An Introduction to IMS Database

Session I03

Central Canada

Db2 Users Group

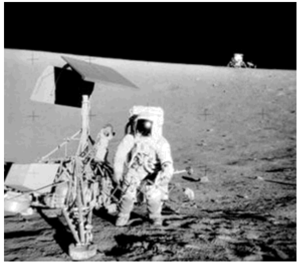
Dennis Eichelberger
IT Specialist
IMS Support – Washington System Center
deichel@us.ibm.com

Information Management System (IMS)

….. Somewhere Beyond the moon

→ originally created for the NASA Apollo Project
to put a man on the moon

Success!



Where IMS Runs the World

- Manufacturing
  - Aerospace, equipment, automobile, …
- Banking
- Telecommunications
- Transportation
- Travel
- Health Providers
- Insurance
- Finance/Credit Card Processing
- Package delivery
- Government
- Retail
- …

IMS Architecture

- Multiple region online system
  - Applications run in their own regions
    - Also called partitions, address spaces, dependent regions…

- Database management (IMS DB component)
  - Separated from application programming
    - IMS realized that moving the definitions out of the program would have substantial benefits when many programs (not just IMS application programs) could access the same database independently

- Transaction management (IMS TM component)
  - Separated from application programming
    - The sending and receiving of input and output messages was removed from programs
      - Allows the same program to access the message regardless of what type of device was used to send in the message
      - IMS's message queues were critical to its initial design

Compatibility

- Application compatibility
  - Programs written in 1968 (1964) still work
  - IMS does not require recompiles of applications for new IMS releases

- Database compatibility
  - Databases do not require upgrades for new IMS releases

- Program is independent of database is independent of access method

- IMS has allowed users to grow their applications instead of requiring major conversions. e.g. recompiles or rewrite

Adaptability

- And yet IMS also keeps up with technological trends
  - Provides a TCP/IP socket server – IMS Connect
    - For access to IMS transactions and databases
  - Allows IMS applications to be either clients or servers
  - Delivers adapters that run in JEE, SOAP, etc. environments
    - Pluggable components that provide easy interfaces to IMS resources
  - Supports Java type-2 and type-4 universal drivers for access to IMS DB
  - Leverages zOS Connect for API – rapid deployments
- Provides application programming interfaces
  - Unique to IMS
    - DL/I calls to access IMS resources: message queues and data
    - IMS applications can be written in: COBOL, PL/I, Assembler, C/C++, Pascal, ADA, REXX, Java
  - Portable – Java classes and JDBC

Programming Interface

- Database definitions outside of programs
  - DBDs
    - Defines the database structure
  - PSBs
    - Views of segments within databases
    - Access definitions (get, insert, delete, replace)
- Programs
  - DL/I calls used for database access
    - Program not aware of data sets and physical characteristics of data
- DL/I calls used for input/output messages
  - Program independent of communication protocols
  - IMS applications get messages from a queue, process them, and insert them back to the queue

And …

- That is why IMS continues to be current
  - Application programs that were written for IMS 5, 10, 15, 20, 40… years ago can continue to run today
    - Even when they are now invoked from the web
      - Or continue to be invoked from older 3270 devices or emulators
      - The **same** application, without change, can be accessed from any type of connection mechanism or network that can access IMS in a modernized infrastructure
- The IMS architecture allows communication components and even database components to be exchanged or replaced

Without affecting the application program in IMS

What New Users of IMS Need to Understand

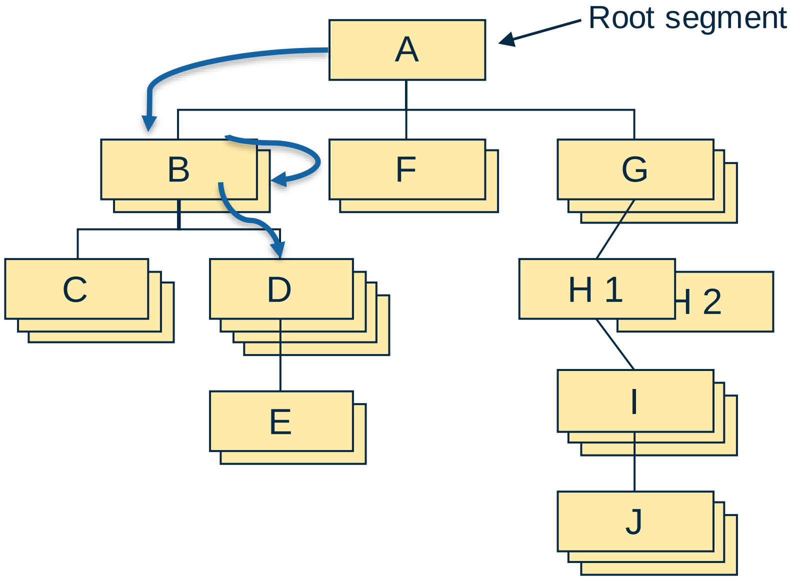
- IMS Databases
- IMS Online Systems and Transaction Management
- IMS Connect
- Application Programming
- IMS in a Parallel Sysplex®

DIAMETER
Handyman

IMS Database Structure

- IMS databases are hierarchical (and Relational)

Tree structure



– Segments have parents and children
  - Segments are accessed through their parents
  - Parents and Children are related
    D is accessed through A and B
– Segments have twins
  - Multiple B segments under one A segment
  - Twins are related
- H2 is accessed via H1
- Segment pointers are updated at the same time the segment is updated

A segment type (e.g. A, B, or C) is similar to a relational table.
A segment occurrence, one of the segments, is similar to a row in a relational table.
An IMS database is similar to several tables which are joined.

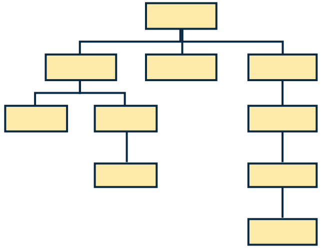
IMS Database Structure …

- Full Function database types
  - Hierarchical indexed direct (HIDAM)
  - Partitioned HIDAM (PHIDAM)
  - Generalized Sequential (GSAM)
  - Hierarchical indexed sequential (HISAM)*
  - Hierarchical Sequential (HSAM) *
  - Hierarchical direct (HDAM)
  - Partitioned HDAM (PHDAM)

* Simple version available

- Fast Path database types
  - Data Entry Database (DEDB)
  - Main Storage Database (MSDB)
    - MSDBs are root-only and stored & accessed only in memory

Structure is chosen based on what makes most sense for access patterns, e.g., sequential versus random



- **Indexed databases**
  - HIDAM, PHIDAM, HISAM
  - Roots are ordered in key sequence
- **Non-indexed databases**
  - HDAM, PHDAM, DEDB
  - Randomization routines are used to order roots

Value of Hierarchical Structures

- GSAM

Usage Characteristics	
Insert sequentially – time order	Segment Delete unavailable

No specific structure
Used for IMS BMP and Batch jobs requiring restartability

Value of Hierarchical Structures

- HSAM

Usage Characteristics	**Not available**
Insert in key sequence	Secondary Indexing
	Logical relationships
	Variable length segments
Update not available	DLET or REPL call

Used for low volume processing e.g. data archival

Value of Hierarchical Structures

- HISAM

Usage Characteristics	
Same size records	
Complete physical record fits into logical defined record	
Few Deletes or Inserts	Delete / Insert Available
Root Delete	Space is reclaimed in primary area, not reclaimed in overflow
Dependent Segment	Marked deleted, Space not available

Database reorganization required to reclaim physical space

Value of Hierarchical Structures

- HDAM PHDAM
  -

Usage Characteristics	
Segments are stored unordered	Position based on key randomization
Segment relationships via pointers	Dependents may be in separate blocks
Delete removes segment	Space is reclaimed

High activity – multiple users
Speedy segment retrieval and update

Value of Hierarchical Structures

- HIDAM PHIDAM

Usage Characteristics	
Two DBDs required – HIDAM	1 = Index 2 = Data area
Root Segments are sequentially ordered	Index may be processed separately
Segment relationships via pointers	Dependents may be in separate blocks
Delete removes segment	Space is reclaimed

High activity – Multiple users
Speedy segment retrieval and update
Fast sequential processing

Value of Hierarchical Structures

- MSDB

Usage Characteristics	
Root Only	Terminal or Non- Terminal related
Root Segments are sequentially inserted	Index may be processed separately
Segment relationships via pointers	Forward pointers only
Delete & Insert calls maybe	If non-Terminal related
DLI calls	Within parent not accepted
Changes only made at IMS restart	MSDBLOAD parameter

High activity – Limited users
Speedy segment retrieval and update
Direct and Binary search
Sequential calls may provide unexpected results

No Longer Recommended

Value of Hierarchical Structures

- DEDB

Usage Characteristics	
Sequential Dependents	Rapid insert time of insert e.g. journaling
Direct Dependents	Hierarchical structure
Replace calls	May result in moved segments
Delete removes segment	Space is reclaimed

Combination of both Direct and Sequential processes
Multiple Areas allow large amount of data in a structure
Online Changes allowed

Large Database Sizes

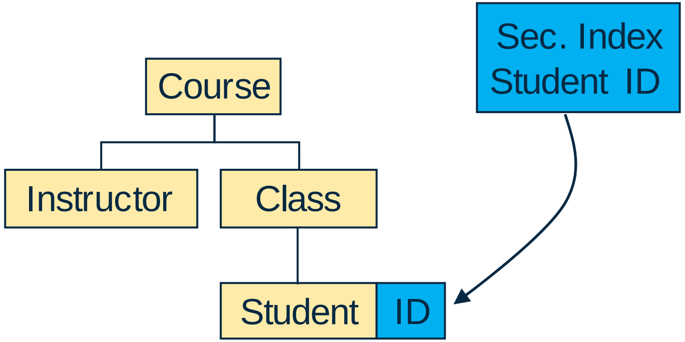
- HALDB (PHDAM and PHIDAM) and DEDB databases
  - Database is composed of multiple database data sets
    - HALDB (High Availability Large Database) calls these datasets partitions
    - DEDB calls these datasets areas
  - HALDB may have up to 1001 partitions
    - Each partition may have up to 10 4GB data sets
      - Over 40 terabytes
    - In IMS 14 and later the OSAM data sets may be up to 8GB
      - Over 80 terabytes
  - DEDBs may have up to 9999 areas
    - Each area has one data set up to 4GB
      - Over 40 terabytes

Application programs are unaware of the partitions or areas

The end user accesses their data regardless of position or residence

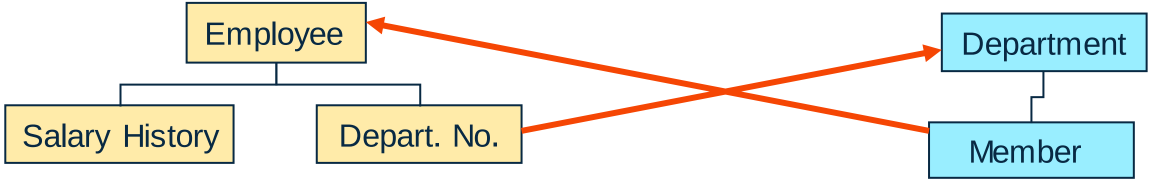
Secondary Indexes

- IMS databases may also have Secondary indexes
  - Indexes are based on different fields in any segment
  - Example:
    - Secondary index into course database based on student ID
      - One could find all the classes in which a student is enrolled

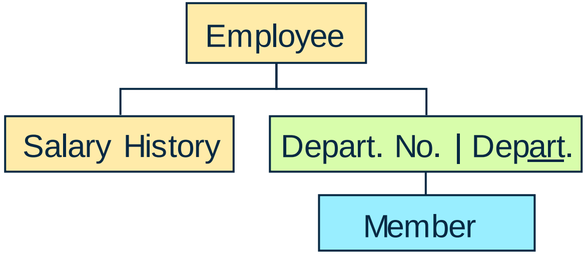


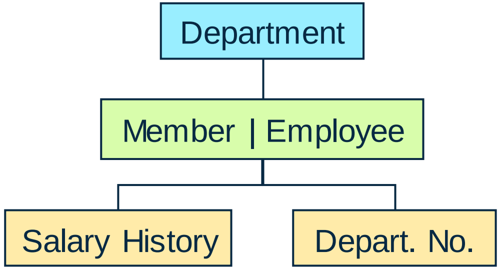
Logical Relationships

- Full Function database may have logical relationships
  - Logical databases are formed from multiple physical DBs
  - Example:
    - Logical relationship between employee and department DBs



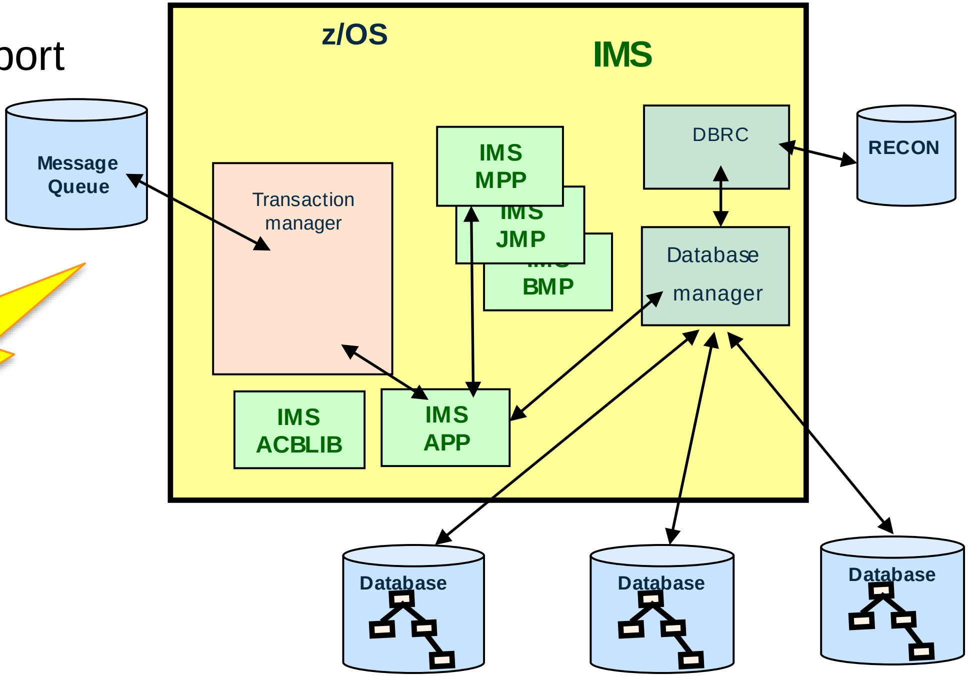
    - Examples of logical databases





Accessing IMS Databases

- Multiple dependent regions
  - Non-serial database access
  - Multiple programming language support
- Full database integrity
  - Processes logging for audit and recovery
  - DBRC for access authorization
- Application independence from Data management & access



Defining Databases

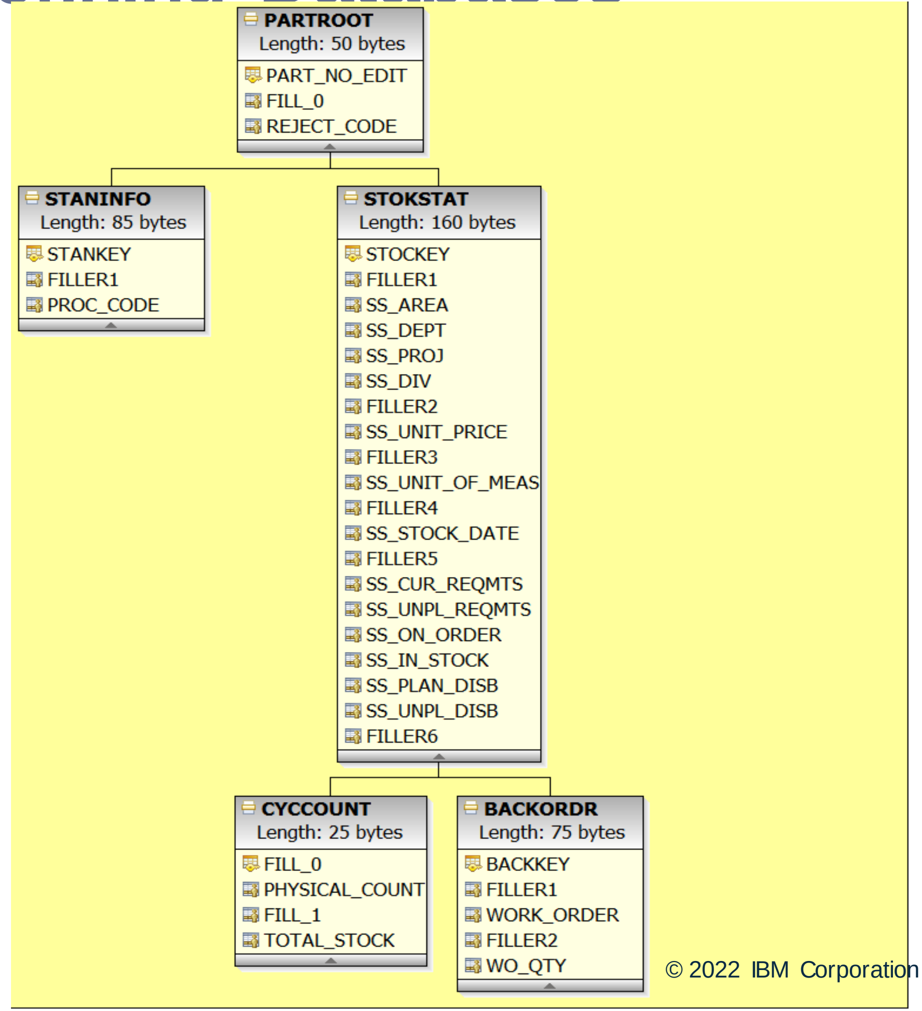
- Database Definition - DBD
  - Database definition is used to define:
    - Database type
    - Dataset type
    - Number of Segments
    - Hierarchical relationship of the segments
    - Fields within a Segment
    - Key fields
    - Data types of the fields
    - Logical relationships within or outside of the database
    - Indexing

Defining Databases

```
DBD   NAME=DI21PART,                                                   X
              ACCESS=(HISAM,VSAM),                                     X
              PASSWD=NO,
              DATXEXIT=NO,
              ENCODING=CP1047
        DATASET DD1=DI21PART,
              DEVICE=3380,
              OVFLW=DI21PARO,
              BLOCK=(100,10),
              SIZE=(4096),
              RECORD=(678)
        SEGM  NAME=PARTROOT,
              EXTERNALNAME=PARTROOT,
              PARENT=0,
              BYTES=50,
              RULES=(LLL,LAST),
              ENCODING=Cp1047
        FIELD NAME=(PARTKEY,SEQ,U),
              EXTERNALNAME=PARTKEY,
              BYTES=17,
              START=1,
              TYPE=C,
              DATATYPE=CHAR
        SEGM  NAME=STANINFO,
              EXTERNALNAME=STANINFO,
              PARENT=PARTROOT,
              BYTES=85,
              RULES=(LLL,LAST),
              ENCODING=Cp1047
```

```
          SEGM  NAME=STANINFO,                                         X
                EXTERNALNAME=STANINFO,                                 X
                PARENT=PARTROOT,                                       X
                BYTES=85,                                              X
                RULES=(LLL,LAST),                                      X
                ENCODING=Cp1047
          FIELD NAME=(STANKEY,SEQ,U),                                  X
                EXTERNALNAME=STANKEY,                                  X
                BYTES=2,                                               X
                START=1,                                               X
                TYPE=C,                                                X
                DATATYPE=CHAR
          DFSMARSH ENCODING=Cp1047,                                    X
                INTERNALTYPECONVERTER=CHAR
          SEGM  NAME=STOKSTAT,                                         X
                EXTERNALNAME=STOKSTAT,                                 X
                PARENT=PARTROOT,                                       X
                BYTES=160,                                             X
                RULES=(LLL,LAST),                                      X
                ENCODING=Cp1047
          FIELD NAME=(STOCKEY,SEQ,U),                                  X
                EXTERNALNAME=STOCKEY,                                  X
                BYTES=16,                                              X
                START=1,                                               X
                TYPE=C,                                                X
                DATATYPE=CHAR
          DFSMARSH ENCODING=Cp1047,                                    X
```

Defining Databases



PARTROOT
Length: 50 bytes
- PART_NO_EDIT
- FILL_0
- REJECT_CODE

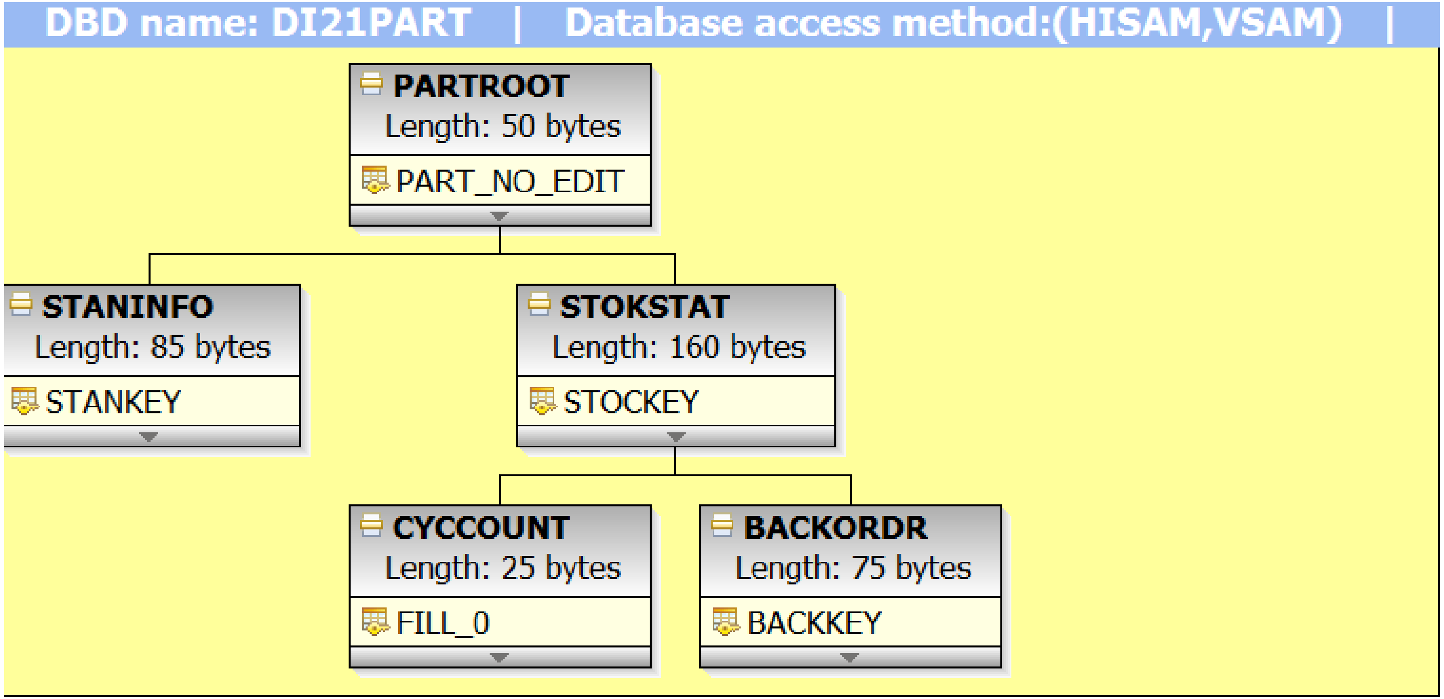
STANINFO
Length: 85 bytes
- STANKEY
- FILLER1
- PROC_CODE

STOKSTAT
Length: 160 bytes
- STOCKEY
- FILLER1
- SS_AREA
- SS_DEPT
- SS_PROJ
- SS_DIV
- FILLER2
- SS_UNIT_PRICE
- FILLER3
- SS_UNIT_OF_MEAS
- FILLER4
- SS_STOCK_DATE
- FILLER5
- SS_CUR_REQMTS
- SS_UNPL_REQMTS
- SS_ON_ORDER
- SS_IN_STOCK
- SS_PLAN_DISB
- SS_UNPL_DISB
- FILLER6

CYCCOUNT
Length: 25 bytes
- FILL_0
- PHYSICAL_COUNT
- FILL_1
- TOTAL_STOCK

BACKORDR
Length: 75 bytes
- BACKKEY
- FILLER1
- WORK_ORDER
- FILLER2
- WO_QTY

Defining Databases



DBD name: DI21PART | Database access method:(HISAM,VSAM) |

PARTROOT
Length: 50 bytes
PART_NO_EDIT

STANINFO
Length: 85 bytes
STANKEY

STOKSTAT
Length: 160 bytes
STOCKEY

CYCCOUNT
Length: 25 bytes
FILL_0

BACKORDR
Length: 75 bytes
BACKKEY

Defining Programs

Program Specification Block - PSB

- Program Specification is used to define Program Control Blocks - PCB:
- Program Control Blocks define
  - Database(s) to be accessed
  - Processing options to access it with
    - Get, Insert, Delete, Replace
  - Segments the program is aware of
  - Fields within the segment the program is aware of
  - Program view of the accessed database(s)

Defining Databases

```
DBD    NAME=DI21PART,                                                   X
               ACCESS=(HISAM,VSAM),                                     X
               PASSWD=NO,
               DATXEXIT=NO,
               ENCODING=CP1047
      DATASET DD1=DI21PART,
               DEVICE=3380,
               OVFLW=DI21PARO,
               BLOCK=(100,10),
               SIZE=(4096),
               RECORD=(678)
      SEGM   NAME=PARTROOT,
               EXTERNALNAME=PARTROOT,
               PARENT=0,
               BYTES=50,
               RULES=(LLL,LAST),
               ENCODING=Cp1047
      FIELD NAME=(PARTKEY,SEQ,U),
               EXTERNALNAME=PARTKEY,
               BYTES=17,
               START=1,
               TYPE=C,
               DATATYPE=CHAR
      SEGM   NAME=STANINFO,
               EXTERNALNAME=STANINFO,
               PARENT=PARTROOT,
               BYTES=85,
               RULES=(LLL,LAST),
               ENCODING=Cp1047
```

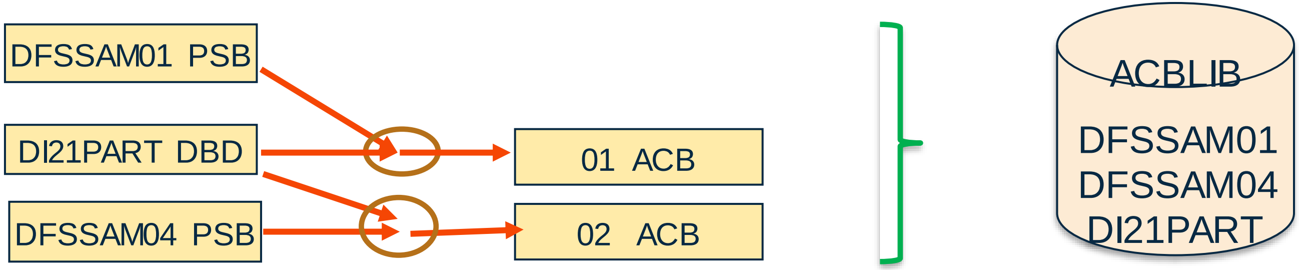
```
          SEGM   NAME=STANINFO,                                         X
                 EXTERNALNAME=STANINFO,                                 X
                 PARENT=PARTROOT,                                       X
                 BYTES=85,                                              X
                 RULES=(LLL,LAST),                                      X
                 ENCODING=Cp1047
          FIELD  NAME=(STANKEY,SEQ,U),                                  X
                 EXTERNALNAME=STANKEY,                                  X
                 BYTES=2,                                               X
                 START=1,                                               X
                 TYPE=C                                                 X
```

```
DBPCB00    PCB    TYPE=DB,DBDNAME=DI21PART,PROCOPT=GRI,KEYLEN=44
           SENSEG NAME=PARTROOT,PARENT=0                                X
           SENSEG NAME=STANINFO,PARENT=PARTROOT
           SENSEG NAME=STOKSTAT,PARENT=PARTROOT                         X
           SENSEG NAME=CYCCOUNT,PARENT=STOKSTAT                         X
           SENSEG NAME=BACKORDR,PARENT=STOKSTAT                         X
           PSBGEN LANG=ASSEM,PSBNAME=DFHSAM04,MAXQ=1                    X
           END                                                          X
```

```
DBPCB01    PCB    TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=44
           SENSEG NAME=PARTROOT,PARENT=0                                X
           SENSEG NAME=STANINFO,PARENT=PARTROOT                         X
           SENSEG NAME=STOKSTAT,PARENT=PARTROOT                         X
           PSBGEN LANG=COBOL,PSBNAME=DFSSAM01,MAXQ=1                    X
           END                                                          X
```

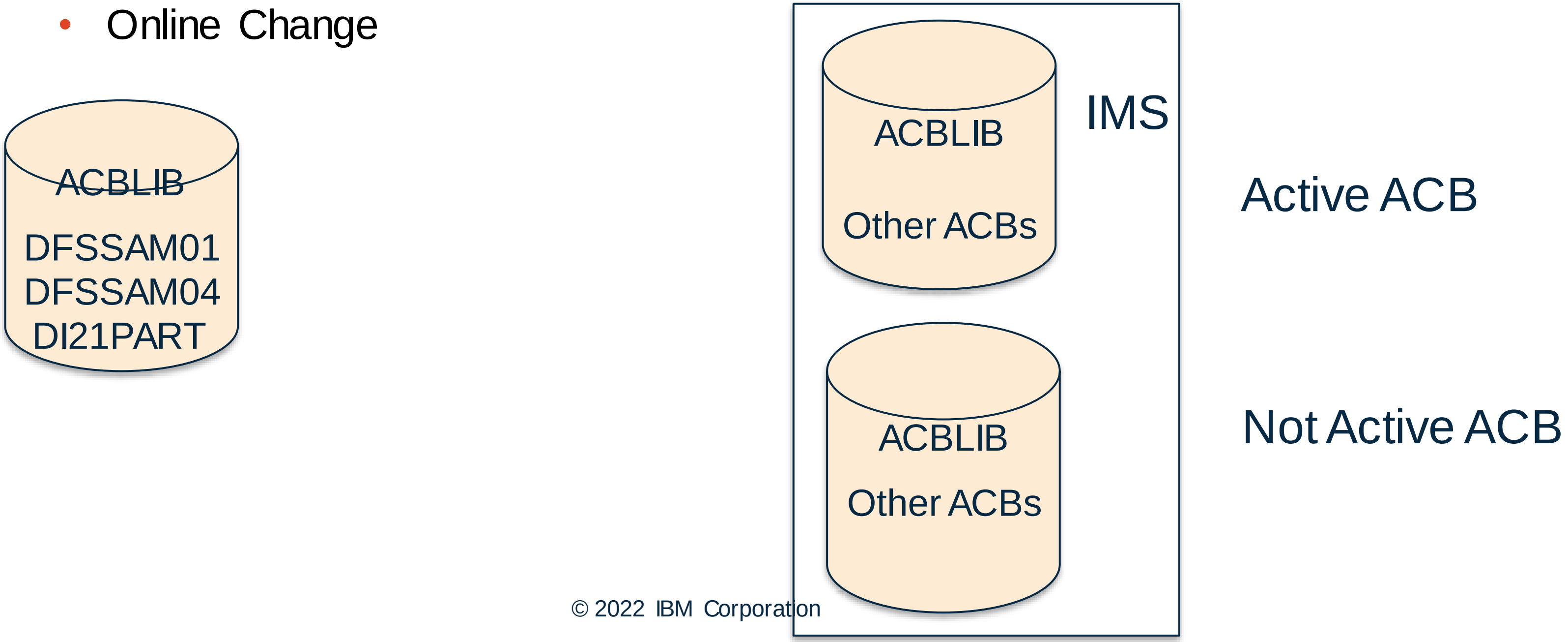
Identifying Databases to Program relationships

- DataBase generation - DBD
- Program Specification Block generation - PSB
- Application Control Block - ACB
  - Connects DBDs and PSBs
  - A PSB may access multiple databases
  - Multiple PSBS may access the same database
  - Application changes or additions may be implemented without an IMS restart.
    - Affected resources may need to be quiesced for control block updates
    - Online Change



Identifying Databases to Program relationships

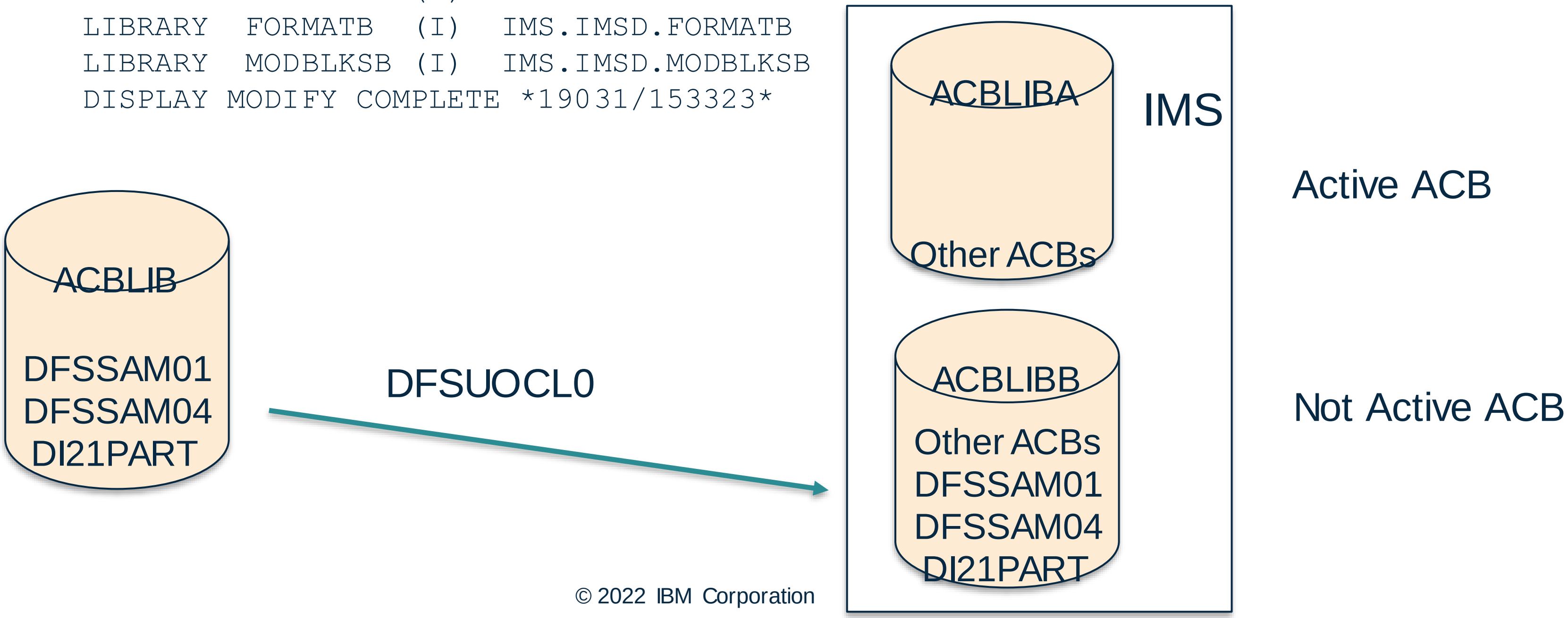
- Application Control Block - ACB
  - Application changes or additions may be implemented without an IMS restart.
    - Affected resources may need to be quiesced for control block updates
    - Online Change



Identifying Databases to Program relationships

- Online Change

```
R 294,/DIS MODIFY
DFS000I      LIBRARY  IMSACBA  (A)  IMS.IMSD.ACBLIBA
DFS000I      LIBRARY  FORMATA  (A)  IMS.IMSD.FORMATA
DFS000I      LIBRARY  MODBLKSA (A)  IMS.IMSD.MODBLKSA
DFS000I      LIBRARY  IMSACBB  (I)  IMS.IMSD.ACBLIBB
DFS000I      LIBRARY  FORMATB  (I)  IMS.IMSD.FORMATB
DFS000I      LIBRARY  MODBLKSB (I)  IMS.IMSD.MODBLKSB
DFS000I      DISPLAY MODIFY COMPLETE *19031/153323*
IMSD
```

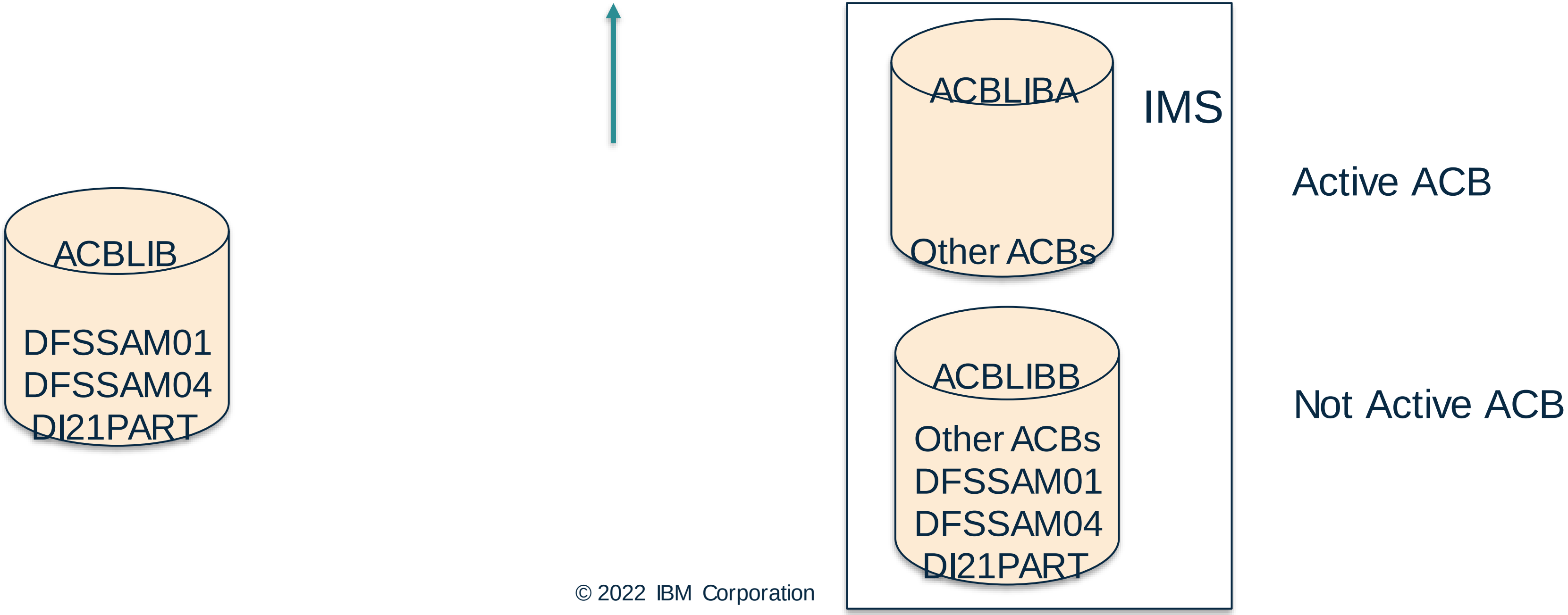


Identifying Databases to Program relationships

- Online Change

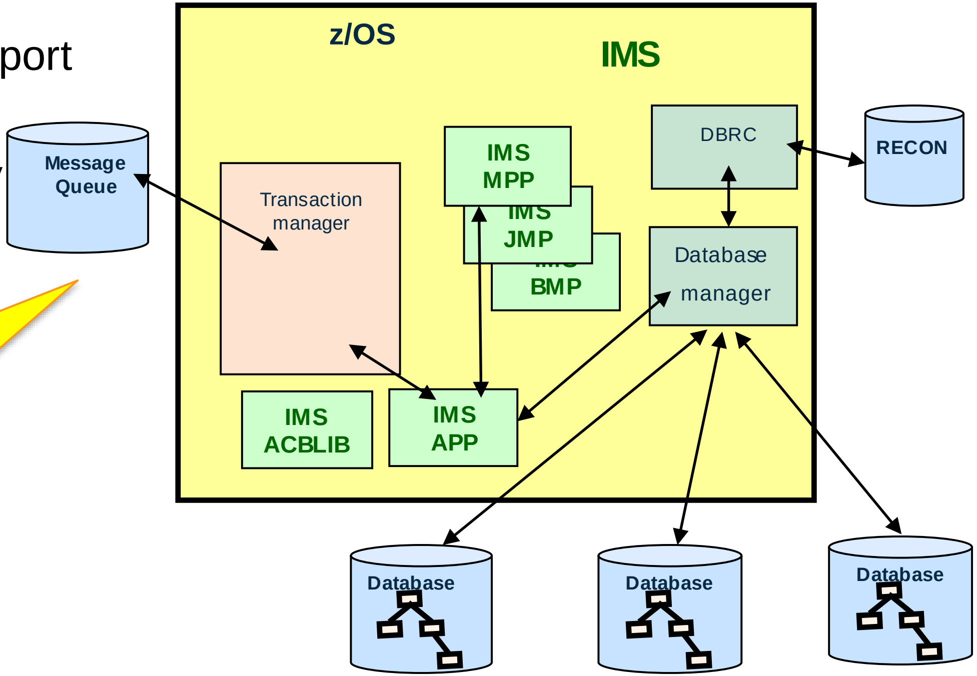
```
R 297,/MODIFY PREPARE ACBLIB
DFS3499I ACTIVE DDNAMES: MODBLKSA IMSACBA  FORMATA  MODSTAT ID:   22    IMSD

R 298,/MODIFY COMMIT
DFS3499I ACTIVE DDNAMES: MODBLKSA IMSACBB  FORMATA  MODSTAT ID:   23
```

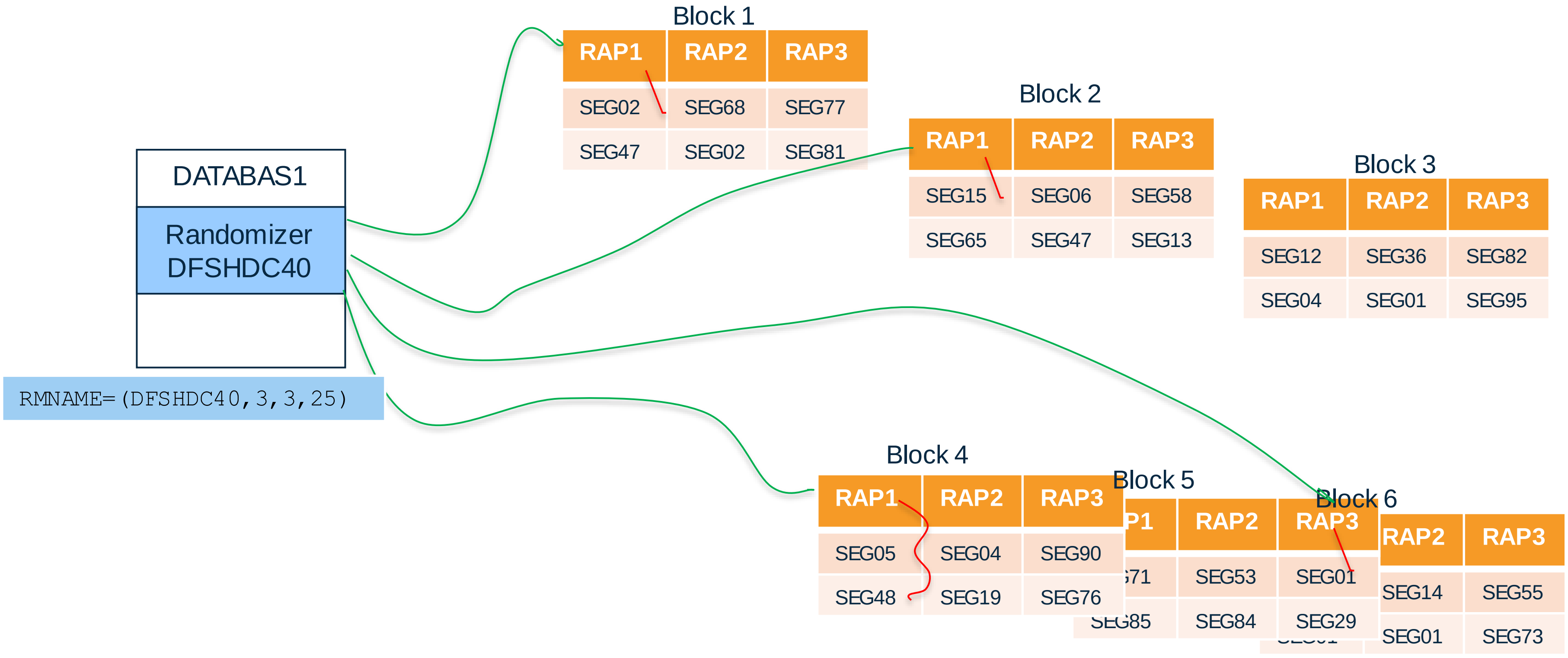


Accessing IMS Databases

- Multiple dependent regions
  - Non-serial database access
  - Multiple programming language support
- Full database integrity
  - Processes logging for audit and recovery
  - DBRC for access authorization
- Application independence from Data management & access



HDAM Database Randomization - Insert



DATABAS1

Randomizer DFSHDC40

```
RMNAME=(DFSHDC40,3,3,25)
```

Block 1

RAP1	RAP2	RAP3
SEG02	SEG68	SEG77
SEG47	SEG02	SEG81

Block 2

RAP1	RAP2	RAP3
SEG15	SEG06	SEG58
SEG65	SEG47	SEG13

Block 3

RAP1	RAP2	RAP3
SEG12	SEG36	SEG82
SEG04	SEG01	SEG95

Block 4

RAP1	RAP2	RAP3
SEG05	SEG04	SEG90
SEG48	SEG19	SEG76

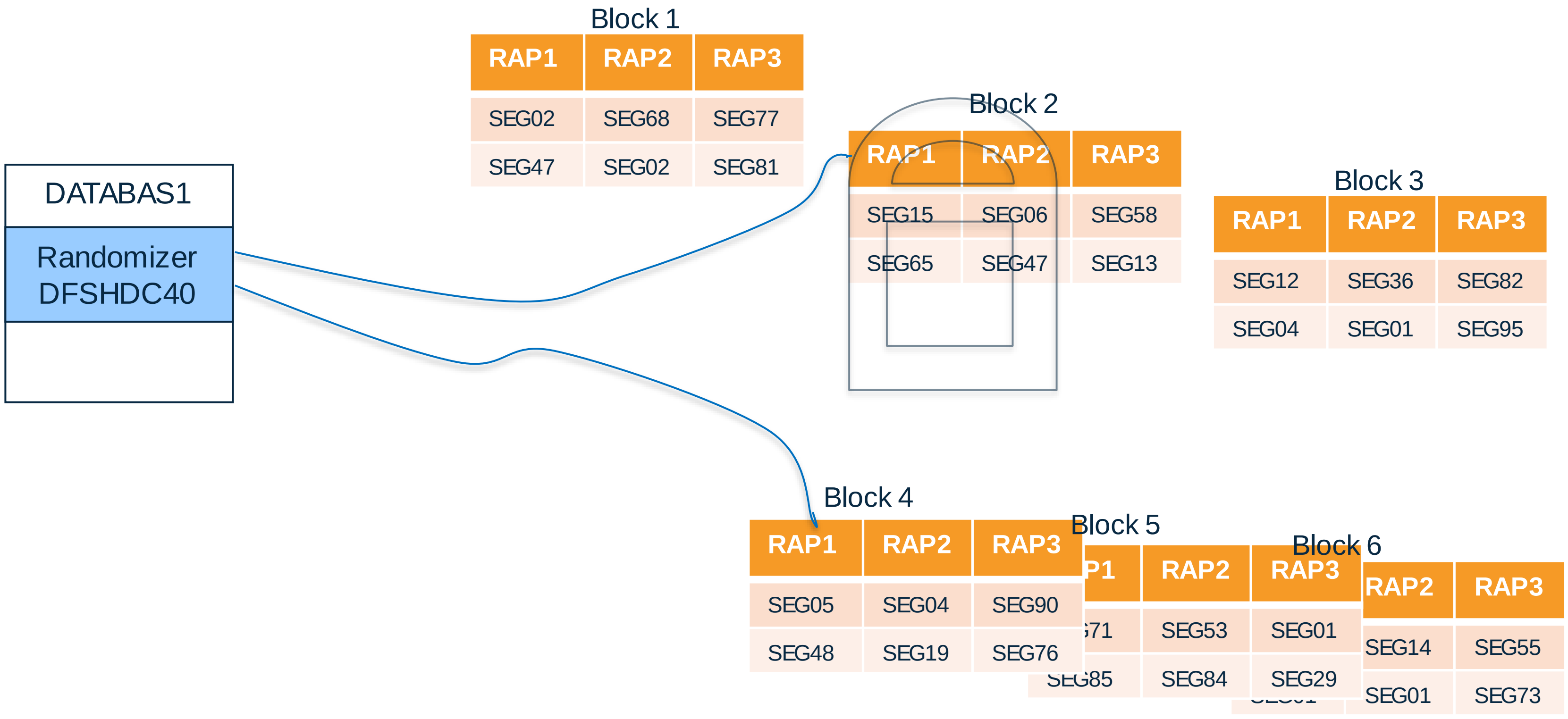
Block 5

P1	RAP2	RAP3
;71	SEG53	SEG01
SEG85	SEG84	SEG29

Block 6

RAP2	RAP3
SEG14	SEG55
SEG01	SEG73

Database Randomization - Lookup



Block 1

RAP1	RAP2	RAP3
SEG02	SEG68	SEG77
SEG47	SEG02	SEG81

Block 2

RAP1	RAP2	RAP3
SEG15	SEG06	SEG58
SEG65	SEG47	SEG13

Block 3

RAP1	RAP2	RAP3
SEG12	SEG36	SEG82
SEG04	SEG01	SEG95

Block 4

RAP1	RAP2	RAP3
SEG05	SEG04	SEG90
SEG48	SEG19	SEG76

Block 5

P1	RAP2	RAP3
71	SEG53	SEG01
SEG85	SEG84	SEG29

Block 6

RAP2	RAP3
SEG14	SEG55
SEG01	SEG73

HIDAM Database - Access

Initial Load

Seg	Prev Seg	Next Seq
SEG02		SEG03
SEG03	SEG02	SEG05
SEG05	SEG03	SEG06
SEG06	SEG05	SEG07
SEG07	SEG06	SEG08
SEG08	SEG07	SEG20
SEG20	SEG08	SEG21
SEG21	SEG20	SEG22
SEG22	SEG21	

Insert Segment

Seg	Prev Seg	Next Seq
SEG02		SEG03
SEG03	SEG02	SEG05
SEG05	SEG03	SEG06
SEG06	SEG05	SEG07
SEG07	SEG06	SEG08
SEG08	SEG07	SEG20
SEG20	SEG08	SEG21
SEG21	SEG20	SEG22
SEG22	SEG21	SEG30
SEG30	SEG22	

Insert Segment

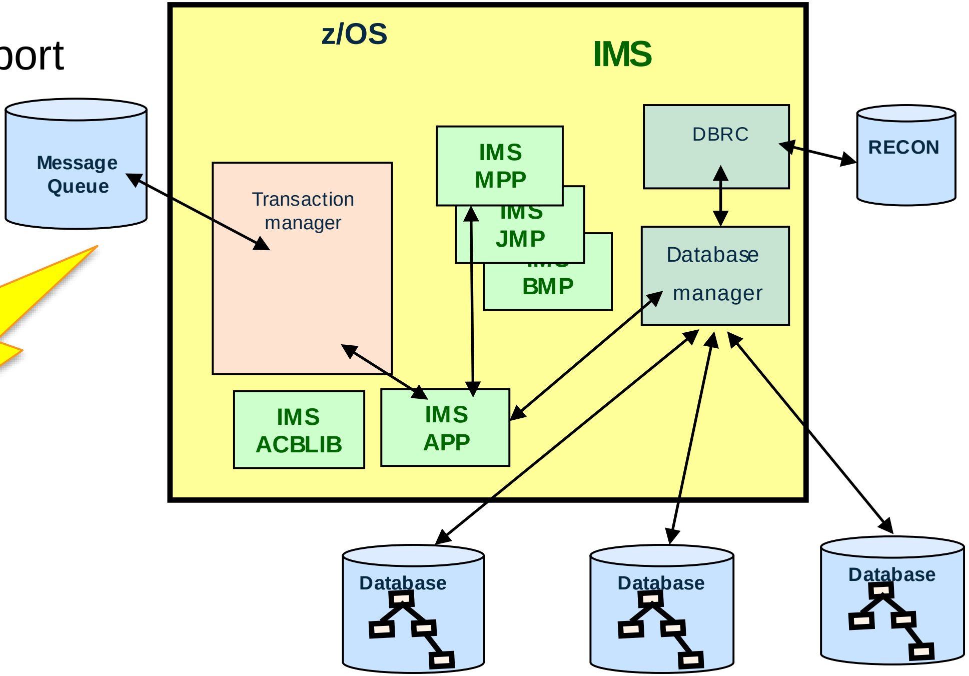
Seg	Prev Seg	Next Seq
SEG02		SEG03
SEG03	SEG02	SEG05
SEG05	SEG03	SEG06
SEG06	SEG05	SEG07
SEG07	SEG06	SEG08
SEG08	SEG07	SEG17
SEG20	SEG17	SEG21
SEG21	SEG20	SEG22
SEG22	SEG21	SEG30
SEG30	SEG22	
SEG17	SEG08	SEG20

Delete Segment

Seg	Prev Seg	Next Seq
SEG02		SEG03
SEG03	SEG02	SEG05
SEG05	SEG03	SEG07
~~SEG06~~	~~SEG05~~	~~SEG07~~
SEG07	SEG05	SEG08
SEG08	SEG07	SEG17
SEG20	SEG17	SEG21
SEG21	SEG20	SEG22
SEG22	SEG21	SEG30
SEG30	SEG22	
SEG17	SEG08	SEG20

Accessing IMS Databases

- Multiple dependent regions
  - Non-serial database access
  - Multiple programming language support
- Full database integrity
  - Processes logging for audit and recovery
  - DBRC for access authorization
- Application independence from Data management & access



Database Access

- All IMS Database access is though a standard programming interface
  - DL/1 using DLI
    - Data Language 1 through the Data Language Interface

AIBTDLI	Application Interface – REXX, Java using PCB labels
ASMTDLI	Assembler programs
CTDLI	C programs
CEETDLI	C programs using PCB labels
CBLTDLI	COBOL programs
PASTDLI	Pascal programs
PLITDLI	PL/1 programs

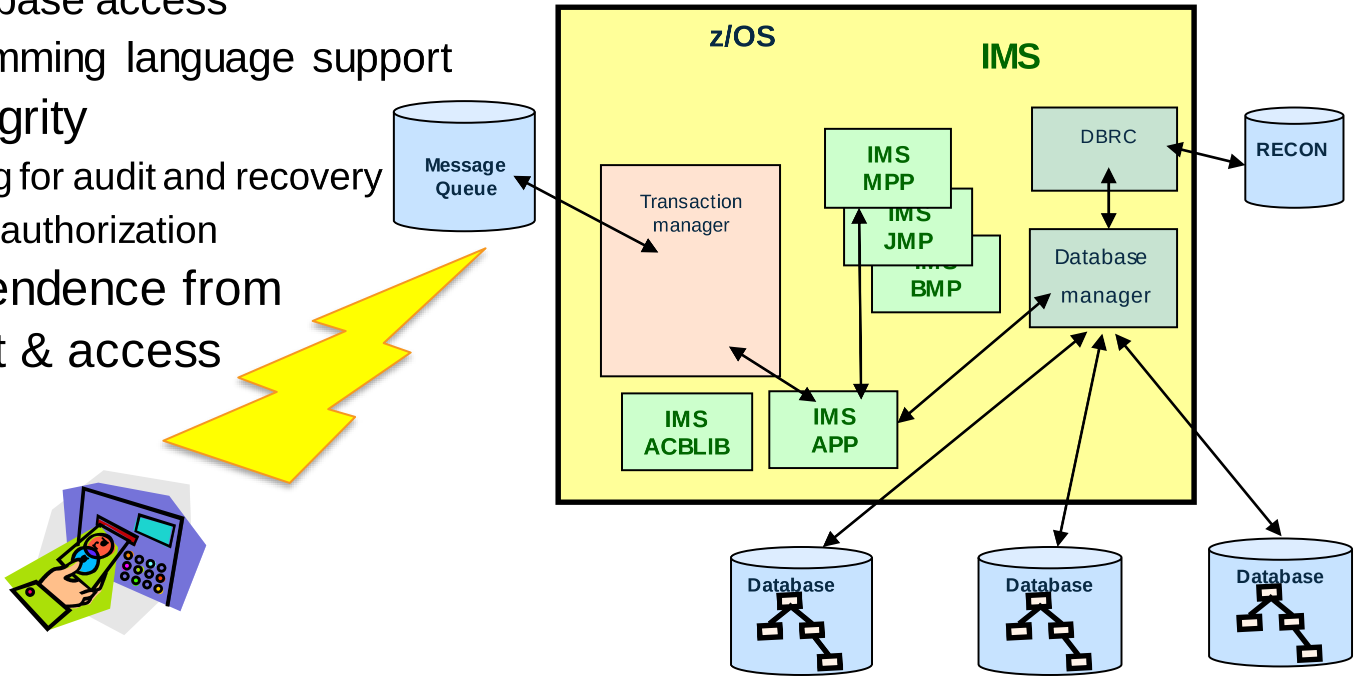
Database Access

- DL/1 call
  - CALL CBLTDLI USING GU *dbpcb i/oarea argument*

  - Translates to:
    - LLZZtrancode ‘search argument’ landing on the Message Queue.
    - IMS sees ‘trancode’ and finds the SMB matching that
    - Loads the application program to run into a dependent region
    - Passes the input message to the program

Accessing IMS Databases

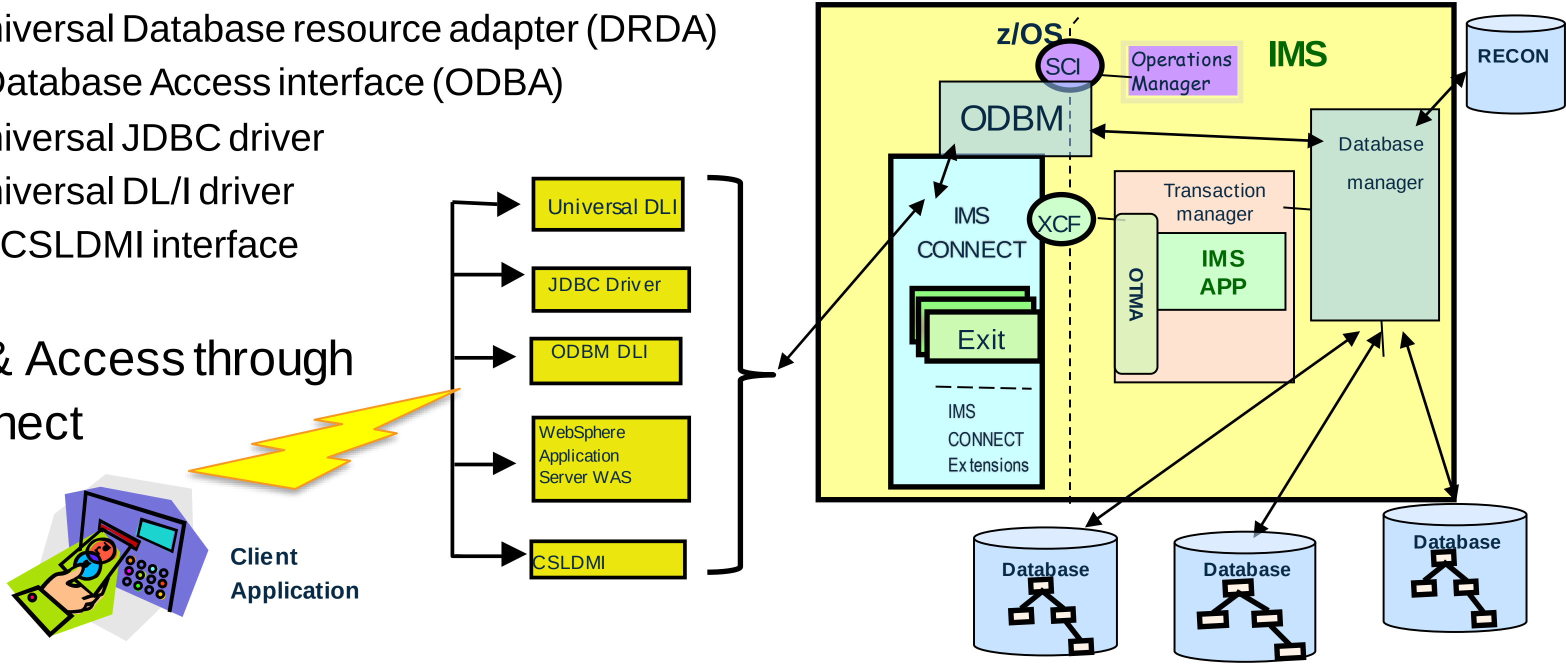
- Multiple dependent regions
  - Non-serial database access
  - Multiple programming language support
- Full database integrity
  - Processes logging for audit and recovery
  - DBRC for access authorization
- Application independence from Data management & access



Open Database Manager Access - ODBM

- Implemented with Common Service Layer
- Allows database access outside of dependent region
  - IMS Universal Database resource adapter (DRDA)
  - Open Database Access interface (ODBA)
  - IMS Universal JDBC driver
  - IMS Universal DL/I driver
  - ODBM CSLDMI interface

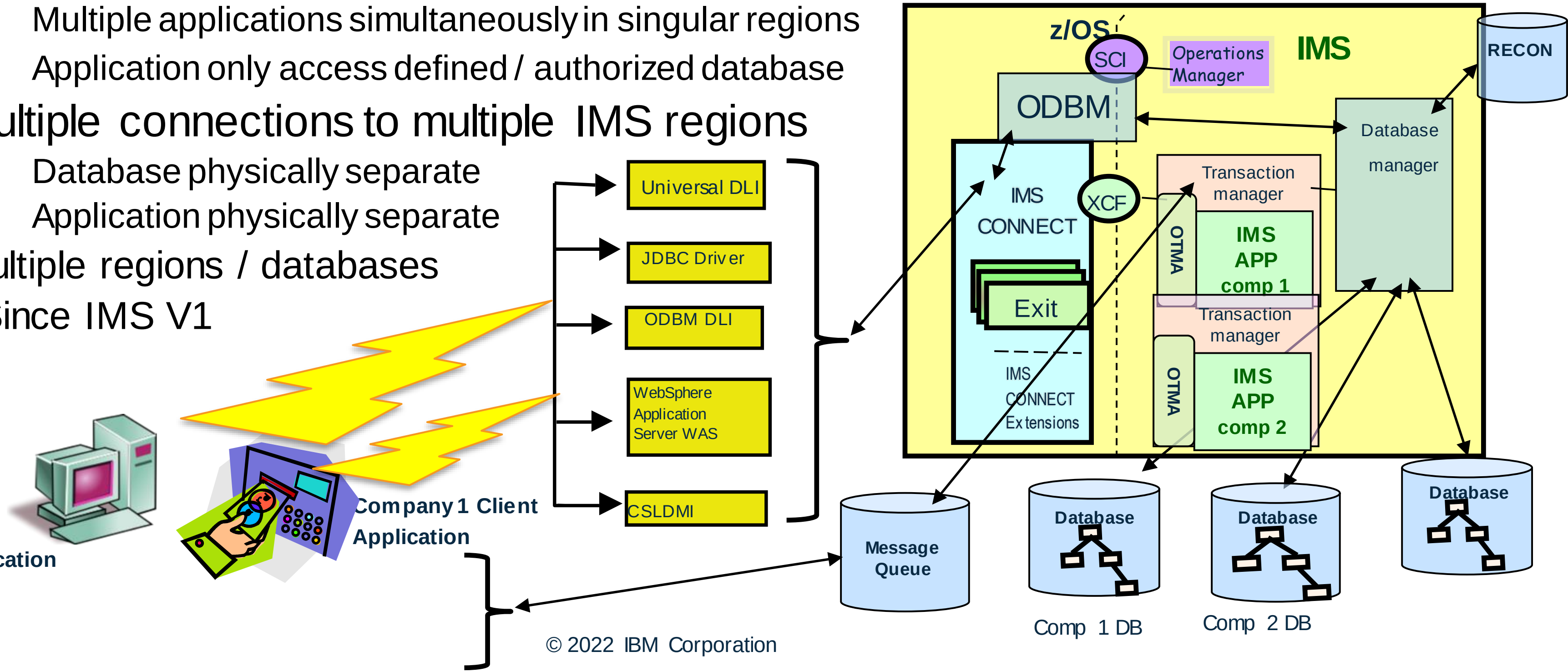
- Security & Access through IMS Connect



Access

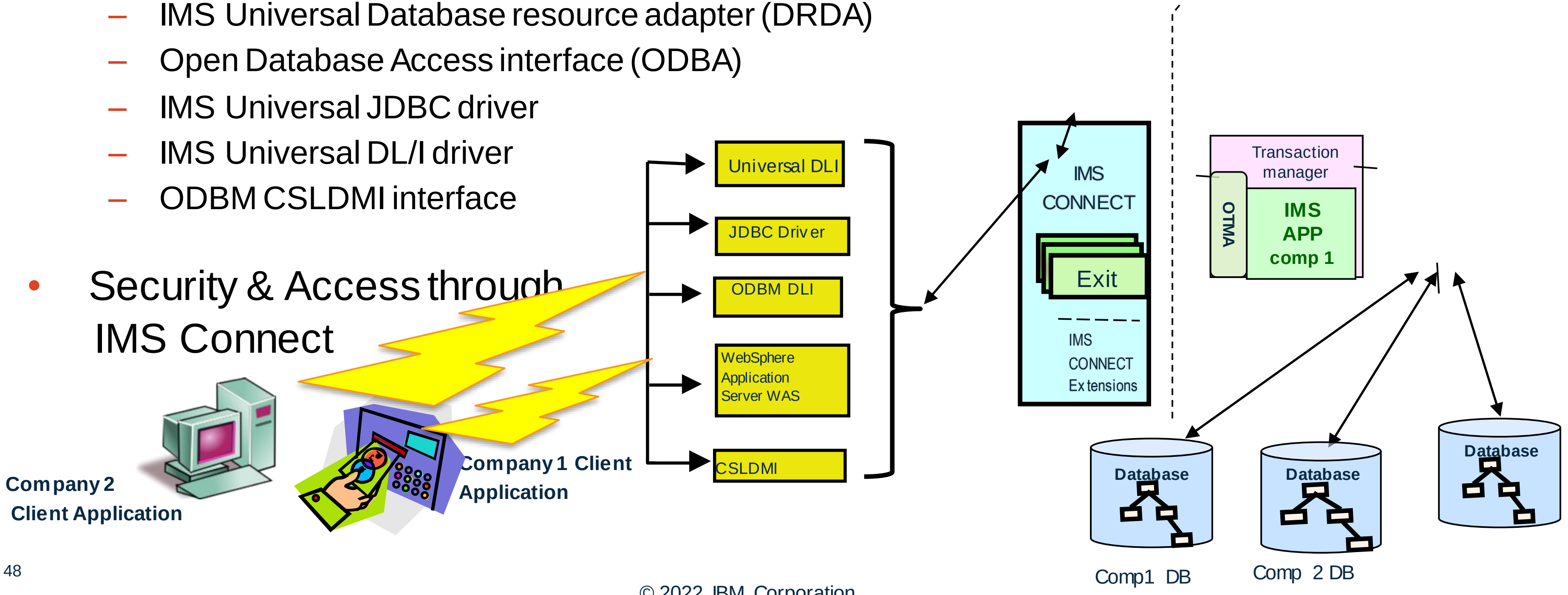
- Multiple connections to single IMS for multiple users
  - Database access independent of application
  - Multiple applications simultaneously in singular regions
  - Application only access defined / authorized database
- Multiple connections to multiple IMS regions
  - Database physically separate
  - Application physically separate
- Multiple regions / databases

  Since IMS V1



Open Database Manager Access - ODBM

- Implemented with Common Service Layer
- Allows database access outside of dependent region
  - IMS Universal Database resource adapter (DRDA)
  - Open Database Access interface (ODBA)
  - IMS Universal JDBC driver
  - IMS Universal DL/I driver
  - ODBM CSLDMI interface
- Security & Access through IMS Connect



Value of Hierarchical Structures

- Performance
  - Efficient navigation
    - Parent/child relationships are implemented with pointers from one data record to another
    - Also true for the sideways relationships from child to child and parent to parent (twins)
    - Therefore
      - After finding the first record, the program does not have to search an index (or do a table scan) to find the next record
      - Application simply needs to follow one of the multiple child record pointers, the single sibling record pointer, or the single parent record pointer to get to the "next" record.
  - Effective Pre-defined structures

- Ease of understanding
  - DB organization is similar to that of a corporate organization chart or family tree
    - A familiar concept to many
  - Moreover, it easily depicts relationships where B is a part of A

(as was the case with the order database we discussed, where each item or segment is a part of an order).

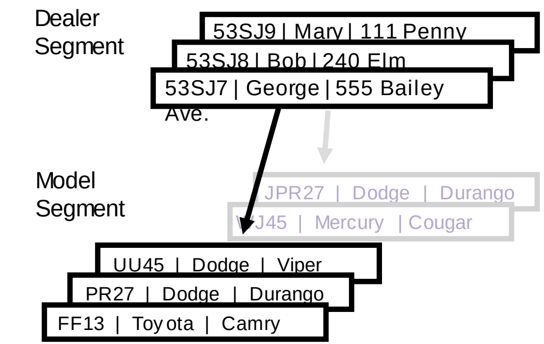
And Yet

- Understanding that the world has moved towards the relational view of databases and SQL to access them
  - IMS also provides the ability to view the hierarchical database structure in a relational way
    - The IMS Enterprise Suite Explorer
      - Creates an SQL View of the Hierarchical database
        - Metadata mapping from the IMS Catalog
    - And a set of java universal drivers that interpret the calls made by java programs
      - Local to z/OS
      - From remote environments

Support for the IMS Universal Drivers: map Hierarchical to Relational

Allows an application to view an IMS hierarchical database as though it was a set of relational tables

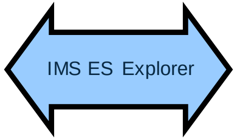
Hierarchical Design



Note: **Segment Names ~ Table Names**
Segment Instances ~ Table Rows
Field Names ~ Column Names

```
SQL:  SELECT Dealer.Name, Dealer.Phone, Order.LastName
       FROM SomePCB.Order
       WHERE Model.MSRP>'50000'
             AND Order.Date>='5/1/2007'
             AND Order.Date<='5/31/2007'

SSA LIST:    DEALER
             MODEL
             ORDER
```



Relational Design

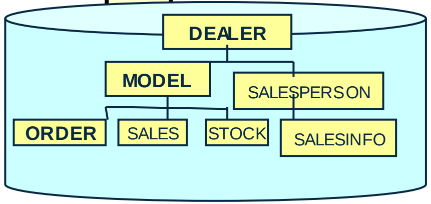
Dealer Table

	DealerID	DealerName	DealerAddress
0	53SJ7	George	555 Bailey Ave.
1	53SJ8	Bob	240 Elm St.
2	53SJ9	Mary	111 Penny Ln.
...	...	...	...

Relational JOIN

Model Table

ID	Make	Model	Dealer	
UU45	Dodge	Viper	53SJ7	0
PR27	Dodge	Durango	53SJ7	0
FF13	Toyota	Camry	53SJ7	0
JR27	Dodge	Durango	53SJ8	1
WJ45	Mercury	Cougar	53SJ8	1
...	...	...		
...				



IMS Catalog

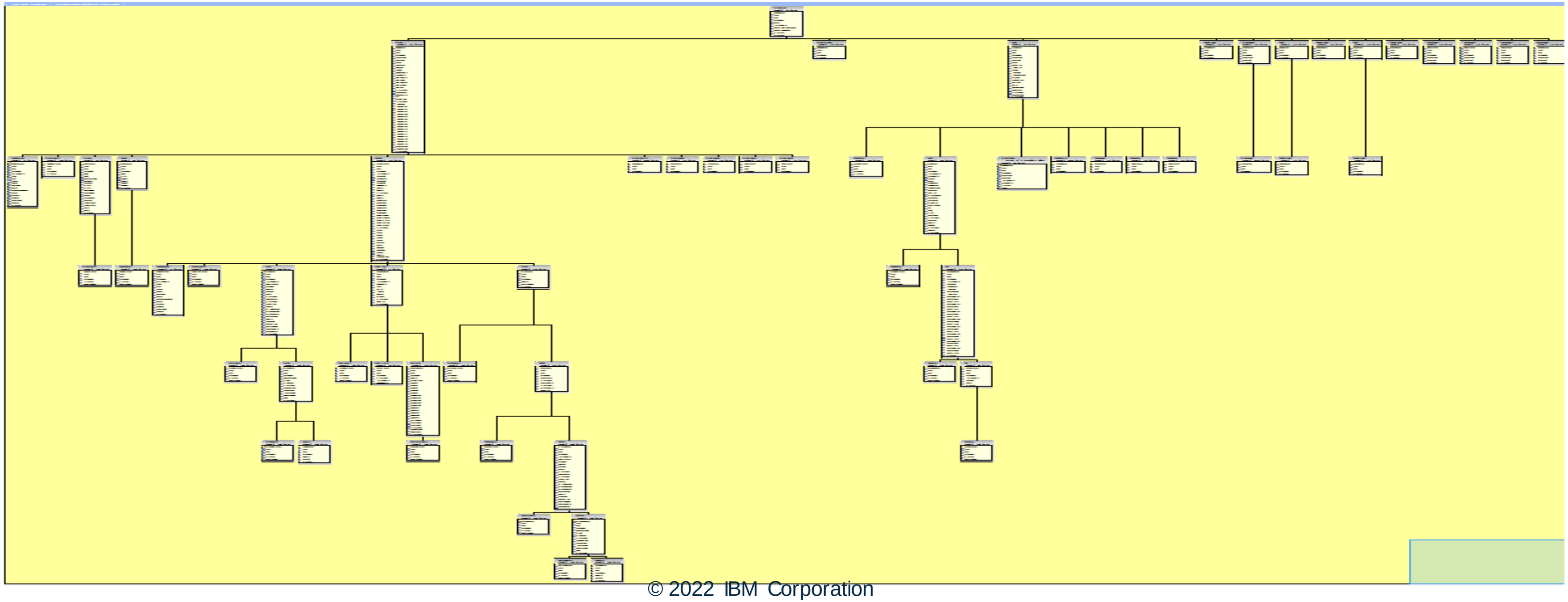
- Database Considerations
  - Traditional database definitions
    - A segment total length
    - A defined Key / Sequence field
    - The remainder is defined as 'space'
    - Relies on the application program to map the 'space' using copy books
  - Relational views of traditional IMS databases using SQL or JDBC
    - Require search fields to be defined in the meta data
      - IMS meta data is stored in the IMS Catalog
  - Adding meta data to the Catalog
    - Hand key all identified fields into the DBD source
      - Regenerate the DBD, the ACB and
      - Online Change to activate
    - Use IMS 14 with the IMS Explorer for Development to create DDL
      - Send DDL directly to the IMS Catalog
      - IMPORT DEFN SOURCE(CATALOG) to activate (TYPE 2 Command)

IMS Catalog

- IMS 12 added an IMS Catalog
  - IMS meta data
    - Single source of Trusted repository of meta data
    - Loaded and updated from IMS ACBLIB
    - Data about data
  - HALDB
    - Structure
      - DBRC registered
      - Four dataset groups
      - Single partition
    - Database maintenance procedures apply
      - Image Copy
      - Reorganization
      - Recovery

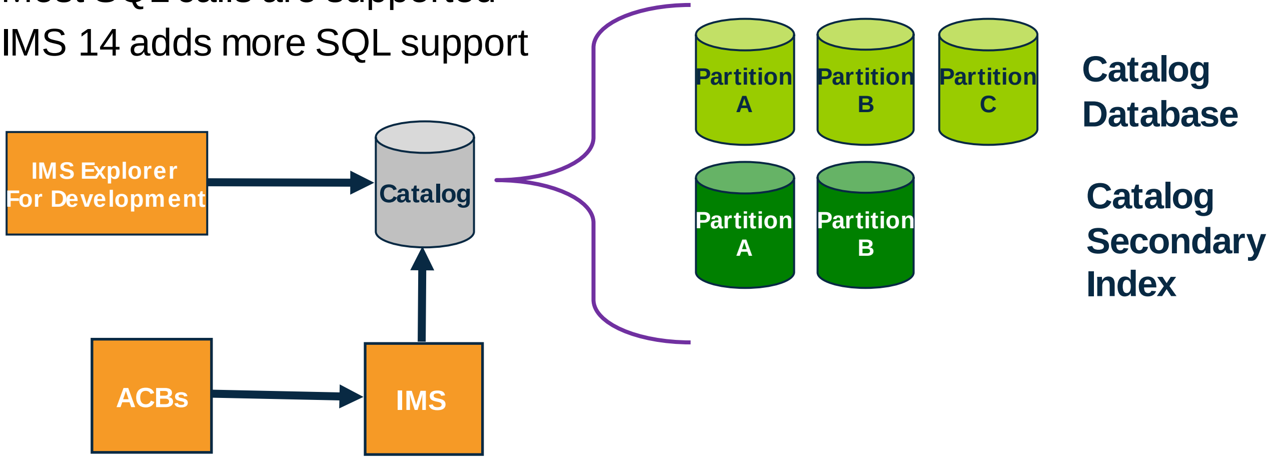
IMS Catalog

- IMS 12 added an IMS Catalog
  - IMS control block meta data
    - Single source of Trusted repository of meta data
    - Loaded and updated from IMS ACBLIB



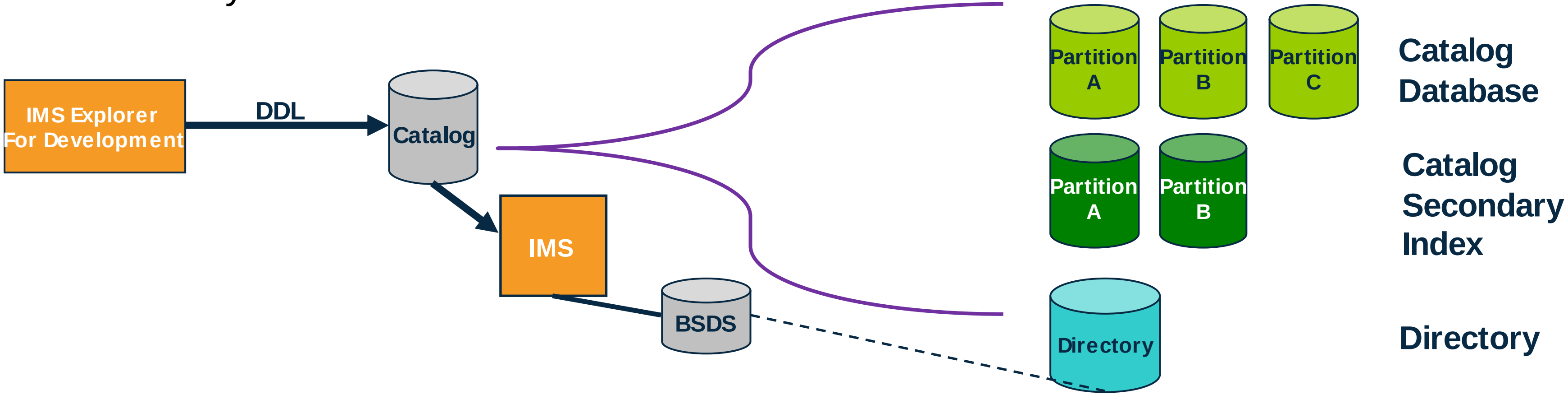
SQL calls to IMS databases

- IMS 13 added direct SQL access using COBOL 5.1 or later
  - Builds upon the catalog trusted meta data
  - Allows a COBOL application program to make SQL calls to access IMS hierarchical databases
    - Most SQL calls are supported
    - IMS 14 adds more SQL support



IMS Managed ACBs and DDL

- IMS 14 added IMS Managed ACBs
  - Allows direct update to an IMS Catalog using Universal Drivers
  - Data Definition Language may be used for updating
  - Allows IMS to Managed its own ACBs
    - IMS Control blocks are loaded from the Catalog
    - Proprietary DBD, PSB and ACB generations are no longer needed
  - Industry standards for IMS resource and schema definitions



Defining Databases

Meta data update

1. Add fields needed to the DBD source to regen

2. FTP Source to DBD
3. Generate the DBD
4. Generate the ACB
5. Online Change

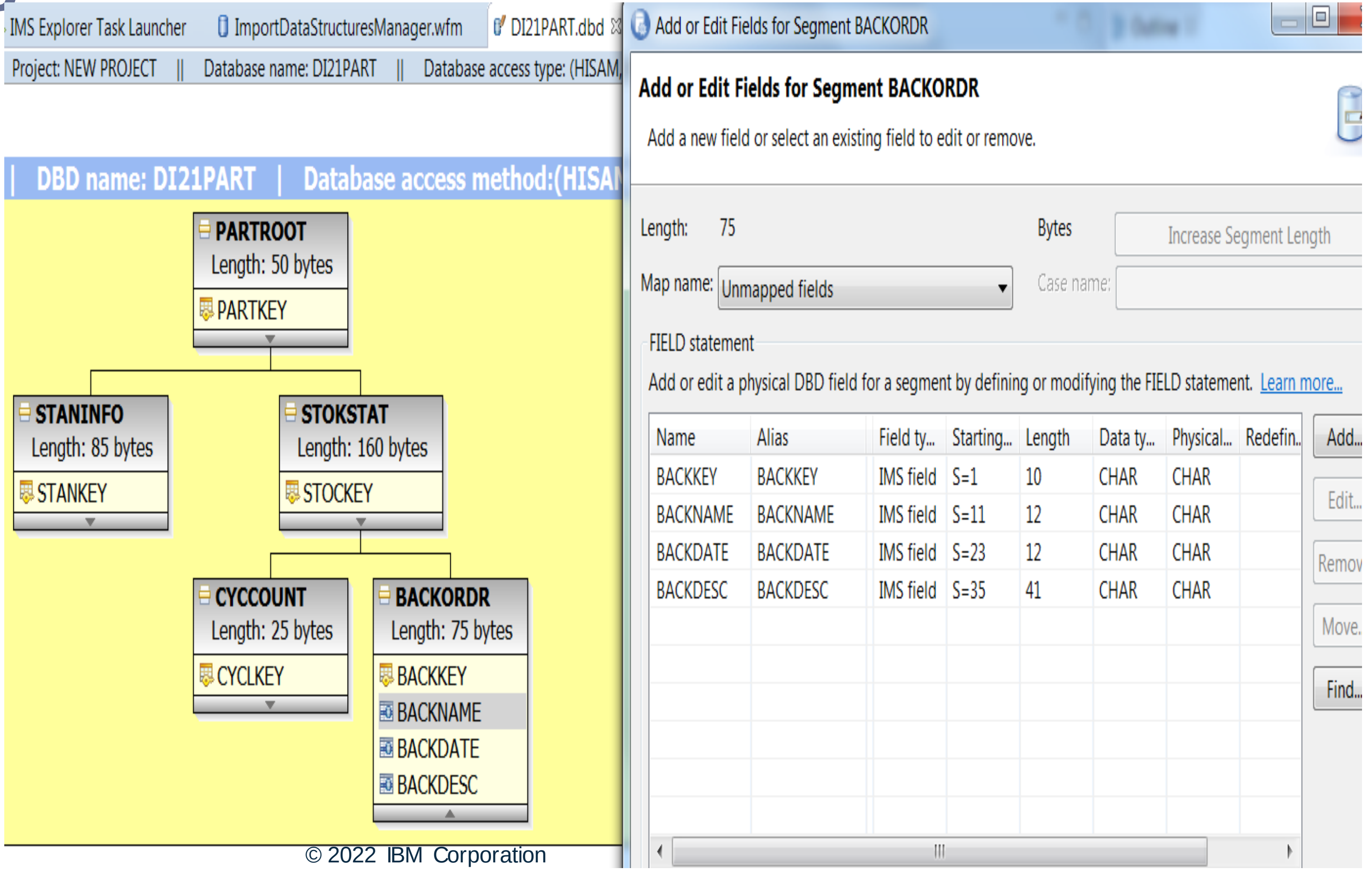
```
DBD    NAME=DI21PART,                                                   X
               ACCESS=(HISAM,VSAM),                                     X
               PASSWD=NO,                                               X
               DATXEXIT=NO,                                             X
               ENCODING=CP1047
         DATASET DD1=DI21PART,                                          X
               DEVICE=3380,                                             X
               OVFLW=DI21PARO,                                          X
               BLOCK=(100,10),                                          X
               SIZE=(4096),                                             X
               RECORD=(678)
         SEGM  NAME=PARTROOT,                                           X
               EXTERNALNAME=PARTROOT,                                   X
               PARENT=0,                                                X
               BYTES=50,                                                X
               RULES=(LLL,LAST),                                        X
               ENCODING=Cp1047
         FIELD NAME=(PARTKEY,SEQ,U),                                    X
               EXTERNALNAME=PARTKEY,                                    X
               BYTES=17,                                                X
               START=1,                                                 X
               TYPE=C,                                                  X
               DATATYPE=CHAR
         FIELD NAME=PARTNAME,                                           X
               EXTERNALNAME=PARTNAME,                                   X
               BYTES=24,                                                X
               START=18,                                                X
               TYPE=C,                                                  X
               DATATYPE=CHAR
         SEGM  NAME=STOCINFO,                                           X
```

IMS Catalog

Meta data update

Use E4D to generation DDL for the Catalog

1. Add fields using E4D Editor



IMS Explorer Task Launcher
ImportDataStructuresManager.wfm
DI21PART.dbd
Project: NEW PROJECT || Database name: DI21PART || Database access type: (HISAM
| DBD name: DI21PART | Database access method:(HISAM
PARTROOT
Length: 50 bytes
PARTKEY
STANINFO
Length: 85 bytes
STANKEY
STOKSTAT
Length: 160 bytes
STOCKEY
CYCCOUNT
Length: 25 bytes
CYCLKEY
BACKORDR
Length: 75 bytes
BACKKEY
BACKNAME
BACKDATE
BACKDESC
Add or Edit Fields for Segment BACKORDR
Add...
Edit...
Remov
Move...
Find...

Add or Edit Fields for Segment BACKORDR

Add a new field or select an existing field to edit or remove.

Length: 75 Bytes Increase Segment Length

Map name: Unmapped fields Case name:

FIELD statement

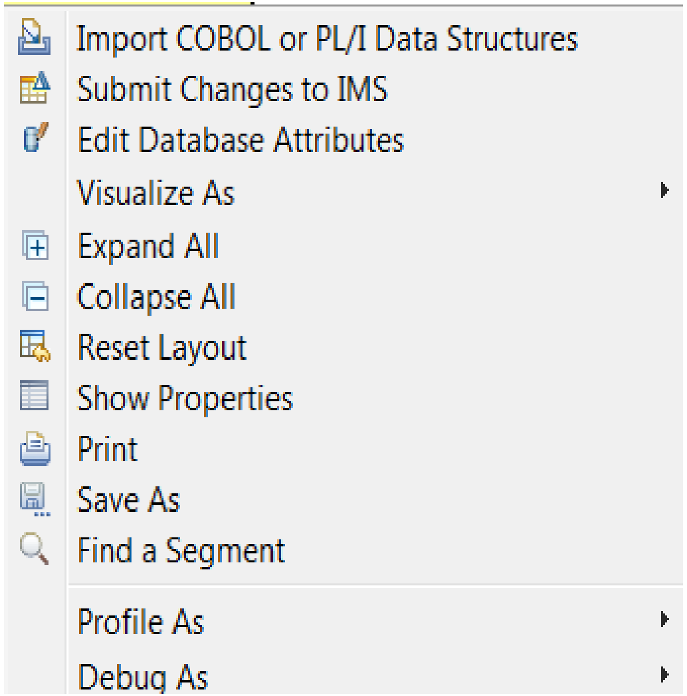
Add or edit a physical DBD field for a segment by defining or modifying the FIELD statement. Learn more...

Name	Alias	Field ty...	Starting...	Length	Data ty...	Physical...	Redefin...
BACKKEY	BACKKEY	IMS field	S=1	10	CHAR	CHAR	
BACKNAME	BACKNAME	IMS field	S=11	12	CHAR	CHAR	
BACKDATE	BACKDATE	IMS field	S=23	12	CHAR	CHAR	
BACKDESC	BACKDESC	IMS field	S=35	41	CHAR	CHAR	

IMS Catalog

Meta data update

- Use E4D to generation DDL for the Catalog

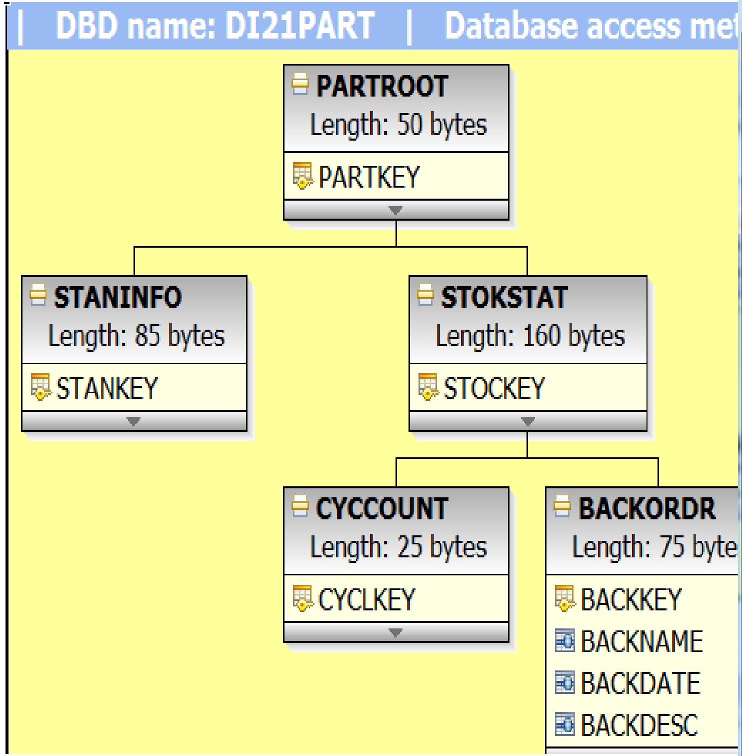


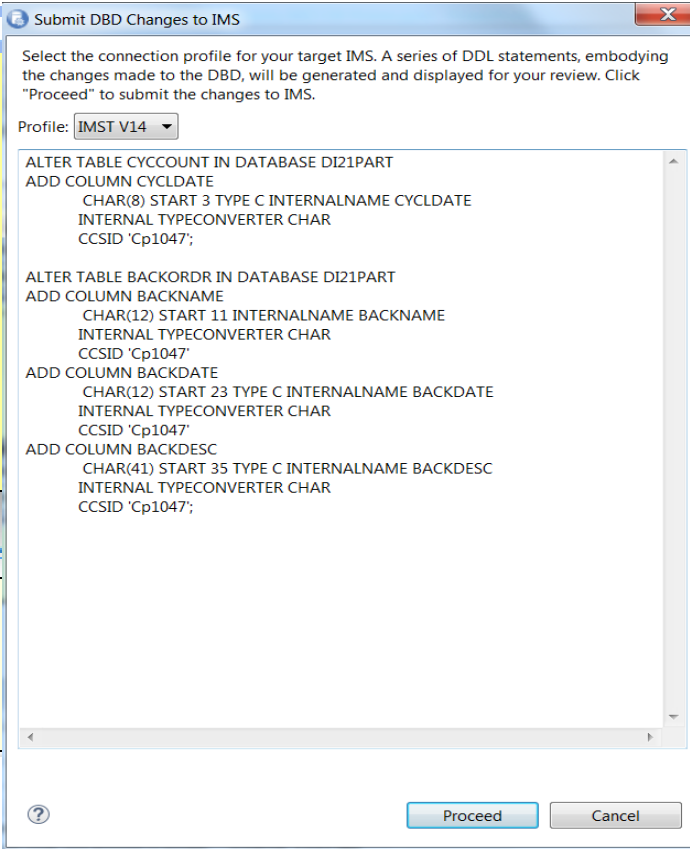
IMS Catalog

Meta data update

Use E4D to generation DDL for the Catalog

2. Generate ALTER DDL statements
3. Submit to Catalog
4. Import definitions





Select the connection profile for your target IMS. A series of DDL statements, embodying the changes made to the DBD, will be generated and displayed for your review. Click "Proceed" to submit the changes to IMS.

Profile: IMST V14

```
ALTER TABLE CYCCOUNT IN DATABASE DI21PART
ADD COLUMN CYCLDATE
          CHAR(8) START 3 TYPE C INTERNALNAME CYCLDATE
         INTERNAL TYPECONVERTER CHAR
         CCSID 'Cp1047';

ALTER TABLE BACKORDR IN DATABASE DI21PART
ADD COLUMN BACKNAME
          CHAR(12) START 11 INTERNALNAME BACKNAME
         INTERNAL TYPECONVERTER CHAR
         CCSID 'Cp1047'
ADD COLUMN BACKDATE
          CHAR(12) START 23 TYPE C INTERNALNAME BACKDATE
         INTERNAL TYPECONVERTER CHAR
         CCSID 'Cp1047'
ADD COLUMN BACKDESC
          CHAR(41) START 35 TYPE C INTERNALNAME BACKDESC
         INTERNAL TYPECONVERTER CHAR
         CCSID 'Cp1047';
```

Summary

- IMS is a database manager
  - Hierarchical databases
  - High Available and large database
  - May be accessed from IMS TM, CICS, WAS, Db2 SP, distributed, any platform
- IMS is a transaction manager
  - Multi-address space architecture
  - Processes messages
  - Schedules and manages programs
  - May be accessed via VTAM, TCP/IP, MQ, …
  - May be accessed from any platform
  - Accesses IMS and DB2 databases

Reference Information

- More information
  - IMS home page
    - www.ibm.com/ims
  - *IMS Primer* SG24-5352
    - Available at www.ibm.com/redbooks
  - *An Introduction to IMS*
    - Available on Amazon.com, textbookx.com, ibmpressbooks.com, etc.