AND PROPER USE OF:
Basic SQL Skill Range	0	You think SQL is a new energy drink
	1	Simple SELECT statements, WITH clause, ORDER BY
	2	WHERE clauses, BETWEEN, LIKE, IN(list), =, >=, >, <, <=, <>, NOT IN(list), NOT LIKE, NOT BETWEEN
	3	Table joins (inner, outer, full), UNION, UNION ALL, CONCAT, static CURSORs, FOR UPDATE OF, ROW_NUMBER
Intermediate SQL Skill Range	4	noncorrelated and correlated subqueries, EXISTS, NOT EXISTS, FETCH FIRST x ROWS ONLY, OPTIMIZE FOR x ROWS
	5	Indexable, Stage1 and Stage 2 predicate evaluation, multirow FETCH/INSERT, GET DIAGNOSTICS, Scalar full SELECT,
	6	Table expressions/common table expressions, GROUP BY, HAVING, IS NOT DISTINCT FROM, EXCEPT/INTERCEPT
	7	CASE expressions, Global Temporary Table (GTT), Declared Temporary Table (DTT), Dynamic Scrollable cursors, SEQUENCE columns
Advanced SQL Skill Range	8	Queries involving > 10 tables, INSERT within SELECT, Star Schema, Snow Flake, GROUP BY expression, IDENTITY columns
	9	MQT (Materialized Query Tables), Recursive SQL, UNION in Views, Native SQL Stored Procedures, > 20 SQL Functions, DENSE_RANK, RANK
	10	Codes effective and efficient SQL applying performance rules and knows when to use each appropriately

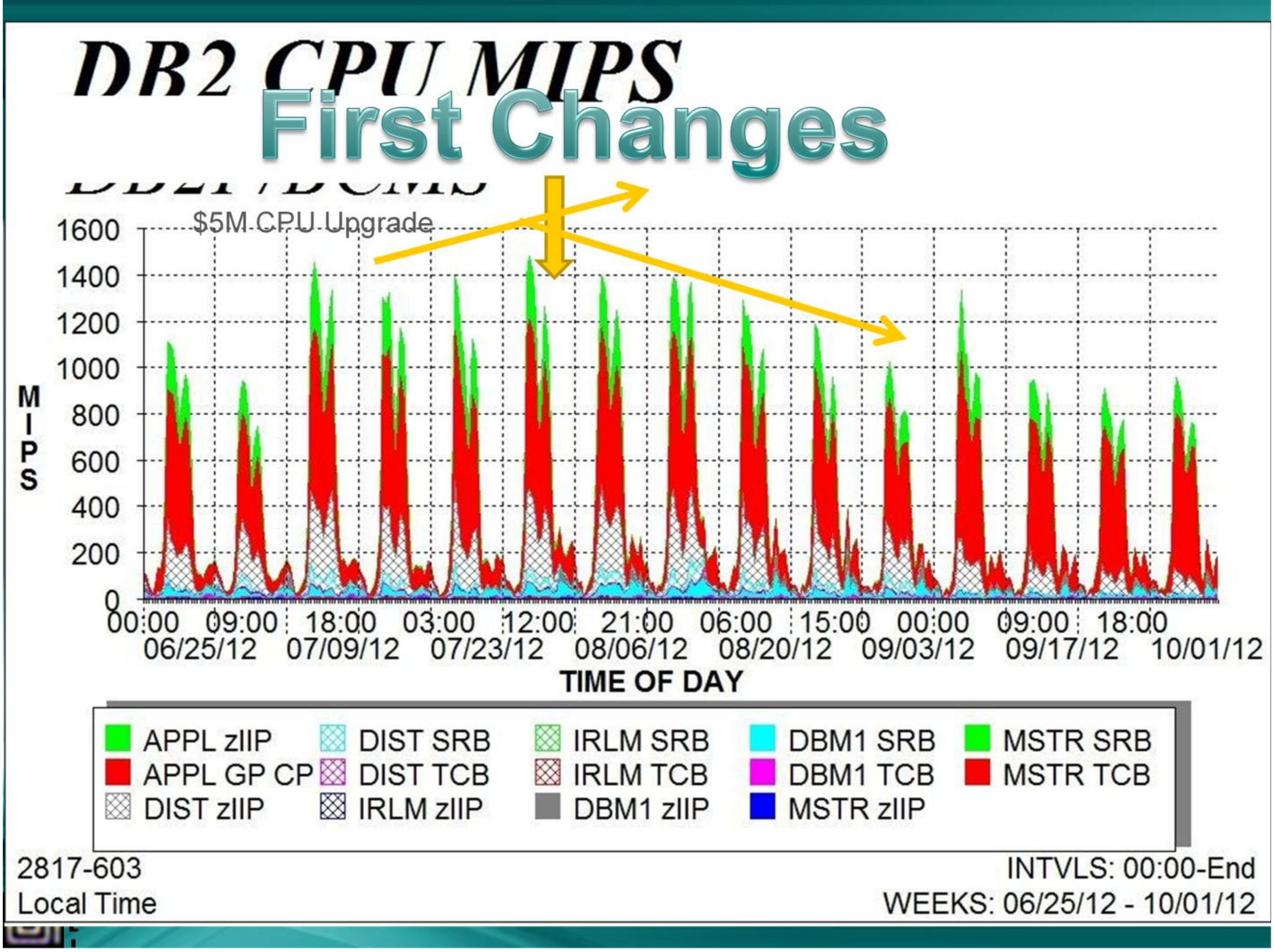
After:_________

Notes:

Dynamic Queries

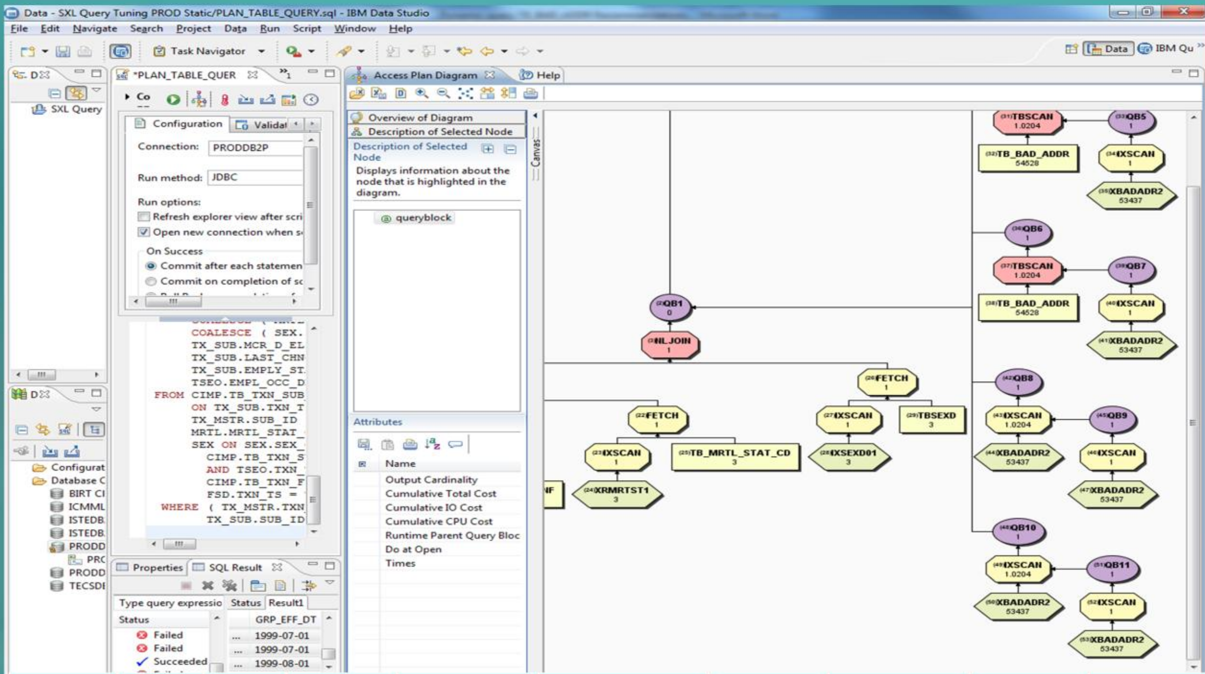
Had no Sheriff

Notes:


DB2 CPU MIPS
First Changes
$5M CPU Upgrade
1600
1400
1200
1000
800
600
400
200
0
MIPS
00:00 09:00 18:00 03:00 12:00 21:00 06:00 15:00 00:00 09:00 18:00
06/25/12 07/09/12 07/23/12 08/06/12 08/20/12 09/03/12 09/17/12 10/01/12
TIME OF DAY
APPL zIIP
APPL GP CP
DIST zIIP
DIST SRB
DIST TCB
IRLM zIIP
IRLM SRB
IRLM TCB
DBM1 zIIP
DBM1 SRB
DBM1 TCB
MSTR zIIP
MSTR SRB
MSTR TCB
2817-603
Local Time
INTVLS: 00:00-End
WEEKS: 06/25/12 - 10/01/12

Notes:

Visual Plan Graphs - Bad



STMT_ID	CPU	EXECS	ELAPSED	GETPAGES	EXAMINED ROWS	PROCESSED	STAT_SORT	STAT_INDX	STAT_RSCN
1022787	4705.7	5258	4722.46	729,475,051	298,690,230	2629	0	283543230	5606

11

Notes:

Removed Bad Scalar Subquery

Query now just returns a 'Y'

First rewrite implemented September 19th

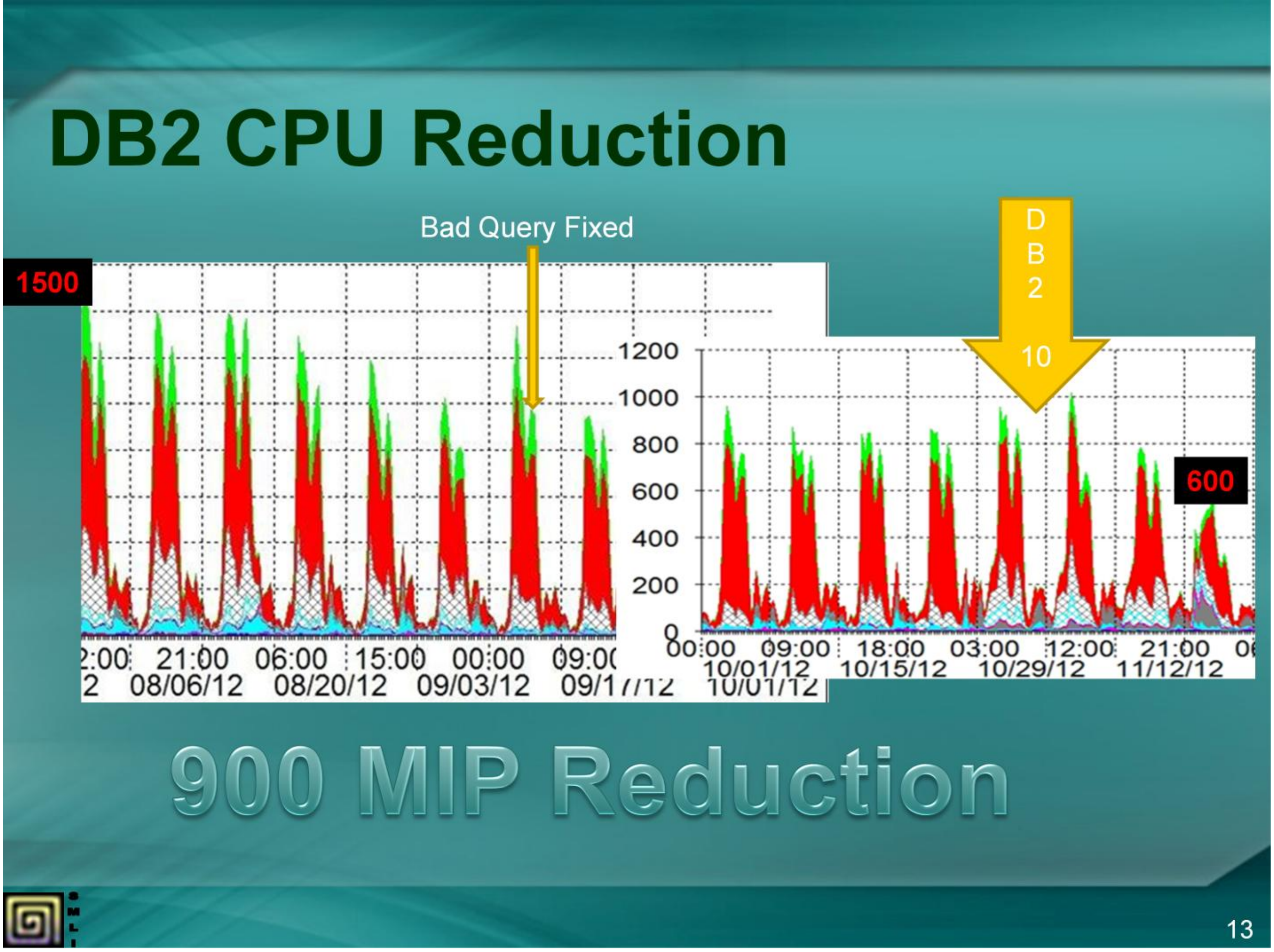
```
SELECT 'Y'
FROM ABCP.TB_BAD_ADDR ADDR
     INNER JOIN
     ABCP.TB_GRP_ELEC_ONLY XGRP
  ON XGRP.GRP_ID = ADDR.GRP_ID
     INNER JOIN ABCP.TB_SUB_CONT SUB
  ON SUB.SUB_ID = ADDR.SUB_ID
     AND SUB.GRP_ID = ADDR.GRP_ID
WHERE ADDR.ADDR_RSLV_FL = 'N'
  AND XGRP.EMPR_ID = 17469
  AND( SUB.SUB_CONT_TERM_DT > (CURRENT
DATE - 90 DAYS) )
FETCH FIRST 1 ROW ONLY
FOR READ ONLY
```

```
USE COUNT          -> 2                  SQL            -> 4
  TIMEPCT            -> .00%               CPUPCT         -> .00%
  INDB2_TIME         -> 00:00.033008       INDB2_CPU      -> 00:00.012755
  GETPAGE            -> 642                GETPFAIL       -> 0
  SYNCREAD           -> 2                  SPFETCH        -> 0
  LPFETCH            -> 0                  DYNPFETCH      -> 85
  PFPAGES            -> 11                 PAGEUPDT       -> 0
```

Notes: Performance rule violations usually result in increased CPU or I/O, time to fix the mistake, and ultimately, a cost to the business unit.

What will the cost be? Depends on the mistake and the frequency of the mistake.

The following presentation will show real case studies of the actual cost of the mistakes.


DB2 CPU Reduction
Bad Query Fixed
1500
D
B
2
10
600
1200
1000
800
600
400
200
0
900 MIP Reduction
13

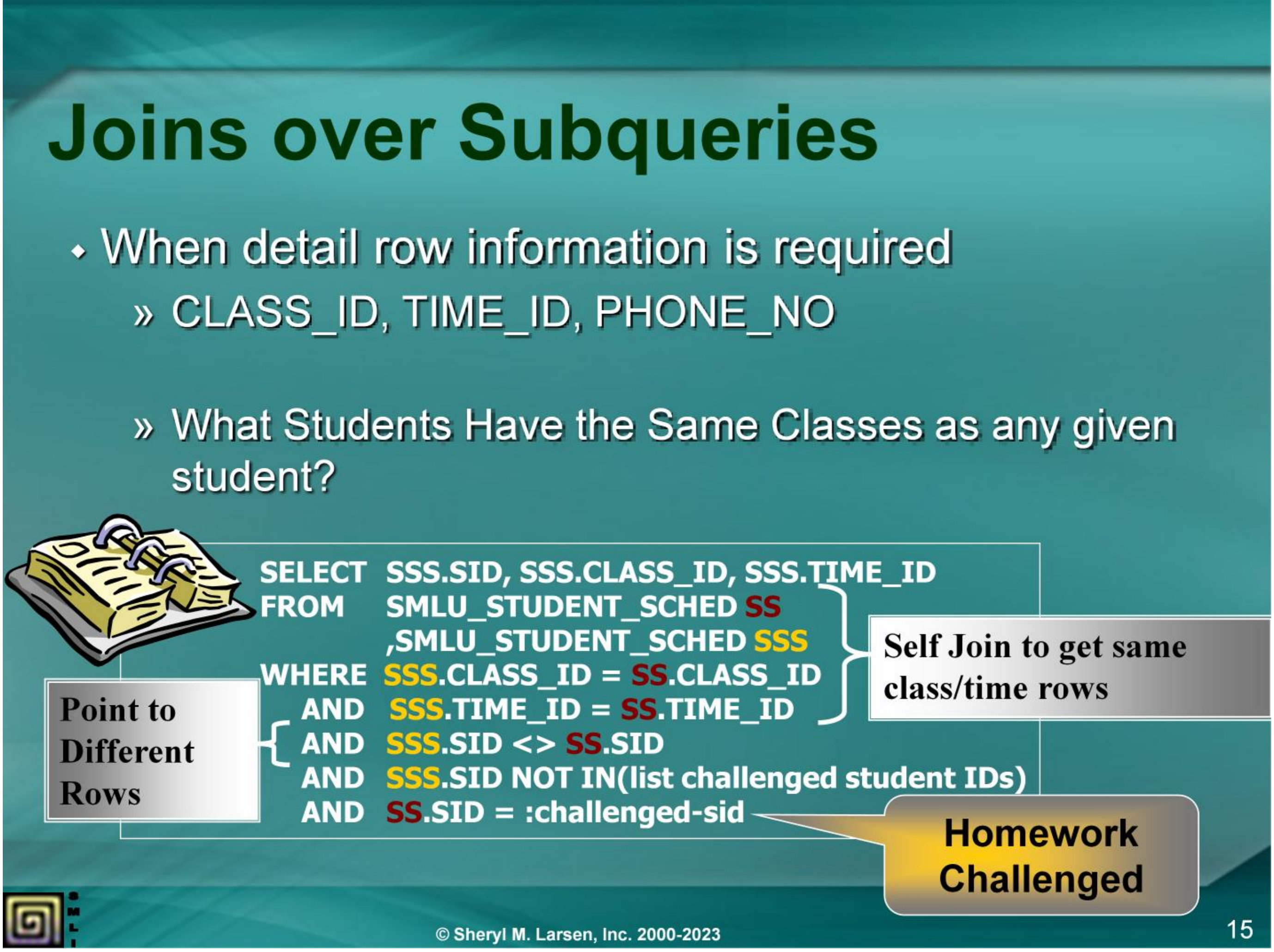
Notes:

Misuse of SQL Costs $$$$$$$$

Misuse has a cost for:
- JOINs
- Subqueries
- LEFT JOINs
- GLOBAL TEMPORARY TABLEs

14

Notes:



Notes: Joins are very good at giving lots of detail from either table.

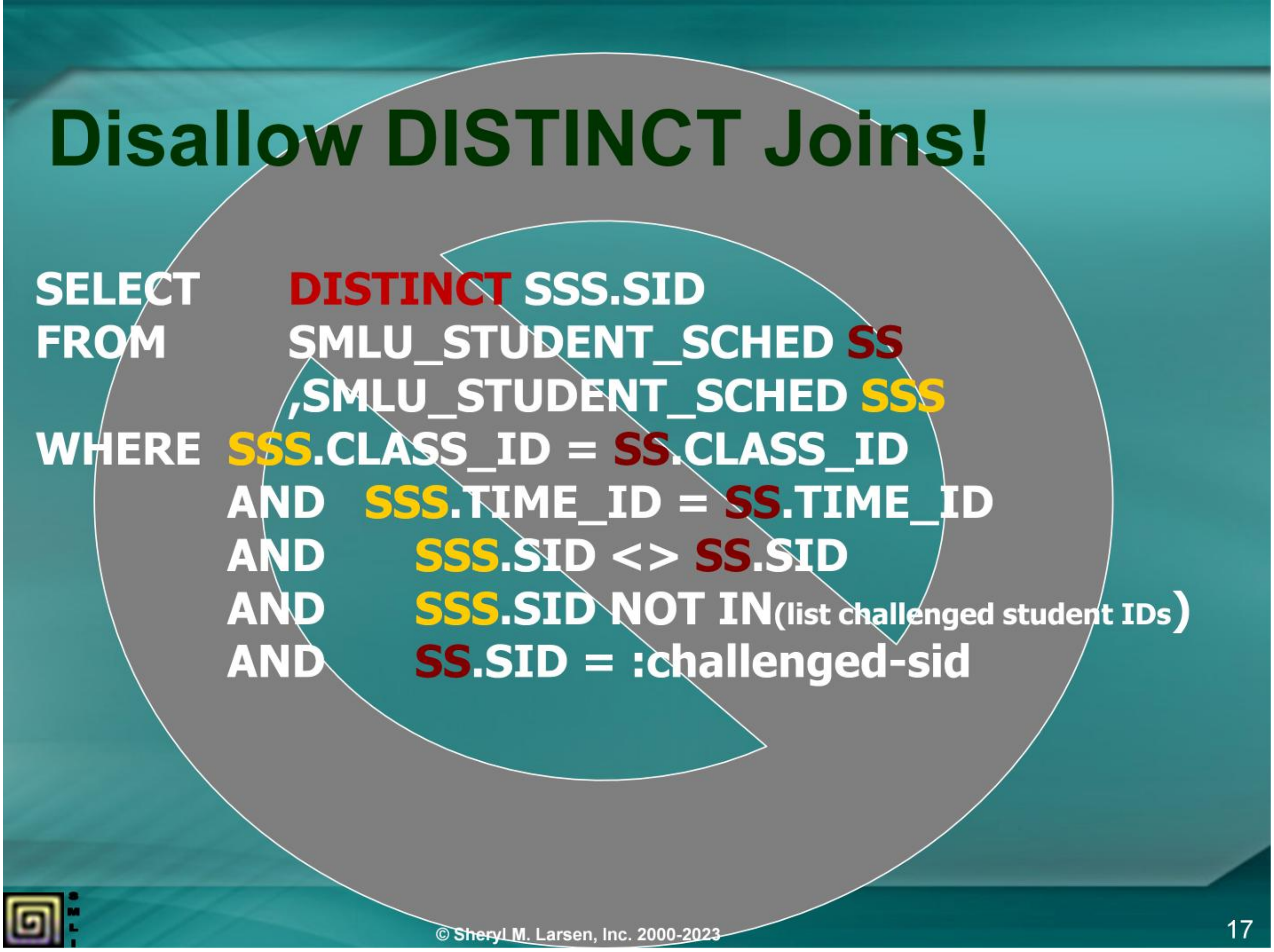
Subqueries over Joins

- When detail row information is not required
  - » Unique SIDs only
  - » What Students Have **at least one** of the same Classes as a given student?

```
SELECT  SS.SID
FROM    SMLU_STUDENT_SCHED SS
WHERE   SS.SID NOT IN(list challenged student IDs)
 AND  EXISTS
          (SELECT 'TRUE OR FALSE'
          FROM SMLU_STUDENT_SCHED SSS
          WHERE  SSS.CLASS_ID = SS.CLASS_ID
              AND   SSS.TIME_ID = SS.TIME_ID
              AND   SSS.SID <> SS.SID
              AND   SSS.SID = :challenged-sid)
```

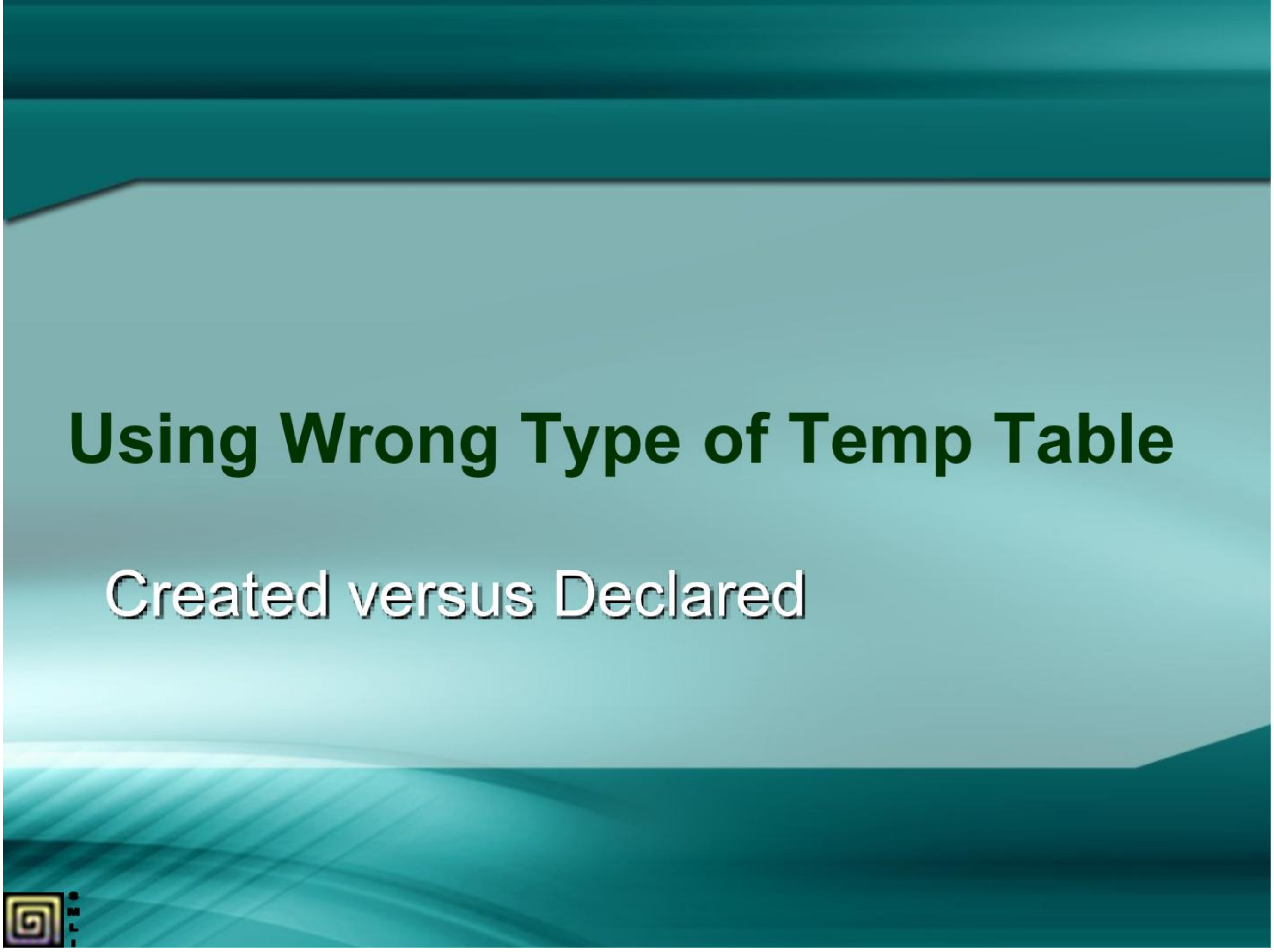
Homework Challenged

Notes: Correlated subqueries are good at bypassing detail.

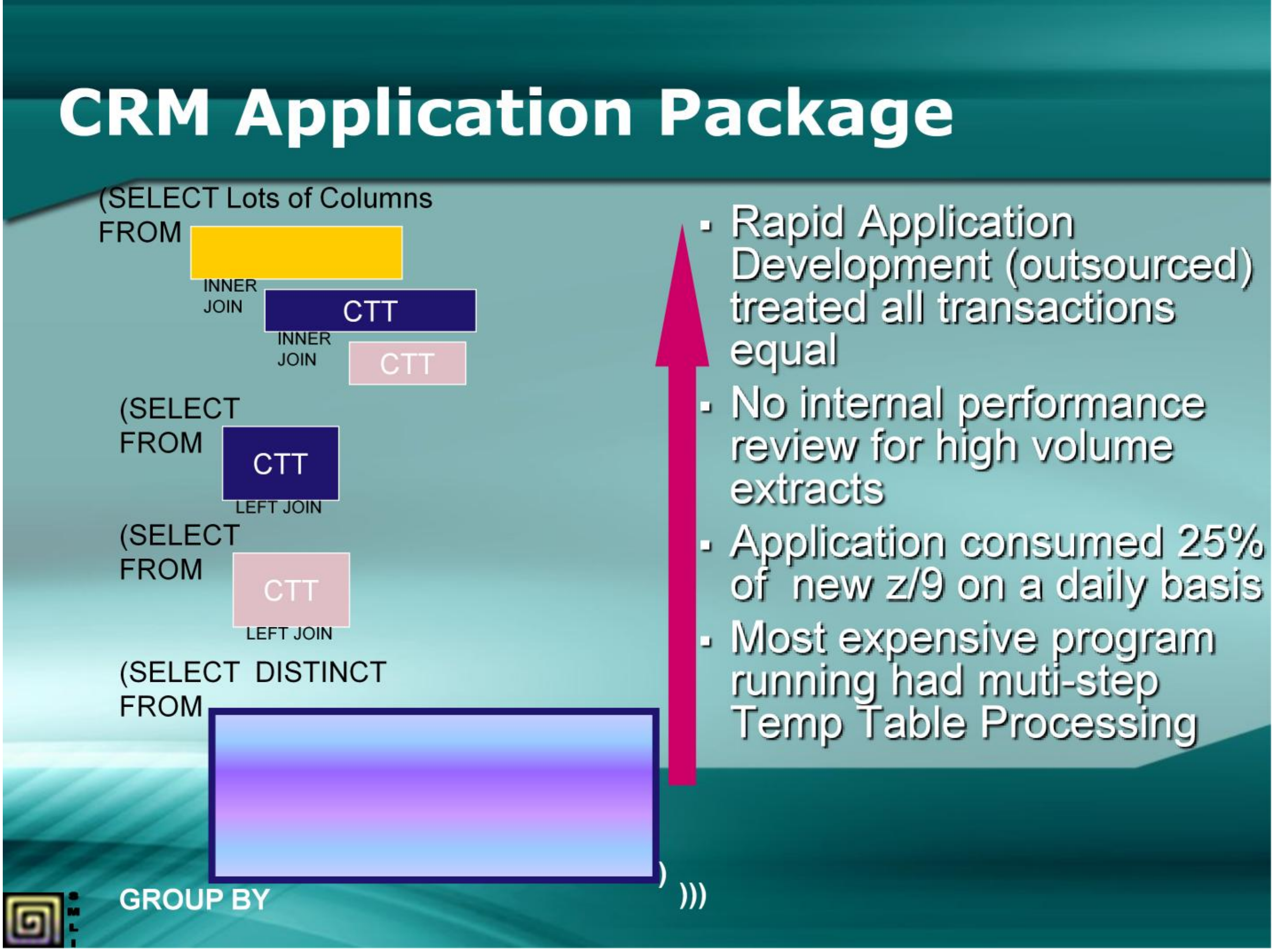


Notes: DISTINCT should never be used with an INNER JOIN. The INNER JOIN produces detail and the DISTINCT removes the detail.

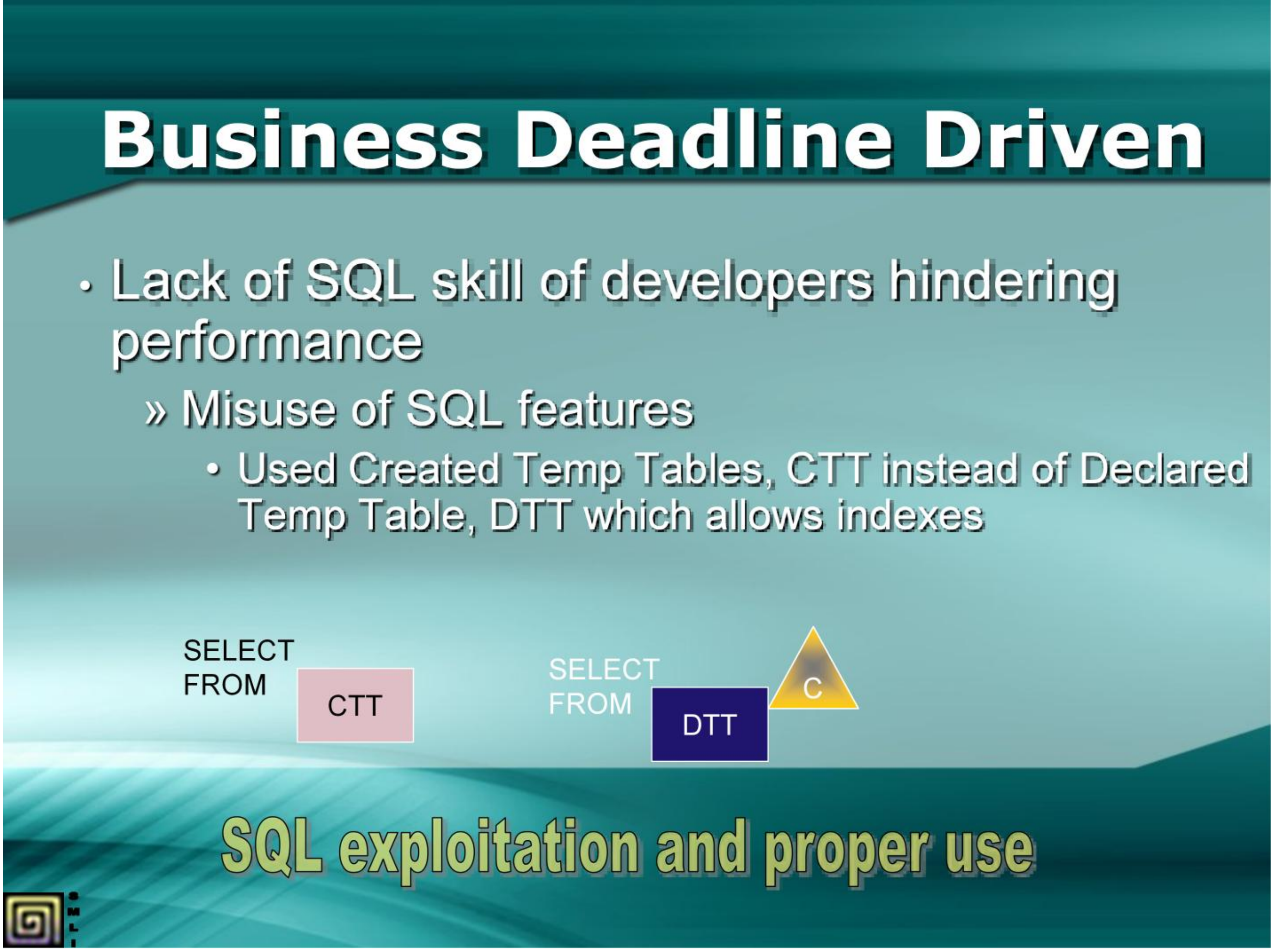
Notes: This INNER JOIN is faster than this LEFT JOIN when there are many rows that do not join. This is because LEFT JOINs have to calculate exceptions to the JOIN.



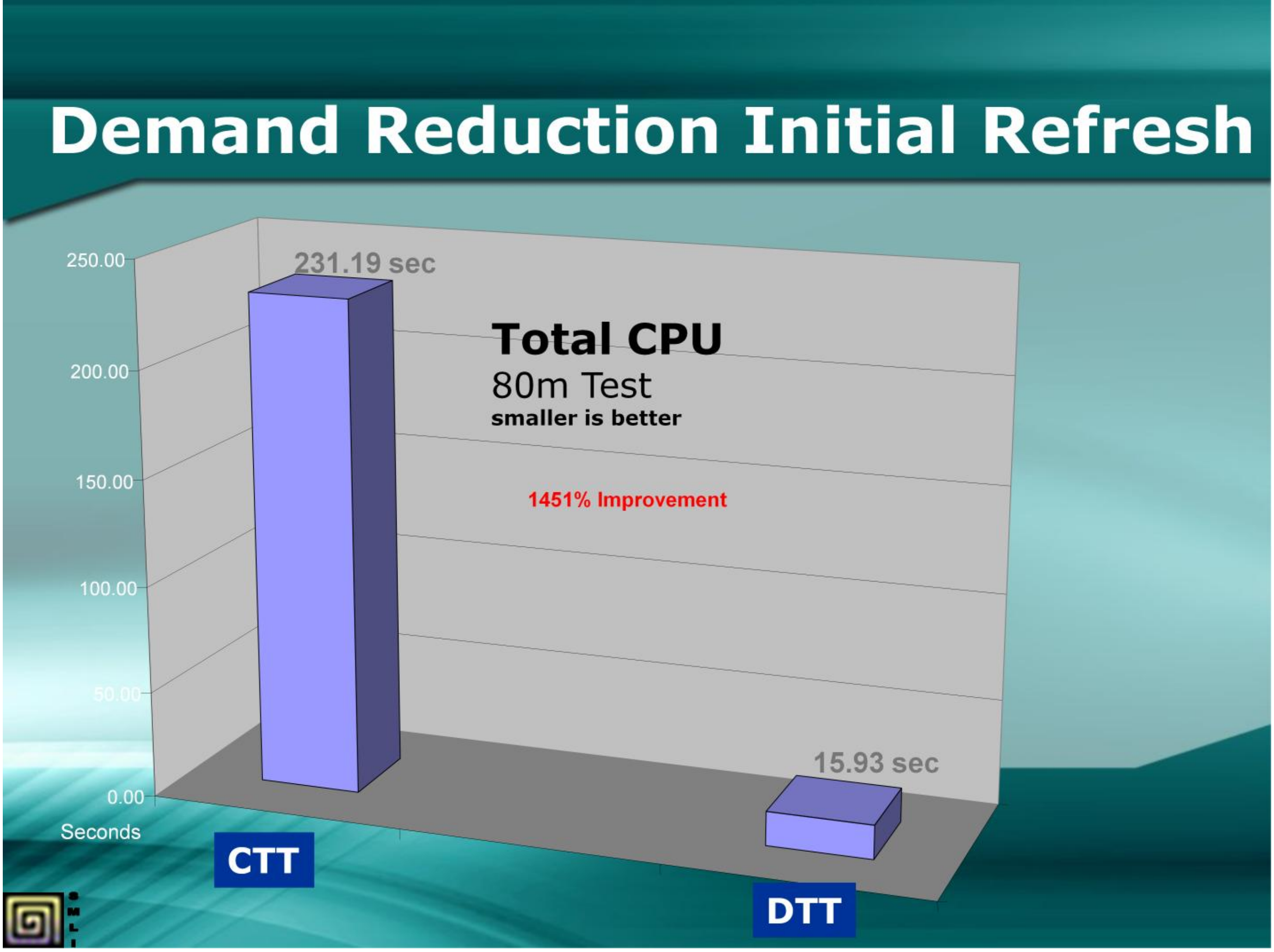
Notes:



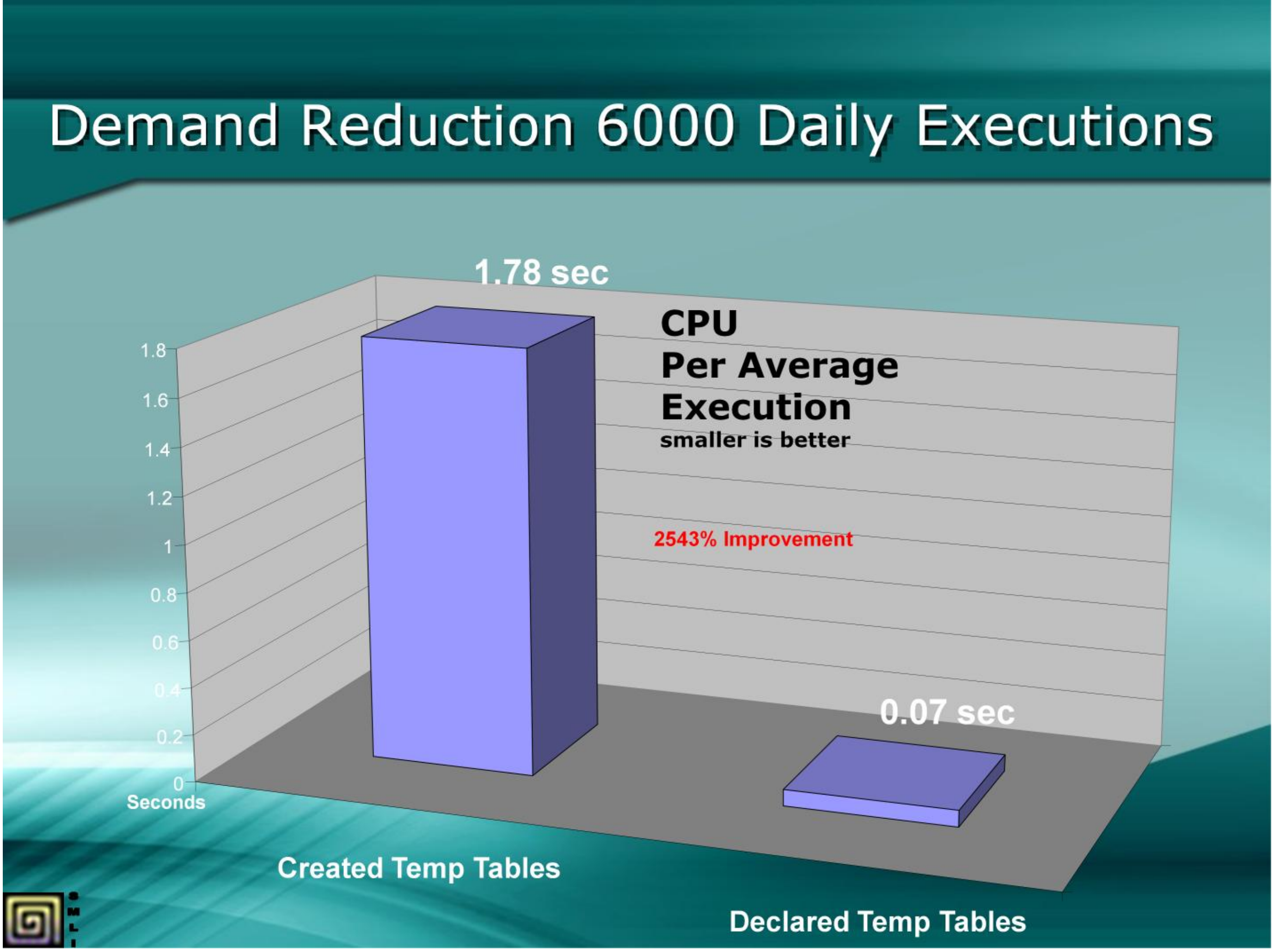
Notes: A problem application consumed computer and people resources when scaling was attempted.



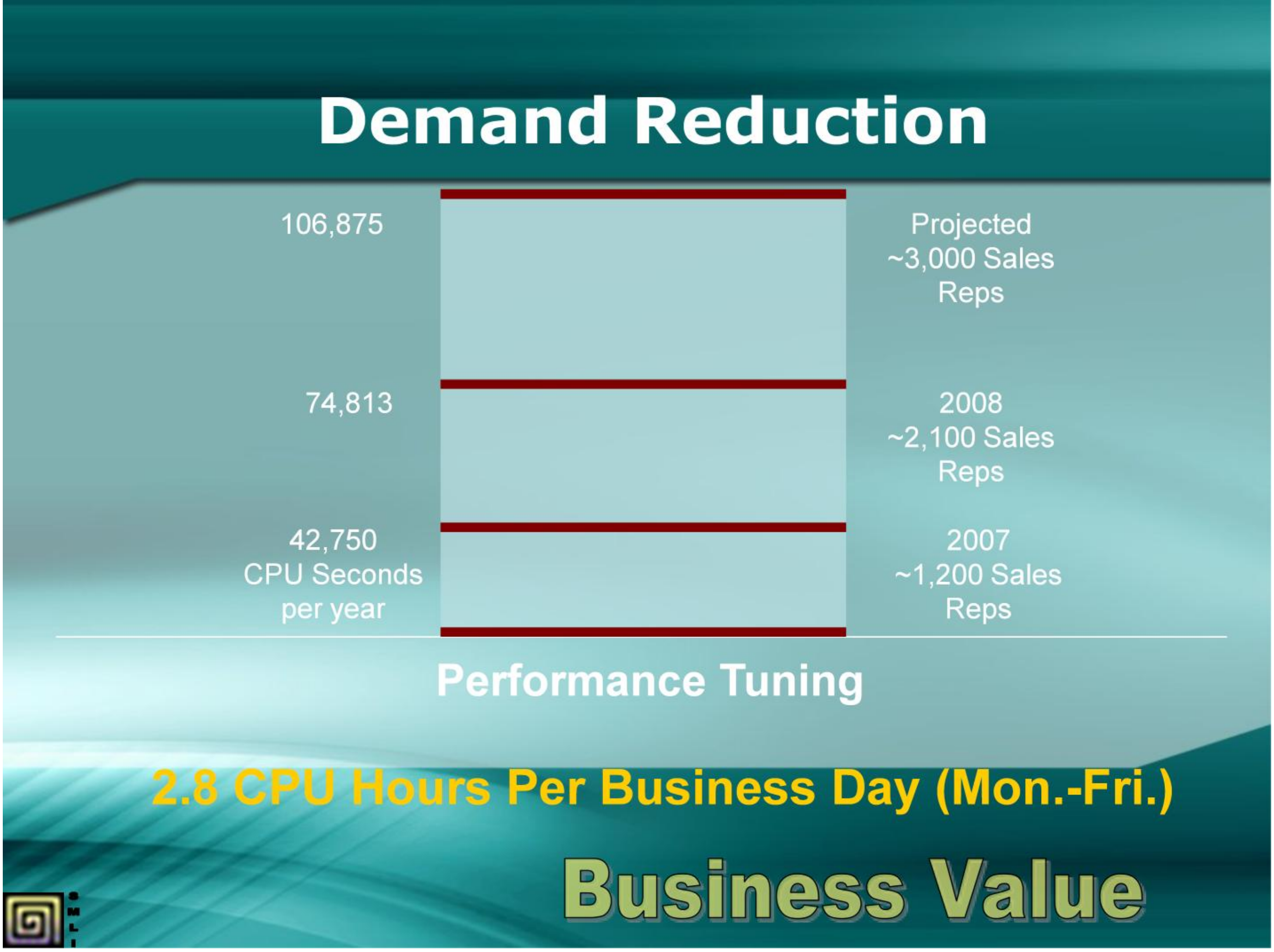
Notes: The most resource intensive program was examined in detail and SQL misuse was quickly detected.



Notes: This problem program extracted a tremendous amount of sales activity. The savings depended on the amount of data being extracted. Initial refreshes for 25% of the sales force were reduced 1451%.



Notes: Smaller daily /hourly refreshes were improved 2543%.

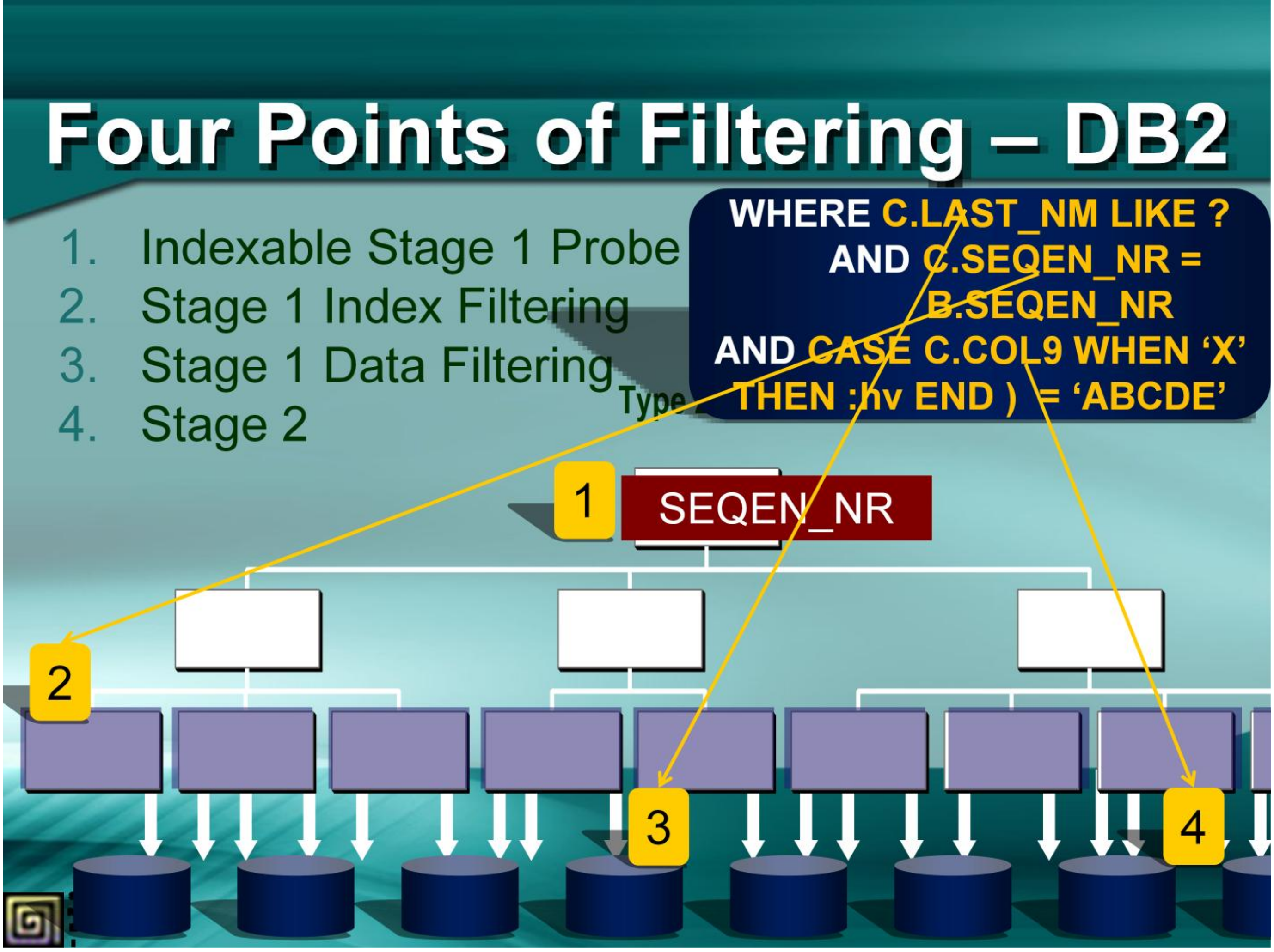


Notes: **These improvements add up in the course of a day when the program executes 6,000 times a day.**

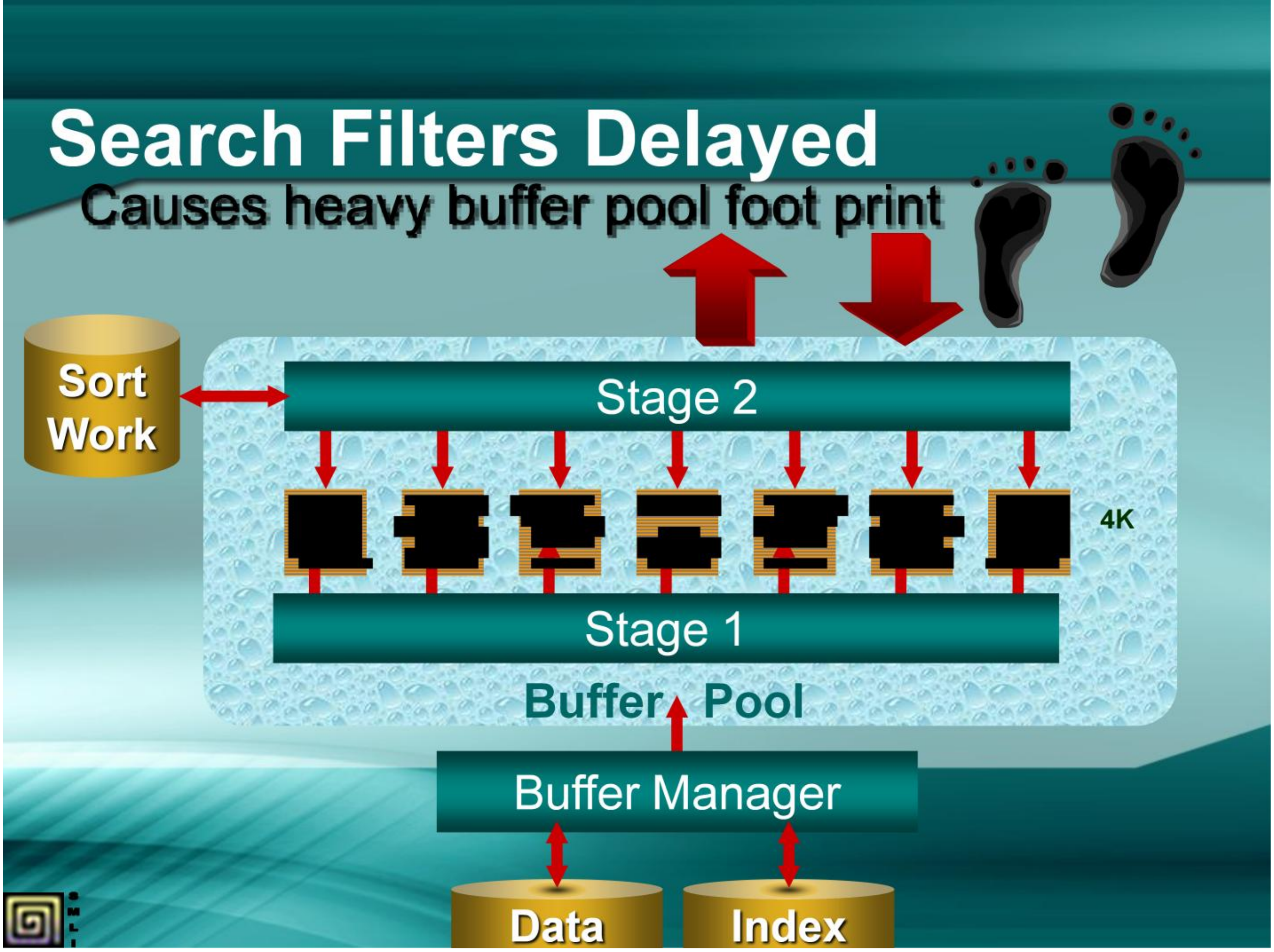
42,750 CPU Minutes Per Year 1 Off of z/OS

712 CPU Hours Per Year 1 Off of z/OS

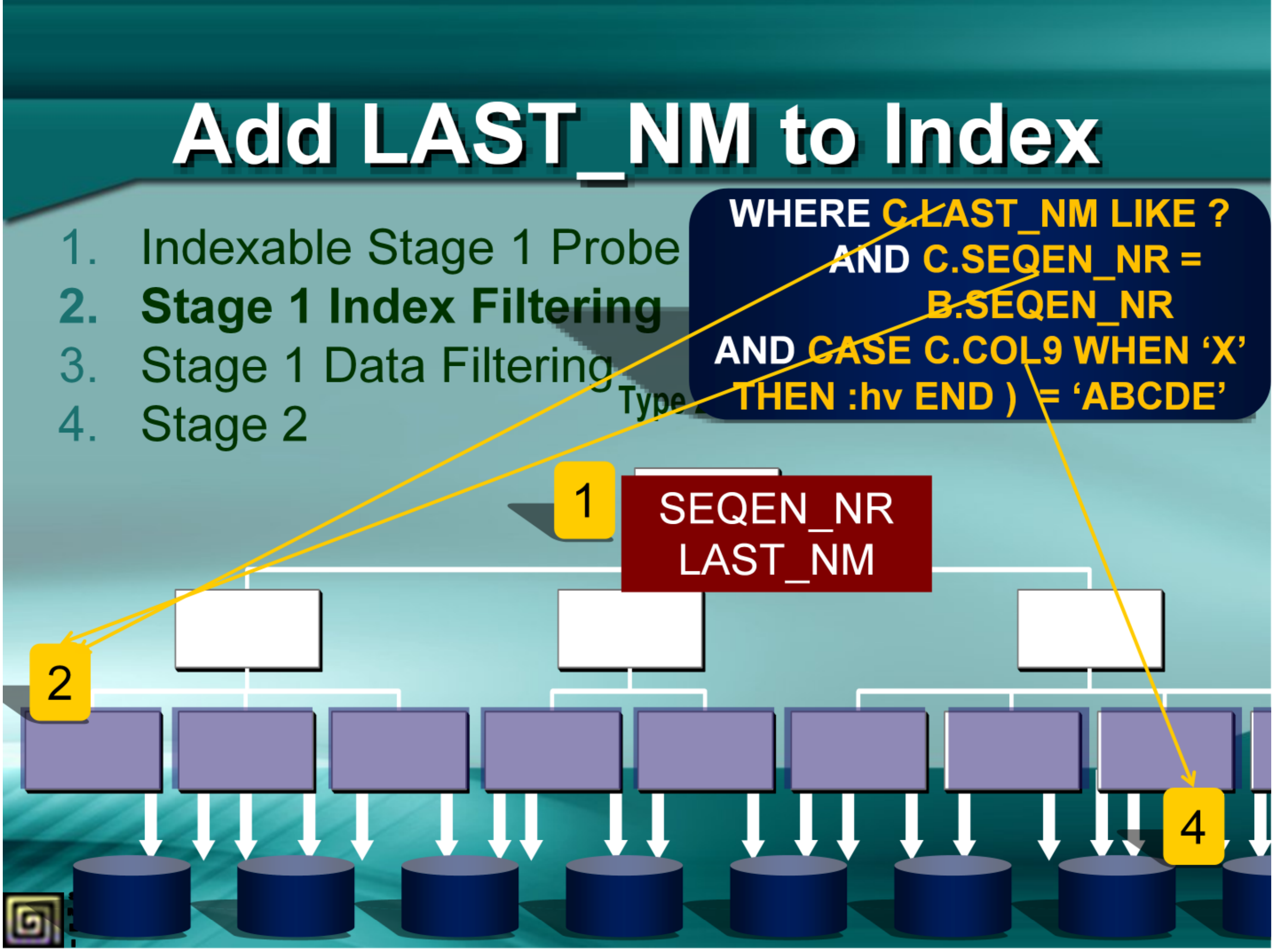
Notes:



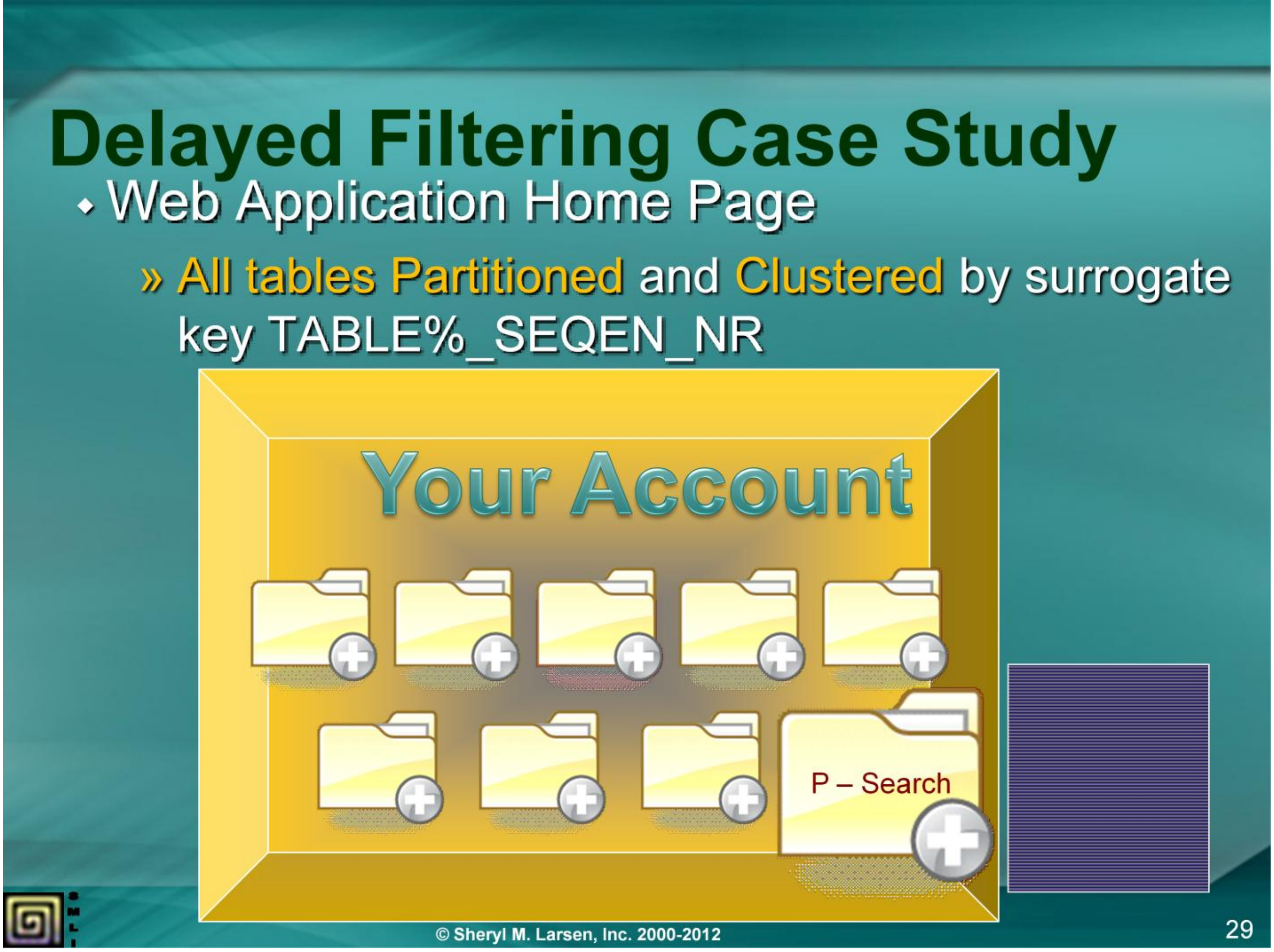
Notes: The first filter to be applied is LAST_NM LIKE at the 3rd point of filtering (after every index page and ever data page was retrieved).



Notes: The more pages brought in to the Buffer Pool the larger the foot print. This increases contention and reduces through put.



Notes: The first filter to be applied is LAST_NM LIKE at the 3rd point of filtering (after every index page and ever data page was retrieved).

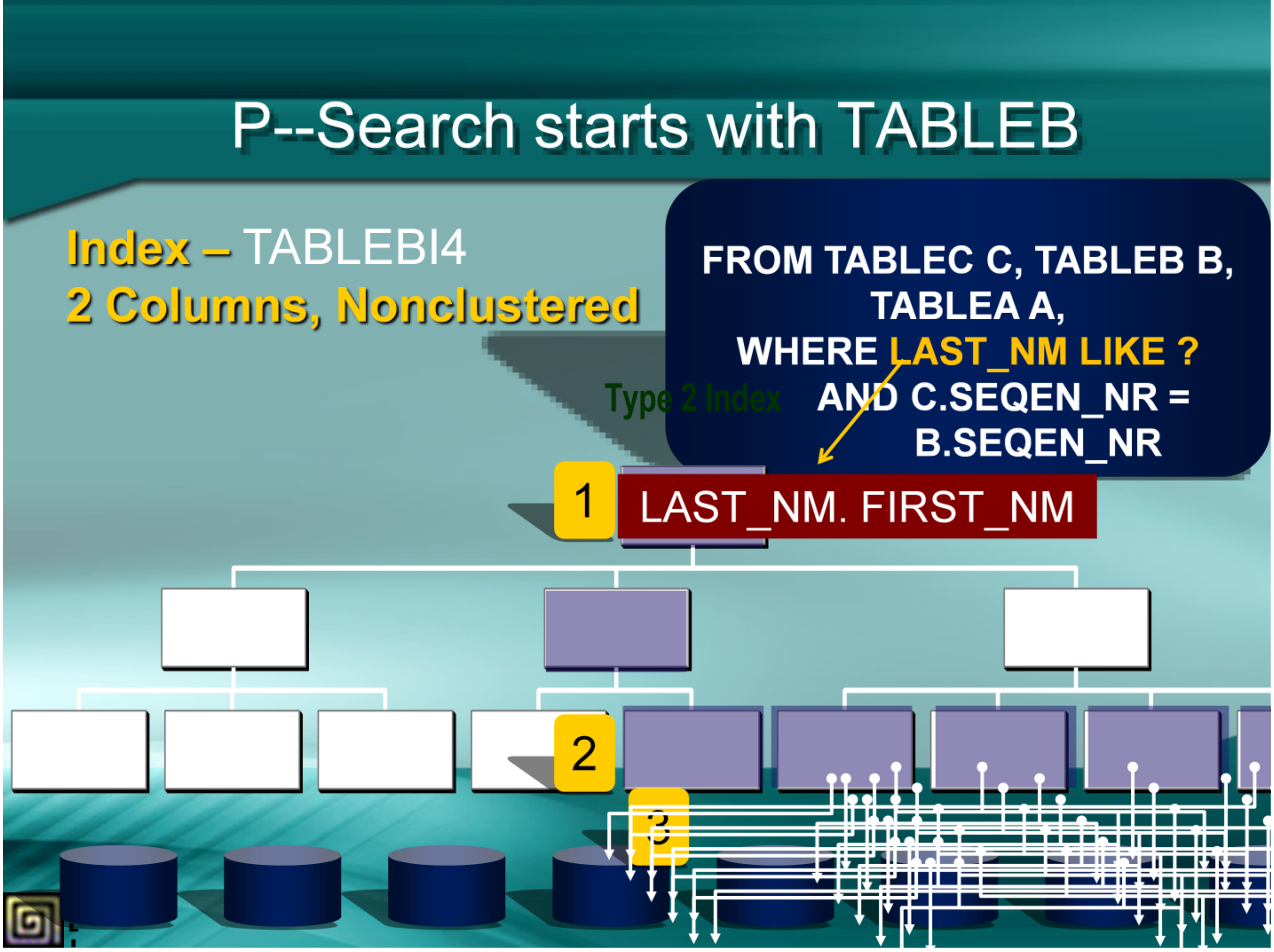


Notes: Architecting web pages to preload all services has a cost but some times can be justified.

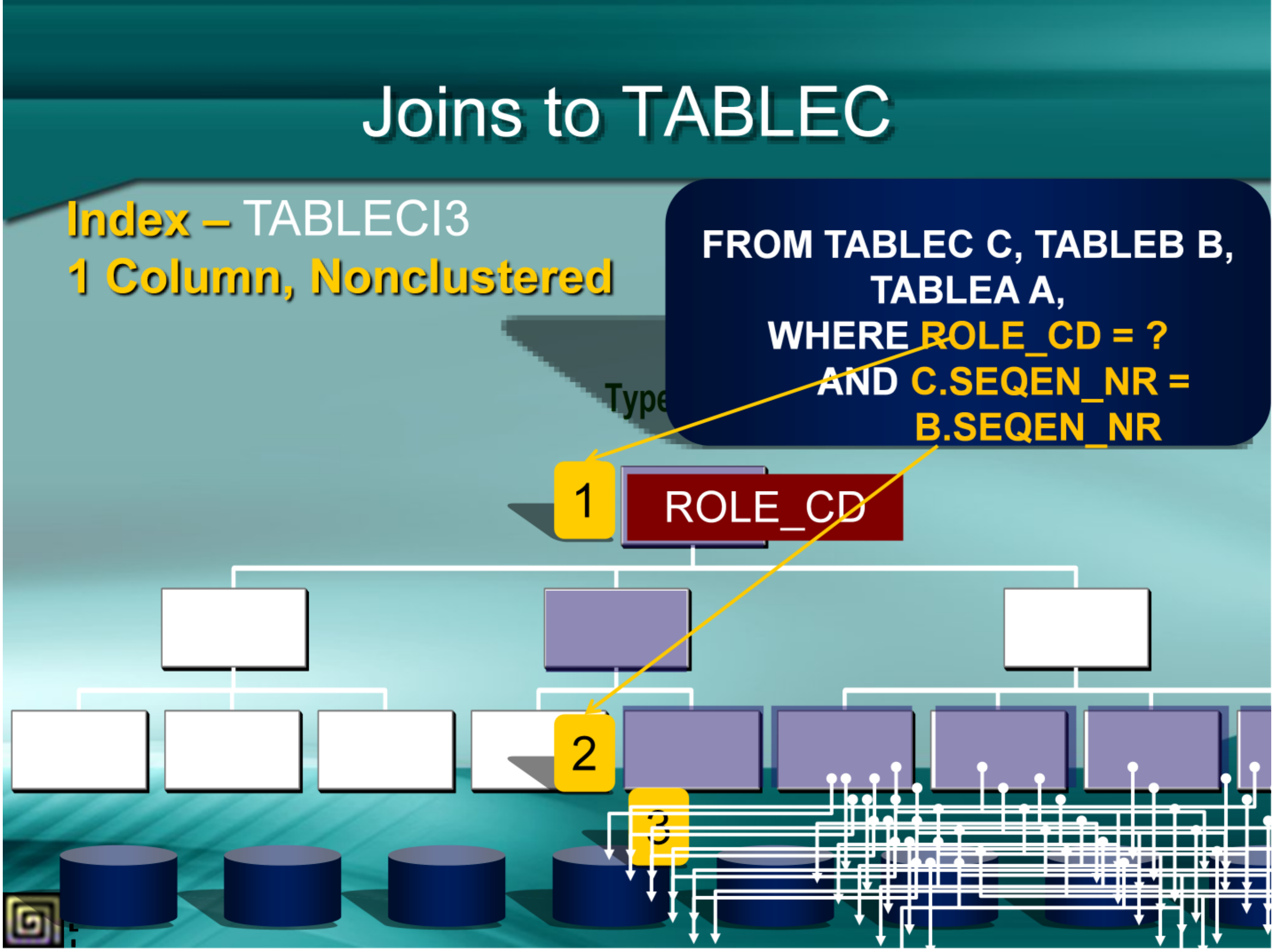
Web Page Details

- 9 Services execute every time
- Dynamic SQL
- Fetches entire result for every service for maximum viewing and sequence agility
- Takes significant wall clock time to load during peak season
- Upcoming slides demo execution of P- Search service

Notes: The most expensive service, P-Search was examined.



Notes: A three table join and DB2 Optimizer starts with TableB and filters LAST_NM but picks up every SEQEN_NR using lots of random I/O.



Notes: The DB2 Optimizer has to choose between a filtering index or a joining index since the combined index does not exist. All sequence numbers are picked up using lots of random I/O and then a merge join puts the tables together.

Notes: The last table is joined using Nested Loop because the optimizer chose the join column index instead of the filtering index as no combined index exists and a local filter is supplied using a TABLEC_SEQEN_NR BETWEEN ? AND ?.   GUAR_DT is not filtered until the 3rd point of filtering and then a sort is issued for the ORDER BY.

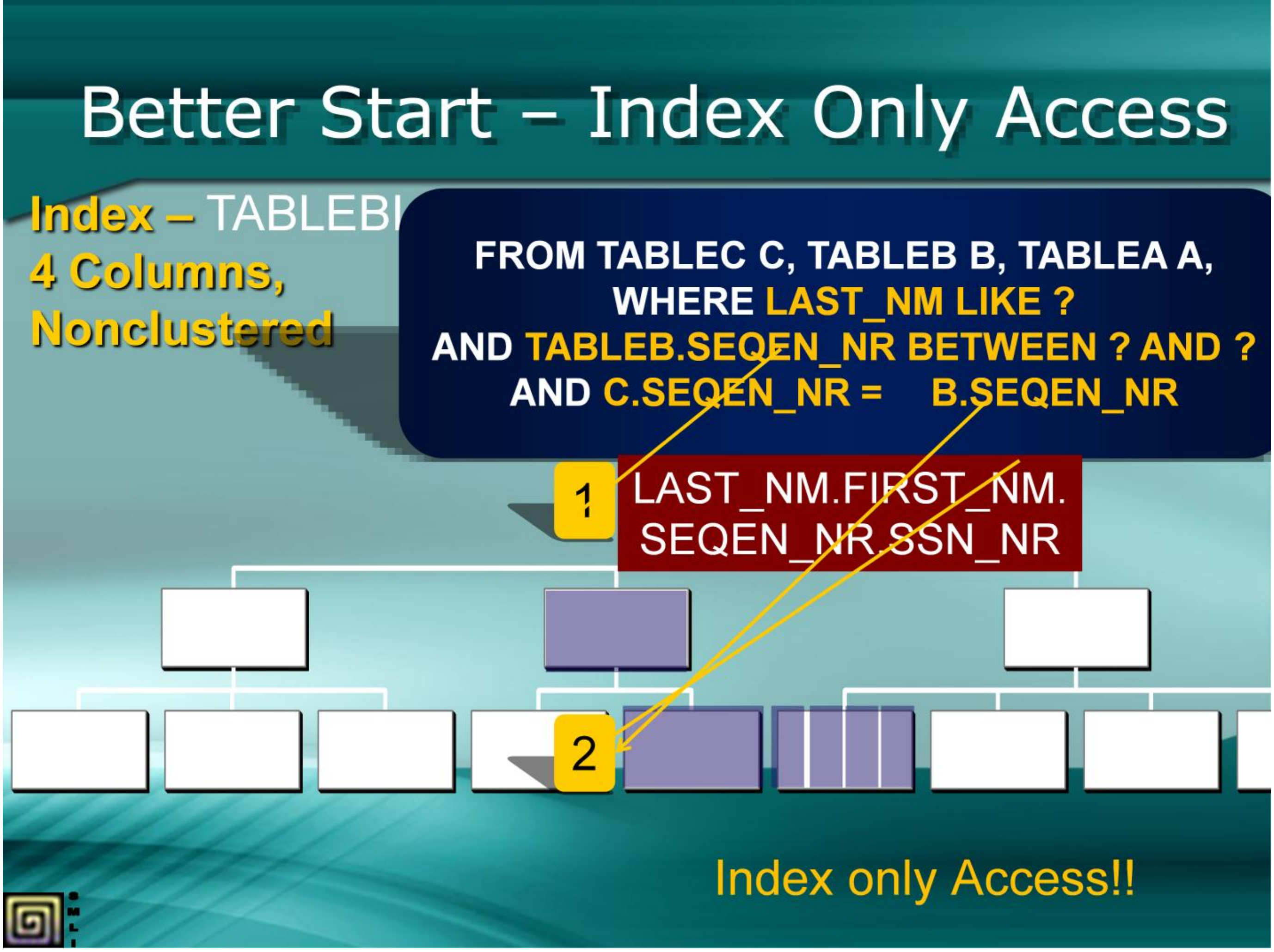
Solutions for P-Search

- Altered index TABLEBI4 added SEQEN_NR, and SSN_NR to the end for index only access *(30% CPU reduction per execution)*
- Recommend altering TABLECI3 to add SEQEN_NR
- Recommend adding new index TABLEAI5
  - » GUAR_DT DESC.SEQEN_NR
- After solutions are implemented
  - » *9.9 CPU seconds reduced to .2 CPU seconds*

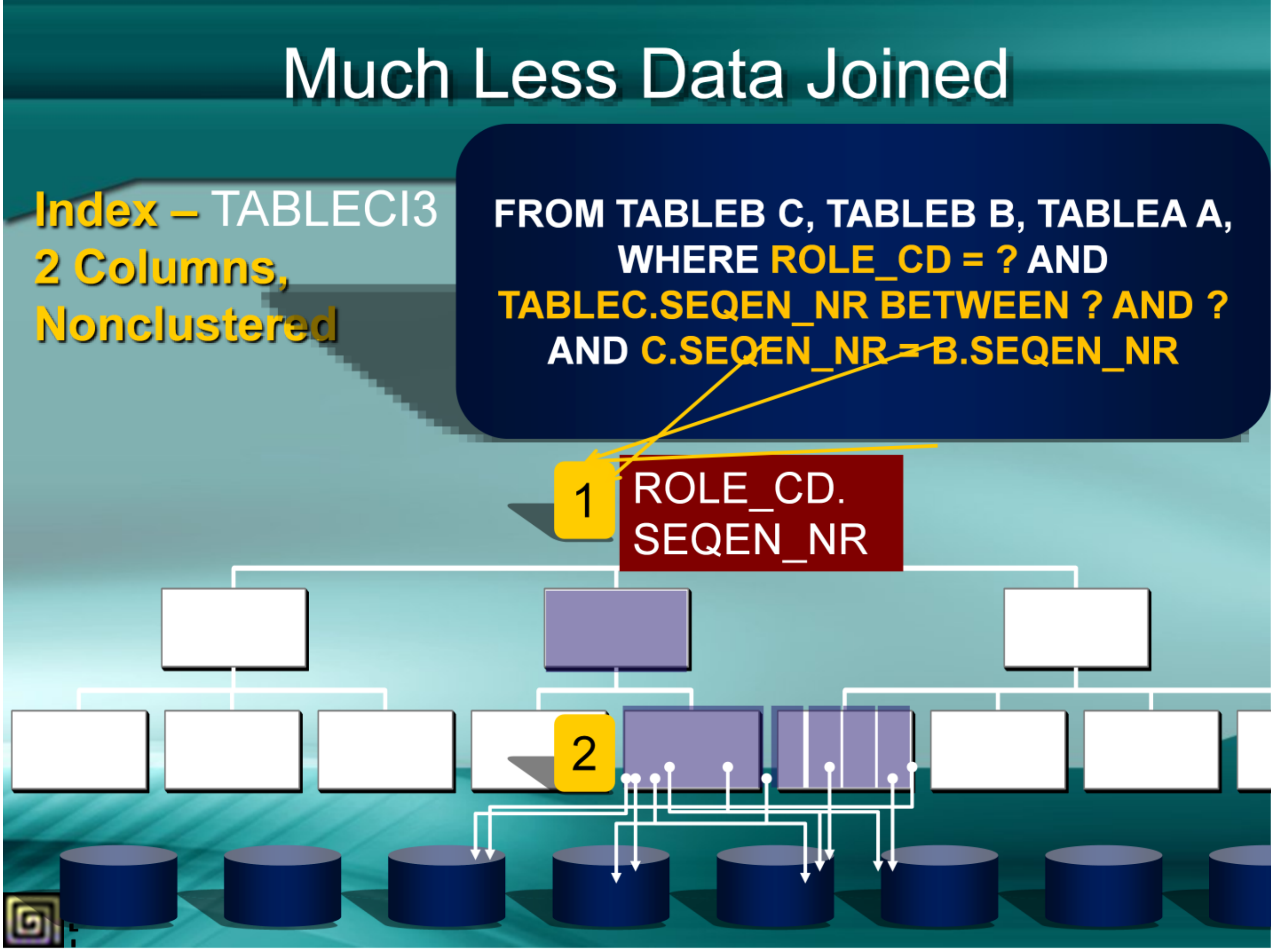
Notes: Performance rule violations usually result in increased CPU or I/O, time to fix the mistake, and ultimately, a cost to the business unit.

What will the cost be? Depends on the mistake and the frequency of the mistake.

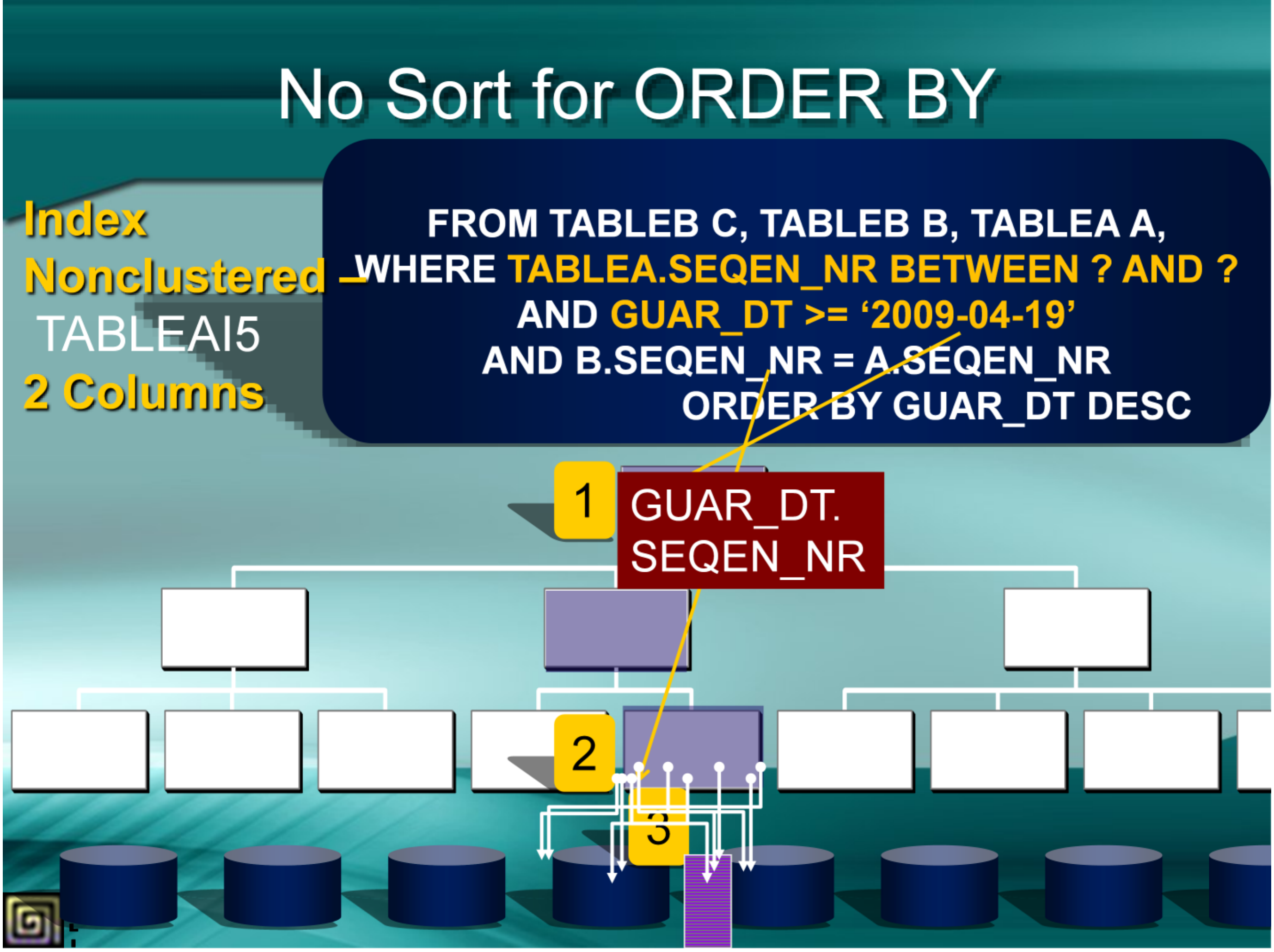
The following presentation will show real case studies of the actual cost of the mistakes.



Notes: Adding SEQEN_NR to the index and manual Predicate Transitive Closure moves the BETWEEN filter to the 2nd point of filtering. Adding SSN_NR to the index eliminates all random I/O to the data pages.



Notes: Adding SEQEN_NR to the filtering index allows the joining to be combined.



Notes: By adding SEQEN_NR to the index the last step combines the joining, filtering and sequencing.

Cost of SQL Inefficient Use

- Cost CPU & I/O
- Cost time to fix
- Cost $$$$$$$$$

How much does it cost?

1. Use joins over subqueries when detail row information is required
2. Use subqueries over joins when detail row information is not required
3. Use INNER JOIN over LEFT JOIN when exceptions are not expected or needed
4. Use CREATE GLOBAL TEMPORARY TABLE when 100% data is infrequently accessed
5. Use DECLARE GLOBAL TEMPORARY TABLE with a clustered index when DTT is large and data is frequently accessed

Notes: The cost depends on the multiplier. That's when performance and scalability issues become evident.

Raise Staff Skills to Reduce CPU Consumption

- Use of Powerful New and Old SQL Features
- Don’t Misuse SQL
- Don’t Use the Wrong Type of Temp Table
- Create Optimal Index Design
- Don’t allow Delayed Filtering
- Follow IBM’s 23 SQL Performance Rules

 39

Notes: Hopefully, you can avoid the mistakes that others have made by following a few performance rules discussed in this presentation.

IBM Db2 SQL Performance Rules

Table 8-4 Best practices for query design From IBM Data Virtualization Manager for z/OS

Category	Description
Efficient SQL	Do not code mathematics on columns in predicates.
	Sort only on the columns that are needed. No need to ORDERBY BY EMPNO,LASTNAME when you can ORDERBY EMPNO.
	Watch out for the LIKE predicate. `Begins With` logic is indexable. `Contains` is not indexable. `Ends With` is not indexible.
	Do not code `Not Between`. Rewrite it as `>HV or <HV`.
	Use `Fetch First XX Rows` whenever possible.
	Make sure cardinality statistics exist for all columns in all tables.
	Code `Not Exists` over `Not In`. Both are stage 2 predicates but `Not Exists` typically outperforms the `Not In`, especially if the list is long.
	When joining two tables the execution is faster if the larger table is on the left side of the join.
	Code `WHERE` clauses with columns that have unique or good indexes.
	Prioritize `WHERE` clauses to maximize their effectiveness. First code the `WHERE` column clauses that reference indexed keys, then the `WHERE` column clauses that limit the most data, and then the `WHERE` clauses on all columns that can filter the data further.

Notes:

IBM Db2 SQL Performance Rules

Good coding practice	When looking for a small set of records, try to avoid reading the full table by using an index and by providing any possible key values. You can also use more `WHERE` clauses so that the fetch goes directly to the actual records.
	All Case logic should have an `else` coded, which eliminates DB2 returning nulls by default if all the Case conditions are not met.
	Stay away from `Not` logic if possible.
	Minimize the number of times cursors are opened and closed.
	Code stage 1 predicates only. Rewrite any stage 2 predicates.
	Use `FOR FETCH ONLY` on all read only cursors.
	Reduce the number of rows to process early by using Sub-selects and `WHERE` predicates.
	Avoid joining two types of columns and lengths when joining two columns of different data types or lengths. One of the columns must be converted to either the type or the length of the other column.
	Limit the use of functions against large amounts of data.

Notes:

IBM Db2 SQL Performance Rules

Category	Description
Reduce impacts to the DVM server	Do not code functions on columns in predicates.
	Minimize the number of times DB2 SQL statements are sent.
	Only select the columns that are needed.
Virtualization	Instead of using multi-level views, try to optimize your SQL queries. Creating views that call other views that call other views can result in joining to the same table multiple times when you only need it once. It creates millions of records in an underlying view where you are interested only in a handful of records.

Notes:

SQL Tuning Confidence Level

0% -- 100%

Speaker: Sheryl Larsen

Email Address:
Sheryl.Larsen@kyndryl.com

Phone: (630) 399-3330

Thank you and please share with friends!

Notes: