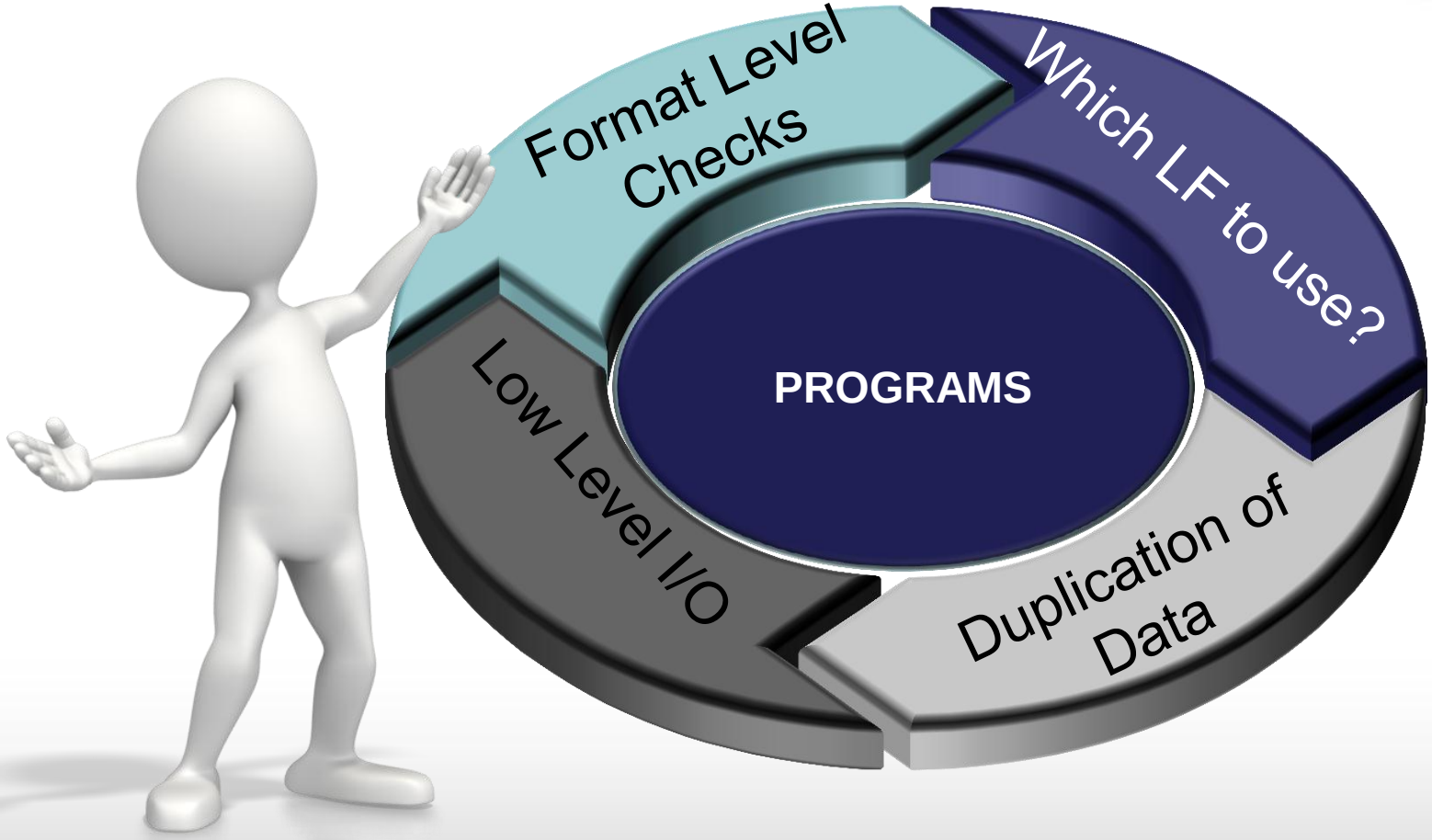
Data Centric Application Architecture

Jim Ritchhart

Data Centric Application Architecture

- **AS/400 shops Application Architecture**



Data Centric Application Architecture

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

Data Centric Application Architecture

GOALS OF DATA CENTRIC PROGRAMMING

- Drive as much work down into the database management system as possible.

Consistency

- Define business rules as part of the database
  - Rules apply to all application interfaces

Consistency

- Take advantage of SQL only capabilities
  - DDL modifications without affecting programs
  - Index Advisor, etc.

Flexibility

Performance

- Database evolves to:
  - Meet new requirements
  - Take advantage of new technology

Expandability

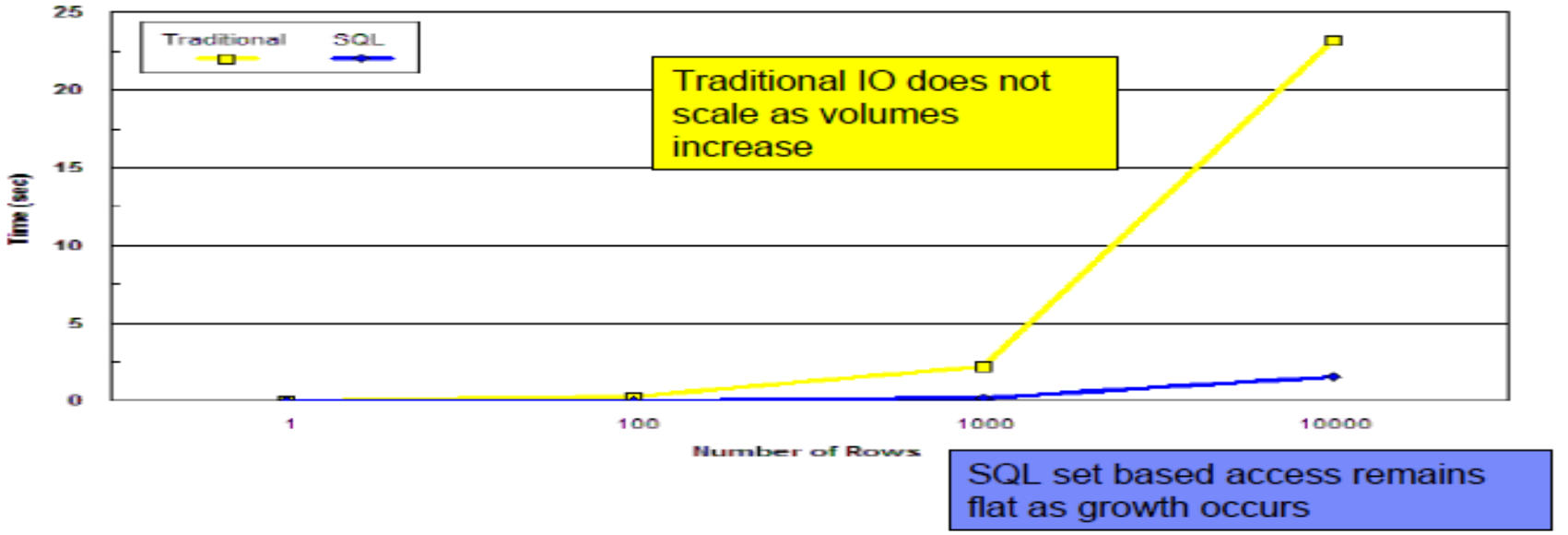
Data Centric Application Architecture

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



Data Centric Application Architecture

Most AS/400 Shops

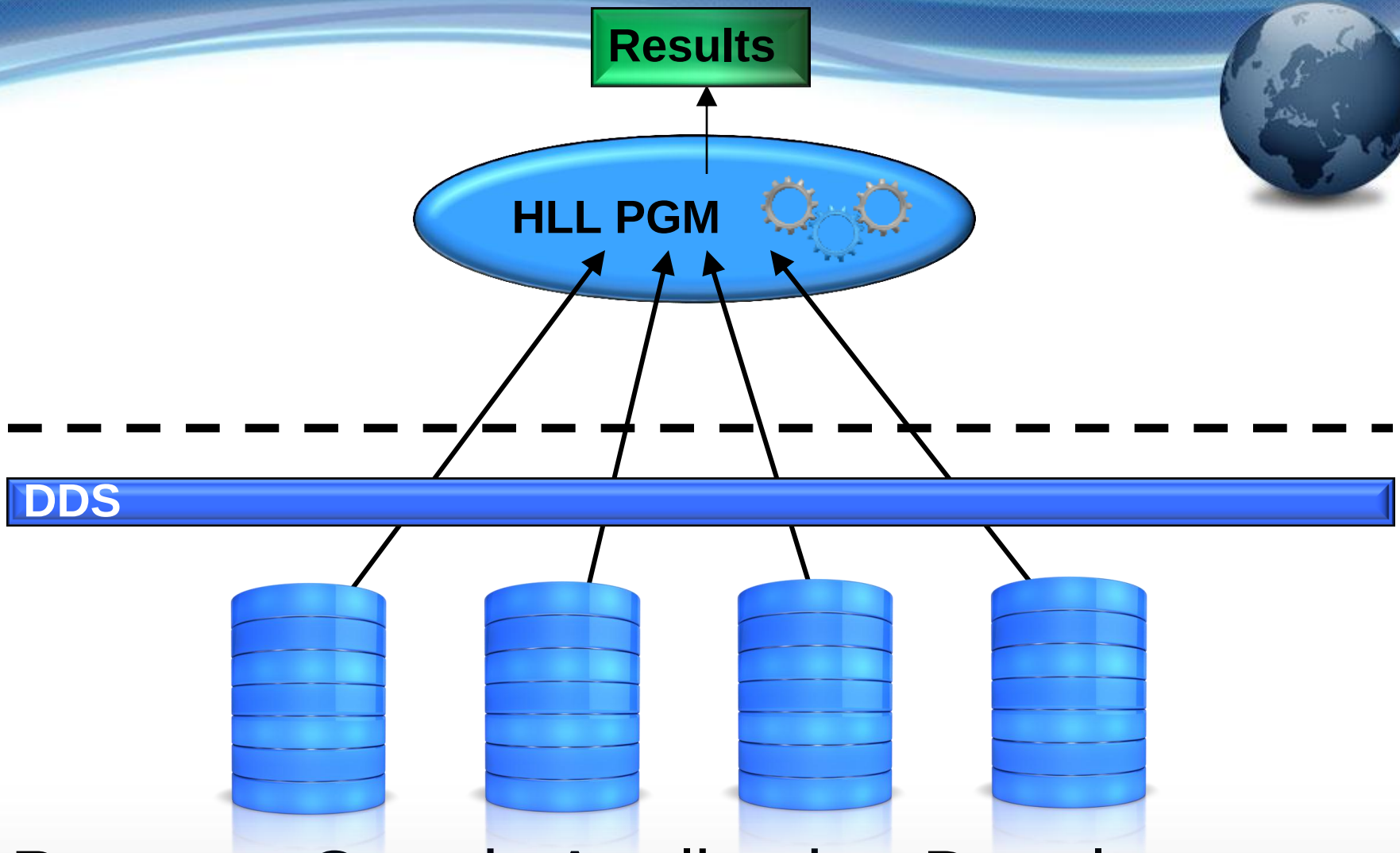
Most HLL / OO Shops

Program Centric → **Data Centric**

- Traditional I/O based
- Slows Down as # of rows increase
- Less Efficient
- Less Flexible
- Programs Determine Access Method of Data
- Single Layer Architecture
- Row Based Data Access

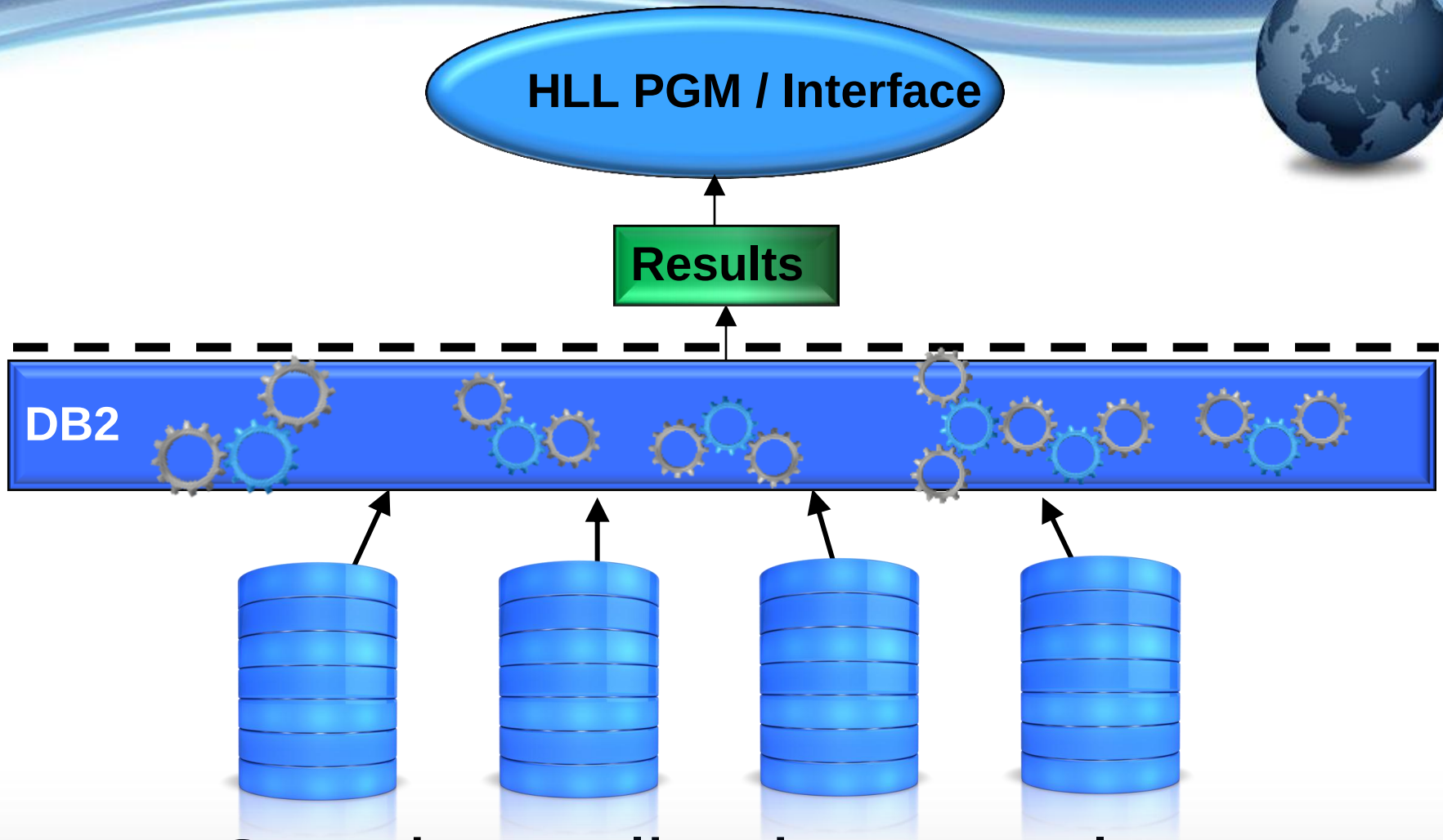
- SQL Based
- Speeds up as # of rows increase
- Very Efficient
- Very Flexible to Changing Business
- DataBase Determines Access Method of Data
- Multi Layer Architecture
- Set Based Data Access

Data Centric Application Architecture



Program Centric Application Development

Data Centric Application Architecture



Data Centric Application Development

Data Centric Application Architecture

Customer Master
Customer Number
Name
Address
City
State
Zip
Type

Order Header
Order Number
Customer Number
Total Order Dollars
Ship Date
Ship Via

- Program Understands Relationships DBMS Does Not
- Programmer must account for Data Integrity
- Programmer Determines what Keyed Access Path to chain to
- To see what Items Customer has ordered, program must do 4 chains (12 disk I/O s)
- Any Change to PF, requires recompile of all programs to prevent Level Checks
- Any change to data that is used for chaining requires a change to a key field

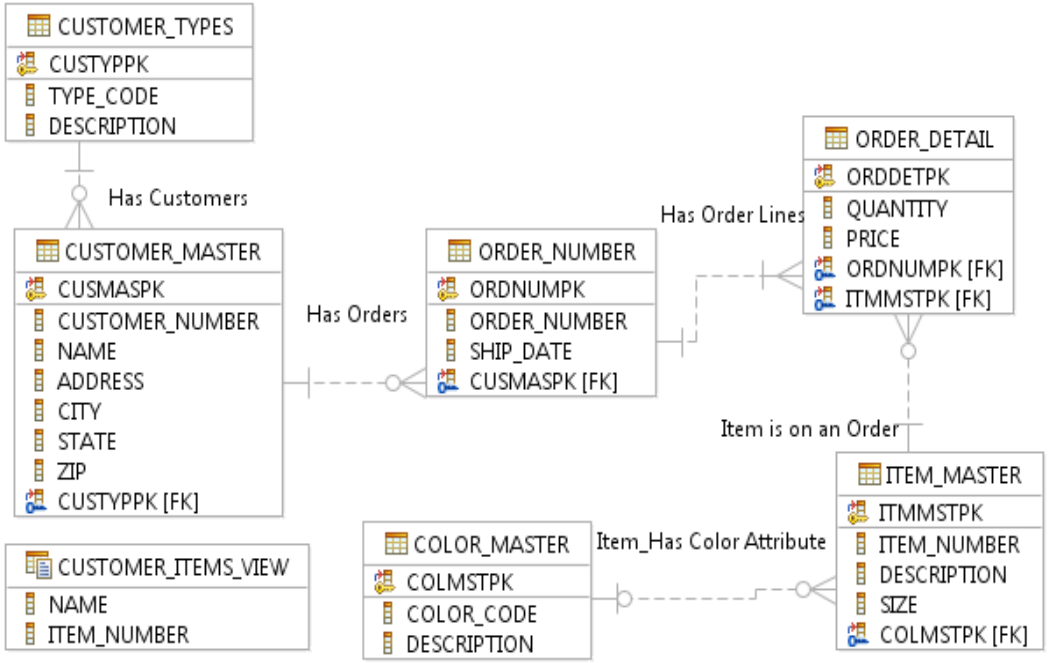
Order Detail
Order Number
Line Number
Item Number
Quantity
Price

Items
Item Number
Description
Color
Size

Program Centric File System

Data Centric Application Architecture

- DBMS understand relationship. Program does not care.
- DBMS accounts for Data Integrity
- DMBS determines how data is accessed. (i.e. what access path)
- To see what Items Customer has ordered, use CUSTOMER_ITMES view. (1 logical I/O. 6 physical I/Os)
- Any Change to PF has no effect on programs
- Keys are identity keys. They never change. Change can be done to all data w/o issues.



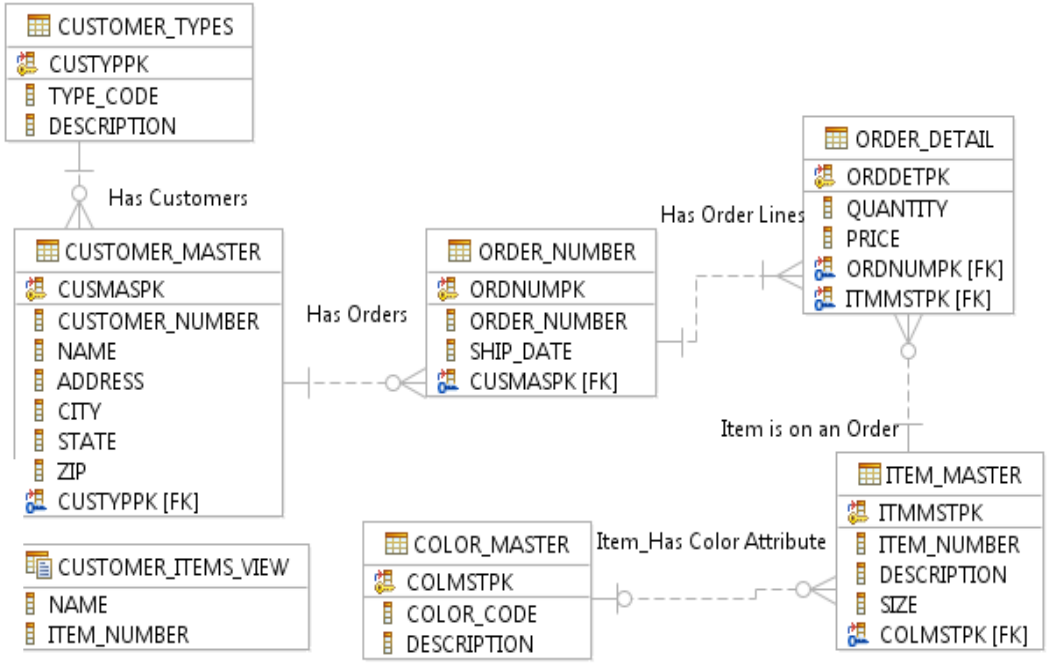

```
SELECT A.NAME,  D.ITEM_NUMBER
FROM CUSTOMER_MASTER      A
INNER JOIN ORDER_NUMBER    B
ON A.CUSMASPK  =  B.CUSMASPK
INNER JOIN ORDER_DETAIL       C
ON B.ORDNUMPK =  C.ORDNUMPK
INNER JOIN ITEM_MASTER       D
ON C.ITMMSTPK   =  ITMMSTPK
```

Data Centric File System

Data Centric Application Architecture

APPLICATION DESIGN RULES:

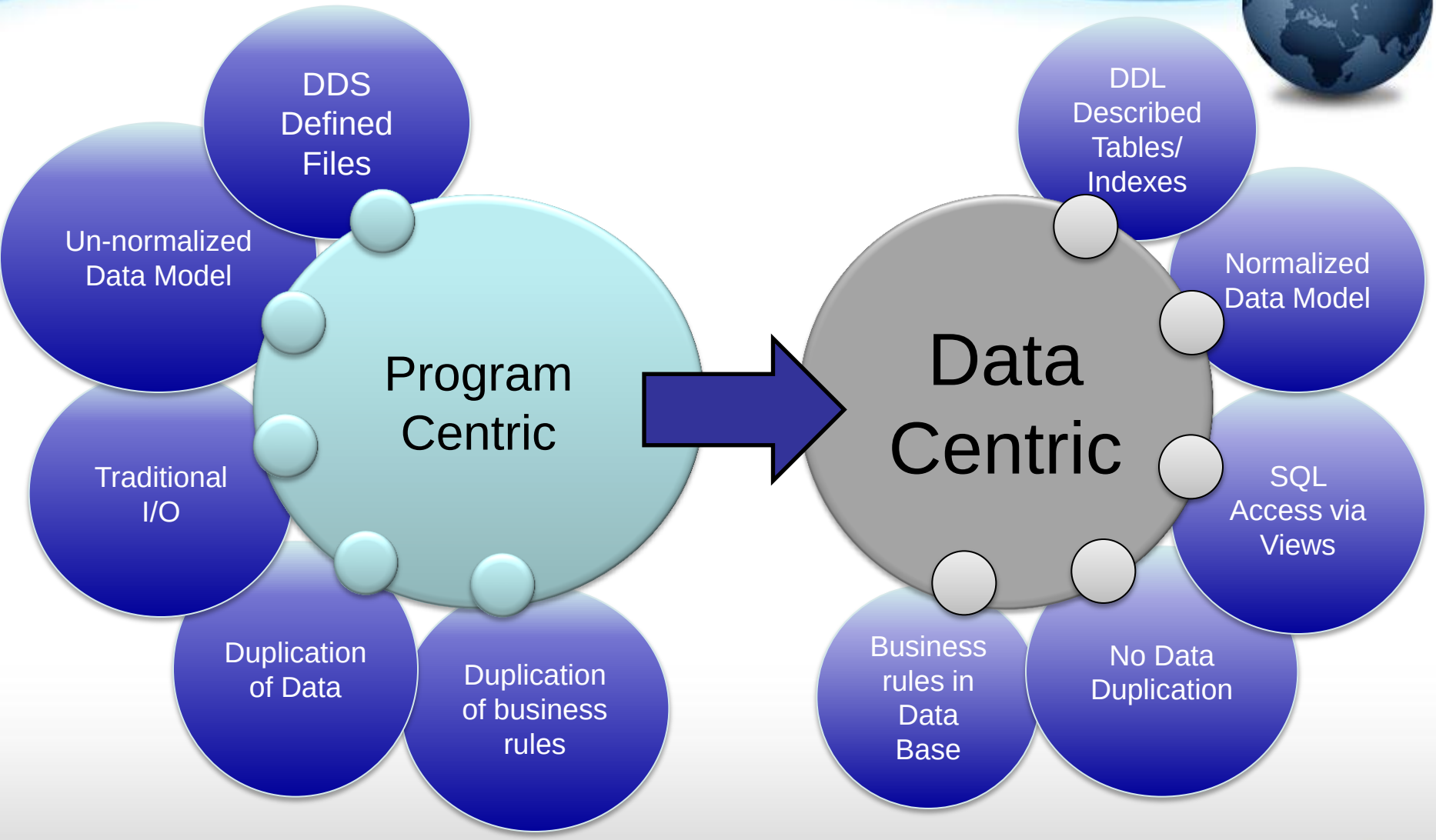
	CONSTRAINT	RULE
✓	Primary Key	IDs Parent Key
✓	Unique	IDs unique public key
✓	Foreign Key	Matches Primary key. Identifies rules for insert /Update/Delete.
✓	Check	Ensures proper values.




```
SELECT A.NAME,  D.ITEM_NUMBER
FROM CUSTOMER_MASTER      A
INNER JOIN ORDER_NUMBER    B
ON A.CUSMASPK  =  B.CUSMASPK
INNER JOIN ORDER_DETAIL       C
ON B.ORDNUMPK =  C.ORDNUMPK
INNER JOIN ITEM_MASTER        D
ON C.ITMMSTPK   =  ITMMSTPK
```

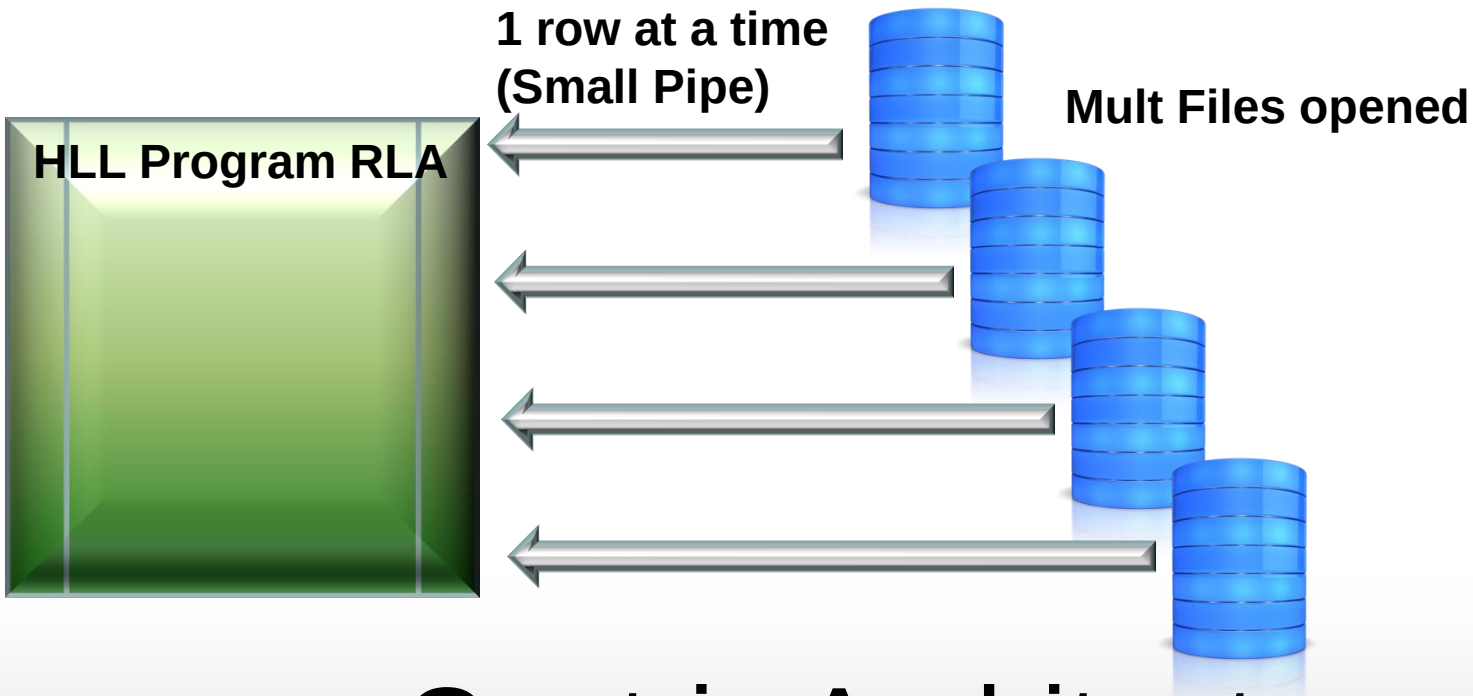
Data Centric File System

Data Centric Application Architecture



Data Centric Application Architecture

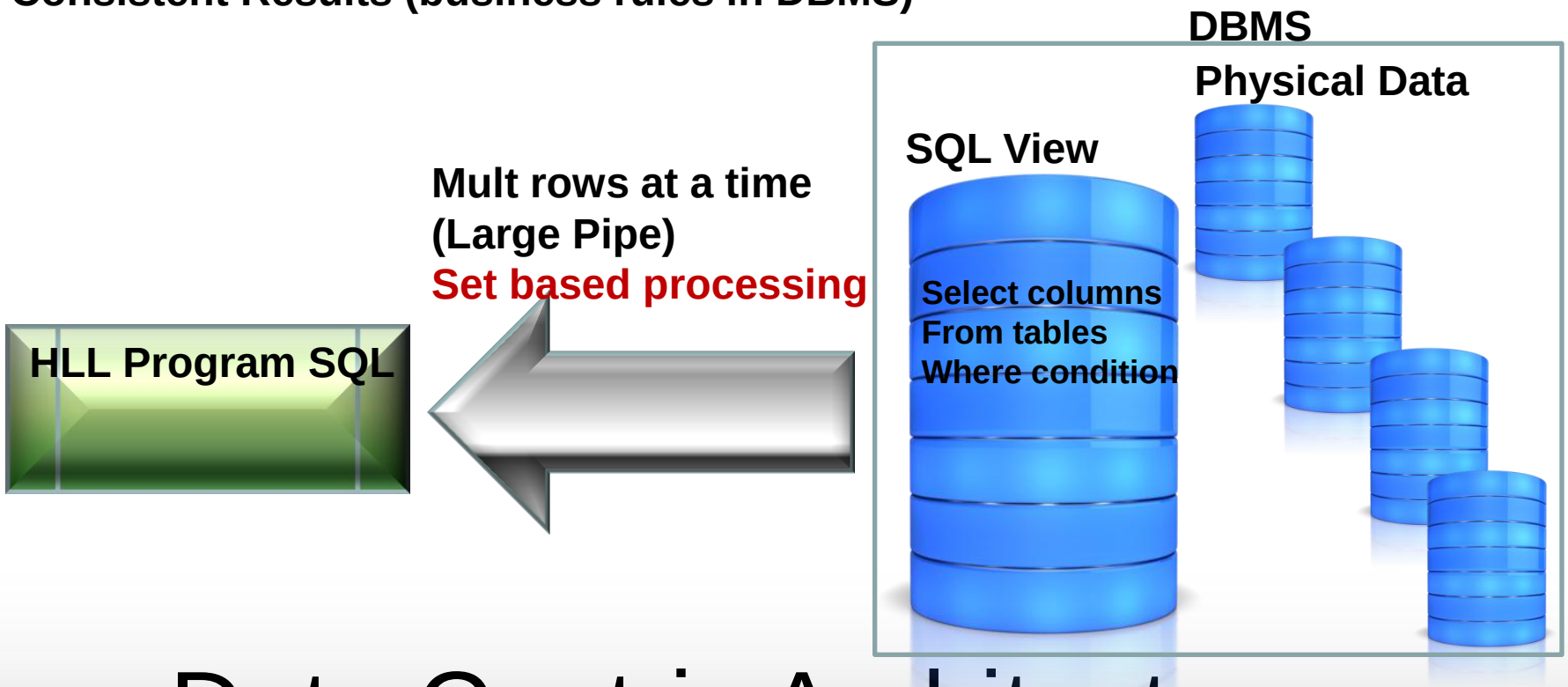
- **Slower Processing**
- **Larger Application (more lines of code)**
- **DBMS resources are not doing much work**
- **Inconsistent results (business rules in each program**



Program Centric Architecture

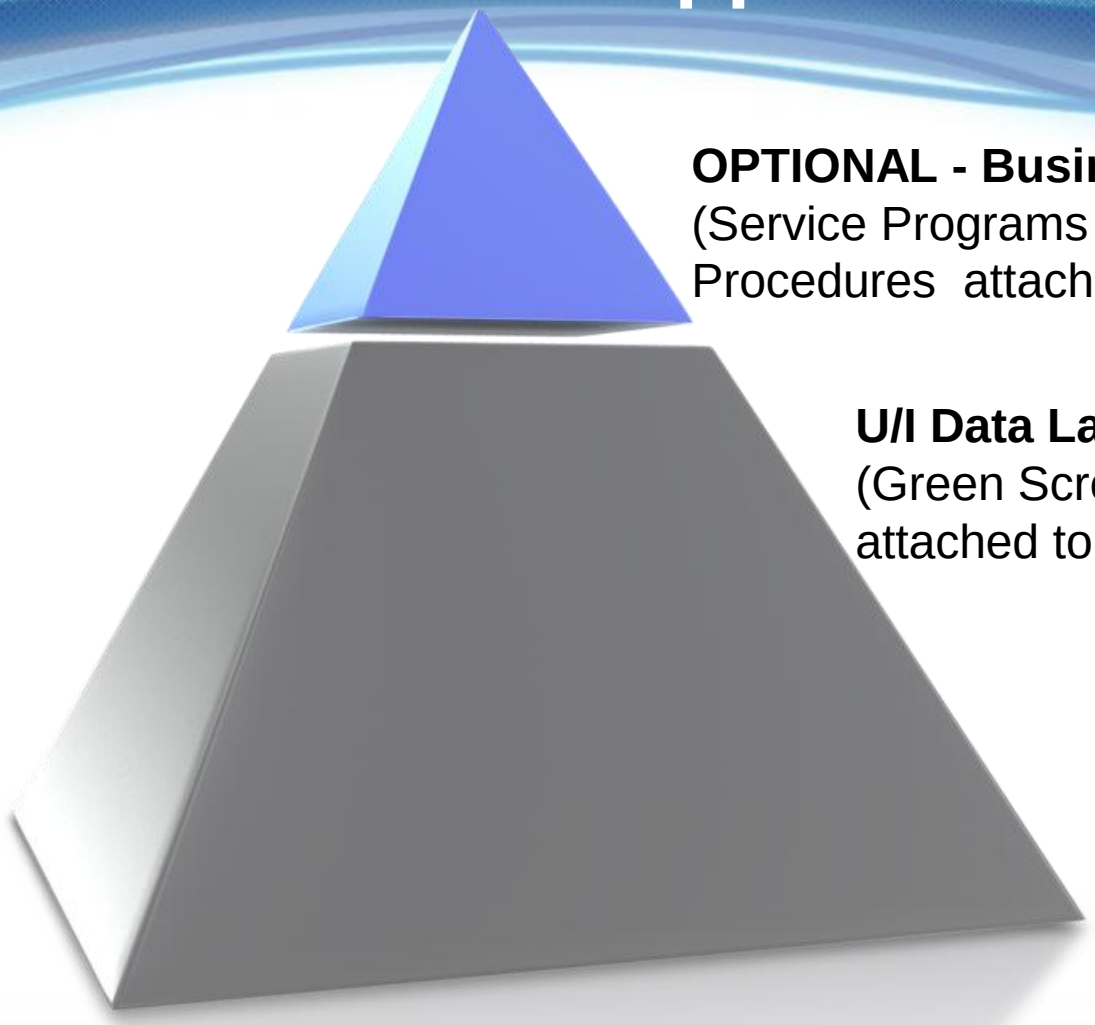
Data Centric Application Architecture

- **Faster Processing**
- **Smaller Application (less lines of code)**
- **DBMS resources are doing lots of work for you**
- **Consistent Results (business rules in DBMS)**



Data Centric Architecture

Data Centric Application Architecture

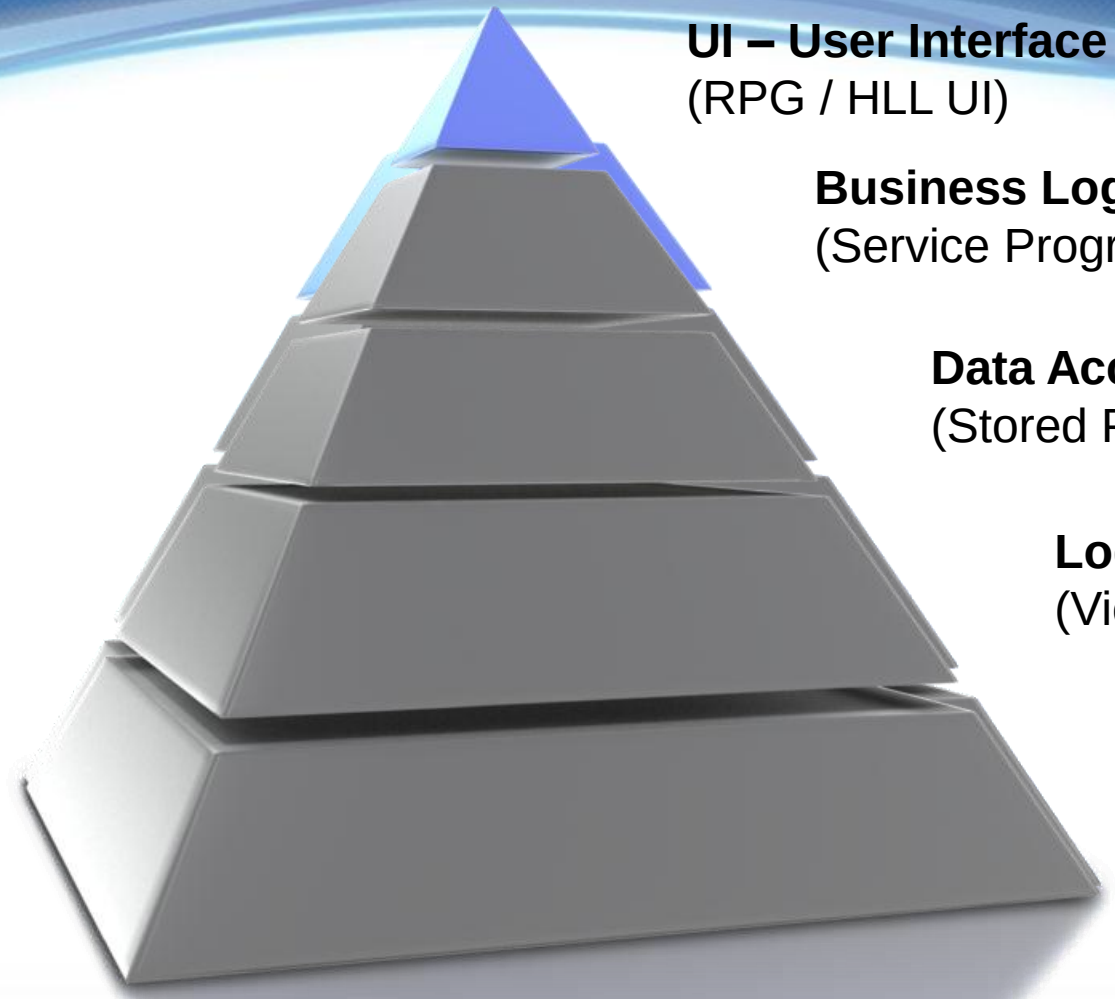


OPTIONAL - Business / Data Layer
(Service Programs or Stored Procedures attached to Data)

U/I Data Layer
(Green Screen programs attached to Data)

Program Centric Architecture

Data Centric Application Architecture



UI – User Interface Layer
(RPG / HLL UI)

Business Logic Layer
(Service Programs)

Data Access Layer
(Stored Procedures)

Logical Data Layer
(Views)

Physical Data Layer
(Tables / Indexes)

Data Centric Architecture

Data Centric Application Architecture

Physical Data Layer
(Tables / Indexes)

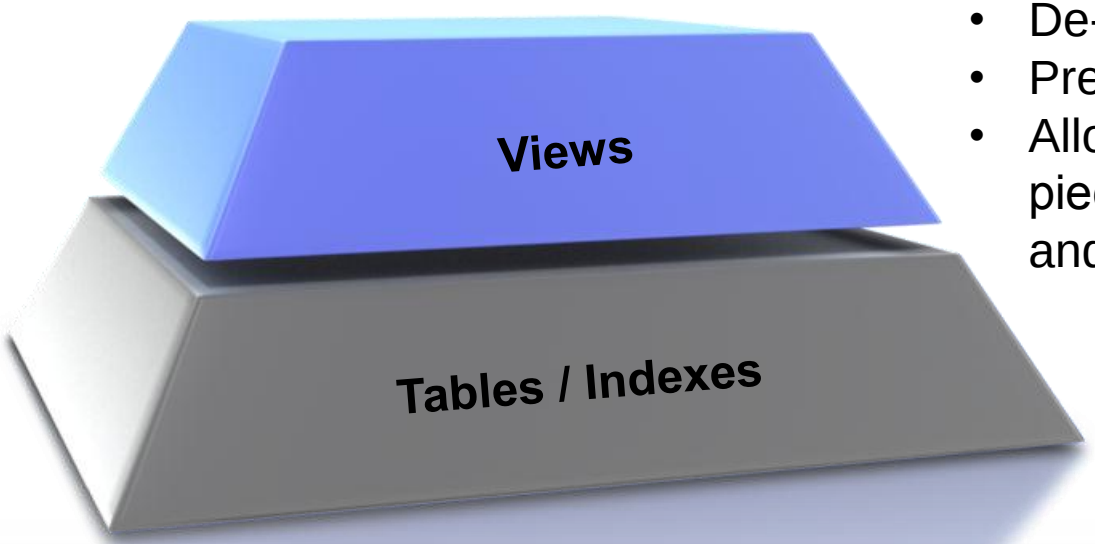
- Normalized Data Model (3^{rd} normal Form)
- Primary Keys
- Foreign Key constraints
- Common Business Logic Constraints

Data Centric Architecture

Data Centric Application Architecture

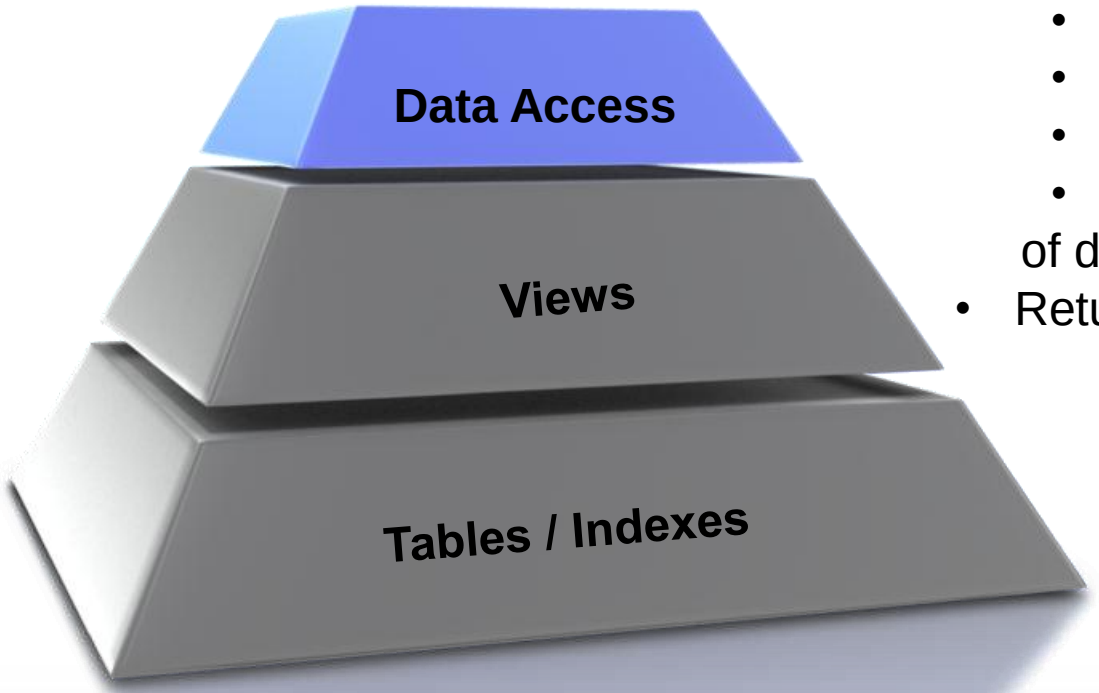
Logical Data Layer
(Views)

- Brings data together across tables
- Give **program specific** “View” into data.
- De-Normalizes Data
- Pre-calculated / Translated Data
- Allows you to secure specific pieces of data by “Hiding them” and not including them in views.



Data Centric Architecture

Data Centric Application Architecture



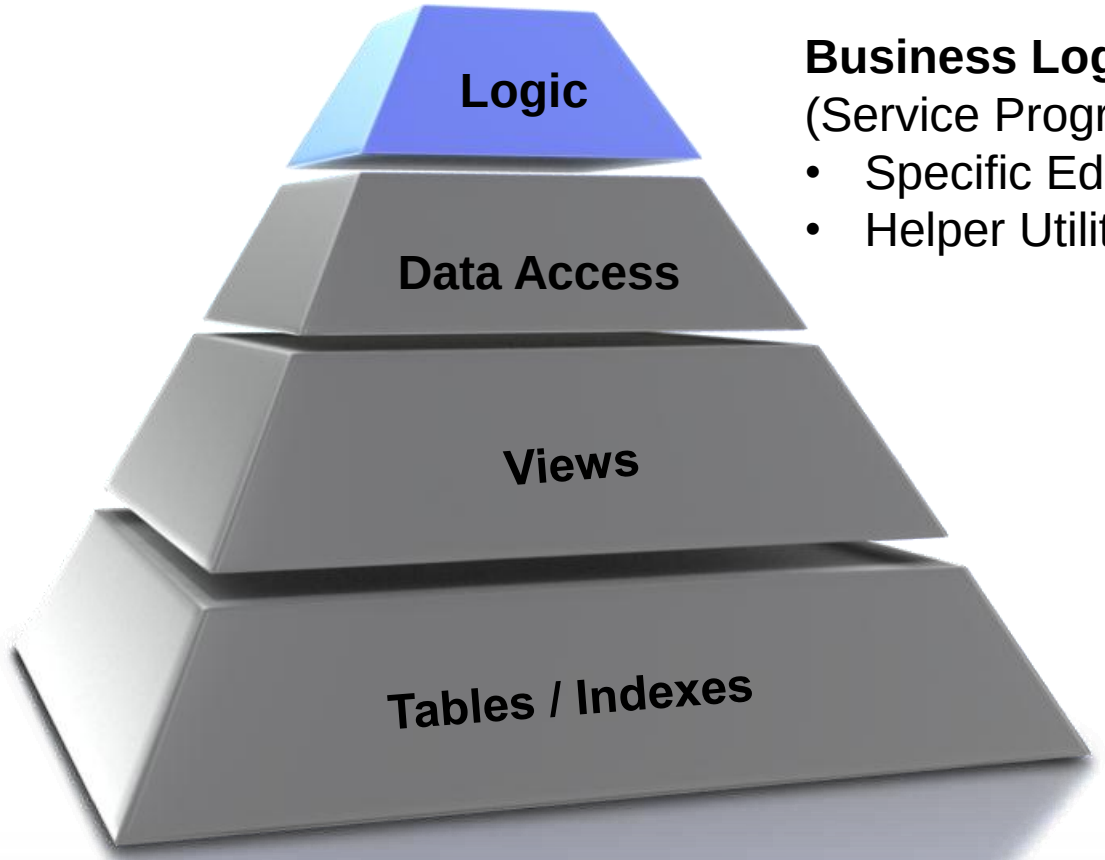
Data Access Layer
(Stored Procedures)

- Responsible for :
  - Select
  - Update
  - Insert
  - Delete

  of data.
- Returns result set of data (7.1)

Data Centric Architecture

Data Centric Application Architecture

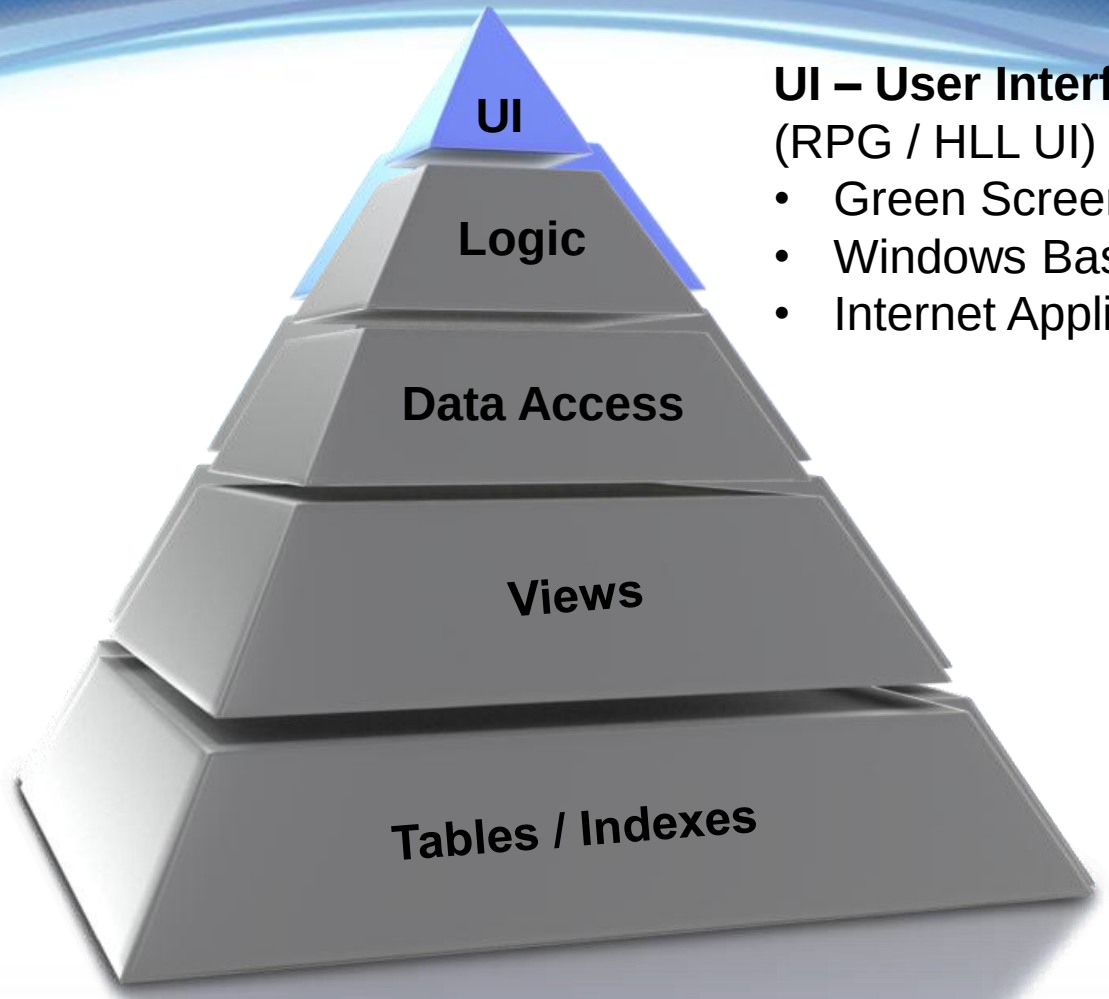


Business Logic Layer
(Service Programs)

- Specific Edits and Business Rules
- Helper Utilities

Data Centric Architecture

Data Centric Application Architecture

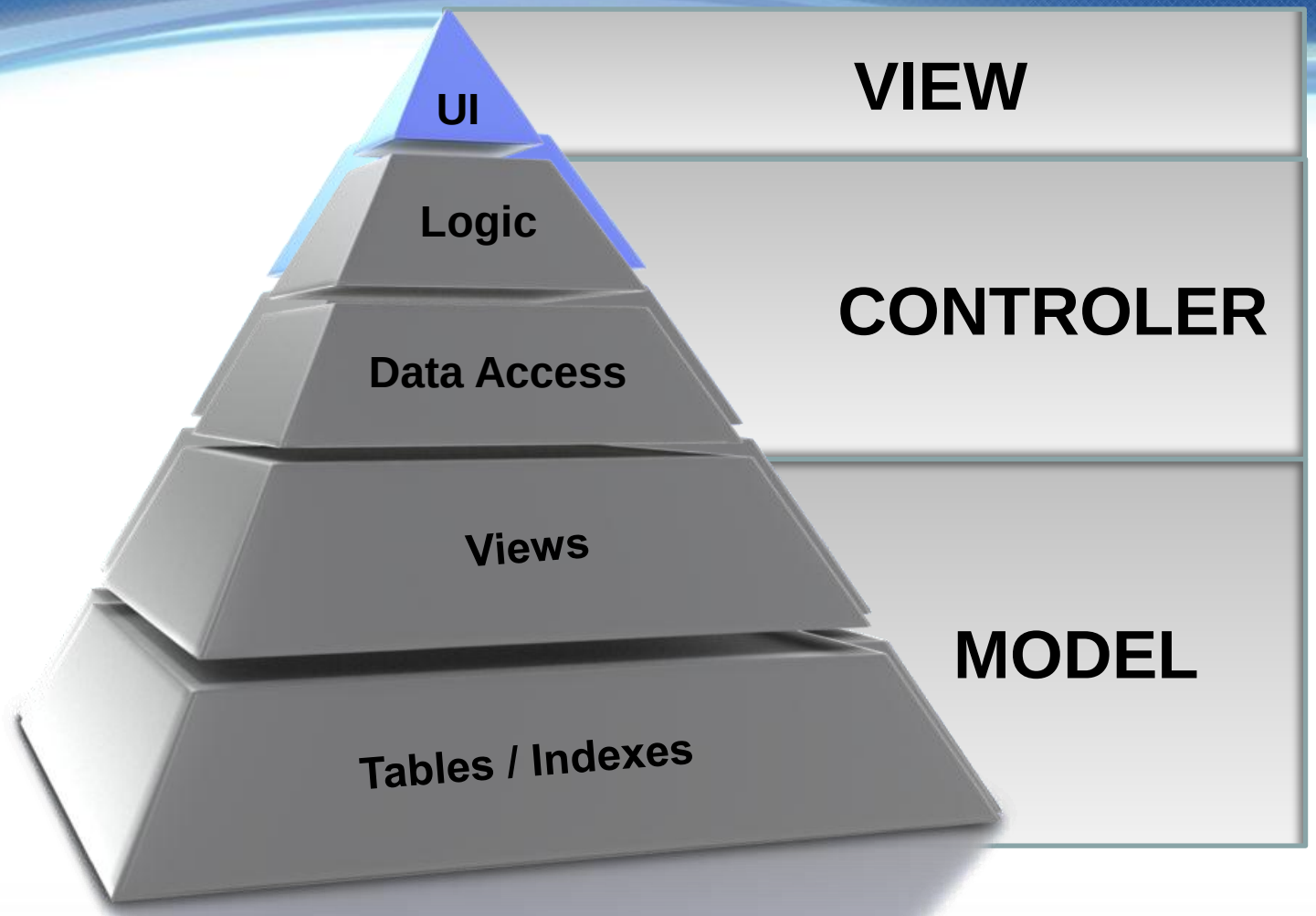


UI – User Interface Layer
(RPG / HLL UI)

- Green Screen Programs
- Windows Based Graphical Interface
- Internet Applications

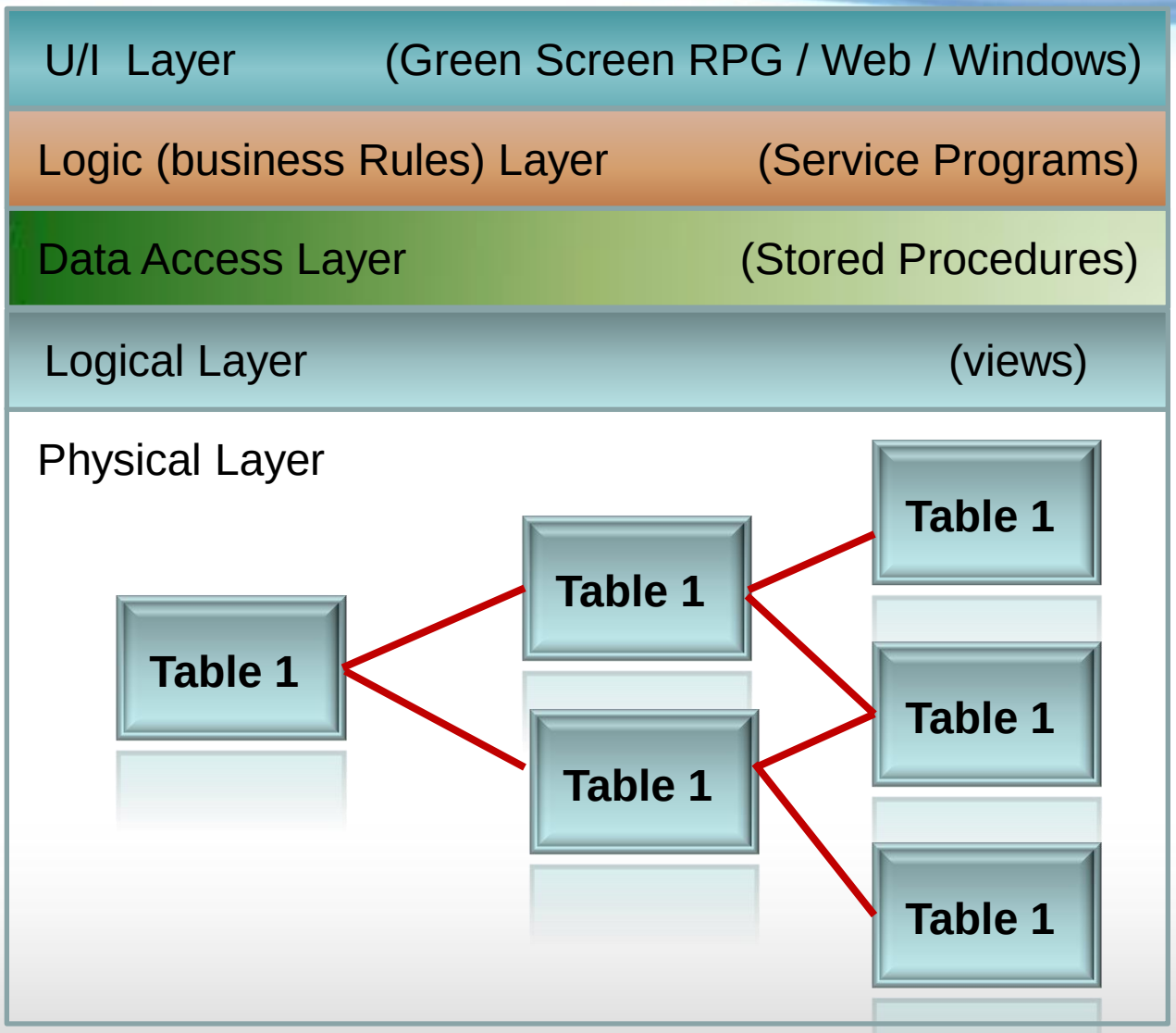
Data Centric Architecture

Data Centric Application Architecture

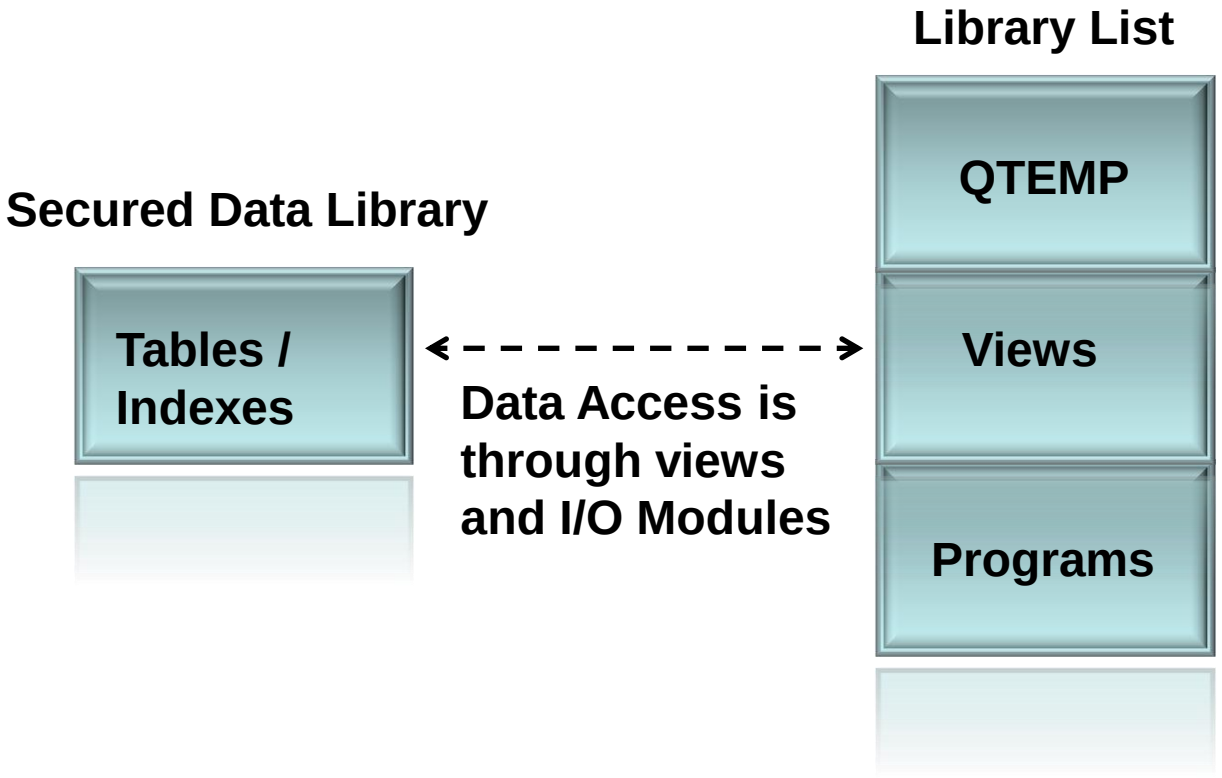


Data Centric Architecture

Data Centric Application Architecture



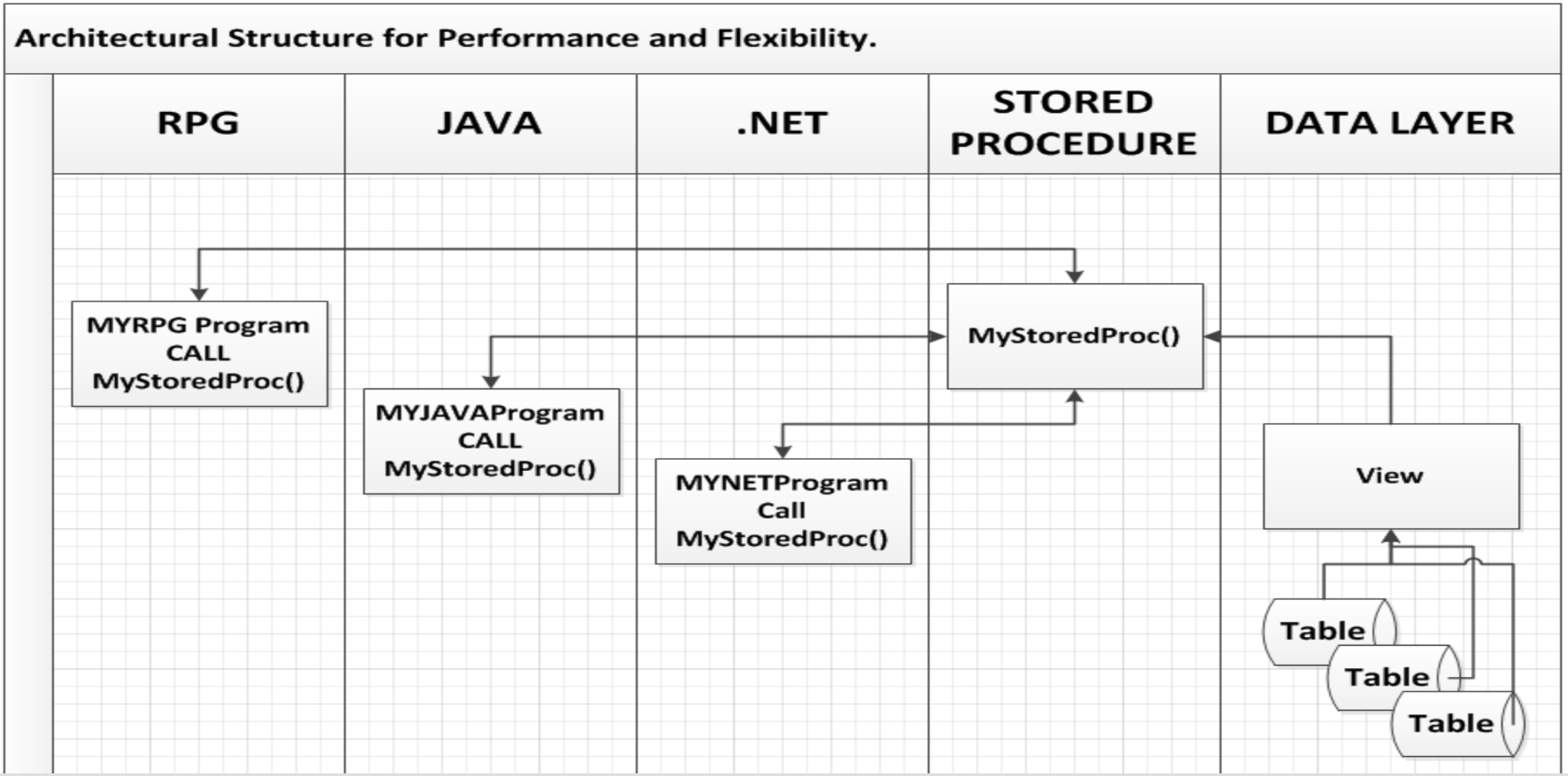
Data Centric Application Architecture



Data Centric Application Architecture

Stored Procedures

- Architectural Structure:



Data Centric Application Architecture

Best Practices

- **Do not use “SELECT * FROM ….” in SQL Statements.**
- **No DDL objects opened in “F” specs.**
- **The optimum number of files accessed in a program is 1.(i.e. use Views to bring data together into your programs)**
- **Use views between your programs and the physical table. Do Not Access Physical Tables Directly.**
- **Use indexes to support your views.**

Data Centric Application Architecture

Best Practices

- **Every SQL statement (in views or programs) should be analyzed by the index advisor to determine index needs.**
- **Compile recursively called SQLRPGLE modules / programs with *ENDACTGRP or *ENDJOB. (i.e. CRTSQLRPGI)**
- **Use both short names and long names for columns names and Table names.**
- **Use short and long names for views and indexes.**

Data Centric Application Architecture

Best Practices

- **Use INTEGER data type when zero precision is used instead of NUMERIC().**
- **Every tables should have a primary key. The PK should be an identity column.**
- **Create Business key as Unique but not PK.**
- **Do not reference access paths (LF / Indexes) in a SQL query. Always access Table or View. (Preferably View).**
- **Use DATE data type instead of numeric(8,0)**

Data Centric Application Architecture

Best Practices

- **Use defaults when possible. i.e. CURRENT_DATE, CURRENT_TIMESTAMP, ect.**
- **Always GET DIAGNOTSTICS at the end of every fetch. Interrogate SQLCODE at the end of every EXE SQL SELECT ……**
- **Save DDL / SQL source in Source Files just like DDS and / RPG!**
- **Embedded SQL statements should be as simple as possible. Complex SQL statements should be in views and accessed w/in HLL.**

Data Centric Application Architecture

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**

Data Centric Application Architecture

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.**

Data Centric Application Architecture

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;
```

Data Centric Application Architecture

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;
```

Data Centric Application Architecture

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
```

Data Centric Application Architecture

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;
```

Data Centric Application Architecture