

# Supplementary Notes for CA218

## Part 1

Dr. Martin Crane (x8974)

[www.computing.dcu.ie/~mcrane/CA218.html](http://www.computing.dcu.ie/~mcrane/CA218.html)

# Ch 1,2: Basic Definitions

- Database
  - An object or mechanism used to store info or data
  - Users should be able to store data in organised manner
  - Examples: phone book, mobile contacts (web isnt a DB!)
- Data:
  - Collection of one or more bits of information
- DBMS (DataBase Management System)
  - Together with Database forms **Database System** that provides logic to ensure & reinforce necessary standards on the data
  - => is a set of rules that are part of the DB software & dictates logically how the data is stored, treated & accessed
- Schema of a DB system:
  - structure described in formal language supported by the DBMS

# Ch 1,2: Basic Definitions (cont'd)

- DB Catalog:
  - Complete description of DB structure & Constraints
  - Contains data on structure of each file, type/storage format of data item & constraints on data
  - Info stored in catalog = **Metadata**
- DB View:
  - Subset of DB or virtual data derived from DB but not stored
- Program-Data Independence:
  - Changes in file structure don't require changes in all programs accessing it

# Ch 1,2 : DBMS Users: Actors

- DBAs are responsible for
  - DB itself AND DBMS & related
  - Also resp for:
    - Authorising access to DB
    - Co-ordination & Monitoring its use
    - Updates
    - Breach of security and slow response time
- DB Designers are responsible for
  - Identifying data to be stored in DB & choosing appropriate structures to represent & store data
  - Talk to prospective users, understand their requirements & design accordingly

# Ch 1,2: DBMS Users: Actors (cont'd)

- End Users
  - Casual/Occasional
    - Only occ'y use DB but maybe need different info each time
    - Usually Middle/High-Level managers, using High-Level Language
  - Naïve/Canned Transactions
    - Constant querying of DB
    - Typically Bank Tellers, Reservation Staff for airlines etc
  - Sophisticated
    - Need to thoroughly familiarise themselves with DBMS facilities
    - Typically engineers, scientists with complex requirements
  - Stand-alone
    - Maintain personal DBs using COTS s/w with menus or GUIs
    - Example is user of tax package for their own personal purposes.

# Ch1,2:DBMS Users: Workers Behind Scene

- Typically associated with design, devpt, operation of DBMS s/w & system environment
- DBMS Designers & Developers
  - Design/implement DBMS modules/interfaces as a s/w package
  - Modules to implement catalog, query language, interface processors, data access & security
  - Must interface with other sys s/w (eg OS, compilers)
- Tool Developers
  - Optional S/w packages to facilitate DB system design & use
  - For DB design, performance monitoring, GUIs, simulation
- Operators & Maintenance Personnel
  - Responsible for actual running/maintenance of h/w & s/w environment for the DB system

# Ch 1,2: More Basic Definitions....

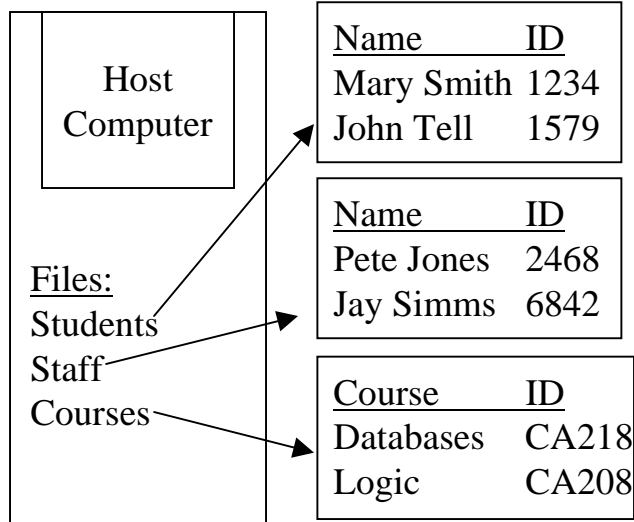
- **View**
  - A user's perspective of the DB that may be a subset of DB or contain virtual data derived from DB files but not stored
  - Example students, library, finance have different views of DB
- **Entity:**
  - A real world object or concept (eg employee or project)
- **Attribute:**
  - A property of interest, further describing an entity (eg name , project)
- **Relationship:**
  - R'ship between 2 or more entities represents an interaction between them
  - Example works-on between employee and project

# Ch 2: Data Models

- Model
  - Used to hide superfluous details while highlighting details relevant to the applications at hand
- => **Data Model** is mechanism to do this for DB applications
  - i.e. *entities* of interest & their *relationships* in the DB
  - Allows conceptualization of association of entities & their attributes
  - Differ in ways of representing associations among entities & attributes
  - Main ones we look at: Hierarchical, Network (& ER) & Relational models
  - ER representation can be mapped into Relational DB
  - All this will become clearer later when we look deeper at ER, Relational

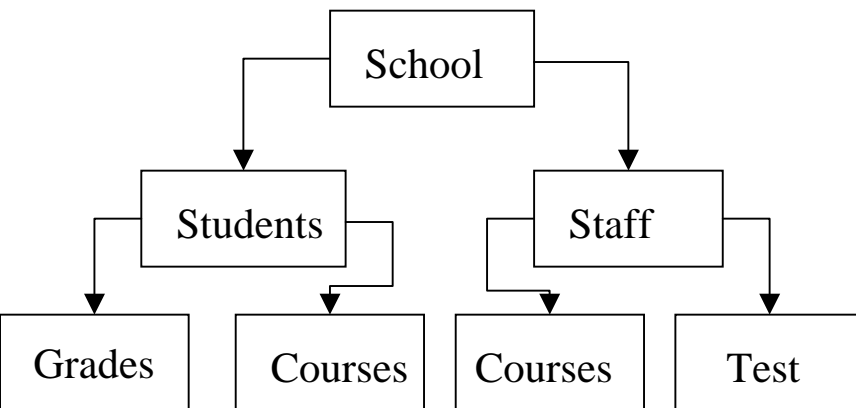
# Ch2: Various Data Models...

## Flat File Example



- *Flat File* (1950s)
  - Single, sequential 2D array of data
  - + Ok for storing data (small lists, etc)
  - - Search: v. inefficient (look at all entries)
  - - No concurrency, probs: crashes, redundancies
- *Hierarchical* (late 1960s)

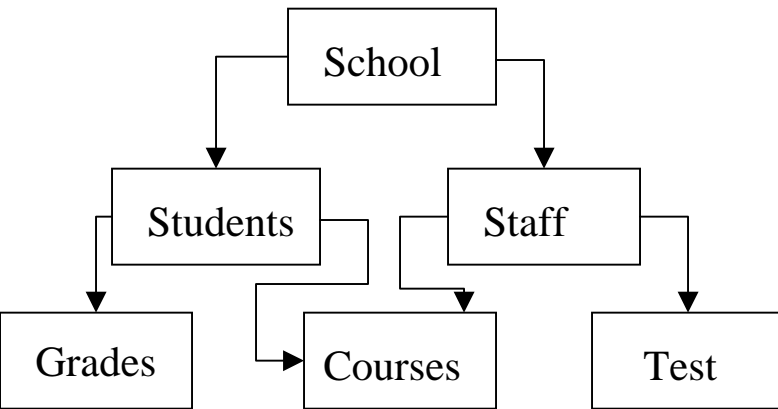
## Hierarchical DB Example



- Parent-child r'ship
- Only 1X1:N r'ship btw. 2 types of data per DB
- Works ok for, e.g. Table of Contents type data
- +
- - Probs with deletion of data e.g. parents
- - duplication of data (eg Courses stored twice)
- - changes (eg to Courses) made multiple times
- - navigation has to go from root down
- - deletion (eg a parent record) causes probs

# Ch2: Various Data Models (cont'd)

## Network DB Example



- *Network* (early 1970s)

- Different to Hierarchical
  - Support N:M r'ships
  - Navigation can start anywhere
- But needs familiarity with structure for changing, navigation & optimisation the DB

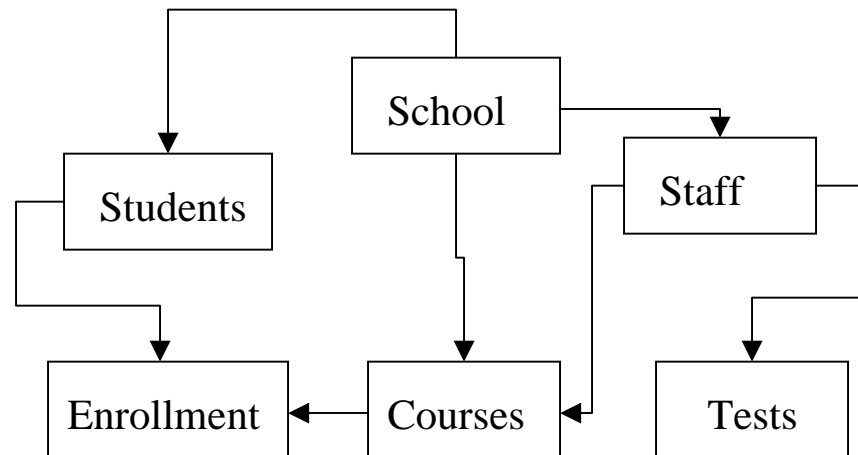
*Network+Hierarchical* evolved into *ER*

- *Relational* (EF Codd, IBM 1970)

- Different to Hierarchical, Network DBs

- Aims to “protect” users from DB structure
- Data Indep means data location unimportant
- Data Storage in *Relations/Tables* (with columns & rows) for data independence
- => data & structural indep (unlike H&N)
- Rows in table id'ed thro *keys* (eg ISBN, PPSN)
- + Tabular view means conceptual simplicity, => easier to design, implement, manage & use.
- + Ad hoc query capability based on SQL
- - Ease of use, mgmt => increased h/w & system s/w overhead

## Relational DB Example



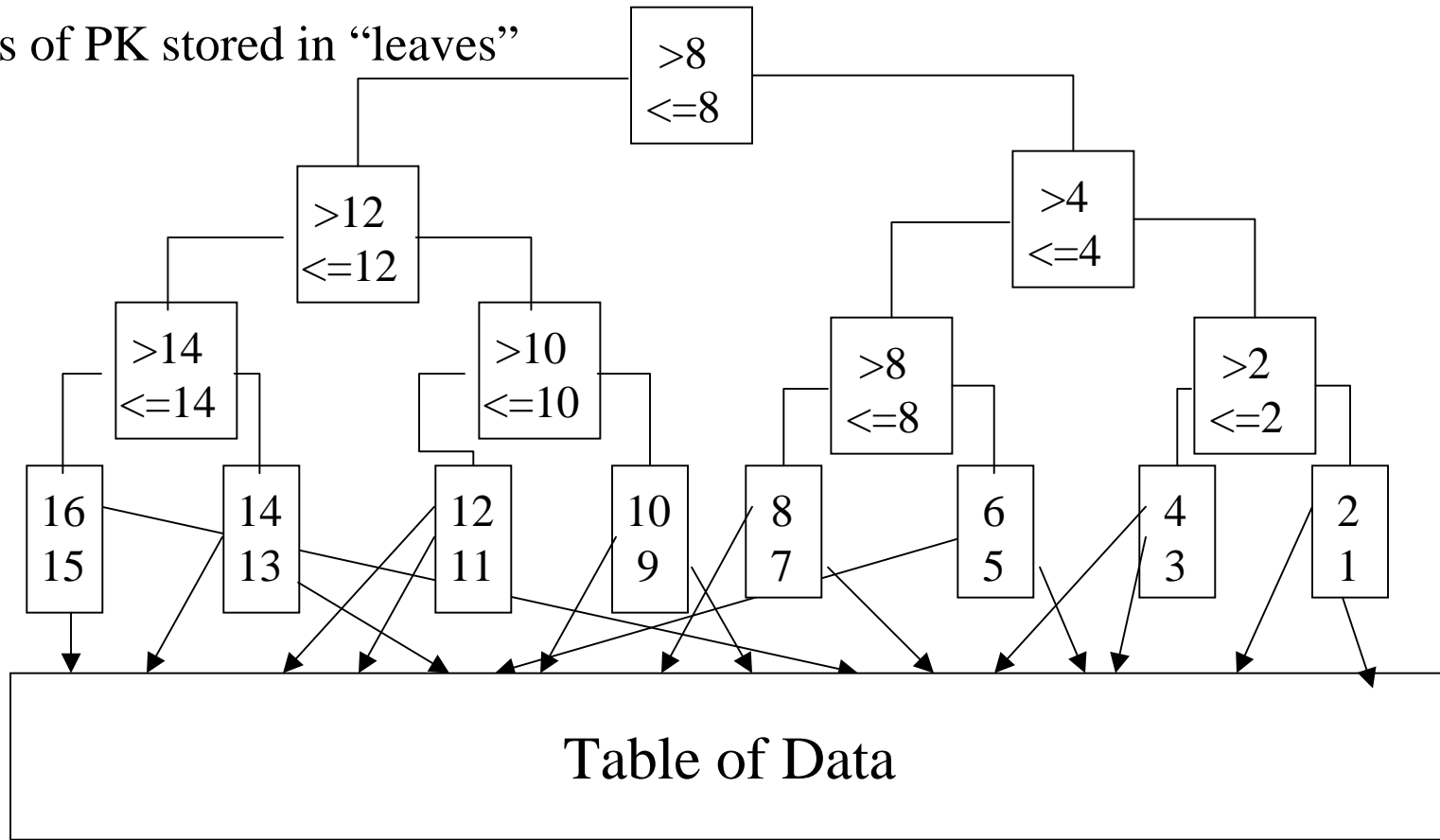
# Ch 3: Definitions of Fields, Indexes....

- Ordering Field
  - Physically order record of files on disk based on one of their fields => ordered or sequential file
  - If *ordering field* is also a *key field* (uniquely i.d.s a row in table) then also called the *ordering key* or *Primary Key* of the file
- Primary Index:
  - Ordered file with fixed length records of two fields:
    1. Same Datatype as ordering key field of data file
    2. Pointer to disk block (i.e. a block address)
- Secondary Index:
  - As with Primary: ordered file, two fields:
    1. Indexing Field: Same Datatype as non-ordering field of data file
    2. Block Pointer *or* Record Pointer

# Ch 3: Definitions of Fields, Indexes....(cont'd)

- Clustering Index:
  - Determines physical ordering of rows based on non-key field without a distinct value for each record (e.g. Phone Book arranges by last names)
  - Only one per table but each can comprise multi-columns (e.g. Phone Book organized by last name & first name)
- Binary Tree Index

- Values of PK stored in “leaves”



# Ch 3: Different File Organisations +/-

	Unsorted	Sorted	Hash
Search	- (linear search)	0 (binary search)	+ (apply Hash function)
Insert	+ (insert at end)	- (reorganise file)	0 (collisions might occur)
Delete	- (find it first)	- (reorganise file)	+ (easy to find, no need to reorganise)
Modify	0 (difficult to find, easy to modify)	- (reorganise if sorting field is affected)	- (reorganise if hash field is affected)