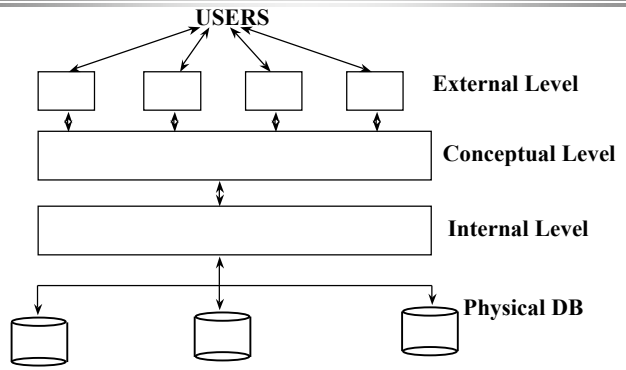
2. ARCHITECTURE OF A DBMS



Internal Level

- closest to the physical storage - concerned with the way in which data is actually stored, including selection of appropriate file organisation and indexing techniques
- *physical* view

External level

- provides users with different ways of viewing the data
- each user or group of users has their own way of viewing the data
- views are designed to meet the needs of users
- not concerned with how data is physically stored
- *logical* views

Conceptual level

- lies at the heart of the DBMS architecture
- provides a mapping between the external and internal views
- represents the community (global) view
- *logical* view

Schemas

- each level consists of one or more views of the underlying data
- views are described by *schemas* (meta-data)
- a DB consists of actual physical data, plus an internal schema, a conceptual schema and several external schemas
- schemas are stored in the system catalogue

External view (COBOL)

```
01  ACC
  02 ACCNO    PIC X(6)
  02 CLIM     PIC 9(6)
  02 BAL      PIC 9(6)
```

External View (PI/1

```
DCL   1 ACCP
        2 ACNUM     CHAR(6)
        2 NAME      CHAR(30)
        2 ADDR      CHAR(40)
        2 CITY      CHAR (2)
```

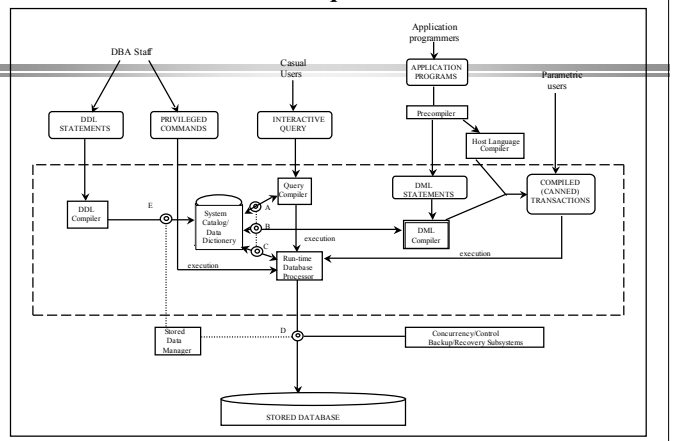
Conceptual view

```
ACCOUNT
        ACCOUNT_NUMBER          CHARACTER (6)
        CUSTOMER_NAME           CHARACTER (30)
        CUSTOMER_ADDRESS        CHARACTER (40)
        ACCOUNT_TYPE            CHARACTER (2)
        CREDIT_LIMIT            NUMERIC (6)
        ACCOUNT_BALANCE         NUMERIC (6)
```

Internal View

```
PHYSICAL_ACCOUNT   LENGTH=18
        ACCNO          TYPE=BYTE(6),OFFSET=0,INDEX=ACCX
        CUSTNM         TYPE=BYTE(30),OFFSET=6
        CUST_ADDR      TYPE=BYTE(40),OFFSET=36
        ACC_TYPE       TYPE=BYTE(2),OFFSET=76
        CREDIT_LIM     TYPE=FULLWORD,OFFSET(78)
        ACC_BAL        TYPE=FULLWORD,OFFSET(82)
```


2.1 DBMS Components



DBMS Components (contd.)

- access to disc controlled by OS
- **stored data manager (SDM)** controls access to DBMS information on disc, including buffer management
- dotted lines and points marked A, B, C, D, and E are under control of SDM
- **DDL compiler** processes schema definitions and stores them in catalogue

DBMS Components (contd.)

- **catalogue** contains details of files, data items, mapping information, constraints, etc.
- **run time DB Processor** handles DB access at run-time
- **query processor** handles high-level interactive queries

DBMS Components (contd.)

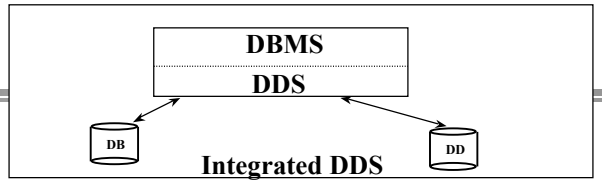
- **pre-compiler** extracts DML commands from application programs written in HLL (high level language, e.g. C, COBOL, C++), passes them to the DML compiler for compilation into object code
- object code for DML commands and the rest of the program are linked forming a **canned transcation** whose executable code calls the run-time processor
- canned transactions are used by parametric users

2.2 System Catalogue and Data Dictionary

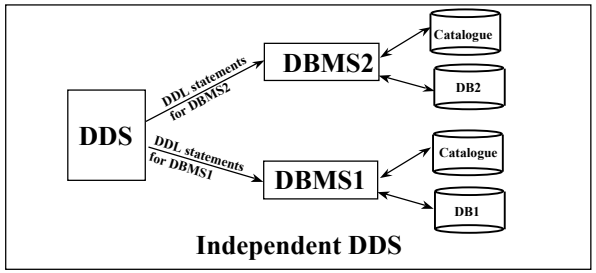
- meaning of data items must be clearly understood
- DDLs and hence system catalogues are primarily concerned with *syntactic* definition of the data
- Data Dictionary Systems (DDS) have been developed to augment the internal DBMS catalogue with *semantic* support
- DDS provide comprehensive support for metadata management

System Catalogue and Data Dictionary (contd.)

- DDSs have grown up alongside DBMSs and there are 2 main ways of coupling the two systems together:
  - integrated DDS
  - independent DDS



Integrated DDS



Independent DDS

Integrated DDS

- DDS is an integral part of DBMS and is effectively identical to the System Catalogue
- it is generally fully *active* i.e. accessed at run-time by DBMS software

Independent DDS

- DDS is an independent, free-standing system performing its own data management functions
- normally *passive* i.e. no run-time link between DDS and DBMS: hence DBMS has to have its own Catalogue
- often generate metedata automatically for variety of DBMSs in the form of DDL; helps to ensure consistency of metadata between DD and catalogue

Fully Functional DDS

Fully functional DDS should provide:

(a) Control of the data resource by providing standardised inventory: facilitates data sharing across computing environments

(b) Assistance in establishing effective communication between the designers and users

(c) control of the costs of developing and maintaining applications

Fully functional DDS (contd.)

(d) Assistance in achieving data independence by permitting applications to access data without knowledge of the location or storage characteristics of the data (through the 3-level architecture)

(e) independence of metadata across computing environments

2.3 Classification of DBMSs

- main method of classification is via the conceptual data model
- choice of model affects virtually all other components in the system, but particularly the external schemas and associated DML
- 4 main approaches

1. hierarchical 60's - 70's
2. network 70's - 80's
3. relational 80's - 90's
4. object-oriented 90's
5. Object-Relational late 90s – 2000s