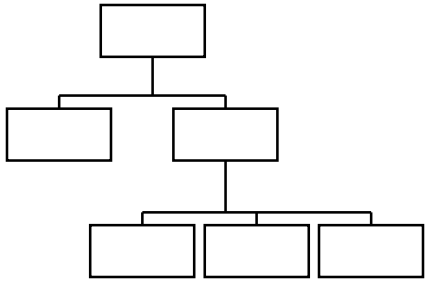
Unit 4

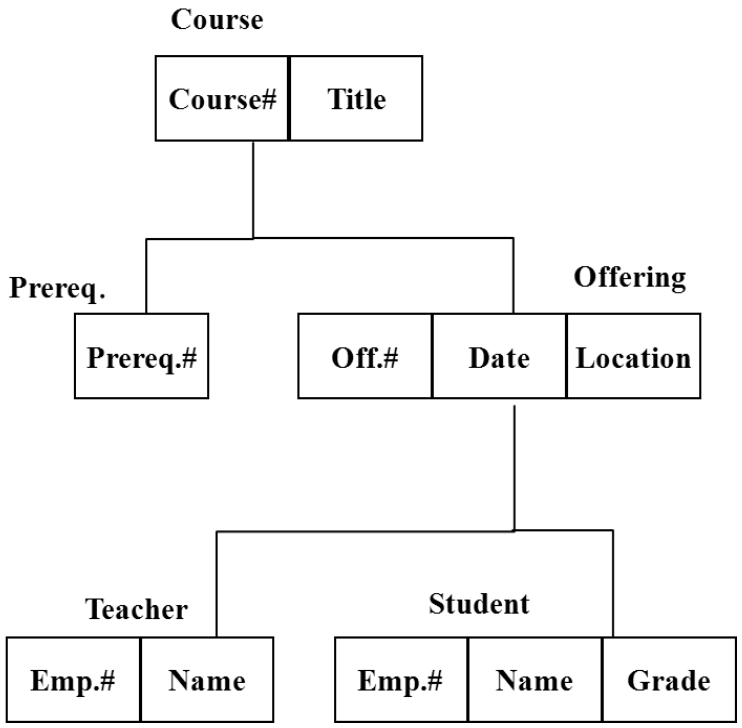
The Hierarchical Model

- 4.1 The Hierarchical Model
- 4.2 IMS



Hierarchical Database Model

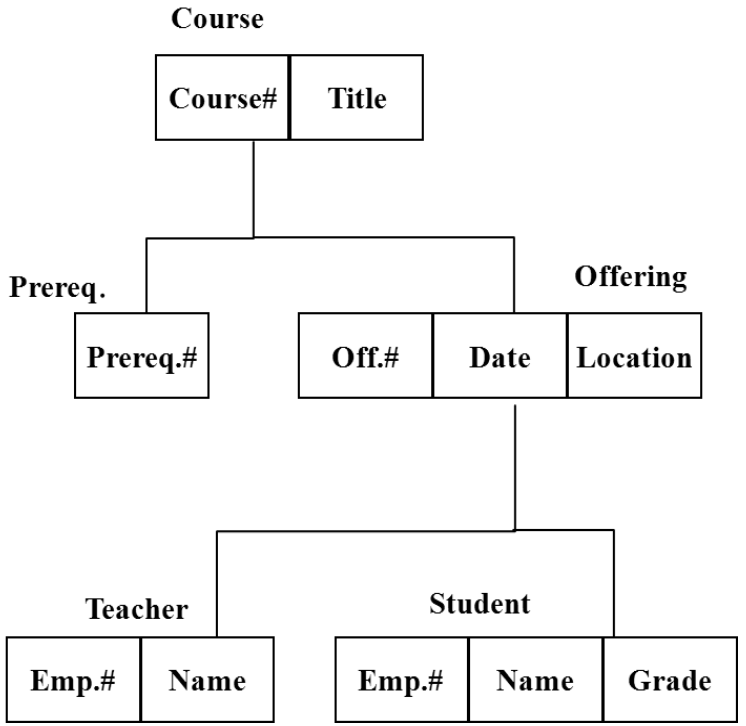
- A **hierarchical database model** is a data model in which the data is organized into a tree-like structure.
- Example of a hierarchical model



Source: From Wikipedia, the free encyclopedia

The First Database Model

- The structure allows representing information using parent/child relationships: each parent can have many children, but each child has only one parent (also known as a 1-to-many relationship).
- This model is recognized as the first database model created by IBM in the 1960s.
- Currently the most widely used hierarchical databases are IMS developed by IBM and Windows Registry by Microsoft.



Source: From Wikipedia, the free encyclopedia

4.1 The Hierarchical Model

(1) Data Structure

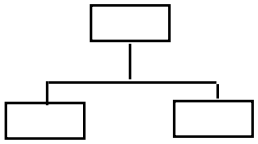
An ordered set of trees, more precisely, an ordered set consisting of multiple occurrences of a single type of tree.

(records)

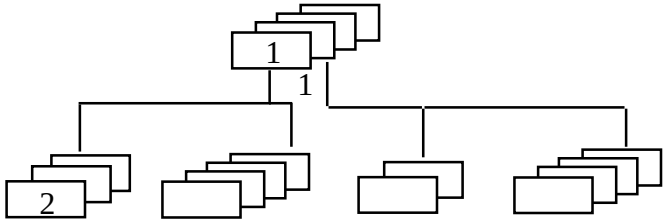
format

S

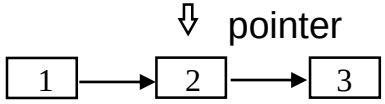
S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens



1. format

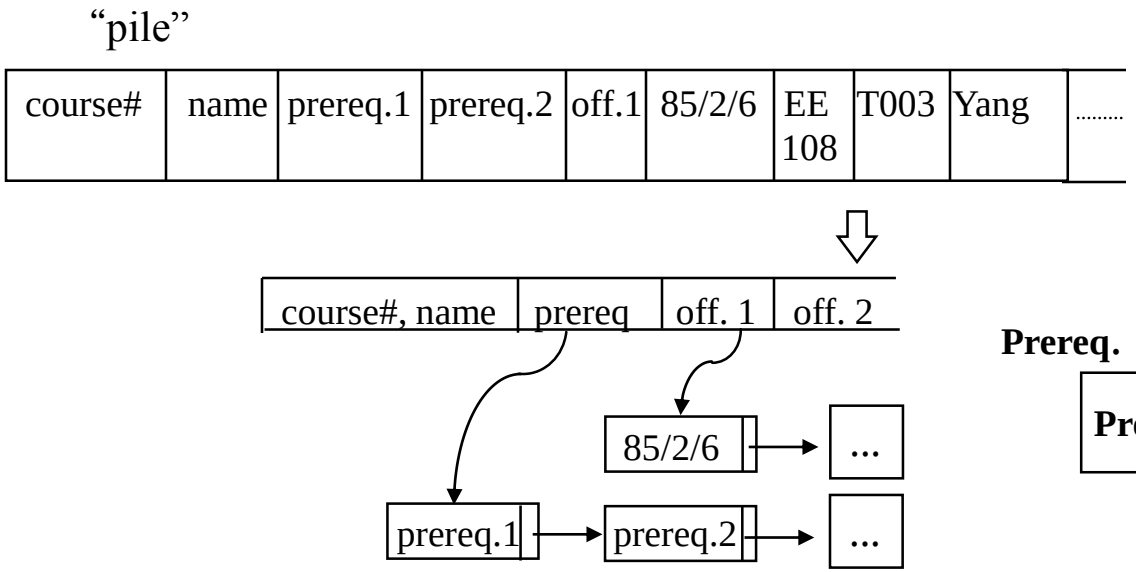


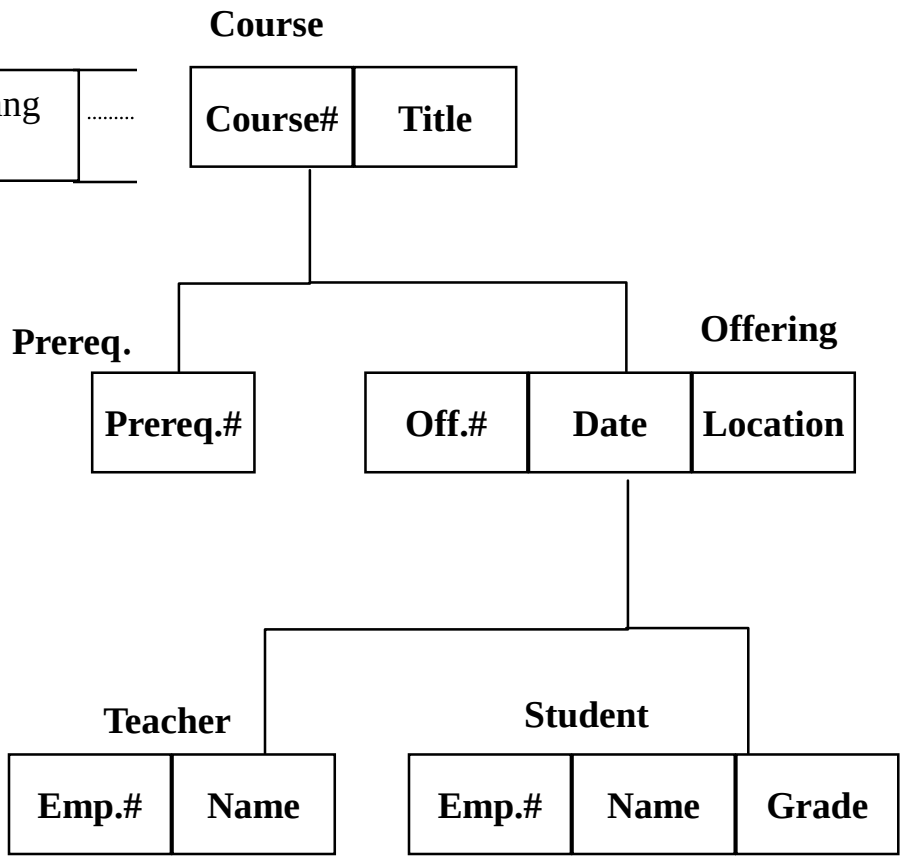
2. occurrences (records)



The Hierarchical Model: Segment

e.g. Tree type of an ~~education database~~





1. **Five record (segment) types:**
   Course, Prereq, Offering, Teacher, student.
2. **Difference with relational model:**
   (i) parent:child vs. foreign key
   (ii) operators

The Hierarchical Model: Occurrence

Occurrence

M23	DBMS	M16	M19	1	85/9/1	EC210	T003	Yang	7617503	...	2	.....

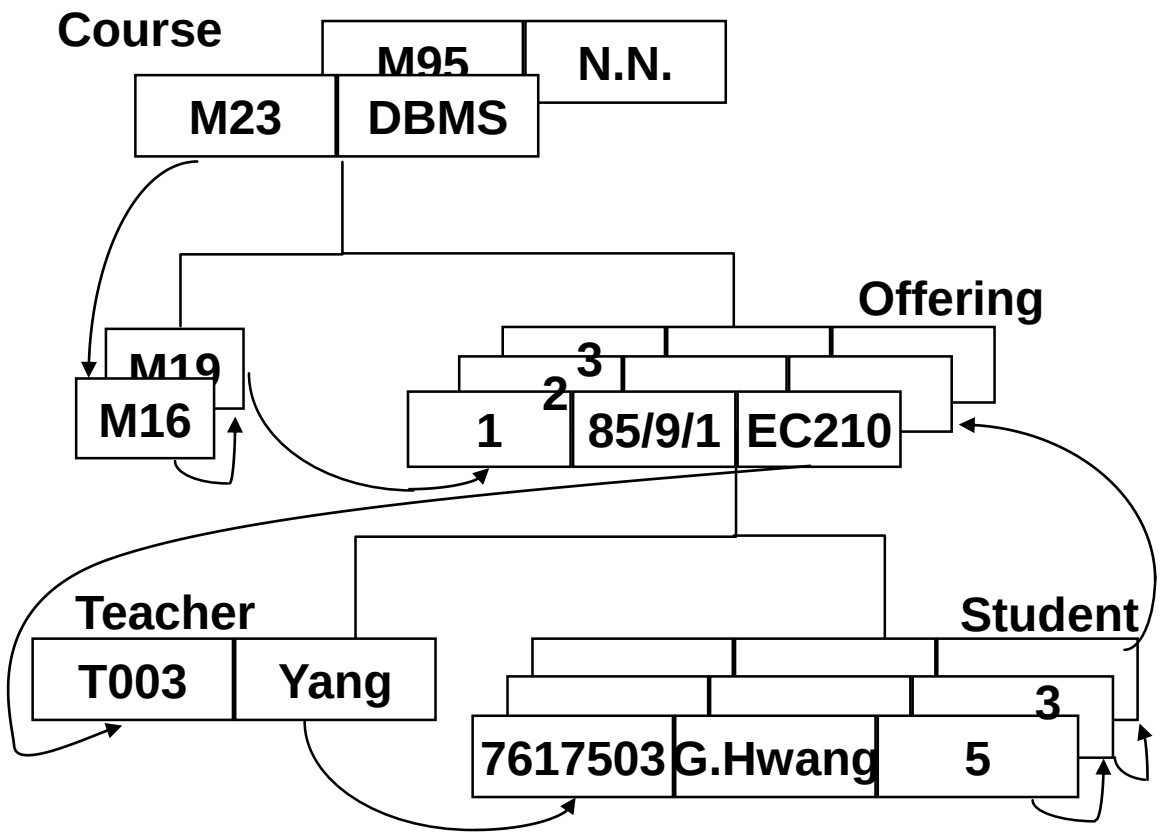
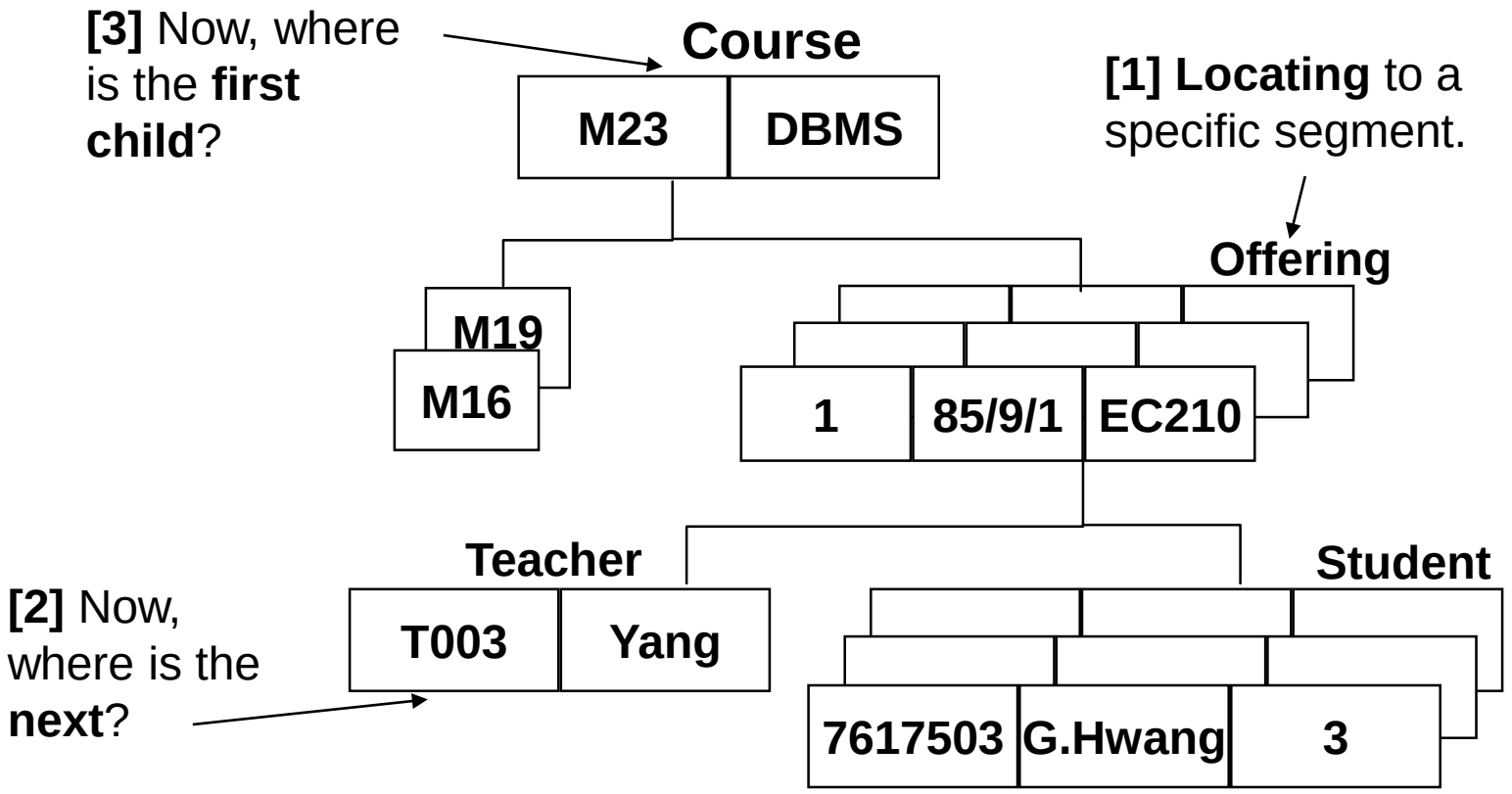


Fig. 4.2: **Hierarchical Sequence**

The Hierarchical Model: Data Manipulation

(2) Hierarchical Data Manipulation

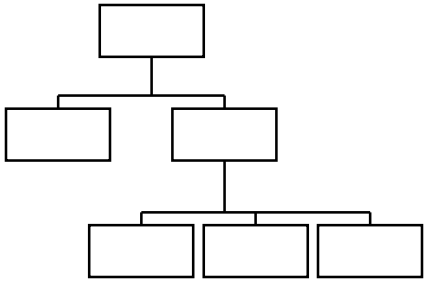
A set of operators for processing data represented in the form of trees



The Hierarchical Model: Data Integrity

(3) Hierarchical Data Integrity

Rule: no child is allowed to exist without its parent, i.e., a kind of referential integrity.



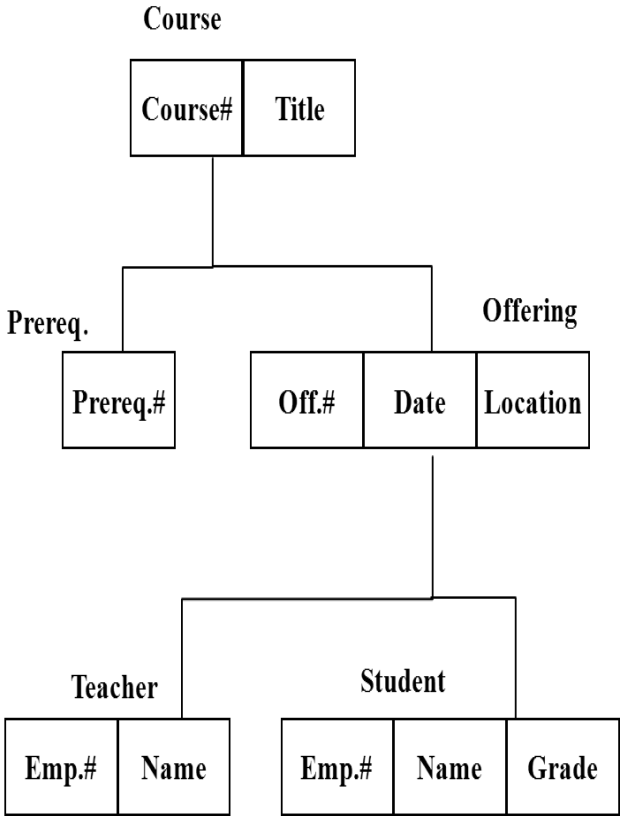
4.2 IMS

- **Overview**
  - **IMS** (**I**nformation **M**anagement **S**ystem)
  - An IBM product
  - One of the first DBMS to be commercially
  - Available in 1968

IMS: Data Definition

(1) Data Definition

e.g. Educational Database (p.4-3)



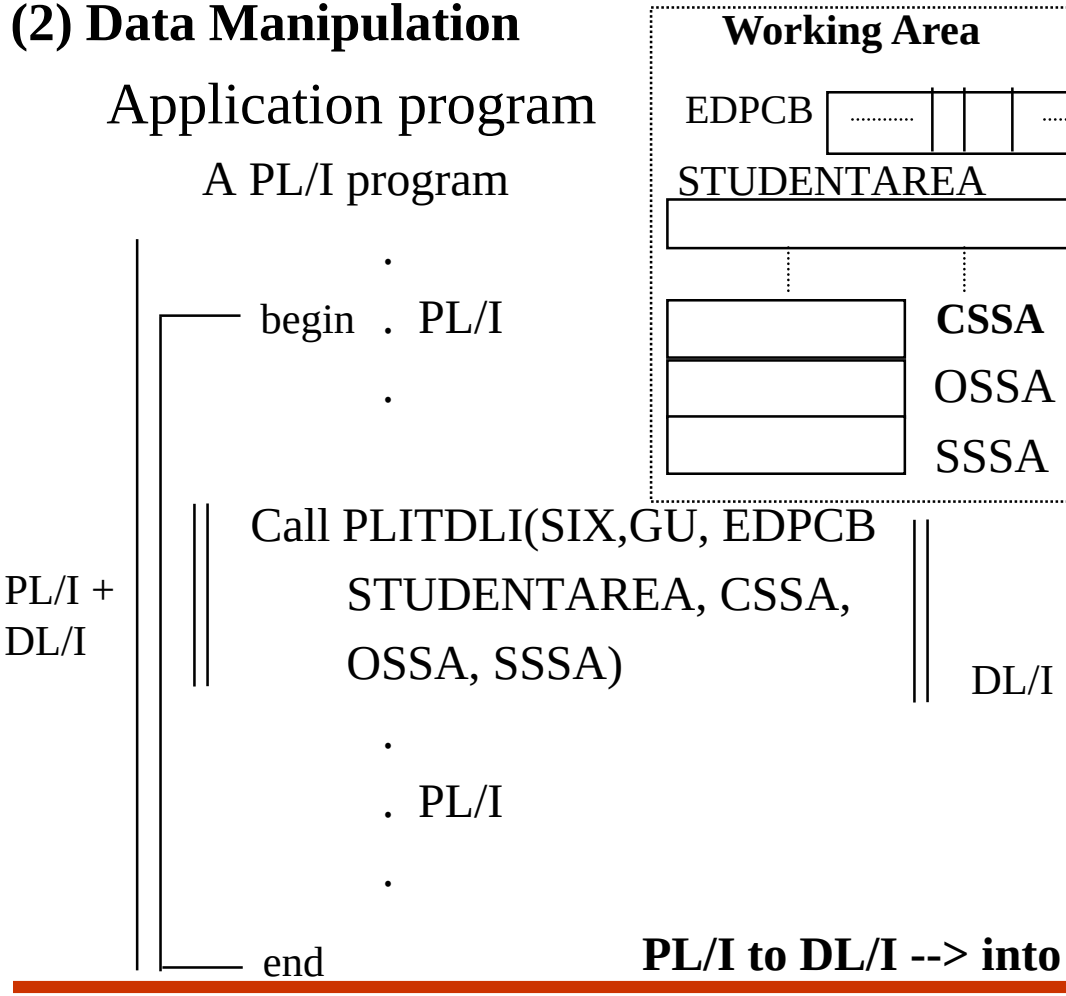

```
  1  DBD     NAME=EDUCPDBD, ACCESS=HSAM
v 2  SEGM    NAME=COURSE, BYTES=36
  3  FIELD   NAME=(COURSE#, SEQ), BYTES=3, START=1
  4  FIELD   NAME=TITLE, BYTES=33, START=4
v 5  SEGM    NAME=PREREQ, PARENT=COURSE, BYTES=3
  6  FIELD   NAME=(PREREQ#, SEQ), BYTES=3, START=1
v 7  SEGM    NAME=OFFERING, PARENT=COURSE, BYTES=21
  8  FIELD   NAME=(OFF#, SEQ), BYTES=3, START=1
  9  FIELD   NAME=DATE, BYTES=6, START=4
 10  FIELD   NAME=LOCATION, BYTES=12, START=10
v11  SEGM    NAME=TEACHER, PARENT=OFFERING, BYTES=24
 12  FIELD   NAME=(EMP#, SEQ), BYTES=6, START=1
 13  FIELD   NAME=NAME, BYTES=18, START=7
v14  SEGM    NAME=STUDENT, PARENT=OFFERING, BYTES=25
 15  FIELD   NAME=(EMP#, SEQ), BYTES=6, START=1
 16  FIELD   NAME=NAME, BYTES=18, START=7
 17  FIELD   NAME=CRADE, BYTES=1, START=25
```

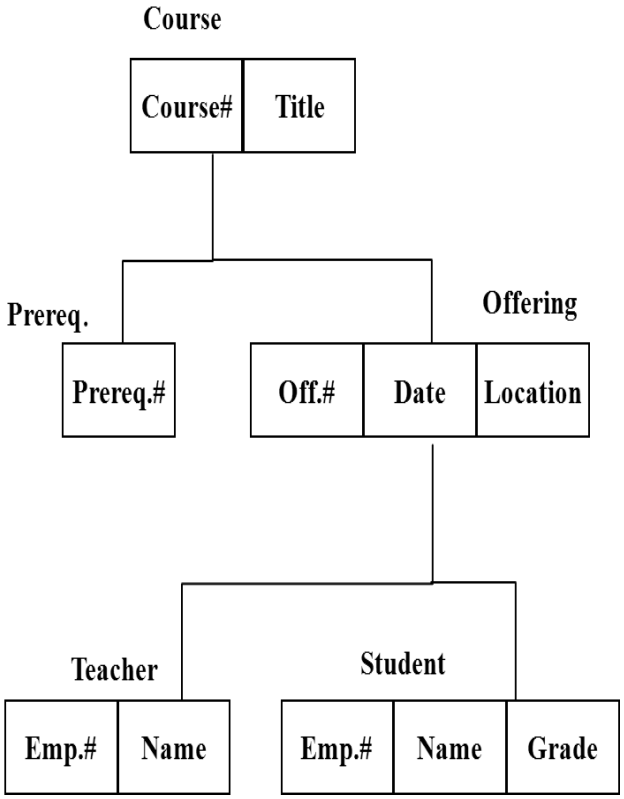
IMS: Data Manipulation

(2) Data Manipulation

Application program

A PL/I program





PL/I to DL/I --> into IMS

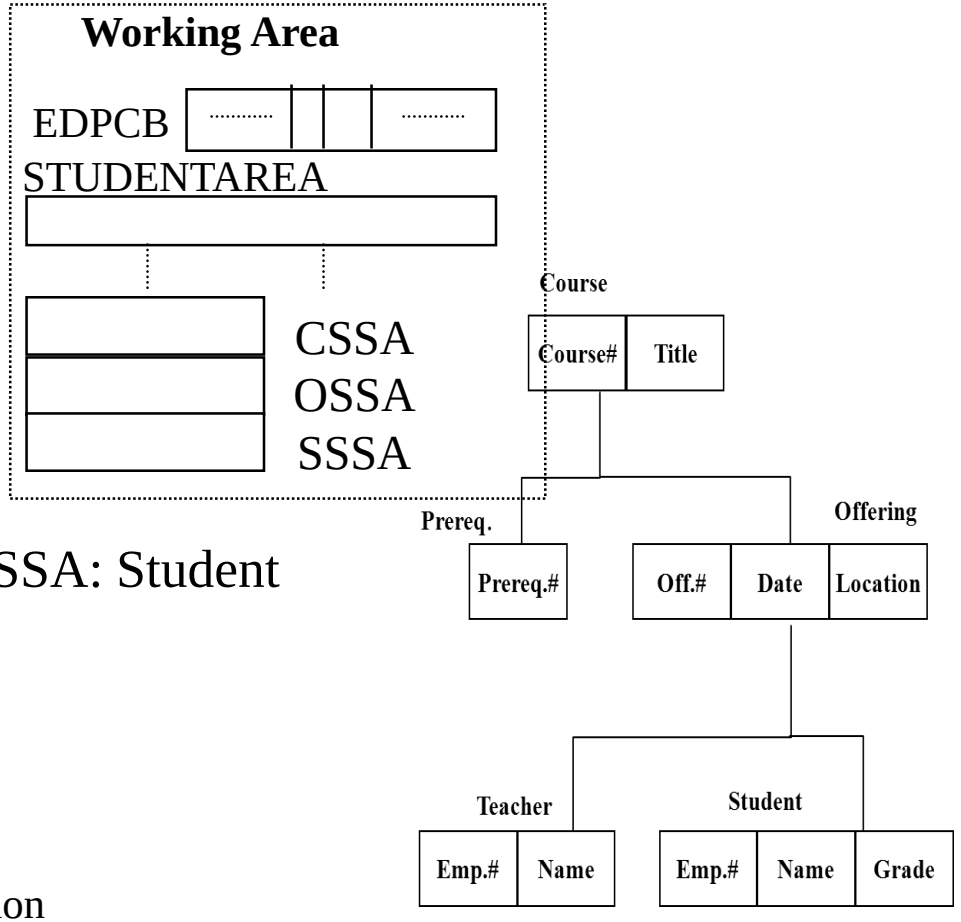
Call PLITDLI(SIX,GU, EDPCB, STUDENTAREA, CSSA, OSSA, SSSA)

- SIX: six arguments will be used
- GU: get unique, and operator
- EDPCB (Communication Block):
  - a cursor and a feedback area (SQLCA in DB2) (ref. p.2-41)
- STUDENTAREA: I/O area, a buffer
- SSA: Segment Search Arguments
- CSSA: Course, OSSA: Offering, and SSSA: Student

e.g.

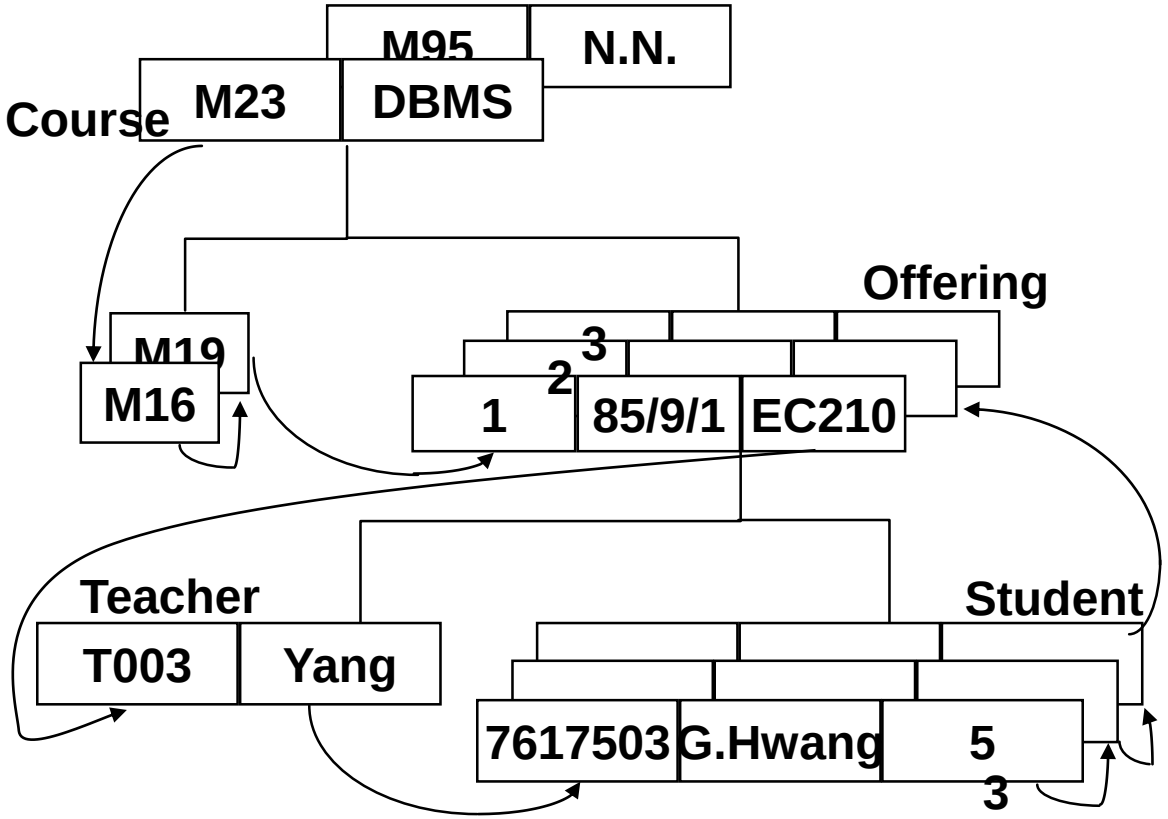
- CSSA: Course where TITLE='DBMS'
- OSSA: Offering where DATE='85/9/1'
- SSSA: Student where GRADE='3'

condition



The Hierarchical Model: Occurrence

Occurrence



e.g.

CSSA: Course where TITLE='DBMS'
OSSA: Offering where DATE='85/9/1'
SSSA: Student where GRADE=‘3'

IMS: Data Manipulation operation

- **IMS operations:**
  - GET Unique (**GU**): Direct Retrieval
  - GET Next (**GN**): Sequential Retrieval
  - GET Next within Parent (**GNP**):
    Sequential Retrieval under current parent
  - GET Hold (**GHU, GHN, GHNP**):
    as above but 'LOCK'
  - Insert (**ISRT**): Add new segment
  - Delete (**DLET**): Delete existing segment
  - Replace (**REPL**): Replace existing segment