The Why and How of Database Modernization

Jim Ritchhart

Thursday, October 17, 2013, 2013

The Why and How of Database Modernization

- **IT Development Goals**
  - Fast delivery
  - Accurate delivery
  - Fast execution of programs
  - **Flexibility to ever changing business needs**

The Why and How of Database Modernization

Why Modernize?

Performance

- SQL scales better as rows to process increase

Flexibility

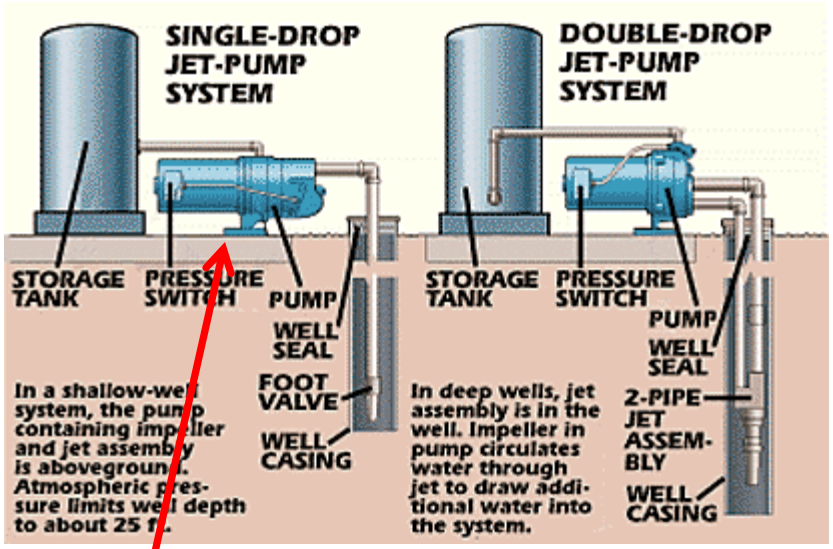
- Faster / Easier reaction to changing business

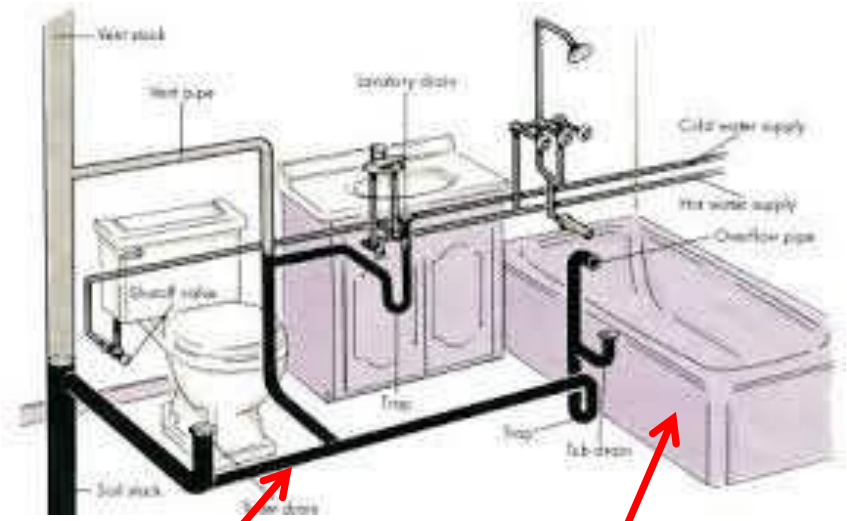
Strategic Direction

- IBM is putting their resources into SQL Enhancements

The Why and How of Database Modernization

Why Modernize?





1960s I/O routines. QDBGET, QDBGETKY

Database

UI

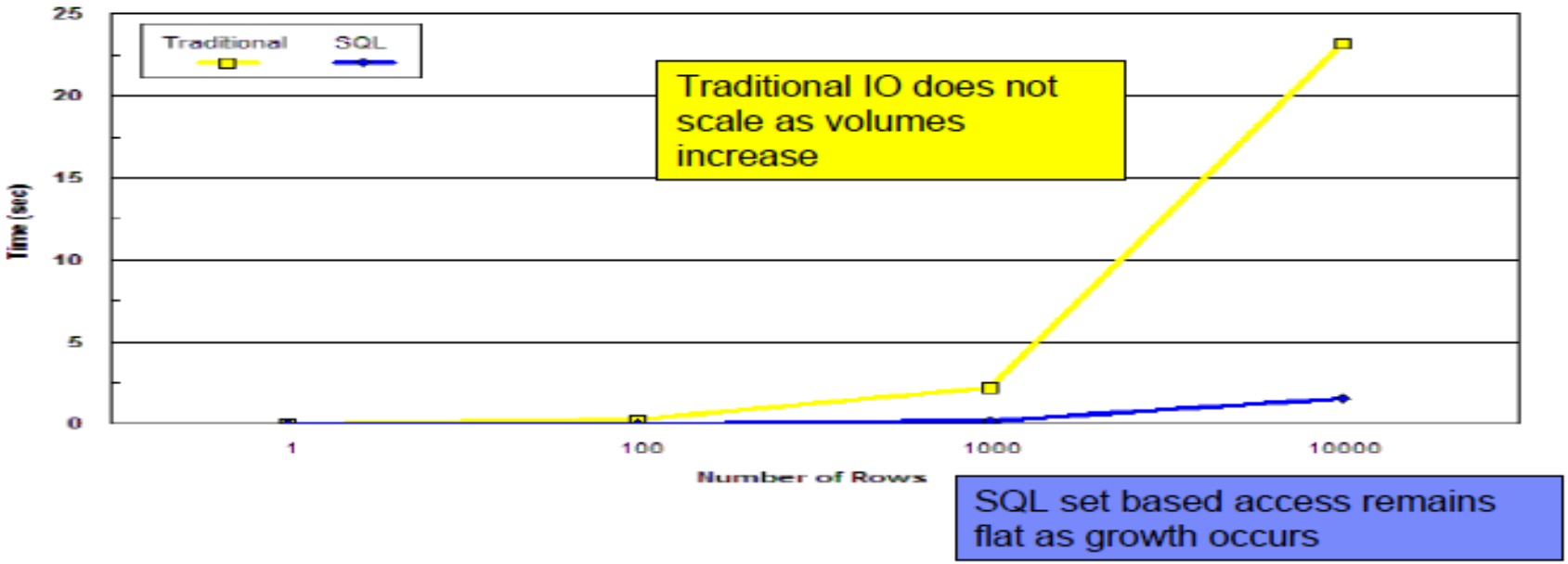
The Why and How of Database Modernization

Traditional I/O

- As the number of rows grows, So does the time to process them!!

SQL and Scalability

- As growth occurs, Native I/O will no longer drive the POWER based processors
- Throwing hardware at a problem is no longer an option
- Application changes are inevitable



10

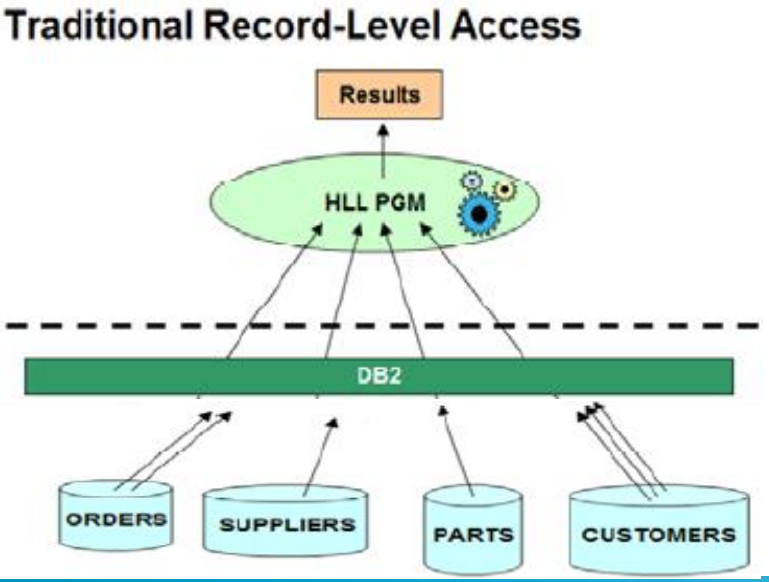
The Why and How of Database Modernization

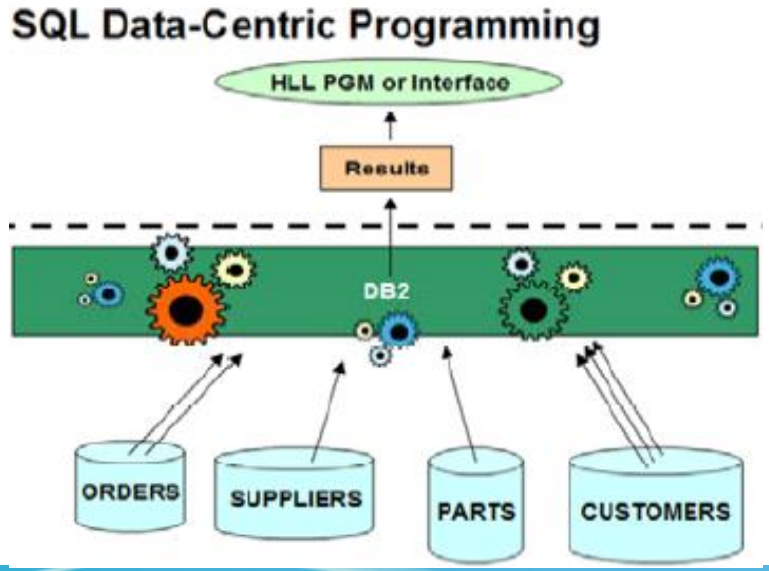
Data-centric programming

Data-centric programming puts specific business rules into the database.

This has many advantages:

- *Consistency – The rules are implemented at a DB level making that rule in effect for any action against the table / column.*
- *Performance – DB level actions are significantly faster then program level actions.*
- *Less programming required – Business rules are already implemented and do not have to be coded for each program.*





The Why and How of Database Modernization

Modernization DB2 Database – 5 Phased Approach

- **Migrate**
  - Reverse engineering the existing DDS database objects to SQL DDL objects
- **Isolate**
  - Accessing the new DB2 database via SQL views and IO data access modules
- **Correct**
  - Eliminating design flaws inherited by DB2 from the legacy database
- **Secure**
  - Securing the DB2 database from unauthorized access
- **Enhance**
  - Enhancing the DB2 with advanced capabilities

The Why and How of Database Modernization

Modernizing DB2 Database – A phased approach

Migrate – Phase1

- **Goal**

Reverse engineer existing DDS files to DDL tables

- **Strategy**

Replace DDS PF with DDL table but retain original record format ID

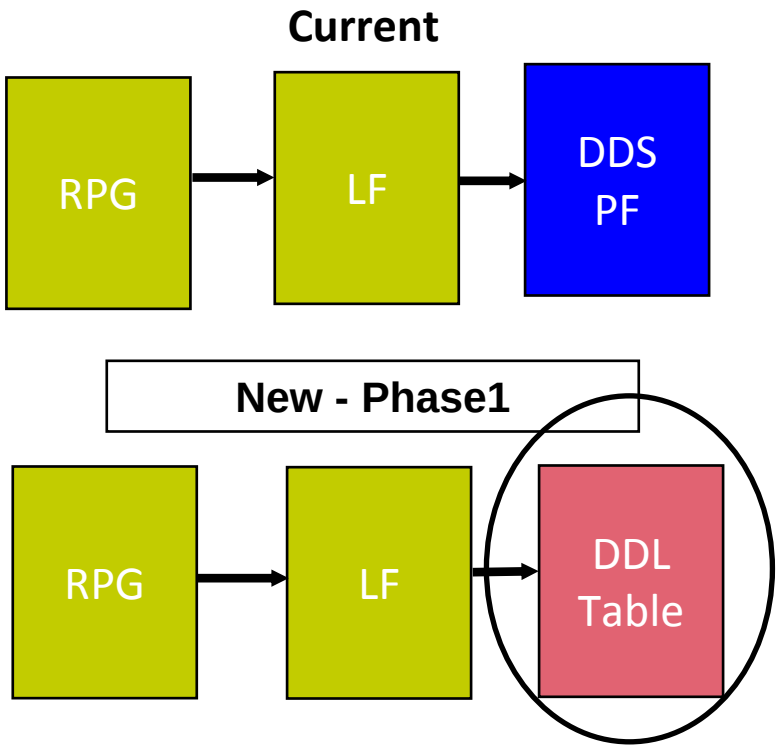
- **Benefits**

• No program changes are required

• Elimination of deleted records

• Faster read

• Concurrent write support



The Why and How of Database Modernization

Modernization DB2 Database

Goals:

- Create SQL Table to redirect queries to faster SQE
- Create Indexes that perform better and allow programs to use them w/o changes
- Create the ability to add/change columns w/o requiring program changes.
- **Retain Format Level IDs so programs are unaffected**

The Why and How of Database Modernization

Modernization DB2 Database

Process:

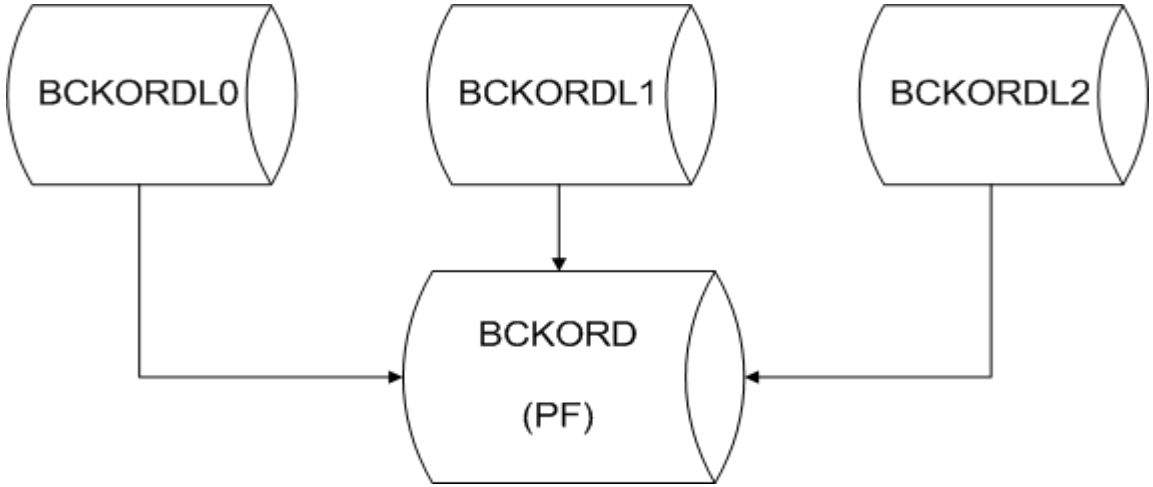
- Convert PF to SQL Table with new name
- Create SQL indexes to replace any implicitly created keyed access paths that exists for DDS files.
  - System Catalog
- Create “Surrogate” LF with same name as original PF retaining the format.
- Modify existing LFs to reference new SQL Table name
- Modify existing LFs to include “FORMAT” keyword getting it’s format information from “Surrogate” LF.

The Why and How of Database Modernization

Modernization DB2 Database

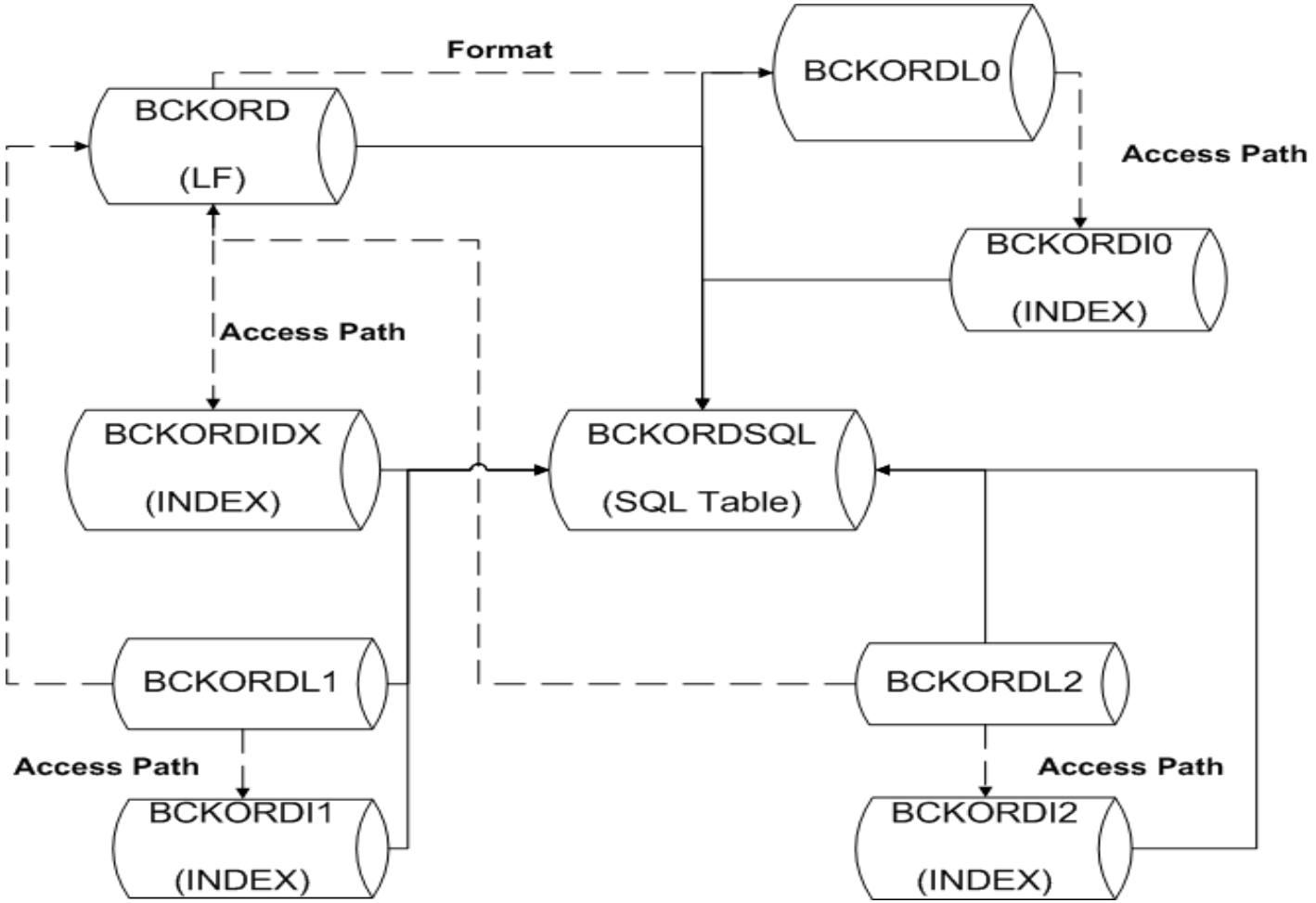
Process:

- Change PF into LF (Name / format / fields remains the same)
- Create New SQL Table as BCKORDSQL
- Create Index for each LF that does not have Joins or Select / Omits (Compile Indexes before LF)
- Add FORMAT keyword to existing LF that get it's format from the old PF i.e. FORMAT(BCKORDRC)



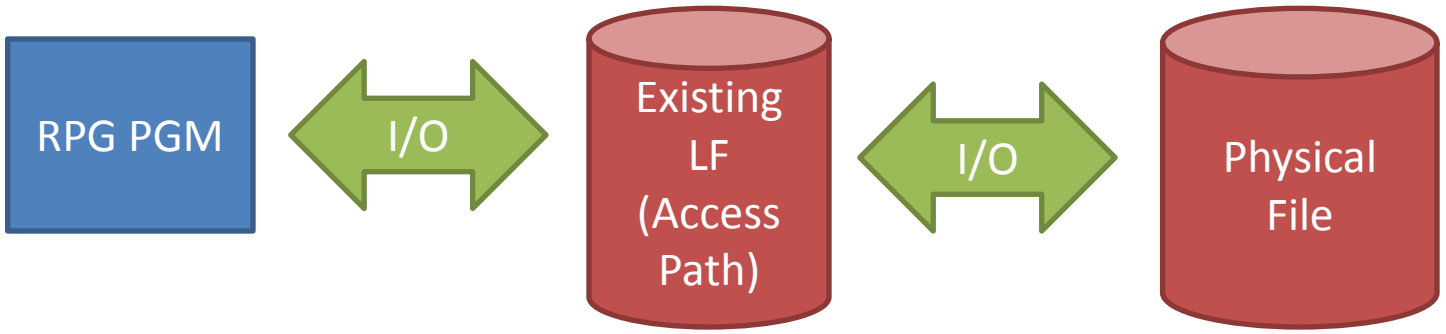
The Why and How of Database Modernization

Modernization DB2 Database



The Why and How of Database Modernization

Modernization DB2 Database – Prior to Modernization

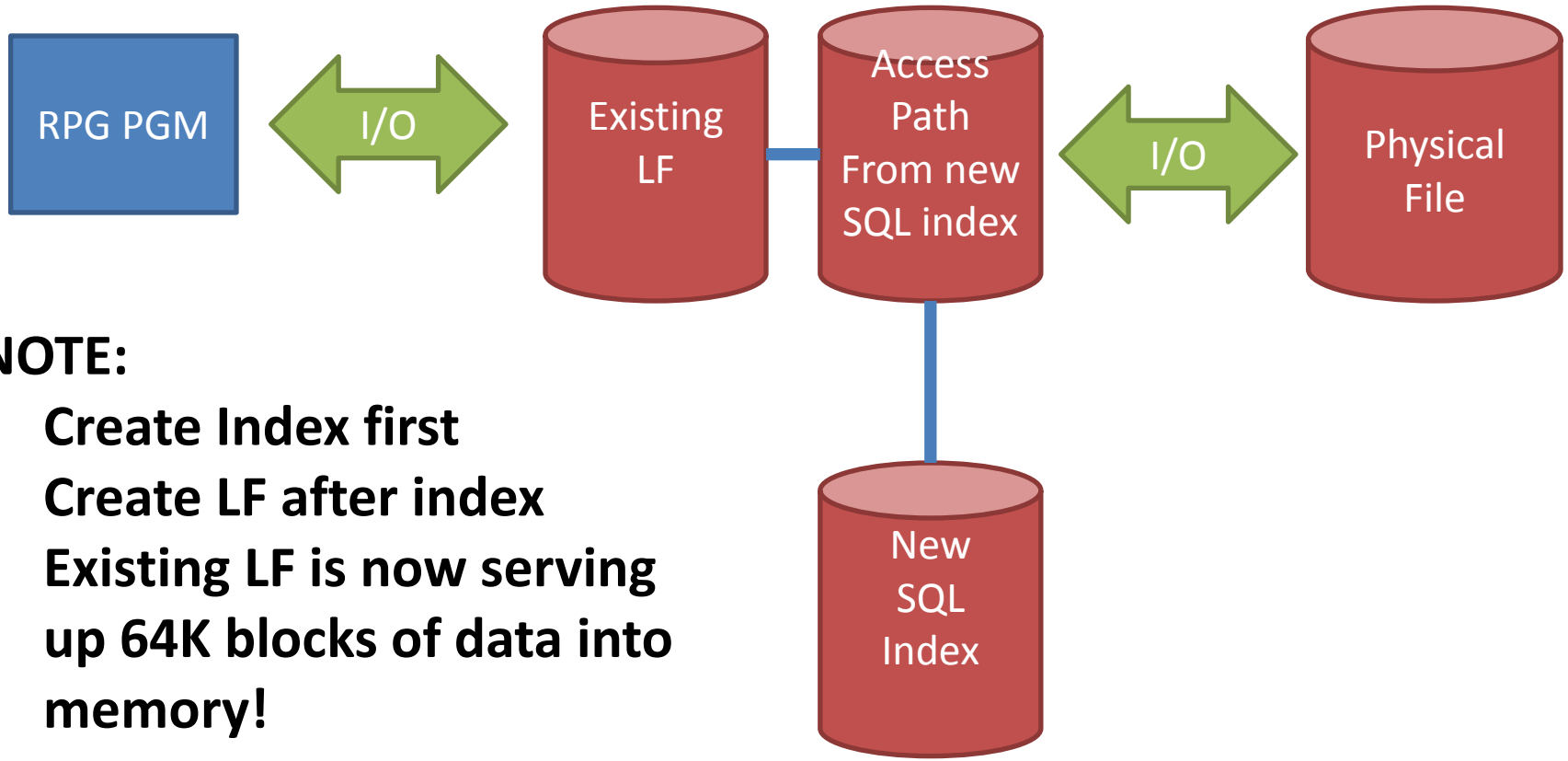


NOTE:

- **Existing LF is serving 8K blocks of data to program in memory.**

The Why and How of Database Modernization

Modernization DB2 Database – After Modernization



NOTE:

- **Create Index first**
- **Create LF after index**
- **Existing LF is now serving up 64K blocks of data into memory!**

The Why and How of Database Modernization

Modernization DB2 Database – Original BCKORD

```
=>                                                                BCKORD
PF  .....A..........T.Name++++++RLen++TDpB......Functions+++++++++++++++++
00        A           R BCKORDRC
00        A             BKORDERN      10          COLHDG('ORDER'  'NUMBER')
01        A             BKWHSECD       4A         COLHDG('WAREHOUSE CODE')
00        A             BKITEMNO      25A         COLHDG('ITEM'  'NUMBER')
00        A             BKVENDORNO    10A         COLHDG('VENDOR'  'NUMBER')
00        A             BKQTYSH        9P 0       COLHDG('QTY'  'SHIPPED')
01        A             BKQTYOR        9P 0       COLHDG('QTY'  'ORDERED')
00        A             BKDTENT        8S 0       COLHDG('DATE'  'ENTERED')
00        A             BKREASON     100A         COLHDG('REASON')
00        A           K BKORDERN
00        A           K BKITEMNO
```

```
 SEU==>                                                           BCKORDL1
 FMT LF  .....A..........T.Name++++++.Len++TDpB......Functions+++++++++++++++++
         ***************** Beginning of data ***************************************
0001.00        A           R BCKORDRC                    PFILE(BCKORD)
0003.00        A           K BKITEMNO
```

The Why and How of Database Modernization

Modernization DB2 Database – New DDL BCKORDSQL

```
0001.00 CREATE TABLE BCKORDSQL
0001.02  BCKORDSQL_PK    FOR BCKORDPK INTEGER
0001.03                  GENERATED BY DEFAULT AS IDENTITY
0001.04                  (START WITH 1  INCREMENT BY 1  NO MINVALUE
0001.07                   NO MAXVALUE  CYCLE  CACHE 20  NO ORDER ),
0002.00 ORDER_NUMBER              FOR BKORDERN    CHAR(10) CCSID 37 DEFAULT '',
0003.00 WAREHOUSE_CODE            FOR BKWHSECD    CHAR(4)  CCSID 37 DEFAULT '',
0004.00 ITEM_NUMBER               FOR BKITEMNO    CHAR(25) CCSID 37 DEFAULT '',
0005.00 VENDOR_NUMBER             FOR BKVENDORNO  CHAR(10) CCSID 37 DEFAULT '',
0006.00 QUANTITY_SHIPPED          FOR BKQTYSH     NUMERIC(9,0)      DEFAULT 0,
0007.00 QUANTITY_ORDERED          FOR BKQTYOR     NUMERIC(9,0)      DEFAULT 0,
0008.00 DATE_ENTERED              FOR BKDTENT     NUMERIC(8,0)      DEFAULT 0,
0009.00 REASON_DESCRIPTION        FOR BKREASON    CHAR(100)         DEFAULT ''
0009.01 ROW_CHANGE_TIMESTAMP      FOR CHANGE_TS   _TIMESTAMP
0009.02                       FOR EACH ROW ON UPDATE AS ROW
0010.00 )BCKORDRC;
0010.01
0010.02   ALTER TABLE BCKORDSQL
0010.03    ADD CONSTRAINT PK_BCKORDSQL_BCKORDPK
0010.04    PRIMARY KEY( BCKORDPK ) ;_
0011.00
0012.00 RENAME BCKORDSQL TO BACKORDERS
0013.00 FOR SYSTEM NAME BCKORDSQL;
```

The Why and How of Database Modernization

Modernization DB2 Database – Original BCKORD

```
SEU==>                                                                BCKORD
FMT  LF  ....A.....T.Name++++++.Len++TDpB......Functions+++++++++++++++++
         ********** Beginning of data **************************************
0009.00      A           R BCKORDRC                      PFILE(BCKORDSQL)
0010.00      A             BKORDERN      10              COLHDG('ORDER' 'NUMBER')
0011.00      A             BKWHSECD       4A             COLHDG('WAREHOUSE CODE')
0012.00      A             BKITEMNO      25A             COLHDG('ITEM' 'NUMBER')
0013.00      A             BKVENDORNO    10A             COLHDG('VENDOR' 'NUMBER')
0014.00      A             BKQTYSH        9P 0           COLHDG('QTY' 'SHIPPED')
0015.00      A             BKQTYOR        9P 0           COLHDG('QTY' 'ORDERED')
0016.00      A             BKDTENT        8S 0           COLHDG('DATE' 'ENTERED')
0017.00      A             BKREASON     100A             COLHDG('REASON')
0018.00      A           K BKORDERN
0019.00      A           K BKITEMNO
```

```
SEU==> _____                                                    BCKORDIDX
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ********** Beginning of data **************************************
0001.00  CREATE INDEX  BCKORDIDX
0002.00  ON BCKORDSQL
0003.00  (BKORDERN, BKITEMNO);
0004.00
0005.00  RENAME INDEX  BCKORDIDX                          _
0005.01  TO  BCKORDER_BY_ORDER_ITEM_NO
0006.00  FOR SYSTEM NAME BCKORDIDX;
```

```
SEU==> _____                                                    BCKORDI1
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ********** Beginning of data **************************************
0001.00  CREATE INDEX  BCKORDI1
0002.00  ON BCKORDSQL
0003.00  (BKITEMNO);
0004.00
0005.00  RENAME INDEX  BCKORDI1                           _
0005.01  TO BACKORDER_BY_ITEMNUMBER
0006.00  FOR SYSTEM NAME BCKORDI1 ;
```

```
SEU==> _____                                                    BCKORDL1
FMT LF .....A.........T.Name++++++.Len++TDpB......Functions++++++++++++++++++
       ********** Beginning of data **************************************
0001.00      A          R BCKORDRC                 PFILE(BCKORDSQL)
0002.00      A                                     FORMAT(BCKORD)
0003.00      A          K BKITEMNO
```

The Why and How of Database Modernization

Modernization DB2 Database – Original BCKORD

```
 SEU==> ____________________________________________________________________ BCKORDV0
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
        ****************** Beginning of data ***************************************
0001.00 CREATE VIEW BCKORDV0 AS
0002.00 SELECT *
0003.00 FROM BCKORDSQL;
0004.00
0005.00 RENAME VIEW BCKORDV0
0006.00 TO BACKORDERS_ALL_COLUMNS
0007.00 FOR SYSTEM NAME BCKORDV0;
```

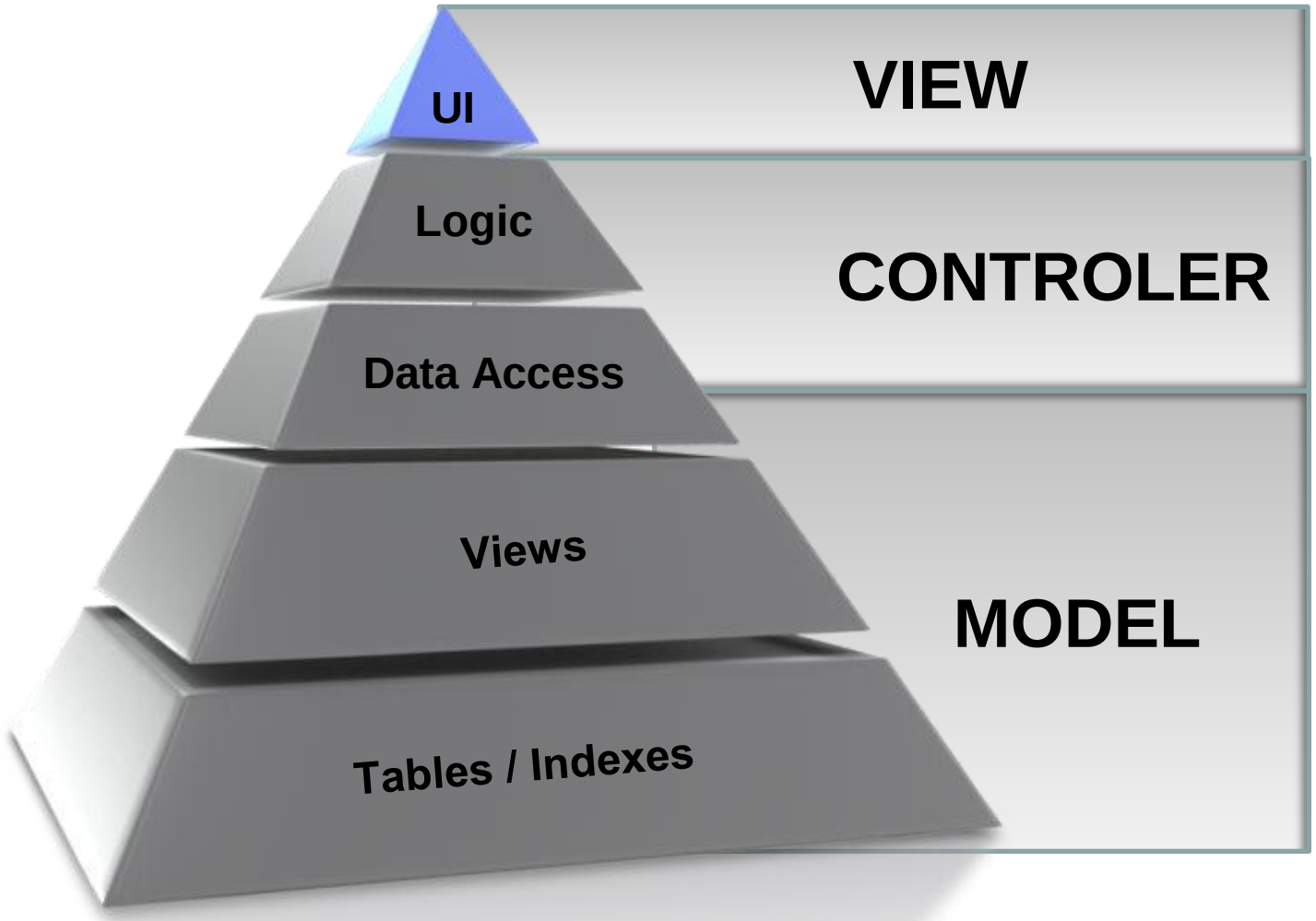
The Why and How of Database Modernization

Modernization DB2 Database – Original BCKORD

Opt	Member	Type	Text
	BCKORD	LF	Backorder LF and format
✓	BCKORDIDX	SQLINDX	Backorder index support for BCKORD
✓	BCKORDI1	SQLINDX	Backorder index support for BCKORDL1
	BCKORDL1	LF	Backorder LF by Item Number
✓	BCKORDSQL	SQLTABL	Backorder Physical Table
✓	BCKORDV0	SQLVIEW	Backorder view - All columns

The Why and How of Database Modernization

Modernization DB2 Database



The Why and How of Database Modernization

Real Life Example #1

Need to add and expand columns in a table

Requirements:

- Expand Name and Address columns in our marketing table in order to account for longer data for our Mexico business.
- Add new language preference column in our contact table to support web.

The Why and How of Database Modernization

Real Life Example #1

Need to add and expand columns in a table

Requirements:

- Expand Name and Address columns in our marketing table in order to account for longer data for our Mexico business.
- Add new language preference column in our contact table to support web.

Issue:

- How do we do this w/o modifying and recompiling 100s of programs?

The Why and How of Database Modernization

Real Life Example #1

Need to add and expand columns in a table

Requirements:

- Expand Name and Address columns in our marketing table in order to account for longer data for our Mexico business.
- Add new language preference column in our contact table to support web.

Issue:

- How do we do this w/o modifying and recompiling 100s of programs?

Solution:

- ALTER TABLE CONTACTS ADD COLUMN LANGPRF CHAR(5) ;
- ALTER TABLE ***library/sqltable*** ALTER COLUMN ADDRESS_LINE1 SET DATA TYPE CHAR (50) CCSID 37 NOT NULL ;

The Why and How of Database Modernization

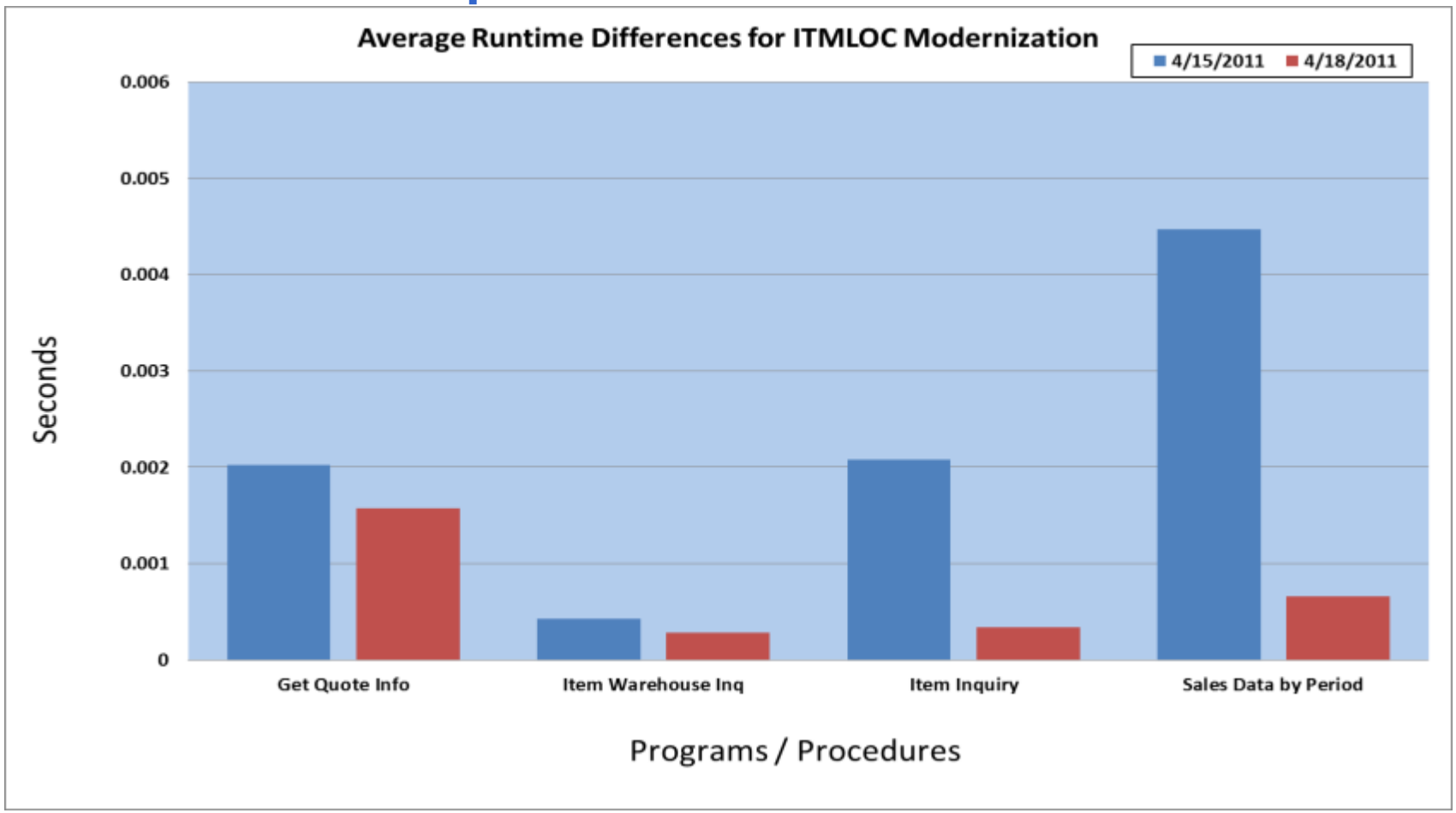
Real Life Example #1

Need to add and expand columns in a table

- You only have to change / recompile the programs that need the new column OR that need the expanded column size.
- Less risk to project
- Flexibility in implementation
  (We made a DB changes 1 week prior to program changes.)
- Less time to implement
- Faster reaction time to business needs!!!

The Why and How of Database Modernization

Real Life Example #2 – Modernization of ITMLOC



The Why and How of Database Modernization

Real Life Example #3

Business Analyst complaining about query taking too much time.

select * from *filename* **where** *datefield* **> 20101001 and** *zip* **= '60031' and** *address* **like 'PO BOX%'**

Prior to Performance Tuning

- File had a LF by Date by Zip code
- Took 25 minutes to query 100 million Rows of data.

Seems Reasonable?

The Why and How of Database Modernization

Real Life Example #3

Business Analyst complaining about query taking too much time.

select * from *filename* **where** *datefield* **> 20101001 and** *zip* **= '60031' and** *address* **like 'PO BOX%'**

Prior to Performance Tuning

- File had a LF by Date by Zip code
- Took 25 minutes to query 100 million Rows of data.

- **EVI or Encoded Vector Index is a new type of Index unique to DDL.**
- **Use EVI for "low cardinality" type of situations. I.E. Low number of unique occurrences in the data relative to the number of rows of data in file.**

The Why and How of Database Modernization

Real Life Example #3

Business Analyst complaining about query taking too much time.

select * from *filename* **where** *datefield* **> 20101001 and** *zip* **= '60031' and** *address* **like 'PO BOX%'**

Prior to Performance Tuning

- File had a LF by Date by Zip code
- Took 25 minutes to query 100 million Rows of data.

- **EVI or Encoded Vector Index is a new type of Index unique to DDL.**
- **Use EVI for "low cardinality" type of situations. I.E. Low number of unique occurrences in the data relative to the number of rows of data in file.**

After EVI Index:
Created Encoded Vector Index by Date
Take 1 minute 45 seconds to run.

Database Modernization - SQL

Think in terms of SETS not records

What NOT to do

```
exec sql declare c1 scroll cursor for mySQL;
exec sql prepare mySQL from : gw_MySQL;
exec sql open c1;
exec sql fetch first from c1 for 0500 rows into :gd_sqldata;
exec sql get diagnostics :Gw_Rows = row_count;

dow Gw_Read = *zeros;
    For I = 1 to Gw_Rows;
        %occur( Gd_sqldata) = I;
        Exec Sql Select Wbactv into :Gw_WbActv
                        from OrderHwbsq
                        Where Wbwbid = :gdf_Wtwbid and
                              Wbsufx = 0;
        If SqlCode = *zeros and Gw_WbActv <> *blanks;
           Iter;
        Endif;
```

Database Modernization - SQL

Think in terms of SETS not records

Here's an alternative

```
Create view OrderhwtV1 as
       select A.Wtcsnr,       A.Wtnam,        A.WtSt,
              A.Wtwbid,       A.Wtinva,       A.Wtuedt,
              A.Wttime,       A.Wtbatc,
              Case
                when B.csnam is not null then b.csnam else A.WtNam
              end as Cusnam,
              Case
                when B.csST  is not null then b.csst  else A.Wtst
              end as Cusst,
              Case
                when B.Cs3lnm is not null then b.cs3lnm else ' '
              end as Cuslnm
       from   OrderhWtsq a
       Left Exception join
              Orderhwbsq C on A.Wtwbid = C.wbwbid and C.wbsufx = 0
       left outer join
```

Database Modernization - SQL

Think in terms of SETS not records

Here's an alternative

(Response Time went from Seconds to Milliseconds)

```
exec sql declare c1 scroll cursor for mySQL;
exec sql prepare mySQL from : gw_MySQL;
exec sql open c1;
exec sql fetch first from c1 for 0500 rows into :gd_sqldata;
Gw_Read = sqlcode;
exec sql get diagnostics :Gw_Rows = row_count;

dow Gw_Read = *zeros;
    For I = 1 to Gw_Rows;
        %occur( Gd_sqldata) = I;
        // .....  Do some work
    Endfor;
    exec sql fetch next  from c1 for 0500 rows into :gd_sqldata;
    Gw_Read = sqlcode;
    exec sql get diagnostics :Gw_Rows = row_count;

Enddo;
```

Database Modernization - SQL

Input primary RPG program with Join Select/Omit LF

PROBLEM:

- **Input Primary RPG program**
- **IP File is Join LF with Select / Omit (Dynamic Selection)**
- **Implemented Level Breaks**
- **Users complaining that it was taking too long to run**

Database Modernization - SQL

Input primary RPG program with Join Select/Omit LF

SOLUTION:

- **Create VIEW with same fields as JLF**
- **Define cursor in one time section of program**
- **Fetch from cursor at each cycle**
- **Look for field value changes for Level Breaks**

- **Total development time was 30 minutes including testing.**

Database Modernization - SQL

Input primary RPG program with Join Select/Omit LF

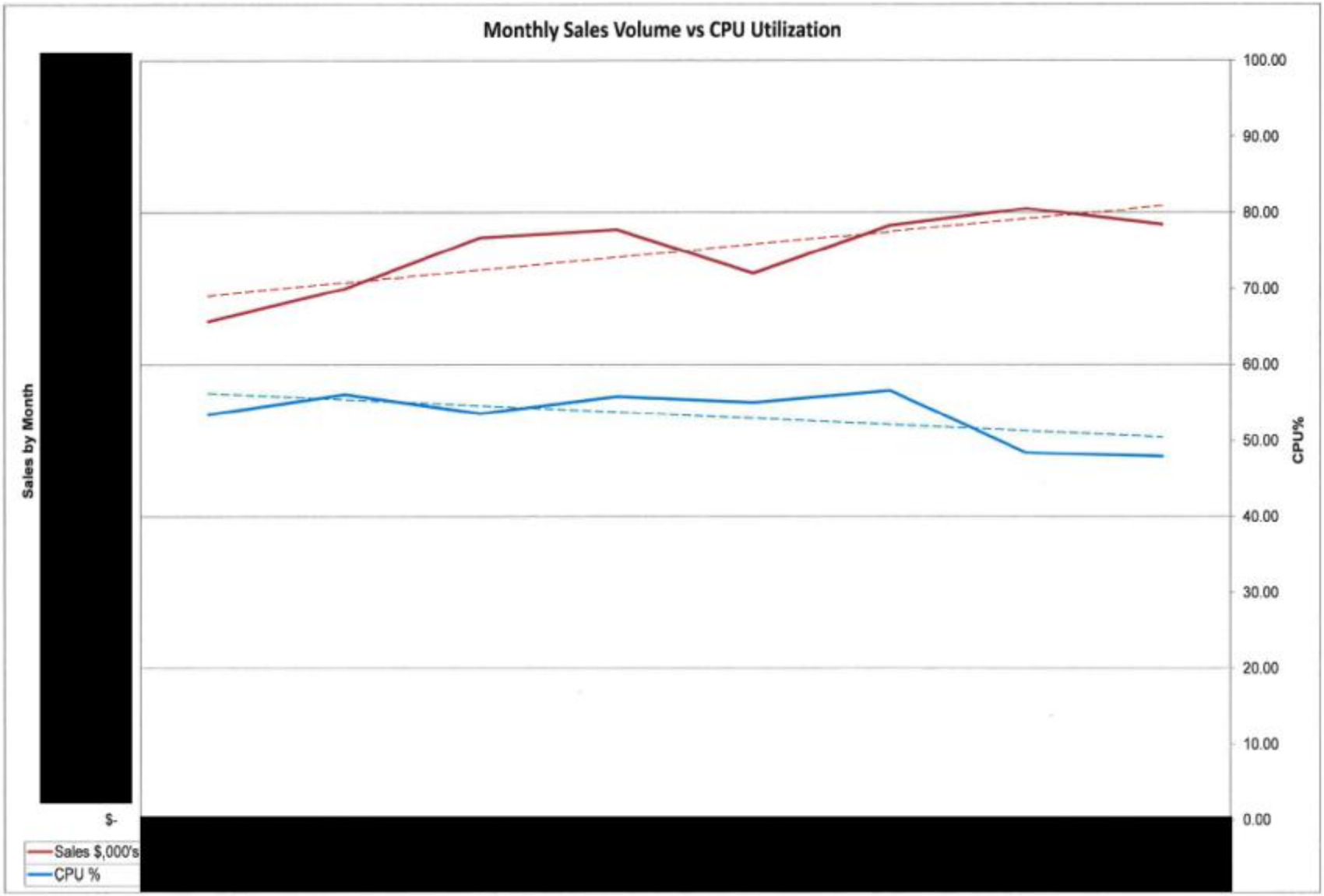
SOLUTION:

- **Run time went from 27 minutes to 2 minutes.**

```
exec sql  fetch next  from c1  into :inrec;
exec sql get diagnostics :wkrows = row_count;
if wkrows = 0   or  sqlcode > 0;
   *inlr = *on;
   return;
endif;

if   THNAM   <>  SHNAM    or
     THCSNR  <>  SHCSNR   or
     THINVN  <>  SHINVN   ;
     *inl1 = *on;
else;                          Level Breaks
     *inl1 = *off;
endif;
```

Database Modernization – System Performance



Database Modernization - Conclusion

Like most technologies, SQL and DDL on the iSeries has a startup cost. However, the benefits far out way the costs.

DDS and Native Record-Level Access are:

- Not Sustainable

Implementing DDL gives you immediate and measurable:

- Performance gains
- Flexibility
- Shorter response times to implement changes.