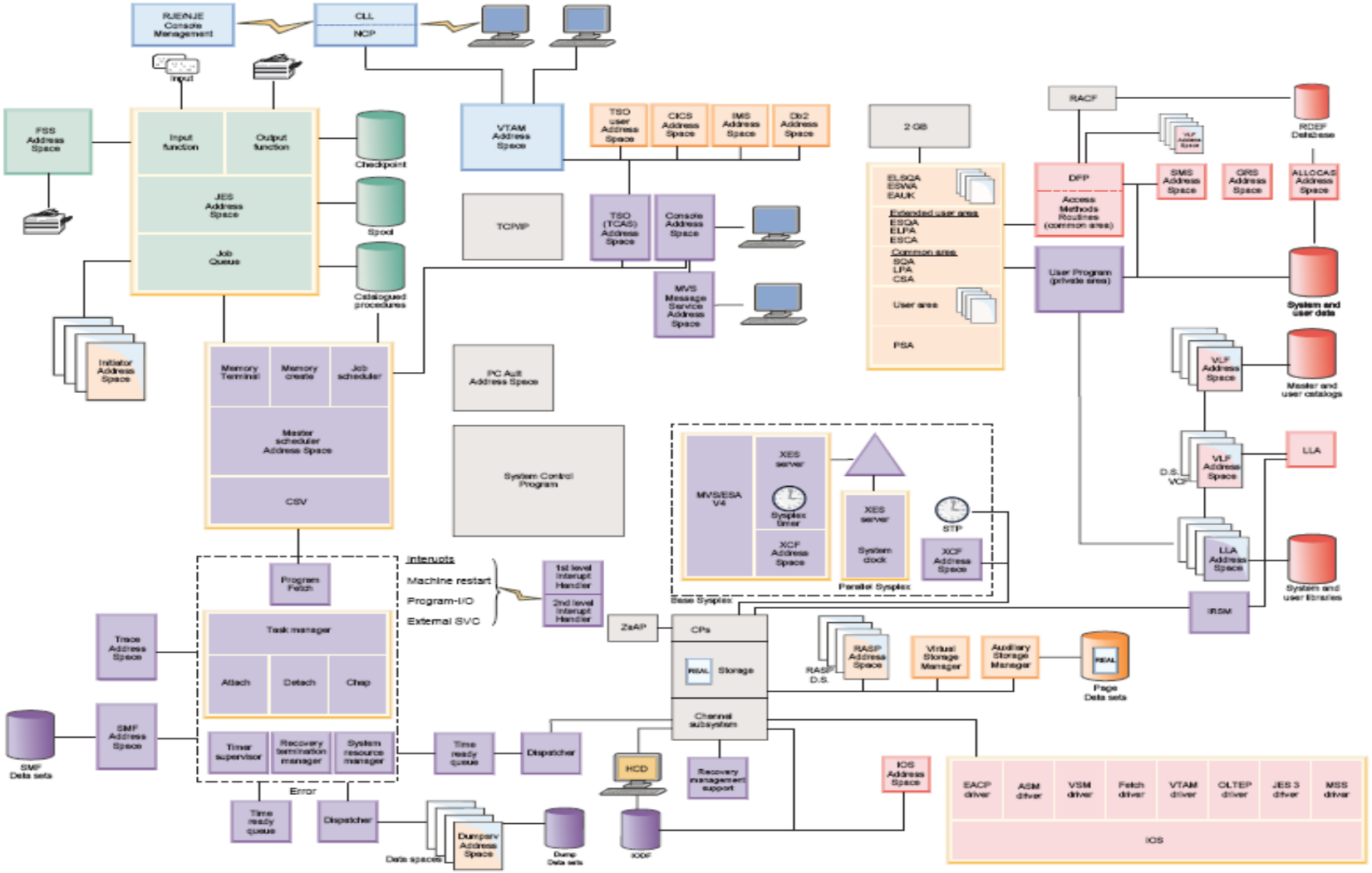
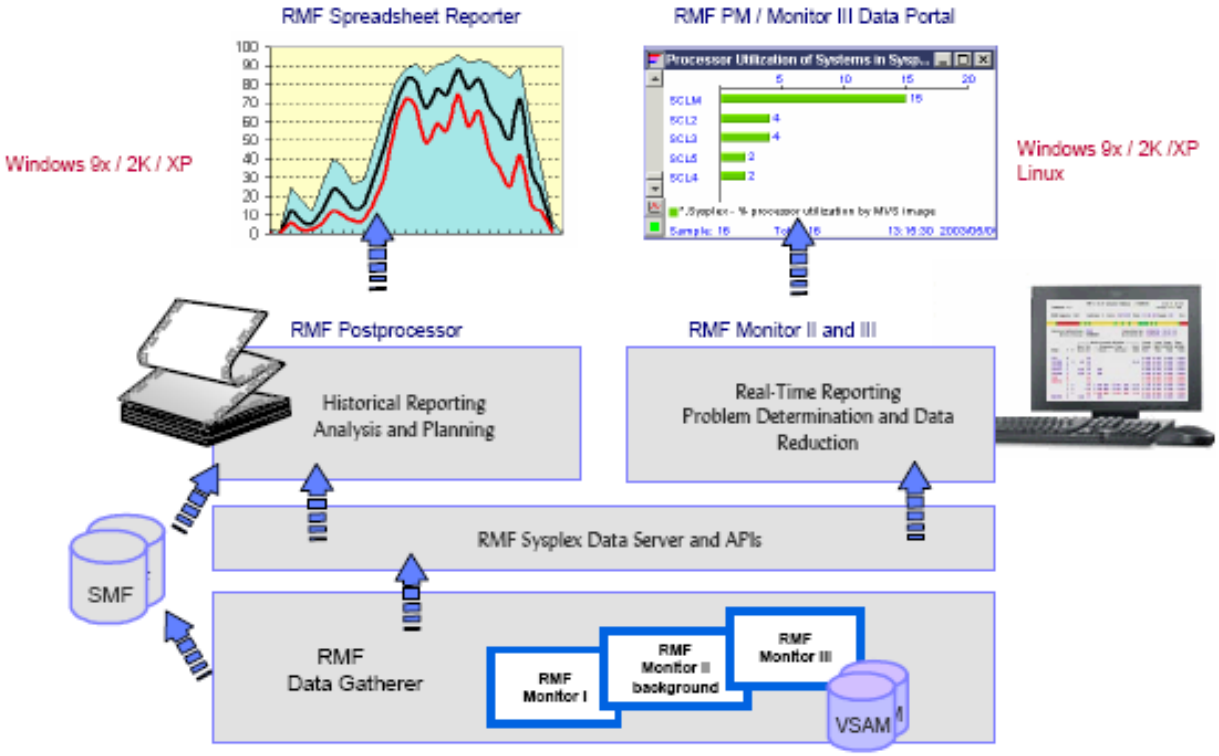
Sistemi Centrali – Modulo 3- Il sistema operativo z/OS (seconda parte)

z/OS poster



Resource Monitor Facility



Workload Manager (WLM)

- Workload Manager (WLM) e' una componente del Sistema Operativo z/OS.

  Esso consente l'esecuzione simultanea di programmi concorrenti in accordo con politiche complessive di utilizzo ed in modo tale che venga raggiunto un obiettivo complessivo di prestazione (Goal).

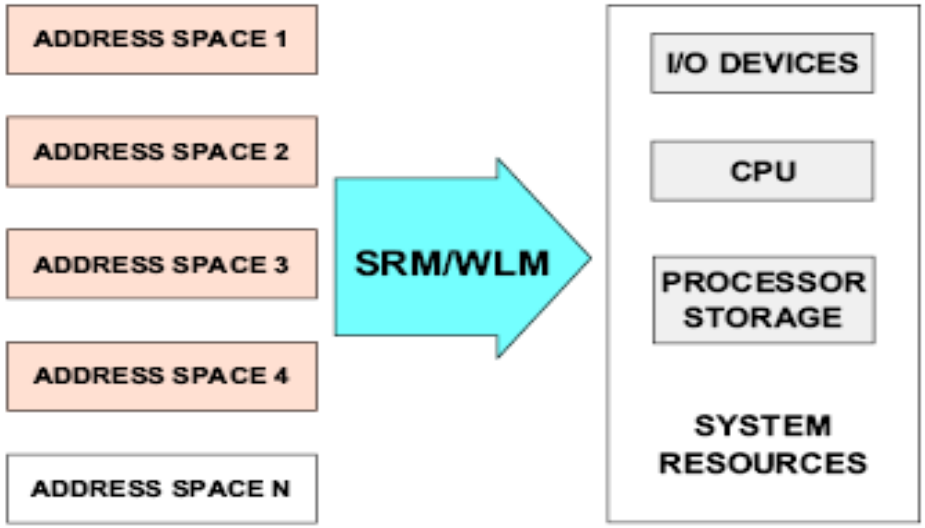
- Il WLM consente di gestire di il comportamento di istallazioni complesse in base a politiche ovvero insiemi di regole che si adattano alle richieste degli utenti dei servizi.

- Si possono usare politiche diverse nell'arco di una giornata o di una settimana .
  Ad esempio favorire le attivita' legate alle richieste di applicazioni Web o CICS in un orario mattutino e privilegiare i lavori di compilazione applicativa per utenti programmatori nel pomeriggio .
  Oppure distinguere politiche per il workload nel week-end e quello dei giorni della settimana dal lunedi' al venerdi'

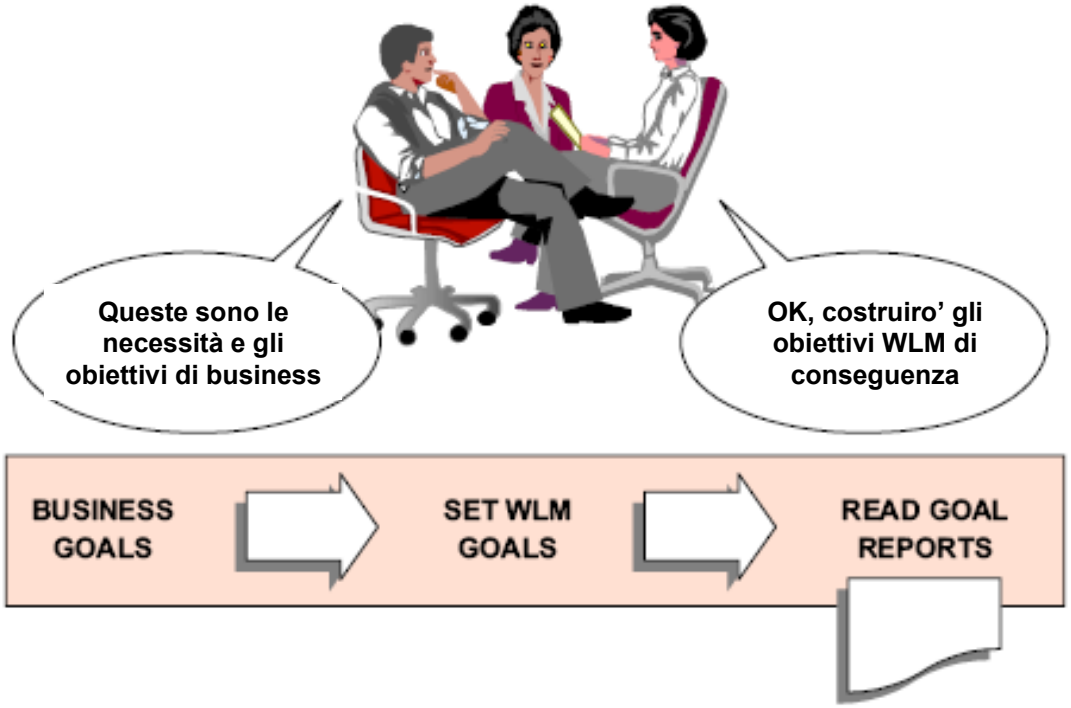
Workload manager (WLM)

il Workload Manager e' in grado di:

- Agire sulle risorse del Sistema (Come CPU e Memoria)
- Modificare l'ordine di esecuzione dei Job, la priorita' dei programmi in esecuzione
- Modificare i meccanismi di uso della memoria Virtuale , della paginazione
- Modificare la priorita' nell'uso dei dispositivi di I/O.

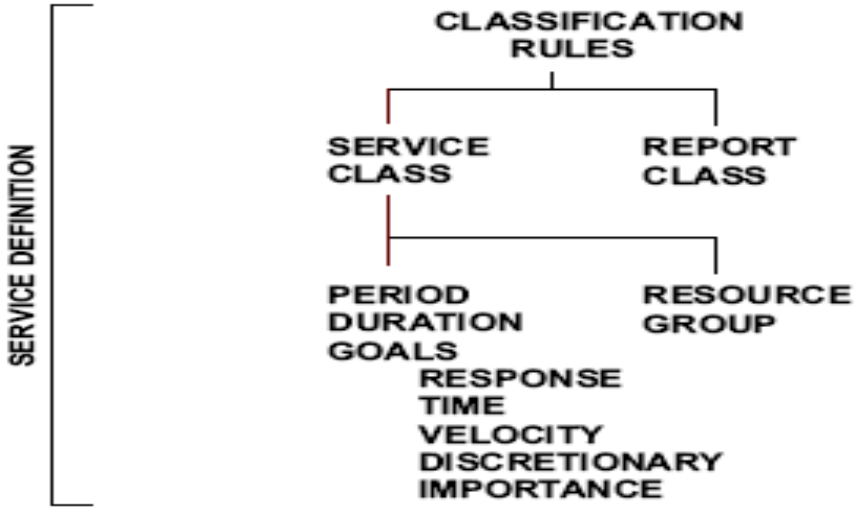


Obiettivi di Business e di WLM



Regole di classificazione e regole di servizio

- I vari tipi di lavoro (workload) vengono raggruppati in classi (service class)
- All'interno di una classe sara' definito un obiettivo prestazionale (performance goal)
- Un insieme di Service Class sarà parte di una Regola di Classificazione (Classification Rule)
- Ad una Regola di Classificazione (CR) corrisponderà una Politica di Servizi (Service Policy) ovvero un ben identificato insieme di obiettivi che l'istallazione intende raggiungere



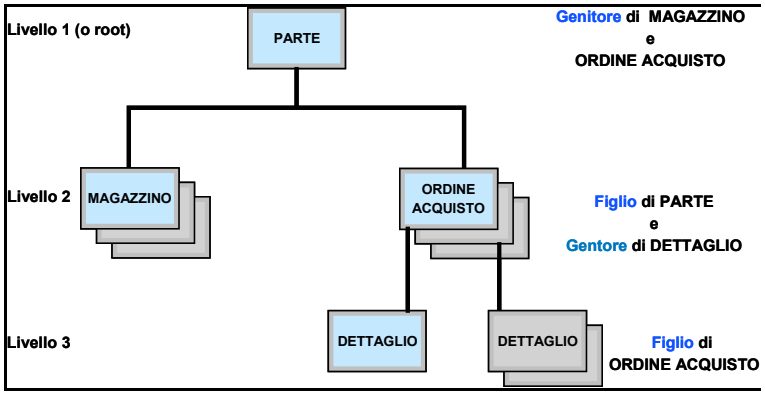
Esempio di definizione di service class

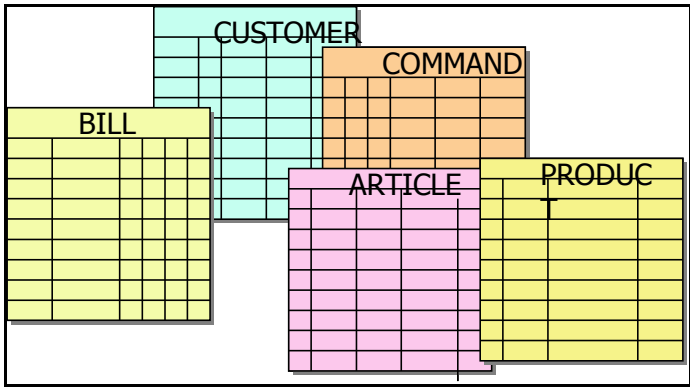
Service Class	Per	Goal	Value	Imp
CICS	1	Avg RT	0.1 s	2
IMS	1	Avg RT	10 s	3
TSO	1	Avg RT	0.1 s	3
	2	Avg RT	1.0 s	3
	3	Avg RT	3.0 s	4
Batch	1	ExVel	10	5

Database– Definizione

Il database e' una collezione di dati correlati o indipendenti memorizzati insieme per servire una o più applicazioni

Dal punto di visto operativo un database fornisce un mezzo per la memorizzazione e il controllo di dati di business. Esso è indipendente dall'applicazione che lo utlizza.
Se correttamente disegnato il database offre una vista consistente dei dati produttivi tali da essere gestiti centralmente.





Data Base Management System– Definizione

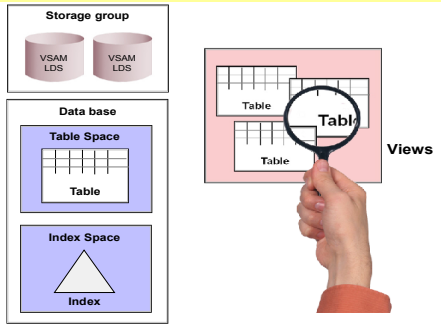
Il DataBase Management System (DBMS) e' una struttura completa per la Gestione di grandi volumi di dati , in grado di mantenere legami di tipo Gerarchico o Relazionale tra di essi.

- Il linguaggio di Consultazione di una Base Dati si basa su un insieme di regole in grado di dare accesso ai dati con l'uso di particolari comandi che fanno riferimento alla struttura di essi.
- DataBase Management Systems (prodotti da IBM) e di piu' frequente uso sotto z/OS sono:

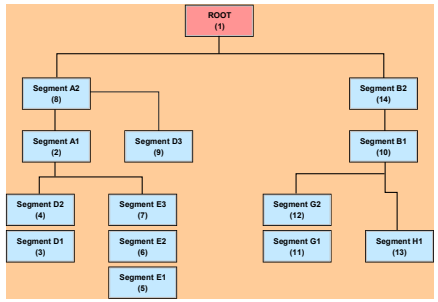
- **DB2 –** Data Base Relazionale caratterizzato dal linguaggio di consultazione Structured Query Language (SQL)

- **IMS/DM** – Data Base Gerarchico caratterizzato dal linguaggio di consultazione Data Language/Interface (DL/1)

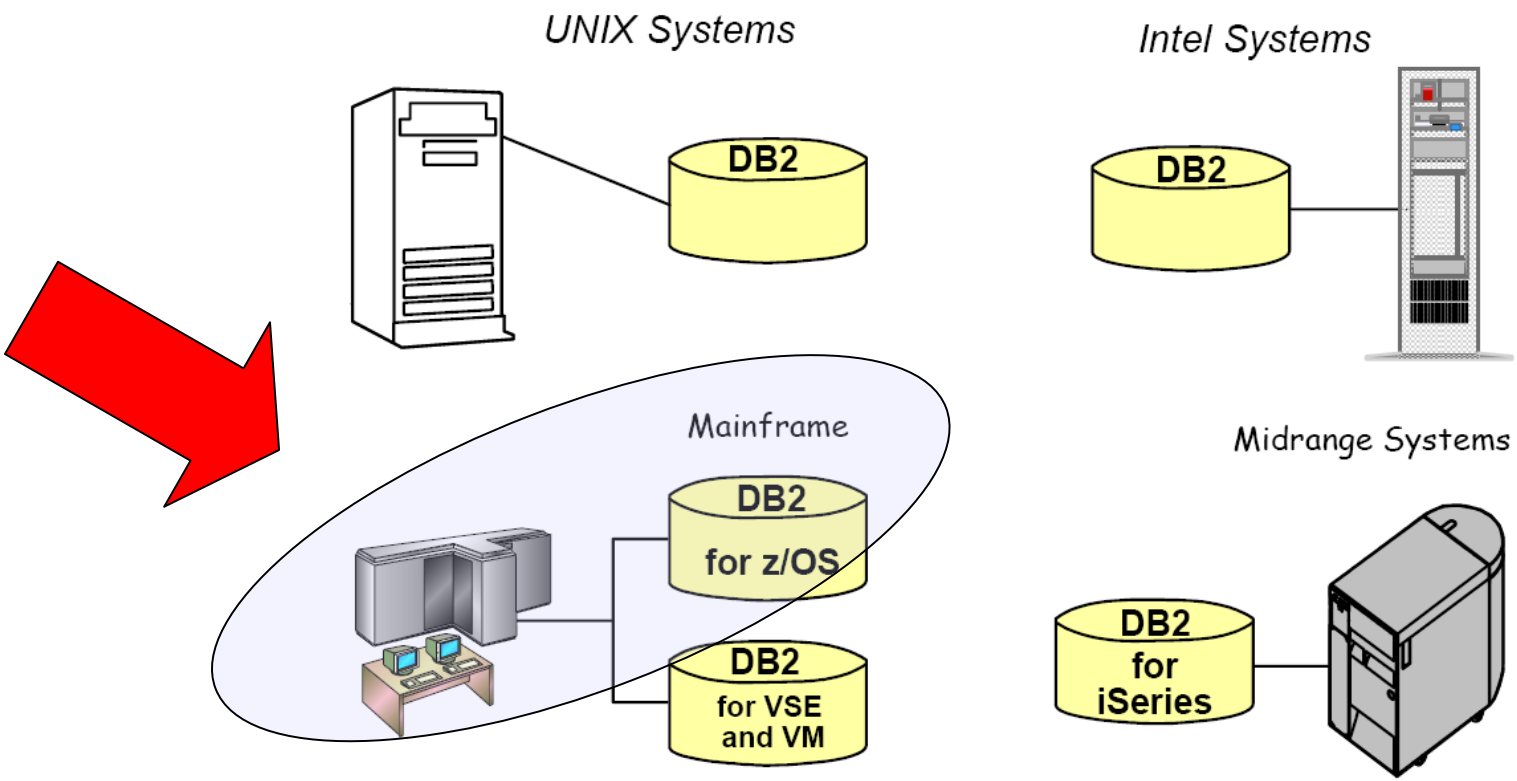
DB2



IMS/DM



Famiglia DB2

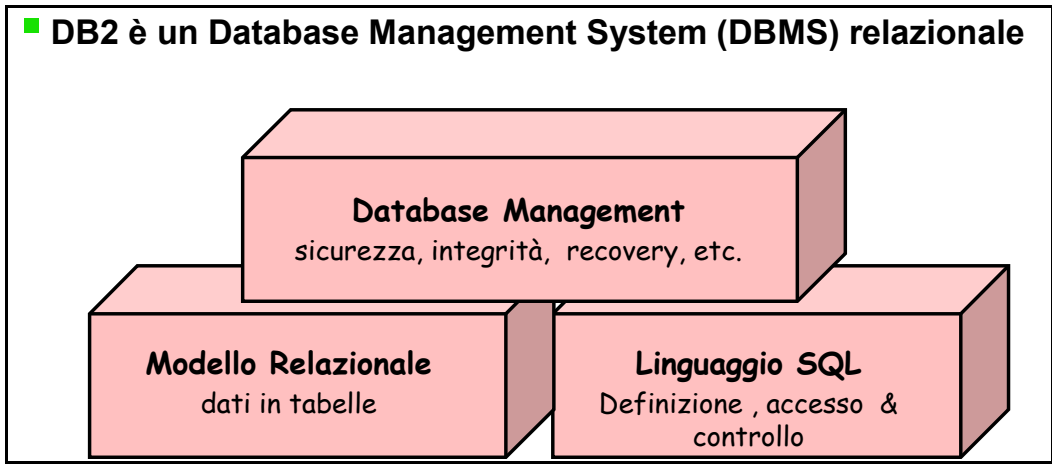


DB2 – Definizione

DB2 e' un sistema di gestione di un data base relazionale (RDBMS) che abilita gli utenti a creare aggiornare e controllare il database usando un linguaggio di interrogazione chiamato Structured Query Language (SQL).

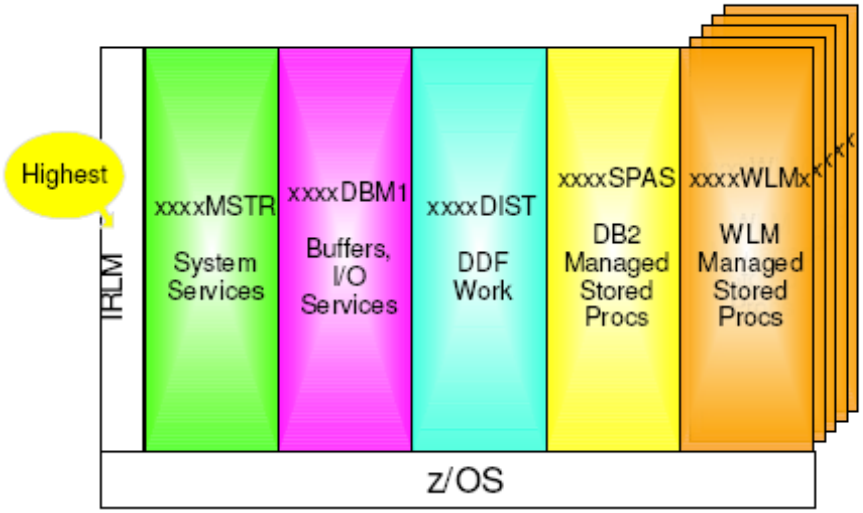
Gli elementi di base sono:

- Oggetti del database
- Cataloghi di sistema
- Directory
- File di configurazione



DB2 - Address Spaces

- **System Service AS (DB2MSTR)** che manipola la maggior parte delle strutture dei database creati dall'utente
- **Database Service AS (DB2DBM1)** dedicato alle funzioni legate al sistema zOS
- **Distributed Data Facility Services AS (DB2DIST)**
da il supporto per le richieste remote
- **Stored Procedures ASs (DB2SPAS e DB2WLMx**
Gestiscono la performance delle Stored Procedures da DB2 o da WLM
- **Internal Resource Lock Manager (IRLM)** la serializzazione degli accessi alle risorse del DB2.

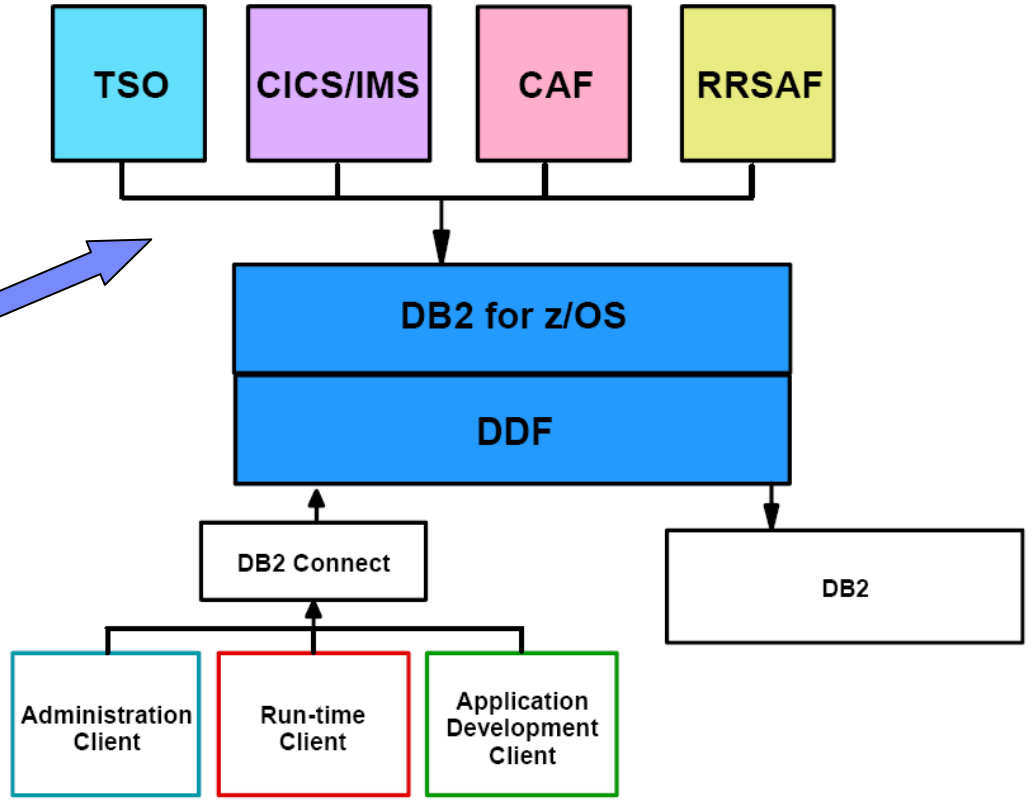


DB2 - Componenti del prodotto

Accesso locale da TSO (da Batch o ISPF)

Accesso locale da Transaction Manager (CICS/IMS)

Accesso remoto via protocollo DRDA



DB2- Gerarchia di oggetti

- **Tabelle**

Dati disposti in righe e colonne

- **Indici**

insiemi ordinati di dati e puntatori alle righe delle tabelle (migliorano la performance)

- **Spazi di Tabelle (Tablespace)**

datasets(files) contenenti tabelle

- **Database**

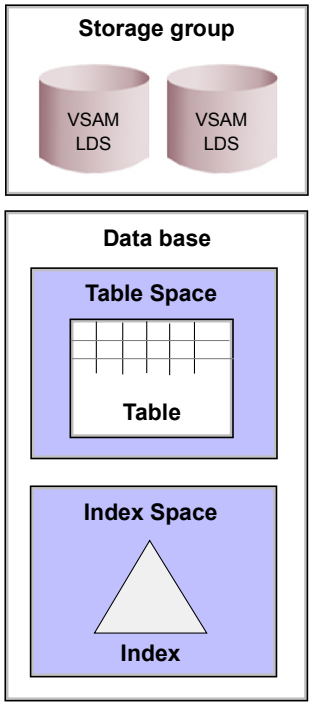
Insieme logico di tabelle e Tablespace

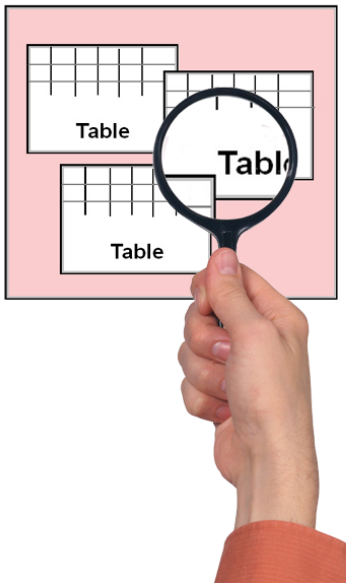
- **Viste**

“ tabelle virtuali" o viste logiche dei dati, o sottoinsiemi di tabelle, o aggregati tra tabelle multiple o database . Le viste non vengono memorizzati su disco. Viste possono essere costruite su altre viste .

- **Chiavi**

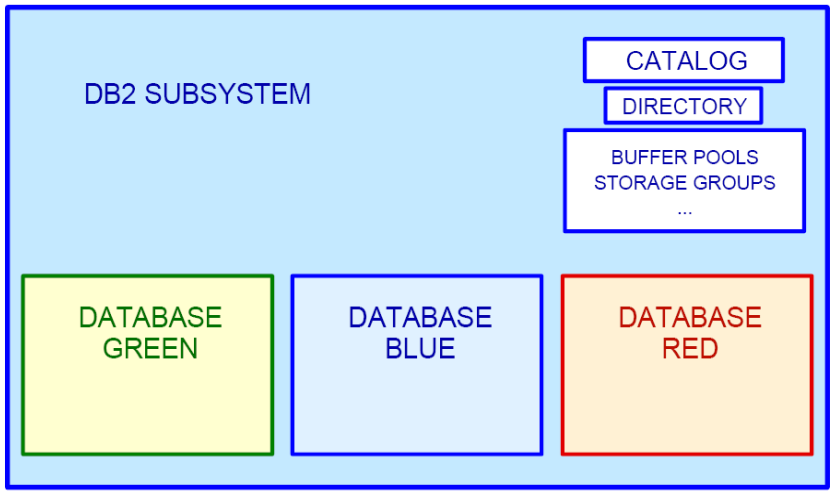
uno o piu’ colonne che sono identificate nella creazione di una tabella o indice e che possono essere utilizzate per identificare una relazione (entità)

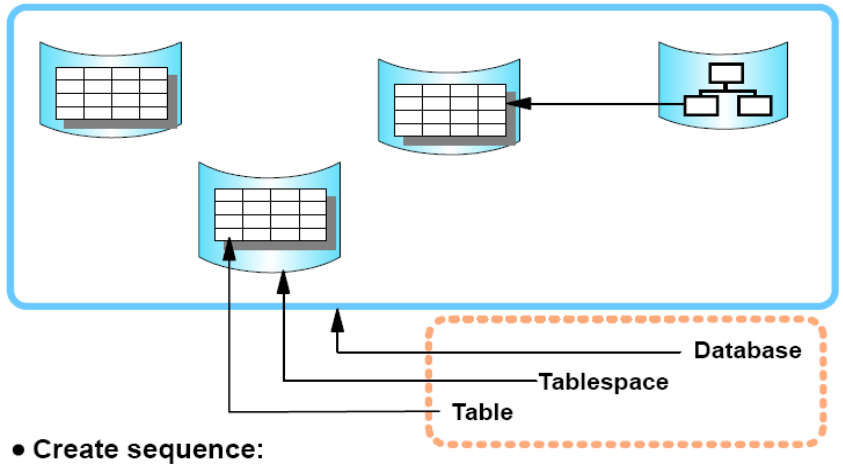




DB2- Gerarchie di oggetti

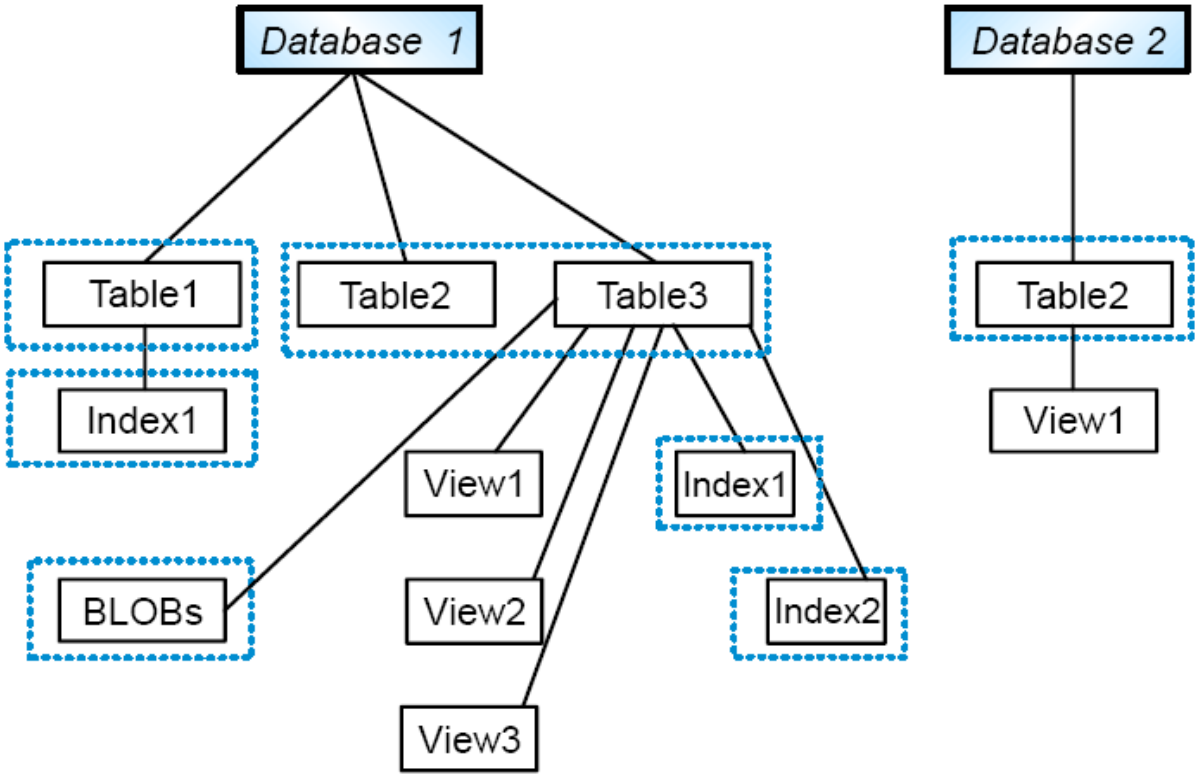
- In ambiente z/OS tipicamente ci sono piu' DB2 (DBMS) ad es. uno di Produzione ed uno di Test.
- Un sottosistema DB2 può avere piu' database
- Il **catalogo** mantiene le informazioni su tutti gli oggetti (tabelle, viste, indici , tablespace ecc)
- La **directory** mantiene le informazioni su i programmi applicativi
- La **log** contiene tutti i cambiamenti ai dati e gli eventi significativi del data base
- **I buffer pools** sono aree di memoria virtuale dove il DB2 mantiene temporaneamente pagine di tablespace o indici
- Ogni DB2 ha il suo catalogo directory Indice , log e buffer pools
- Ogni database può avere molti tablespace



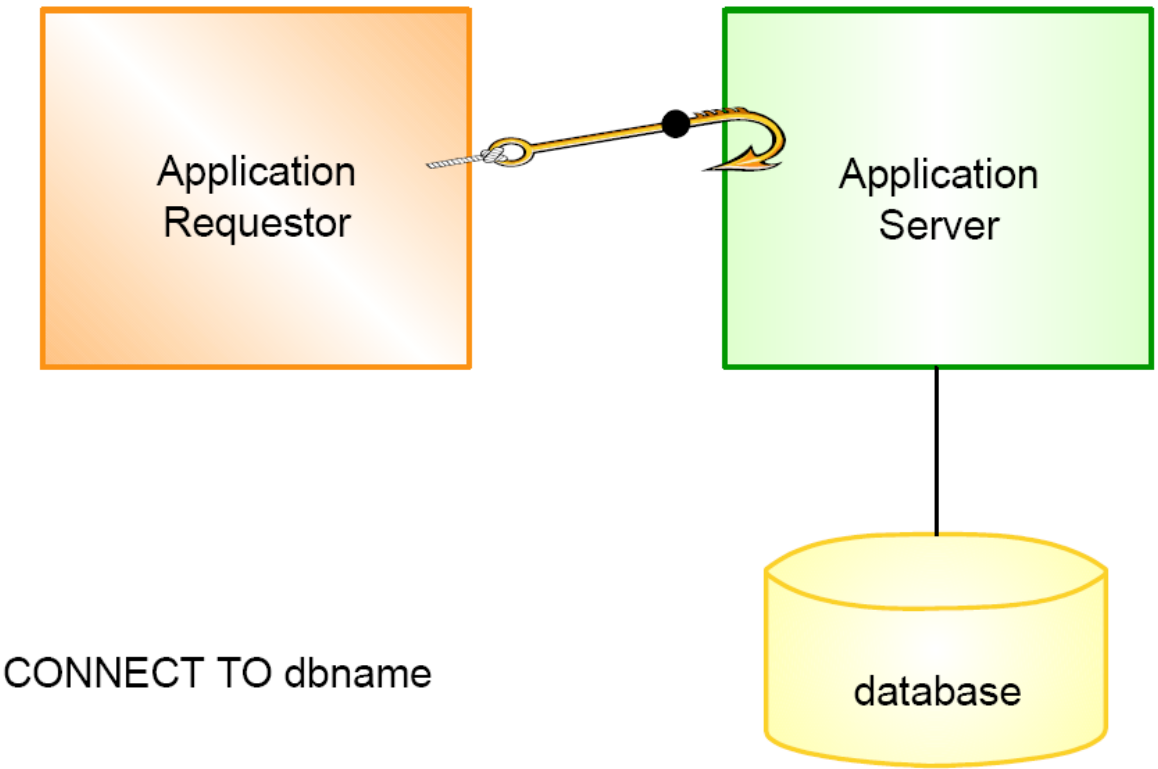


- **Create sequence:**
  - Create database
  - Create tablespace
  - Create table

DB2- Gerarchie di oggetti



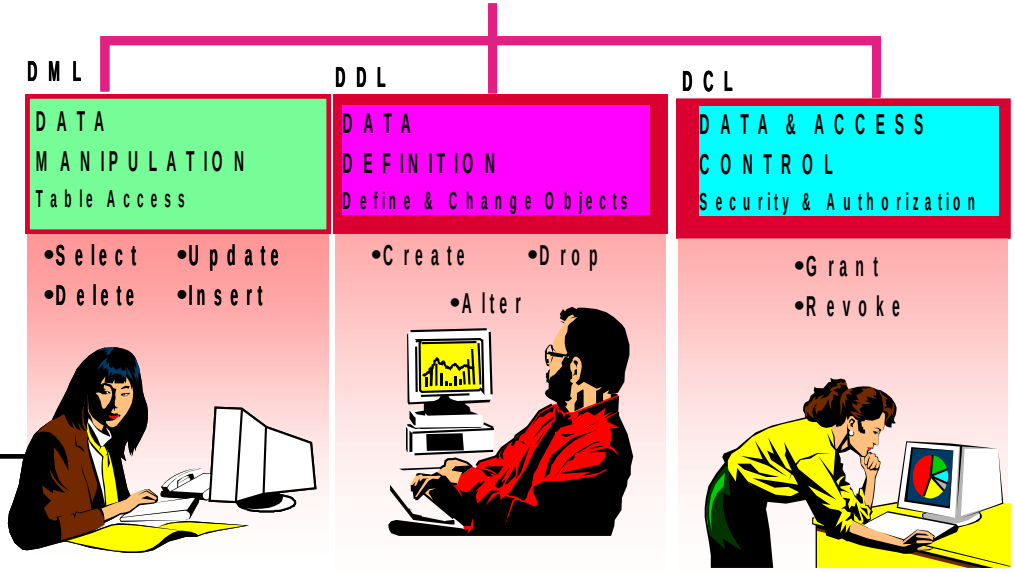
Connessione al Database



CONNECT TO dbname

Tipi di SQL Statement

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)



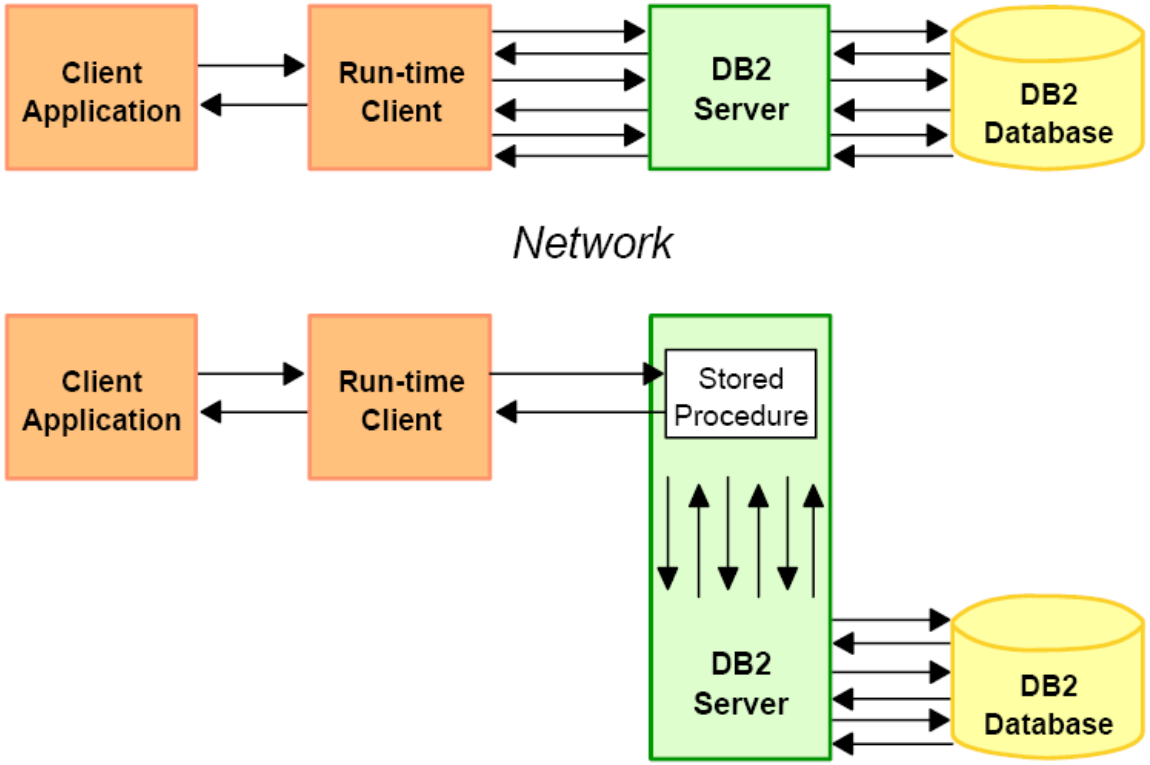

```
SELECT *
    FROM  CUSTOMER
    WHERE CUST_NBR = 239
```

CUST_NBR	CUST_NAME	CUST_REGION	CUST_CREDIT
239	TOTALLY WIRED	NE	A

```
CREATE TABLE CUSTOMER
(CUST_NBR        DECIMAL (9)        NOT NULL,
CUST_NAME        CHAR  (32)         NOT NULL,
CUST_REGION      CHAR(2)            NOT NULL,
CUST_CREDIT      CHAR(3),
CUST_INIT        TIMESTAMP      NOT NULL,   ...
```

```
GRANT ALL ON CUSTOMER
TO  NATIONAL_SALES, MARY, TED

GRANT SELECT ON  SW_CUSTOMER TO  SW_SALES, JOHN

REVOKE  SELECT ON SW_CUSTOMER FROM  NE_SALES, JANE
```

Stored Procedures

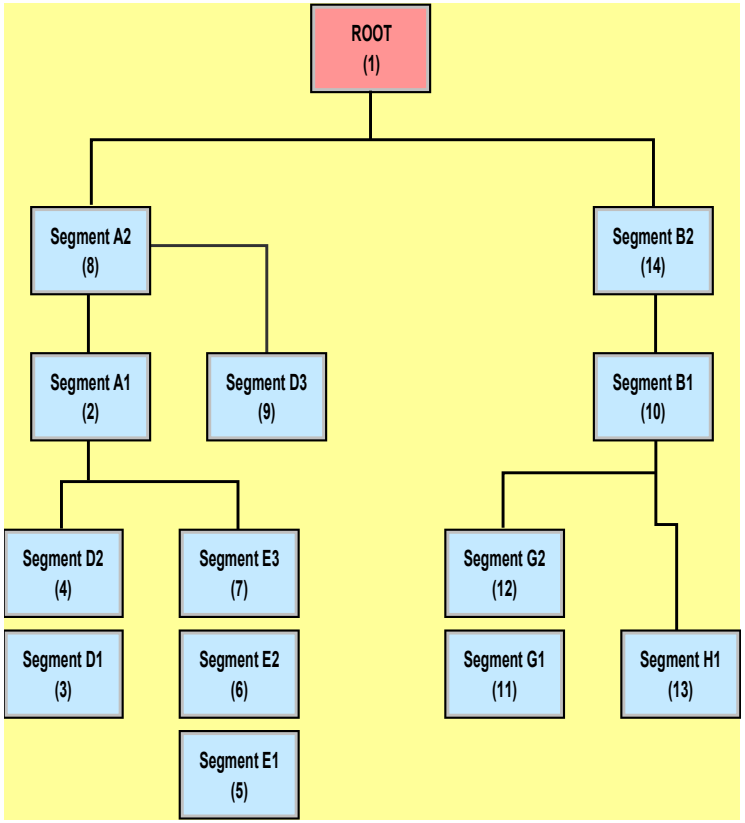


Uso del linguaggio SQL da Terminale : SPUFI

```
                          DB2I PRIMARY OPTION MENU          SSID: DB8H
COMMAND ===> 1_

Select one of the following DB2 functions and press ENTER.

 1  SPUFI                    (Process SQL statements)
 2  DCLGEN                   (Generate SQL and source language declarations)
 3  PROGRAM PREPARATION      (Prepare a DB2 application program to run)
 4  PRECOMPILE               (Invoke DB2 precompiler)
 5  BIND/REBIND/FREE         (BIND, REBIND, or FREE plans or packages)
 6  RUN                      (RUN an SQL program)
 7  DB2 COMMANDS             (Issue DB2 commands)
 8  UTILITIES                (Invoke DB2 utilities)
 D  DB2I DEFAULTS            (Set global parameters)
 X  EXIT                     (Leave DB2I)



 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Funzioni di IMS DM

- IMS usa un modello gerarchico come base di memorizzazzione di dati .
- L'IMS/DM utilizza una interfaccia applicativa
- (ovvero un query langage) chiamato DL/I
- Ie entità individuali sono implementate come segmenti in struttura gerarchica
- Il disegnatore del database determina la gerarchia in base alle relazioni tra entità e percorsi di accesso richiesti dalle applicazioni .
- Rispetto ad un DB2 un database e' equivalente ad una tabella
- DL/I Permette una ampia varietà di strutture dati.
- Un massimo di 15 livelli di struttura.



Applicazione – Definizioni

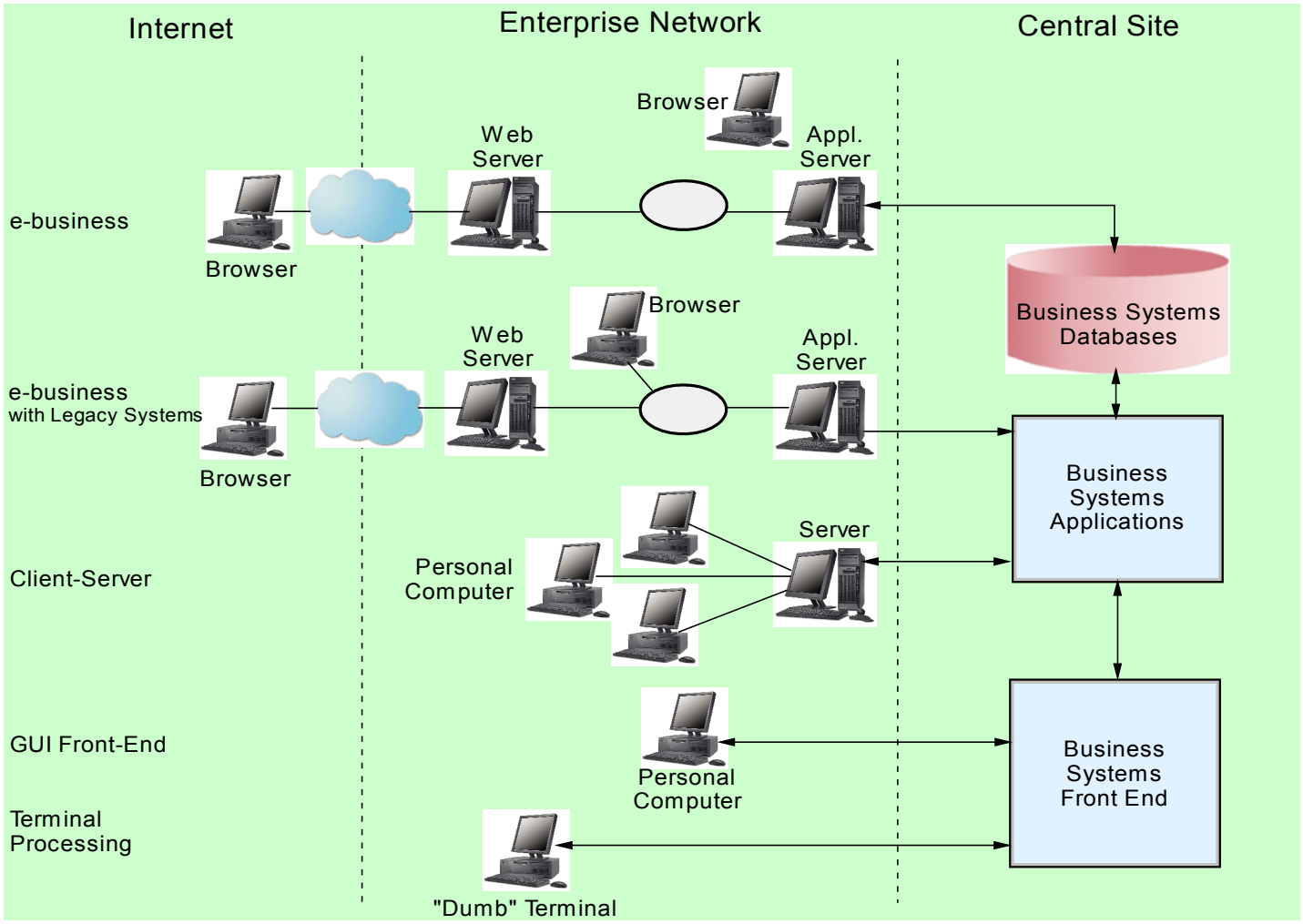
Una Applicazione :

•E’ costituita da uno o piu programmi in grado di soddisfare una specifica richiesta o di risolvere uno specifico problema.

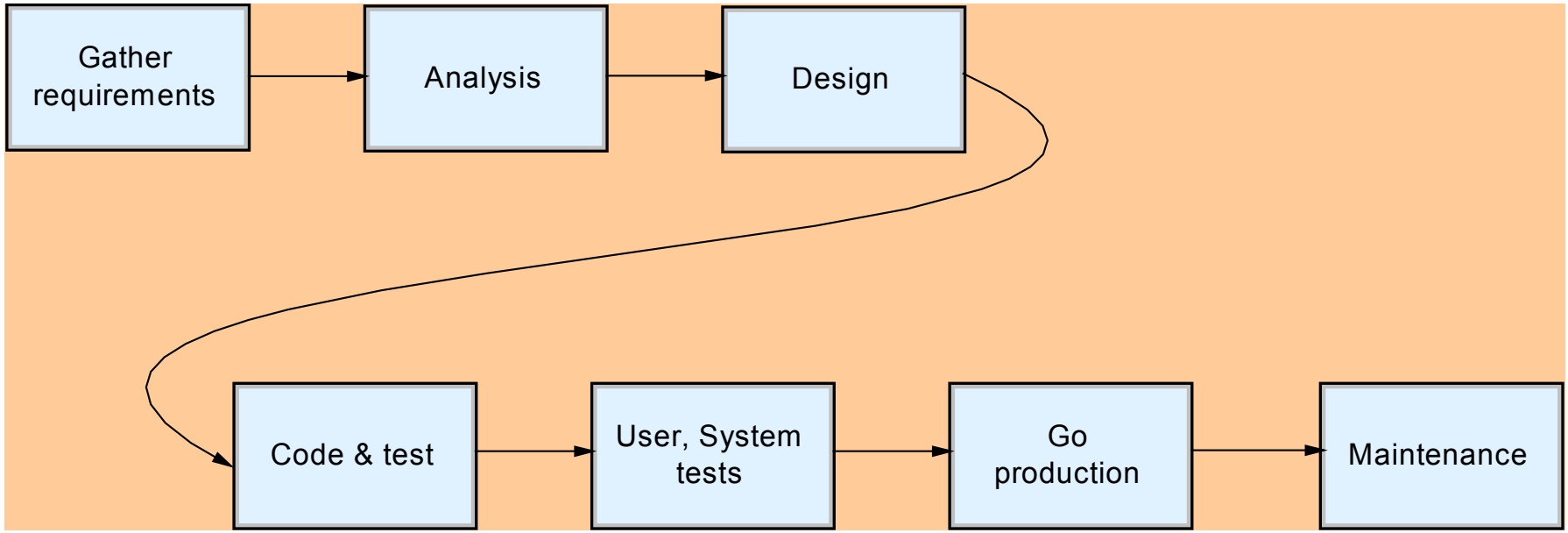
•La Soluzione puo’ risiedere su una o piu’ piattaforme informatiche

- **Application designer:**
  - Determina la migliore soluzione di programmazione.
  - Comprende :
    - Obiettivi della Applicazione
    - Ruoli organizzativi
    - Hardware e Software di Base .
  - Possiede una visione globale del progetto in cui si inserisce l’applicazione.
- **Application programmer:**
  - Costruisce , prova e rilascia l’applicazione .
  - Lavora in base alle specifiche fornite dal designer
  - Usa tutti I ‘tools’ disponibili.
  - La programmazione applicativa e’ costituita da una serie di iterazioni di
  -
    - Code changes and compiles
    - Application builds
    - Unit testing.

Una Applicazione puo' risiedere su Varie Piattaforme



Ciclo di vita di una Applicazione



Overview sui Linguaggi di Programmazione

- **UN PROGRAMMA**

  e' un insieme ordinato di istruzioni date al Calcolatore**.**

- **UN LINGUAGGIO DI PROGRAMMAZIONE**

  e' un insieme di regole che consentono di costruire un programma .

- **UN LINGUAGGIO MACCHINA**

  E' rappresentato in formato binario o direttamente riconducibile ad esso rappresenta le istruzioni cosi' come potrebbero essere date direttamente al processore (CISC).

- **I LINGUAGGI DI PROGRAMMAZIONE AD ALTO LIVELLO HLL Compiler**

  si pongono l'obiettivo di trasformare una o piu' istruzioni in linguaggio macchina utilizzando la semantica dei linguaggi Umani o una rappresentazione grafica delle azioni richieste al Calcolatore:

  Esistono molti linguaggi di programmazione.

Classificazione dei Linguaggi di Programmazione #1

- 1st generation
  - Linguaggio Macchina
  - Specifico per ogni Hardware
- 2nd generation
  - Assembler language
  - Specifico per ogni hardware
  - Istruzioni Simboliche in diretta relazione con le istruzioni di macchina
  - Necessita' di una fase precedente all'esecuzione detta Assemblaggio
- 3rd generation (HLL)
  - Procedural languages
  - Esempi : COBOL,PL/I ,Fortran,
  - Devono essere tradotti (compilati) prima dell'esecuzione
  - Generalmente portabili (con adattamenti) attraverso hardware diversi e diversi SO
- 4th generation – (4GL)
  - Non-procedural languages
  - Report generators
  - Query languages
  - Completamente Portabili
  - Examples:
    - RPG, CSP, QMF, SQL

Classificazione dei Linguaggi di Programmazione #2

- Visual programming languages **(or event-driven languages)**
  - **VisualBasic, VisualC++**
- Object-Oriented language
  - **used in OO technology, e.g. Smalltalk, Java, C++**
- Other languages
  - **3D applications**
- Scripting languages
  - **Perl**
  - **REXX**
  - **HTML**

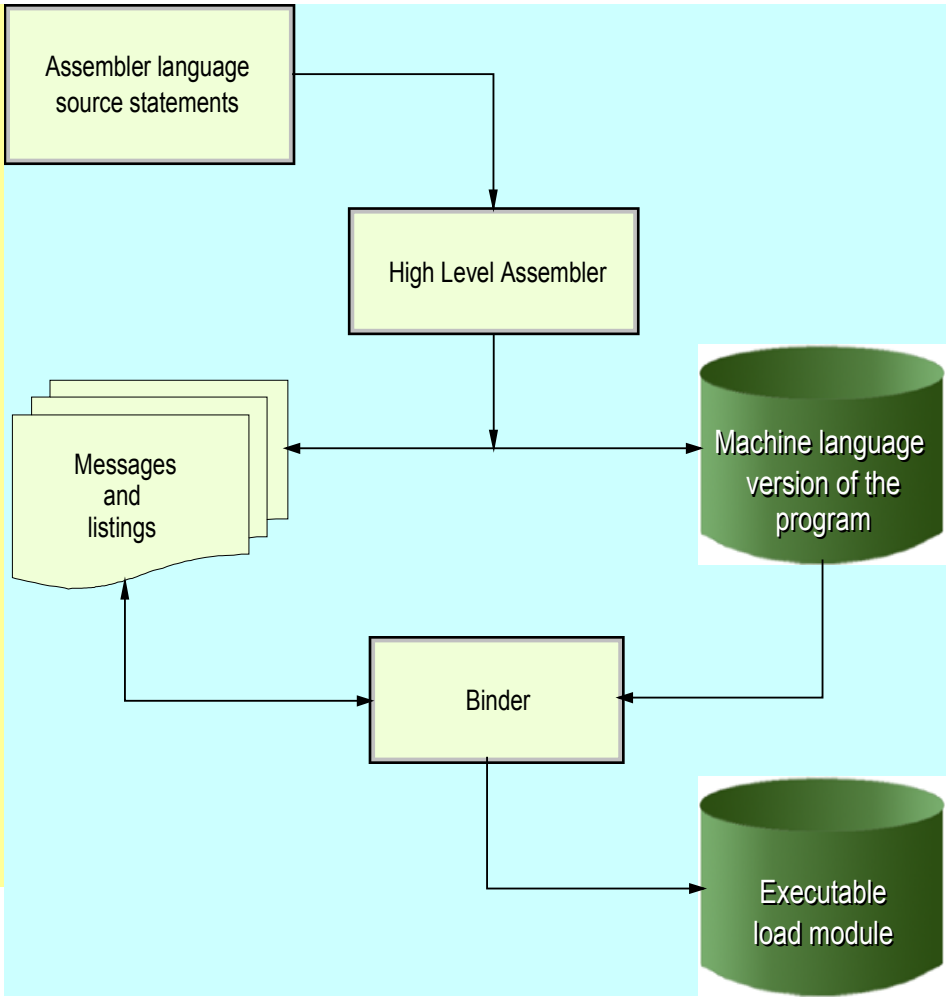
Uso del Linguaggio Assembler su z/OS

Assembler language

- Non usato spesso per lo sviluppo di applicazioni.
- Specifico per ogni Hardware

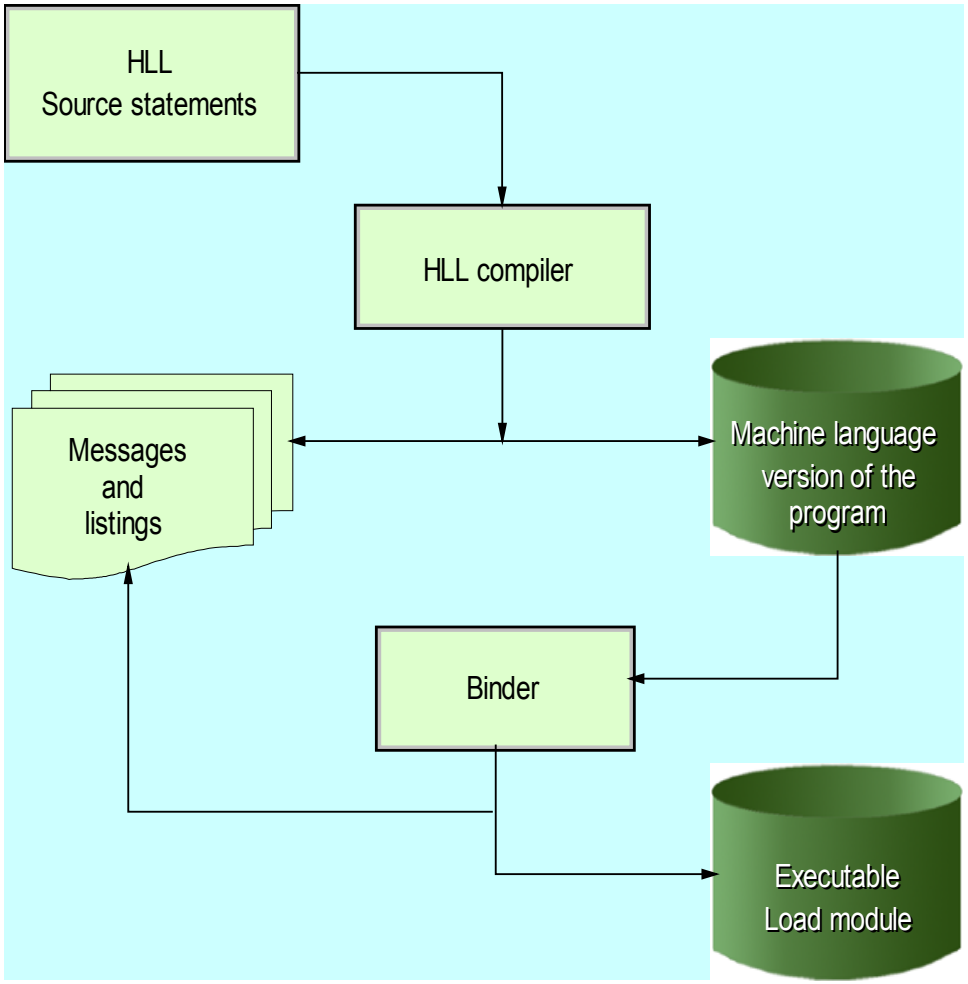
Usato quando:

- Si accedono bits o bytes
- Si debba accedere a un system control blocks
- Sia richiesta una particolare performance
- Pezzi di codice riusabile richiamati all'interno di programmi scritti in linguaggi di alto livello



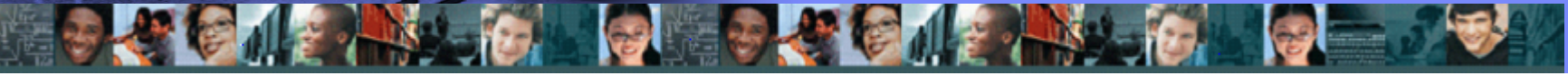
HLL : Esempio COBOL su z/OS

- **COBOL e' un linguaggio di programmazione di terza generazione integrato con nuove potenzialita'.**
- **Usato per applicazioni business-oriented**
- **Possibilita' di IBM Enterprise COBOL for z/OS and OS/390**
  - **Applicazioni COBOL Integrate nei processi Web-oriented**
  - **Interoperabilità con Java**
  - **Parsing di data in formato XML e Unicode**



Struttura di un programma Cobol

- **Identification Division** che identifica il programma con un nome e altre informazioni opzionali
- **Environment Division** dove si descrivono gli aspetti del programma che dipendono dall'ambiente elaborativo
- **Data Division** dove sono definite le caratteristiche dei dati
  - **File Section** per definire i dati nelle operazioni di input output
  - **Linkage Section** per definire i dati ottenuti da altri programmi
  - **Working Storage Section** per allocare staticamente una quantità di memoria
  - **Local Storage Section** per allocare/deallocare una quantità di memoria ogni volta che il programma e' chiamato/rilasciato
- **Procedure Division** dove sono le istruzioni per manipolare i dati e le interfacce con altre procedure



Esempio di Programma Cobol

```
  IDENTIFICATION DIVISION.
 Program-ID. Helloprog.
 Author. A. Programmer.
 Installation.  Computing Laboratories.
 Date-Written.  08/21/2002.
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. computer-name.
OBJECT-COMPUTER. computer-name.
SPECIAL-NAMES.
  special-names-entries.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT [OPTIONAL] file-name-1
        ASSIGN TO system-name [FOR MULTIPLE {REEL | UNIT}]
        [.... .
I-O-CONTROL.
    SAME [RECORD] AREA FOR file-name-1 ... file-name-n.
```

```
 IDENTIFICATION DIVISION.
. . .
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT filename ASSIGN TO assignment-name
    ORGANIZATION IS org ACCESS MODE IS access
    FILE STATUS IS file-status
. . .
DATA DIVISION.
FILE SECTION.
FD  filename
01  recordname
    nn . . . fieldlength & type
    nn . . . fieldlength & type
. . .
WORKING-STORAGE SECTION
01 file-status  PICTURE 99.
. . .
PROCEDURE DIVISION.
    . . .
    OPEN iomode filename
    . . .
    READ filename
    . . .
    WRITE recordname
    . . .
    CLOSE filename
    . . .
```