

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

Lecture Handout

Database Management System

Lecture No. 1

Reading Material

“Database Systems Principles, Design and Implementation” written by Catherine Ricardo, Maxwell Macmillan.	Chapter 1.
--	------------

Summary

- Introduction to the course
- Database definitions
- Importance of databases
- Introduction to File Processing Systems
- Advantages of the Database Approach

Introduction to the course

This course is first (fundamental) course on database management systems. The course discusses different topics of the databases. We will be covering both the theoretical and practical aspects of databases. As a student to have a better understanding of the subject, it is very necessary that you concentrate on the concepts discussed in the course.

Areas to be covered in this Course:

- **Database design and application development:** How do we represent a real-world system in the form of a database? This is one major topic covered in this course. It comprises of different stages, we will discuss all these stages one by one.
- **Concurrency and robustness:** How does a DBMS allow many users to access data concurrently, and how does it protect against failures?

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

- **Efficiency and Scalability:** How does the database cope with large amounts of data?
- **Study of tools to manipulate databases:** In order to practically implement, that is, to perform different operations on databases some tools are required. The operations on databases include right from creating them to add, remove and modify data in the database and to access by different ways. The tools that we will be studying are a manipulation language (SQL) and a DBMS (SQL Server).

Database definitions:

Definitions are important, especially in technical subjects because definition describes very comprehensively the purpose and the core idea behind the thing. Databases have been defined differently in literature. We are discussing different definitions here, if we concentrate on these definitions, we find that they support each other and as a result of the understanding of these definitions, we establish a better understanding of use, working and to some extent the components of a database.

Def 1: A shared collection of logically related data, designed to meet the information needs of multiple users in an organization. The term database is often erroneously referred to as a synonym for a “database management system (DBMS)”. They are not equivalent and it will be explained in the next section.

Def 2: A collection of data: part numbers, product codes, customer information, etc. It usually refers to data organized and stored on a computer that can be searched and retrieved by a computer program.

Def 3: A data structure that stores metadata, i.e. data about data. More generally we can say an organized collection of information.

Def 4: A collection of information organized and presented to serve a specific purpose. (A telephone book is a common database.) A computerized database is an updated, organized file of machine readable information that is rapidly searched and retrieved by computer.

Def 5: An organized collection of information in computerized format.

Def 6: A collection of related information about a subject organized in a useful manner that provides a base or foundation for procedures such as retrieving information, drawing conclusions, and making decisions.

Def 7: A Computerized representation of any organizations flow of information and storage of data.

Each of the above given definition is correct, and describe database from slightly variant perspectives. From exam point of view, anyone will do. However, within this course, we will be referring first of the above definitions more frequently, and concepts discussed in the definition like, logically related data, shared collection

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

should be clear. Another important thing that you should be very clear about is the difference between database and the database management system (DBMS). See, the database is the collection of data about anything, could be anything. Like cricket teams, students, busses, movies, personalities, stars, seas, buildings, furniture, lab equipment, hobbies, hotels, pets, countries, and many more anything about which you want to store data. What we mean by data; simply the facts or figures. Following table shows the things and the data that we may want to store about them:

Thing	Data (Facts or figures)
Cricket Player	Country, name, date of birth, specialty, matches played, runs etc.
Scholars	Name, date of birth, age, country, field, books published etc.
Movies	Name, director, language (Punjabi is default in case of Pakistan) etc.
Food	Name, ingredients, taste, preferred time, origin, etc.
Vehicle	Registration number, make, owner, type, price, etc.

There could be infinite examples, and please note that the data that is listed about different things in the above table is not the only data that can be defined or stored about these things. As has been explained in the definition one above, there could be so many facts about each thing that we are storing data about; what exactly we will store depends on the perspective of the person or organization who wants to store the data. For example, if you consider food, data required to be stored about the food from the perspective of a cook is different from that of a person eating it. Think of a food, like, Karhahi Ghost, the facts about Karhahi ghosht that a cook will like to store may be, quantity of salt, green and red chilies, garlic, water, time required to cook and like that. Where as the customer is interested in chicken or meat, then black or red chilies, then weight, then price and like that. Well, definitely there are some things common but some are different as well. The thing is that the perspective or point of view creates the difference in what we store; however, the main thing is that the database stores the data.

The database management system (DBMS), on the other hand is the software or tool that is used to manage the database and its users. A DBMS consist of different components or subsystem that we will study about later. Each subsystem or component of the DBMS performs different function(s), so a DBMS is collection of different programs but they all work jointly to manage the data stored in the database and its users. In many books and may be in this course sometimes database and database management system are used interchangeably but there is a clear difference and we should be clear about them. Sometimes another term is used, that is, the database system, again, this term has been used differently by different people, however in this course we use the term database system as a combination of database and the database management system. So database is collection of data, DBMS is tool to manage this data, and both jointly are called database system.

Importance of the Databases

Databases are important; why? Traditionally computer applications are divided into commercial and scientific (or engineering) ones. Scientific applications involve more computations, that is, different type of calculations that vary from simple to very

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

complex. Today such applications exist, like in the fields of space, nuclear, medicine that take hours or days of computations on even computers of the modern age. On the other hand, the applications that are termed as commercial or business applications do not involve much computations, rather minor computation but mainly they perform the input/output operations. That is, these applications mainly store the data in the computer storage, then access and present it to the users in different formats (also termed as data processing) for example, banks, shopping, production, utilities billing, customer services and many others. As is clear from the example systems mentioned, the commercial applications exist in the day to day life and are related directly with the lives of common people. In order to manage the commercial applications more efficiently databases are the ultimate choice because efficient management of data is the sole objective of the databases. So such applications are being managed by databases even in a developing country like Pakistan, yet to talk about the developed countries. This way databases are related directly or indirectly almost every person in society.

Databases are not only being used in the commercial applications rather today many of the scientific/engineering application are also using databases less or more. databases are concern of the effectively latter form of applications are more Commercial applications involve The goal of this course is to present an in-depth introduction to databases, with an emphasis on how to organize information in the database and to maintain it and retrieve it efficiently, that is, how to design a database and use it effectively.

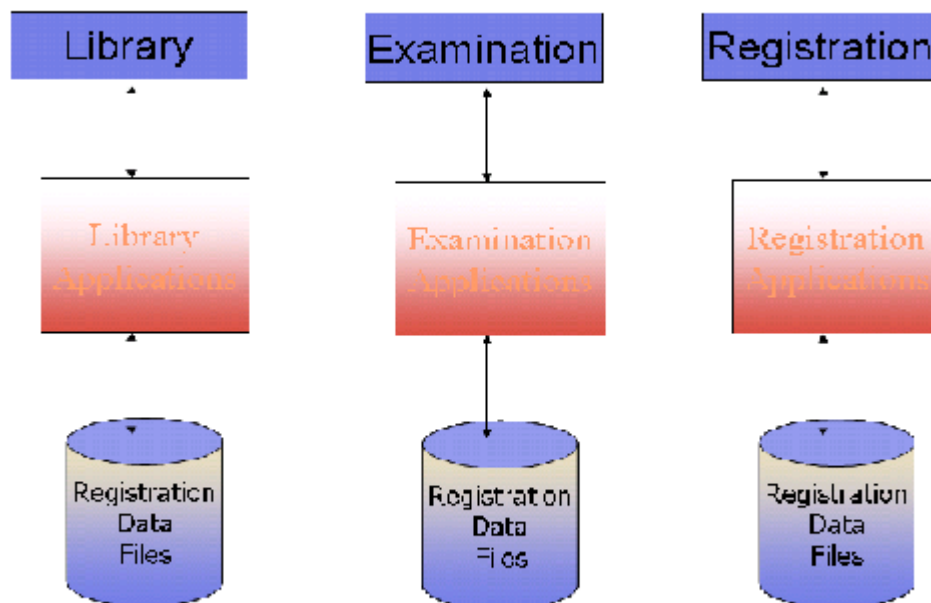
Databases and Traditional File Processing Systems

Traditional file processing system or simple file processing system refers to the first computer-based approach of handling the commercial or business applications. That is why it is also called a replacement of the manual file system. Before the use computers, the data in the offices or business was maintained in the files (well in that perspective some offices may still be considered in the pre-computer age). Obviously, it was laborious, time consuming, inefficient, especially in case of large organizations. Computers, initially designed for the engineering purposes were though of as blessing, since they helped efficient management but file processing environment simply transformed manual file work to computers. So processing became very fast and efficient, but as file processing systems were used, their problems were also realized and some of them were very severe as discussed later.

It is not necessary that we understand the working of the file processing environment for the understanding of the database and its working. However, a comparison between the characteristics of the two definitely helps to understand the advantages of the databases and their working approach. That is why the characteristics of the traditional file processing system environment have been discussed briefly here.

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

File Processing Systems



Program and Data Interdependence

Fig. 1: A typical file processing environment

The diagram presents a typical traditional file processing environment. The main point being highlighted is the **program and data interdependence**, that is, program and data depend on each other, well they depend too much on each other. As a result any change in one affects the other as well. This is something that makes a change very painful or problematic for the designers or developers of the system. What do we mean by change and why do we need to change the system at all. These things are explained in the following.

The systems (even the file processing systems) are created after a very detailed analysis of the requirements of the organizations. But it is not possible to develop a system that does not need a change afterwards. There could be many reasons, mainly being that the users get the real taste of the system when it is established. That is, users tell the analysts or designers their requirements, the designers design and later develop the system based on those requirements, but when system is developed and presented to the users, it is only then they realize the outcome of the effort. Now it could be slightly and (unfortunately) sometimes very different from what they expected or wanted it to be. So the users ask changes, minor or major. Another reason for the change is the change in the requirements. For example, previously the billing was performed in an organization on the monthly basis, now company has decided to bill the customers after every ten days. Since the bills are being generated from the computer (using file processing system), this change has to be incorporated in the system. Yet another example is that, initially bills did not contain the address of the customer, now the company wants the address to be placed on the bill, so here is change. There could be many more examples, and it is so common that we can say

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

that almost all systems need changes, so system development is always an on-going process.

So we need changes in the system, but due to program-data interdependence these changes in the systems were very hard to make. A change in one will affect the other whether related or not. For example, suppose data about the customer bills is stored in the file, and different programs use this file for different purposes, like adding data into the bills file, to compute the bill and to print the bill. Now the company asks to add the customers' address in the bills, for this we have to change the structure of the bill file and also the program that prints the bill. Well, this was necessary, but the painful thing is that the other programs that are using these bills files but are not concerned with the printing of the bills or the change in the bill will also have to be changed, well; this is needless and causes extra, unnecessary effort.

Another major drawback in the traditional file system environment is the non-sharing of data. It means if different systems of an organization are using some common data then rather than storing it once and sharing it, each system stores data in separate files. This creates the problem of redundancy or wastage of storage and on the other hand the problem on inconsistency. The change in the data in one system sometimes is not reflected in the same data stored in other system. So different systems in organization; store different facts about same thing. This is inconsistency as is shown in figure below.

File Processing Systems

Library	Exam	Registration
Reg_Number	Reg_Number	Reg_Number
Name	Name	Name
Father Name	Address	Father Name
Books Issued	Class	Phone
Fine	Semester	Address
	Grade	Class

Duplication of Data Vulnerable to Inconsistency

Fig. 2: Some more problems in File System Environment

Previous section highlighted the file processing system environment and major problems found there. The following section presents the benefits of the database systems.

Advantages of Databases

It will be helpful to reiterate our database definition here, that is, database is a shared collection of logically related data, designed to meet the information needs of multiple

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

users in an organization. A typical database system environment is shown in the figure 3 below:

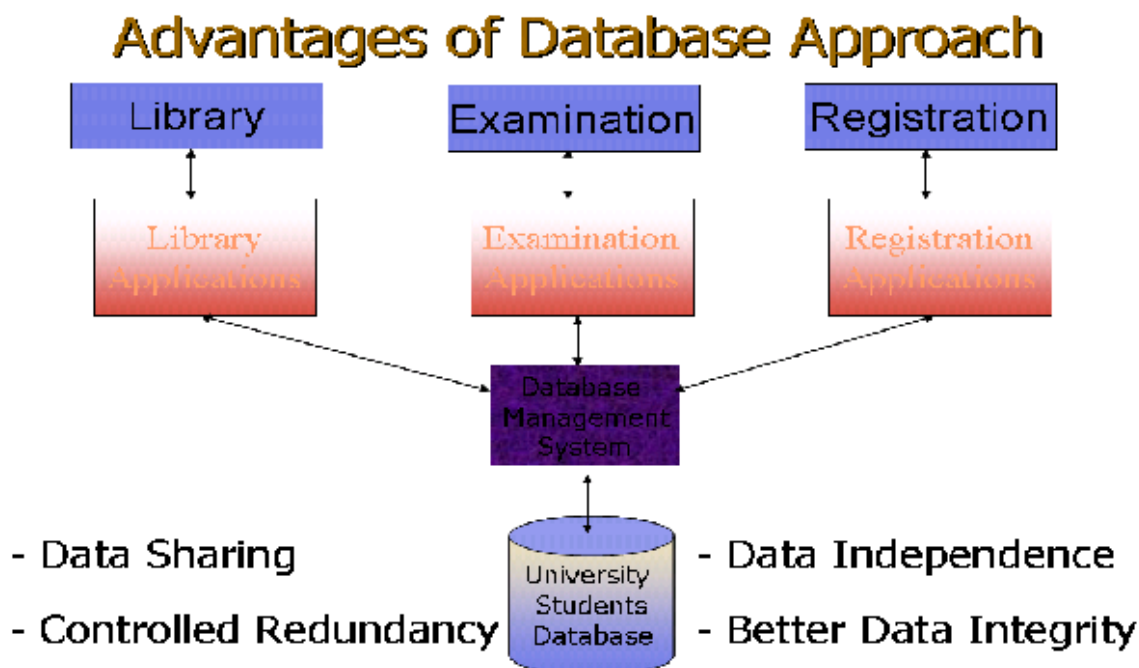


Fig. 3: A typical Database System environment

The figure shows different subsystem or applications in an educational institution, like library system, examination system, and registration system. There are separate, different application programs for every application or subsystem. However, the data for all applications is stored at the same place in the database and all application programs, relevant data and users are being managed by the DBMS. This is a typical database system environment and it introduces the following advantages:

○ **Data Sharing**

The data for different applications or subsystems is placed at the same place. This introduces the major benefit of data sharing. That is, data that is common among different applications need not to be stored repeatedly, as was the case in the file processing environment. For example, all three systems of an educational institution shown in figure 3 need to store the data about students. The example data can be seen from figure 2. Now the data like registration number, name, address, father name that is common among different applications is being stored repeatedly in the file processing system environment, where as it is being stored just once in database system environment and is being shared by all applications. The interesting thing is that the individual applications do not know that the data is being shared and they do not need to. Each application gets the impression as if the data is being stored for it. This brings the advantage of saving the storage along with others discussed later.

○ **Data Independence**

Database Management System	Course Code: CS403
Lecture No. 01	Topic: Introduction to Database
Instructor: Dr. Nayyer Masood	cs403@vu.edu.pk

Data and programs are independent of each other, so change in one has no or minimum effect on other. Data and its structure is stored in the database where as application programs manipulating this data are stored separately, the change in one does not unnecessarily effect other.

- **Controlled Redundancy**

Means that we do not need to duplicate data unnecessarily; we do duplicate data in the databases, however, this duplication is deliberate and controlled.

- **Better Data Integrity**

Very important feature; means the validity of the data being entered in the database. Since the data is being placed at a central place and being managed by the DBMS, so it provides a very conducive to check or ensure that the data being entered into the database is actually valid. Integrity of data is very important, since all the processing and the information produced in return are based on the data. Now if the data entered is not valid, how can we be sure that the processing in the database is correct and the results or the information produced is valid? The businesses make decisions on the basis of information produced from the database and the wrong information leads to wrong decisions, and business collapse. In the database system environment, DBMS provides many features to ensure the data integrity, hence provides more reliable data processing environment.

Dear students, that is all for this lecture. Today we got the introduction of the course, importance of the databases. Then we saw different definitions of database and studied what is data processing then studied different features of the traditional file processing environment and database (DB) system environment. At the end of lecture we were discussing the advantages of the DB approach. There some others to be studied in the next lecture. Suggestions are welcome.

Exercises

- Think about the data that you may want to store about different things around you
- List the changes that may arise during the working of any system, lets say Railway Reservation System

Thanks and good luck

In the name

of

Allah

Database Management Systems

Lecture - 2

Overview of lecture-1

- Database definitions
- File processing systems
- Advantages of database

Today's Lecture

- More advantages
- Some costs
- Levels of data
- Database users

Data & Information

Company: Super Soft

Dept: Sales

Emp Name

Age

Salary

Malik Sharif

23

55

Sh. M. Akmal

24

55

M. A. Butt

20

40

Malik Junaid

19

20

➤ **Schema**

➤ **Database Applications**

➤ **Database Management
System (DBMS)**

Other Advantages

- **Data consistency**
- **Better data security**
- **Faster development of
new applications**

They also provide

- **Economy of scale**
- **Better concurrency control**
- **Better backup and recovery procedures**

BUT

Its not always just the

SUGAR

Disadvantages

- Higher costs
- Conversion cost
- More difficult recovery

Data As Resource

Resource

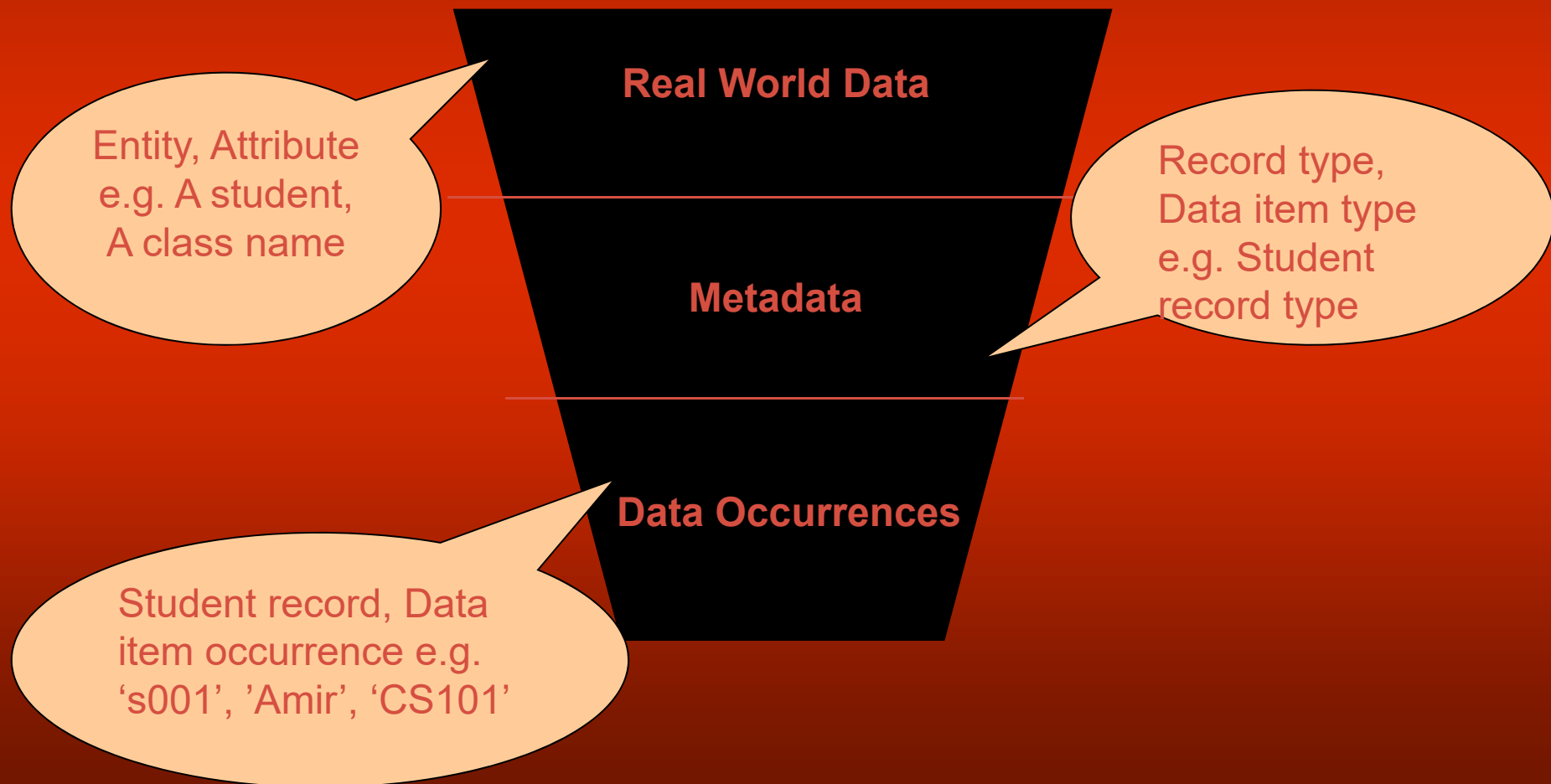
**Any asset that is of
value to an organization
and that incurs cost**

Is data a resource ?

YES

Levels of Data,

Rizwan prepare ppt for front



Employee



name, age,
qual, sal

Emp
Name text
Age number
Sal number

Malik Sharif	23	55
Sh. M. Akmal	24	55
M. A. Butt	20	40
Malik Junaid	19	20

Levels of Data

- **Real-world data**
- **Metadata**
- **Data Occurrence**

Database Users

- Application Programmers
- End Users
 - Naïve
 - Sophisticated

Database Users

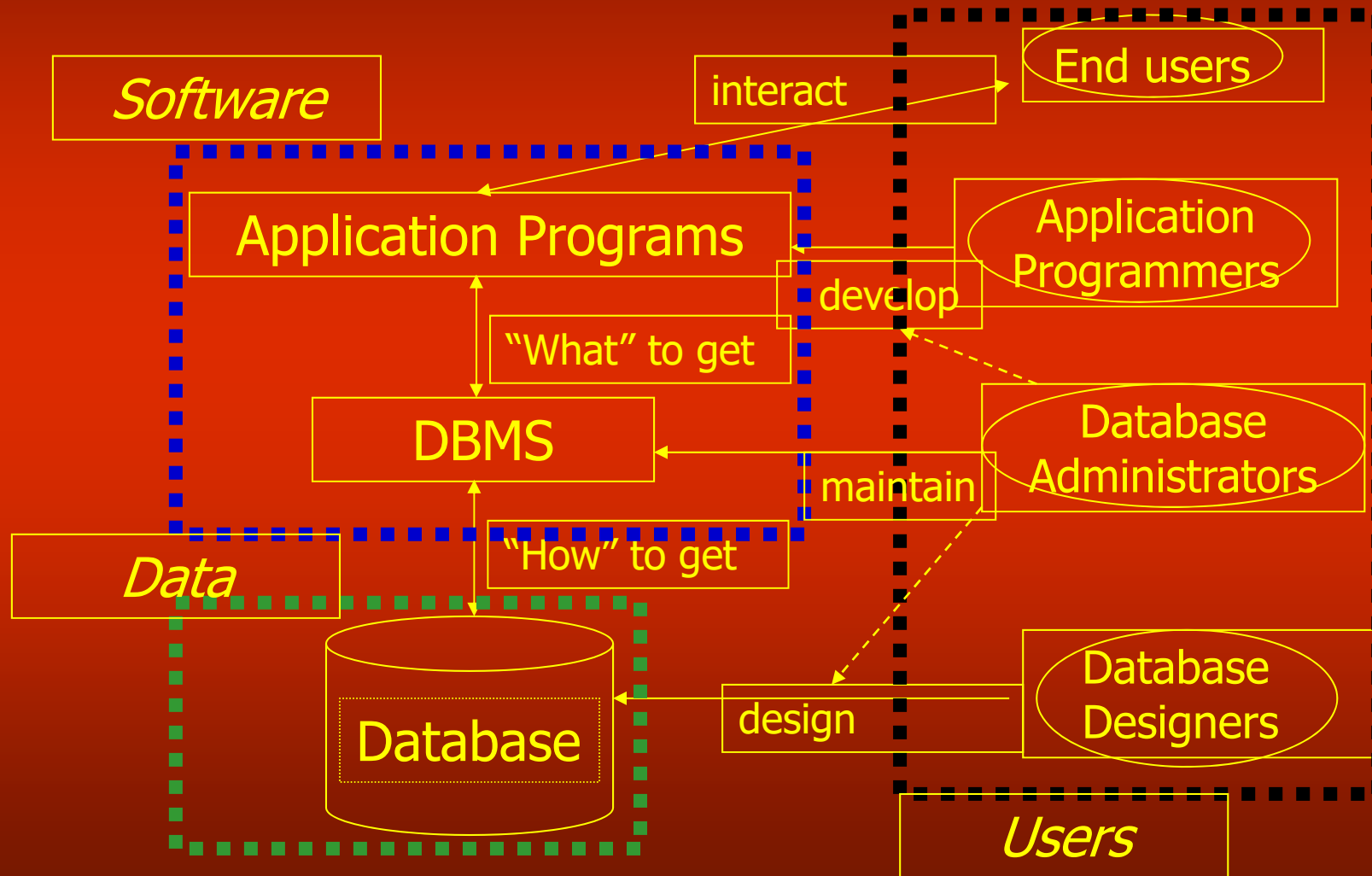
➤ Database Administrator (DBA)

A person who has central control over data and programs that access this data

Functions of DBA

- **Schema definition**
- **Granting data access**
- **Routine Maintenance**
 - **Backups**
 - **Monitoring disk space**
 - **Monitoring jobs running**

Typical Components



Database Architecture

History

- **Initiates in 1960s; IMS, IDS**
- **Conference on DAta Systems Languages (CODASYL)**
- **Data Base Task Group (DBTG)**
- **General architecture in 1971**

History

➤ **DBTG's two layered architecture having**

- **System View- Schema**
- **User View- Sub rschemas**

History

- **American National Standards Institute (ANSI)**
- **Standards Planning and Requirements Committee (SPARC)**
- **ANSI-SPARC's 3-layered Architecture**

Summary

- **Basic Terminology**
- **Pros n Cons of DB Env**
- **Levels of Data**
- **Types of Users**
- **History of 3-L Architecture**

Thanks

Database Management Systems

Lecture - 2

Database Management Systems

Lecture - 3

Lecture Overview

➤ Database Architecture

➤ Data Independence

Database Architecture

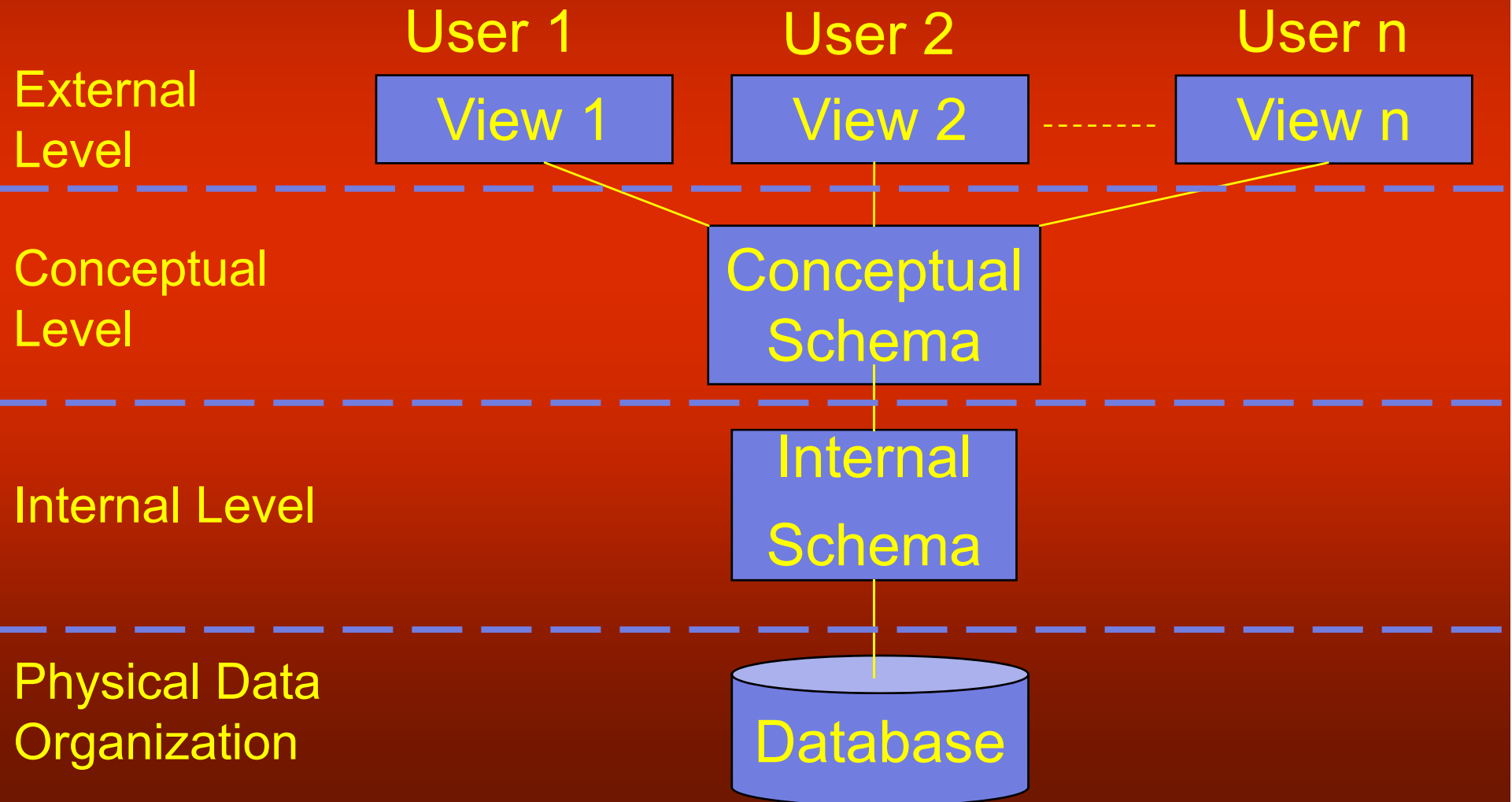
Three Level Architecture

- A basis for understanding DBMS functionalities
- Three levels at which data can be described

Objective

- **Separate users view from the physical representation**
- **Why?**
 - **Different views of same data**
 - **Consolidated representation**
 - **Both ways easy change**

The Three-Level Architecture



The Architecture

- Depicted by three schemas or three models
- Refers to permanent structure or *intention of database*

Level 1

External View

The way users think about data

External View

Each user has a view of the database limited to the appropriate portion of the user's *perspective of reality*.

External View

Users may have *different views* of the same data e.g. date, time etc.

External View

Virtual/calculated data: that is not actually stored in the database but is created when needed e.g. age, statistical data etc.

External View

DBMS uses external views to create user interface for different users which is both the *facility and barrier*

External View

User's external view is created after considering data access, reports, and the transactions needs.

External View

**External schema evolves as
user needs are modified over
time**

Employee Data

First Name: Rana
Last Name: Aslam
Date of Birth:
12 Sep, 1970

Saleema



Workers

Name: R. Aslam
Age: 25y, 10d
Dept: Sales

Saleem

External Layer

Lower Layers

Level 2

Logical or Conceptual View

**A complete description of the
information content of the
database**

Conceptual Schema

- **The entire information structure of the database, as seen by the DBA**
- **The community view of data**

Conceptual Schema

All entities, attributes and their relationships are represented here

Conceptual Schema

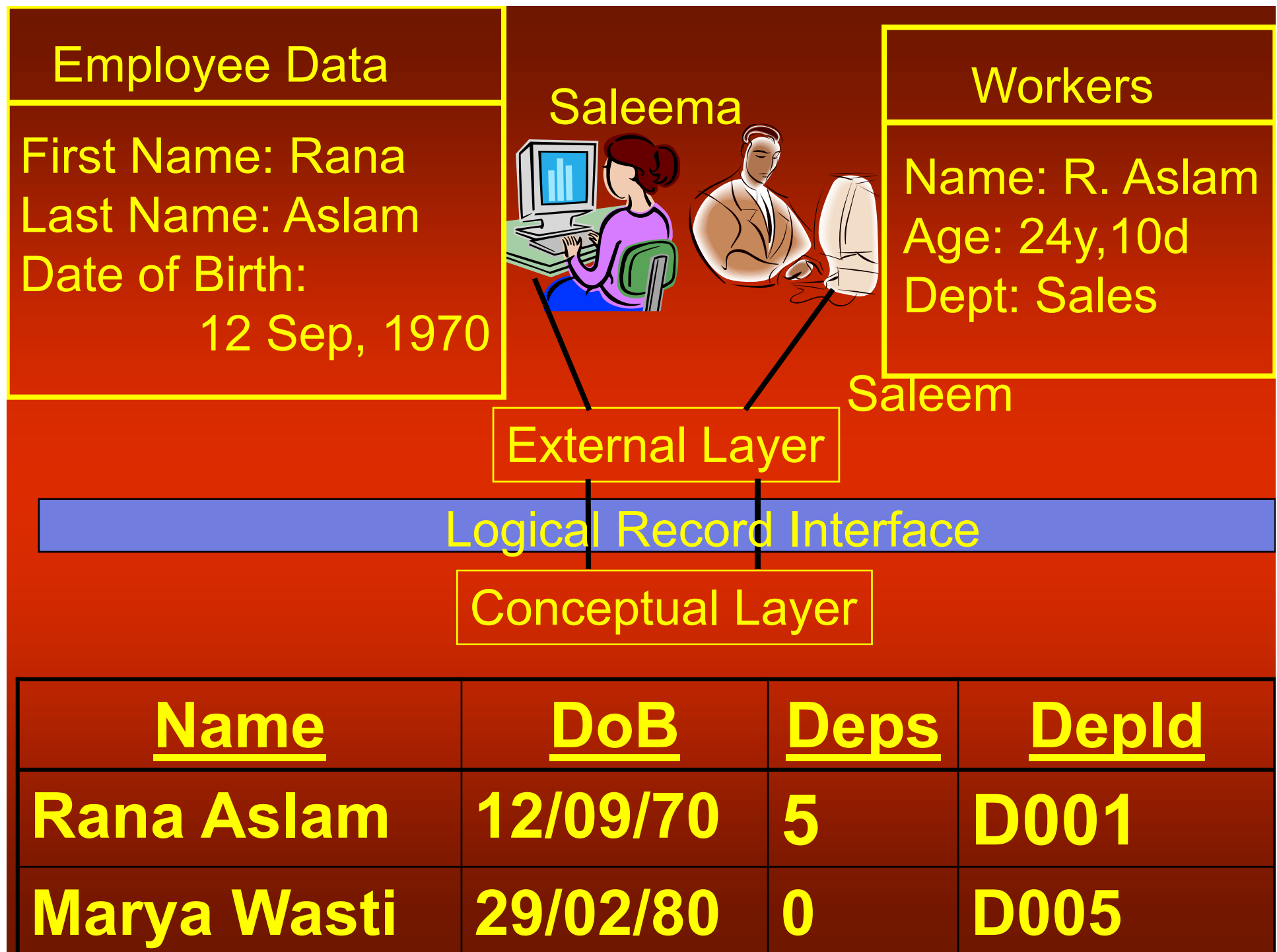
**Contains record types
representing entities, data item
types with their attributes,
relationships and constraints
on data.**

Conceptual Schema

**Contains Semantic information
about the data meaning, security
and integrity information**

Conceptual Schema

Relatively constant: designed
with the present as well as
future needs of an organization



Database Management Systems

Lecture - 3

Database Management Systems

Lecture - 4

Today's Review

- **Continue 3-L architecture**
- **Data Independence**
- **Different aspect of DBMS**

Level 3

Internal or Physical View

**Concerns about the physical
implementation of the database**

Internal View

- **DBMS chooses type of data structures**
- **lays out data on storage devices with operating system access methods**

Internal View

- *Internal record*: a single stored record
- Does not just contain what we see at the conceptual level
- DBMS adds other data

Physical Level

- **Generally same as Internal**
- **Actual representation of data on the storage device**
- **In the binary format**
- **OS responsibility**

Inter-Schema Mappings

- **Also a part of 3-level architecture**
- **External/conceptual mapping**
- **Conceptual/Internal**

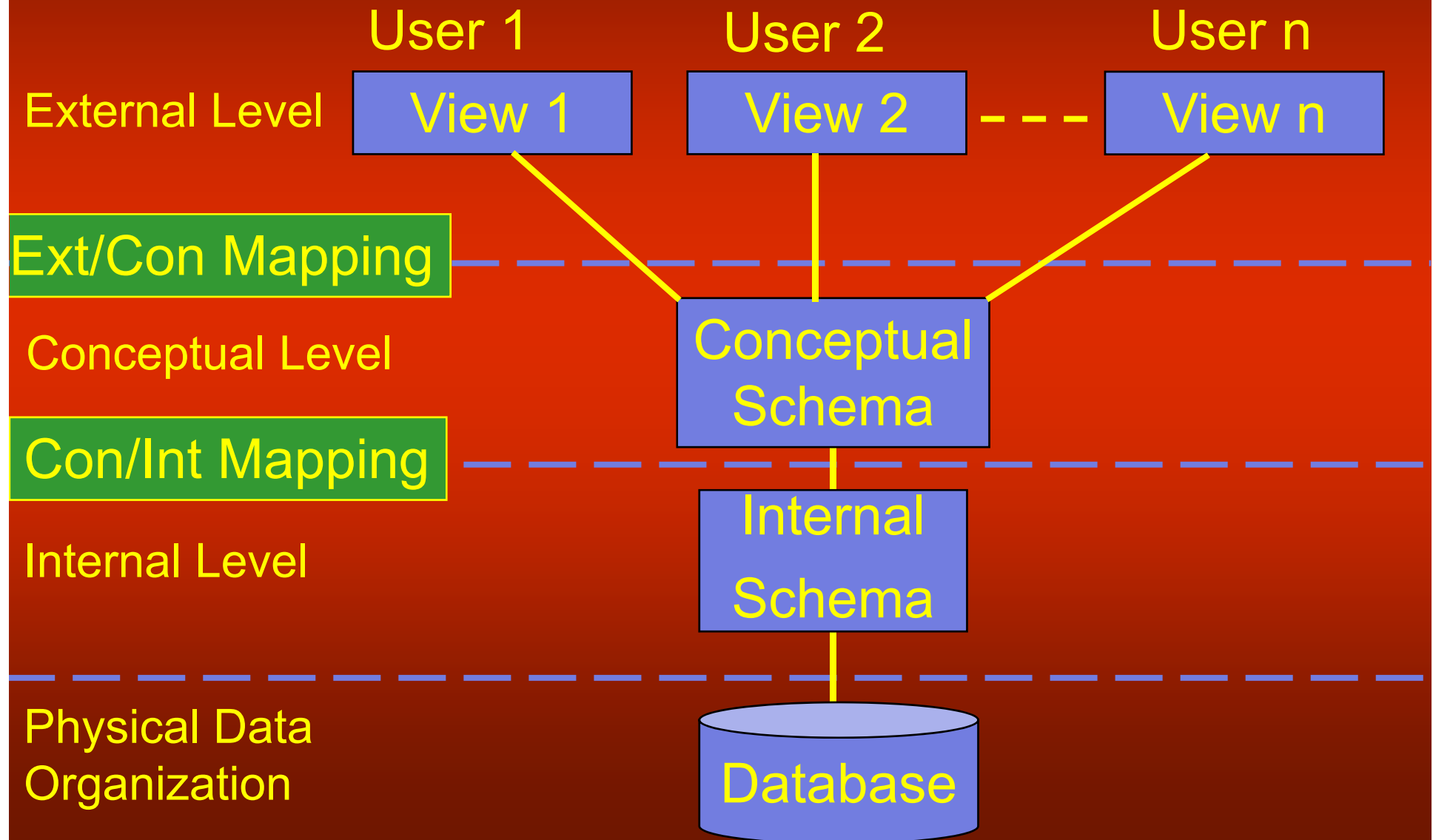
Ext/Con Mapping

**Specifies mapping between
objects in the external view to
those in the logical model**

Con/Int Mapping

**Specifies mapping between
objects in the logical model to
those in the physical model-
data independence**

3-Level Architecture



First Name: Rana

Last Name: Aslam

Date of Birth:

12 Sep, 1970

Saleema



Name: R. Aslam

Age: 24y,10d

Dept: Sales

Saleem

<u>Name</u>	<u>DoB</u>	<u>Deps</u>	<u>DepId</u>
Rana Aslam	12/09/70	5	D001
Marya Wasti	29/02/80	0	D005

BH|RH|Rana Aslam 120970 5 D001|RH|Marya Wasti...

0111001101001110010100101010010100101010010101.....

Data Independence

Data Independence

- **A major outcome of 3-L Arch**
- **The immunity of applications to change in storage structure and access strategy**

Data Independence

- Changes in lower level do not affect the upper levels
- Don't take it word to word
- Mind the direction please



Data Independence Types

- ♦ **Logical Data Independence**
- ♦ **Physical Data Independence**

Logical Data Independence

- Changes in conceptual model do not affect the external views
- Immunity of external level from changes at conceptual level

Types of Changes

- Adding a new file/index etc.
- Adding a new field in a file
- Changing type/size
- Deleting an attribute

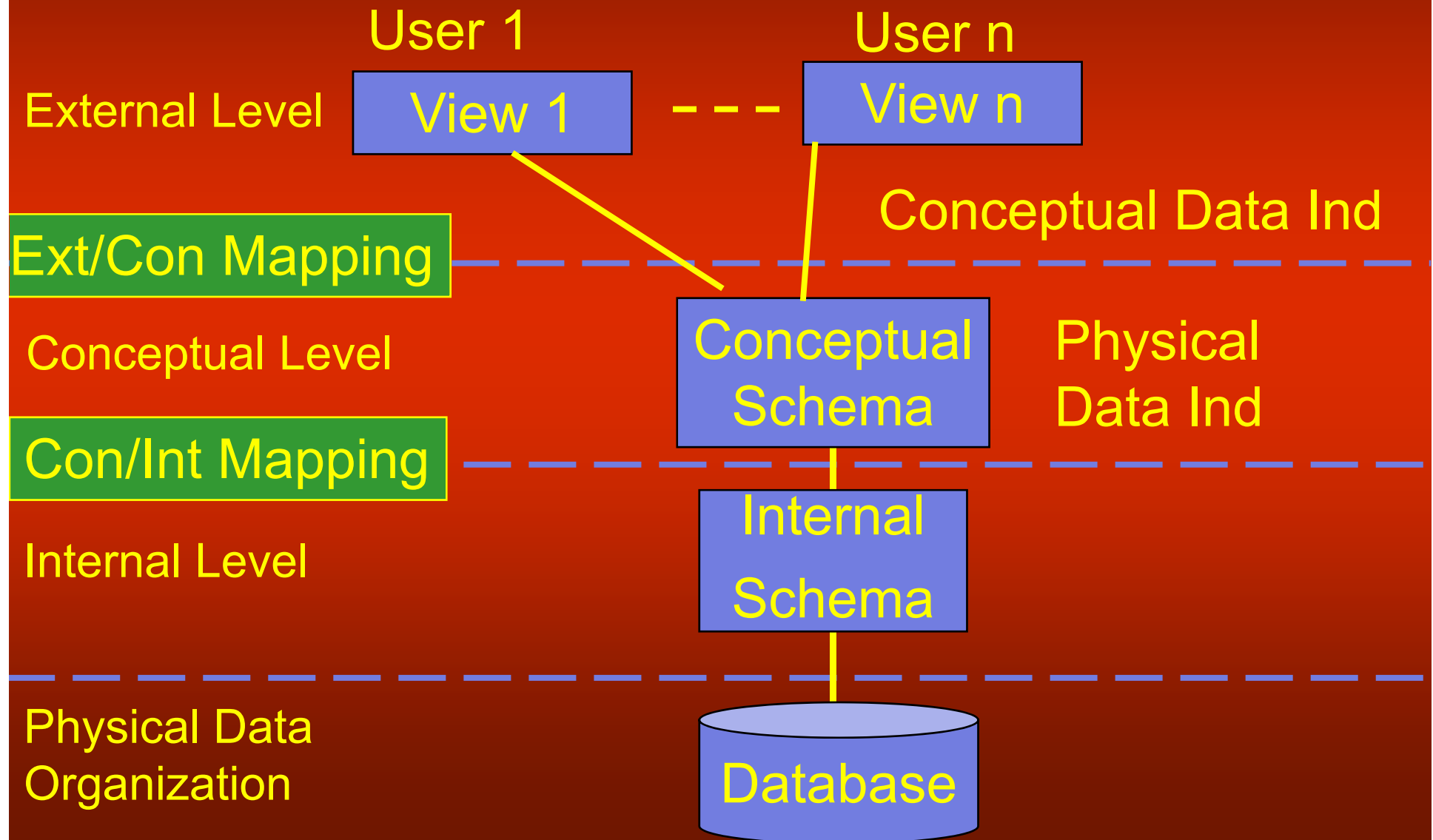
Physical Data Independence

- Changes in the internal model do not affect the conceptual model
- Immunity of Conceptual level from changes at Internal level

Changes Examples

- Changing file organization
- Index implementation, hash, tree etc.
- Changing storage medium

3-Level Architecture



Functions of DBMS

- **Data Processing**
- **A User Accessible Catalog**
- **Transaction Support**
- **Concurrency Control Services**

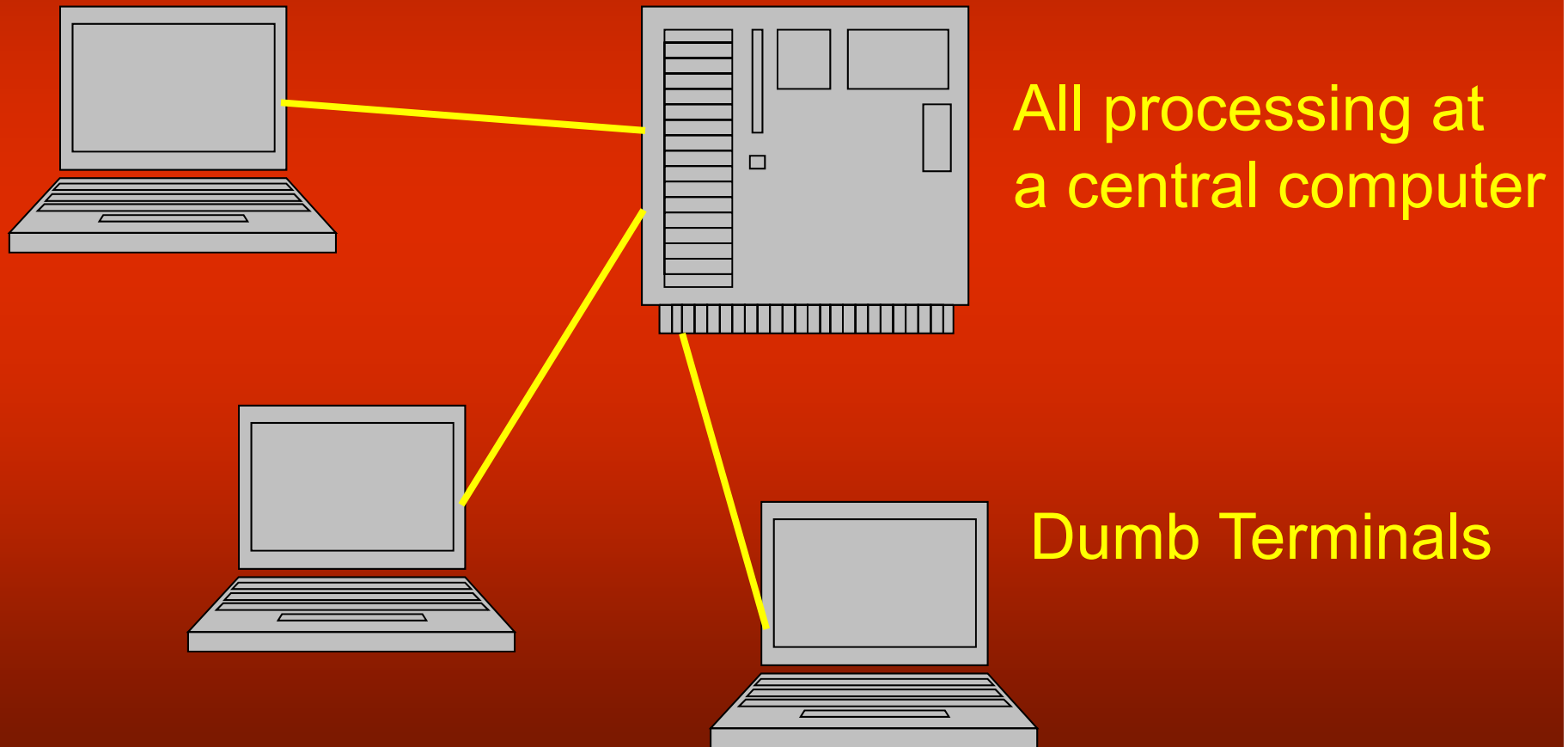
Functions of DBMS

- Recovery Services
- Authorization Services
- Support for Data Communication
- Integrity Services

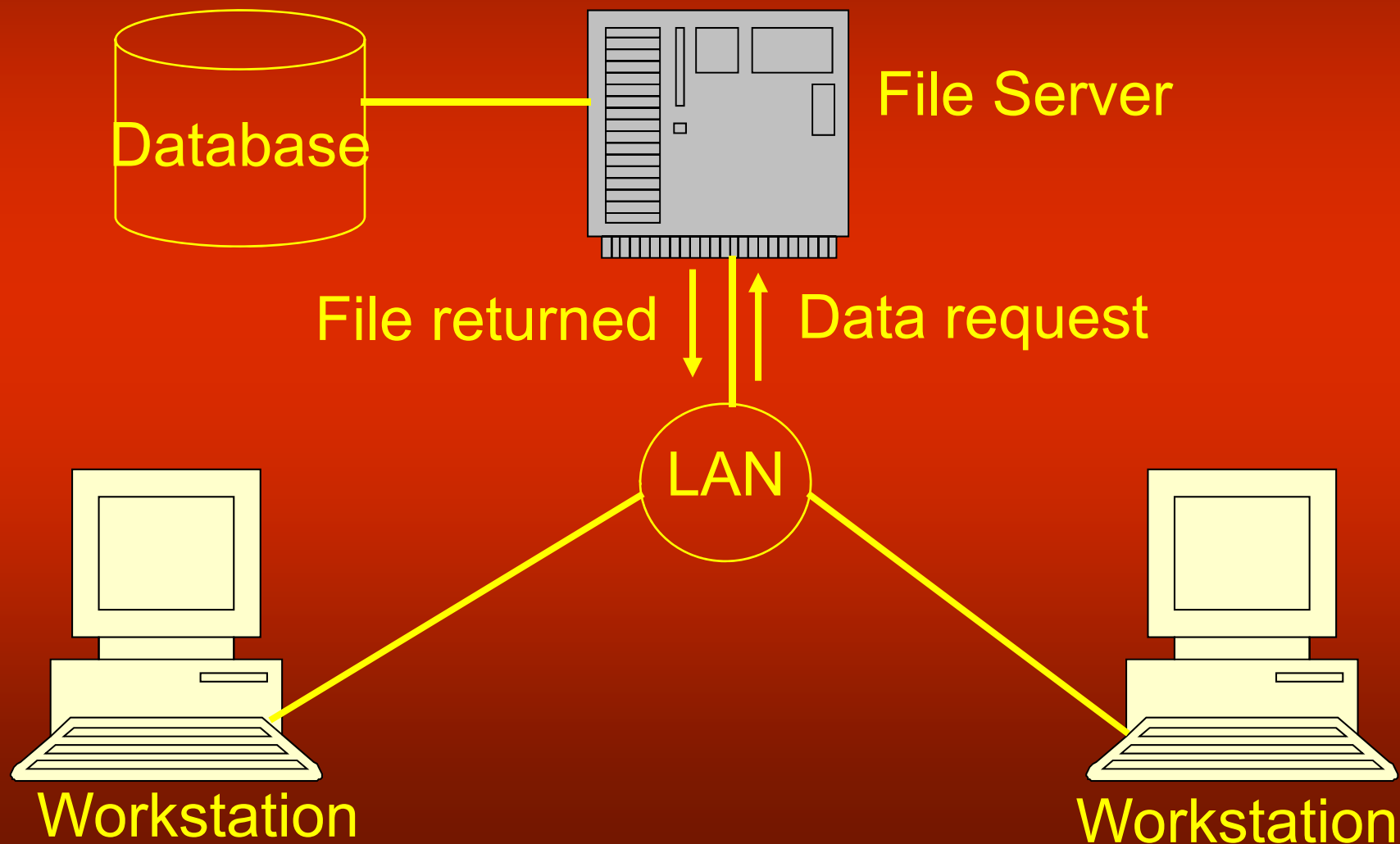
DBMS Environments

- **Single User**
- **Multi-user**
 - **Teleprocessing**
 - **File Servers**
 - **Client-Server**

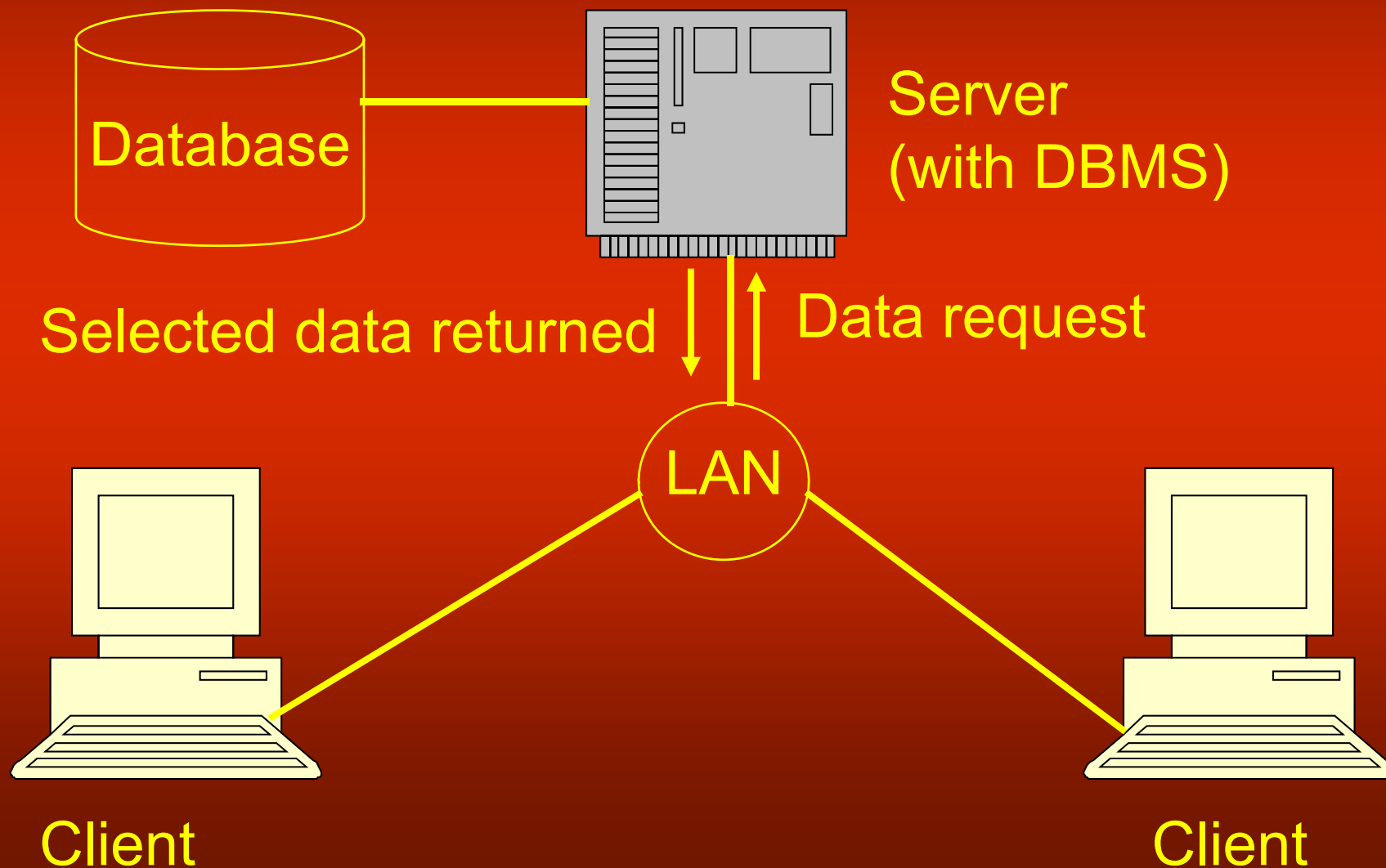
Teleprocessing



File Servers



Client-Server



Today's Summary

- Completed 3-L architecture
- Discussed data independence
- Discussion on DBMS

**Thanks and
Allah Hafiz**

Database Management Systems

Lecture - 4

Database Management Systems

Lecture - 5

Lecture's Overview

- Database Application Development Process
- Preliminary Study of System

Database Application Development Process

Involves

- Database Design
- Application Programs
- Implementation

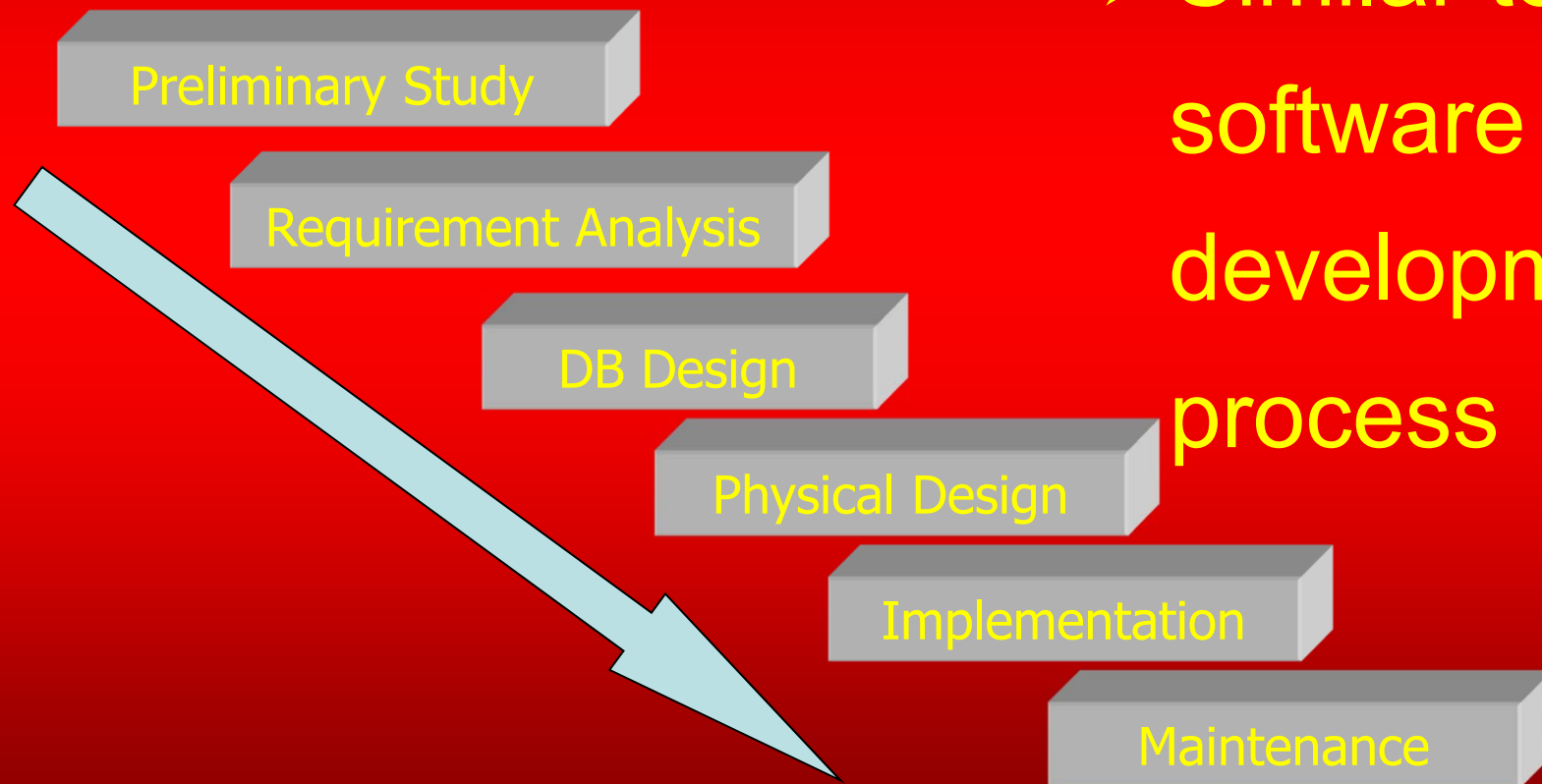
Database Design

- A DB Design is a model of a particular real-world system
- It provides a picture of reality
- Should be simple and self-explanatory

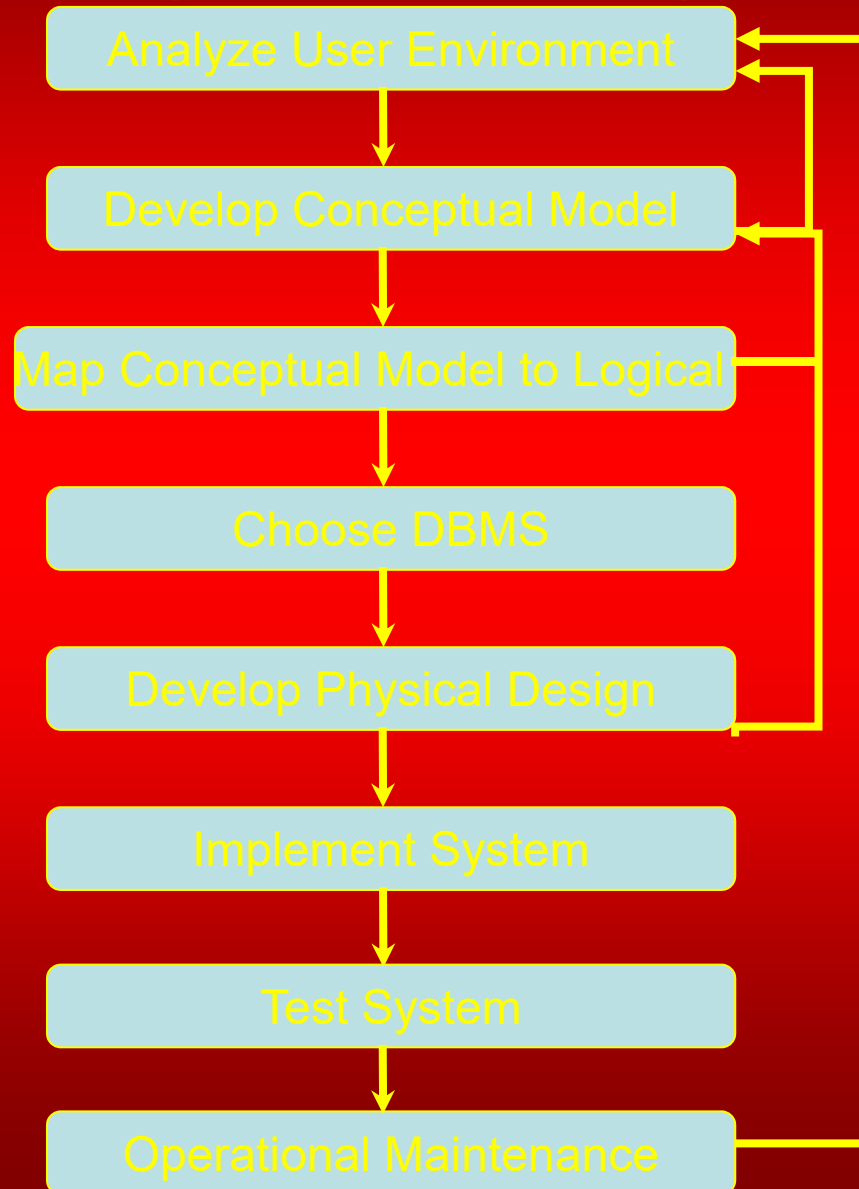
Database Development Process

Database Development Process

➤ Similar to software development process



Design Stages



Analyze Existing System

- Objective: To understand the working of existing system
- Analyze users' requirements

Tool Used

- Why to use at all?
- Data Flow Diagrams

Data Flow Diagrams

Data Flow Diagrams (DFDs)

- Represent the flow of data between different processes within a system

Data Flow Diagrams

- Simple & intuitive, not focusing on details
- To describe, what users do, rather than what computers do

Data Flow Diagrams

➤ Limitations

- Focus only on flows of information
- Decision points/basis not included

DFD-Symbols

Dataflows: pipelines through which packets of information flow. Arrows are labeled with name of the data that moves through



DFD-Symbols

Data Store

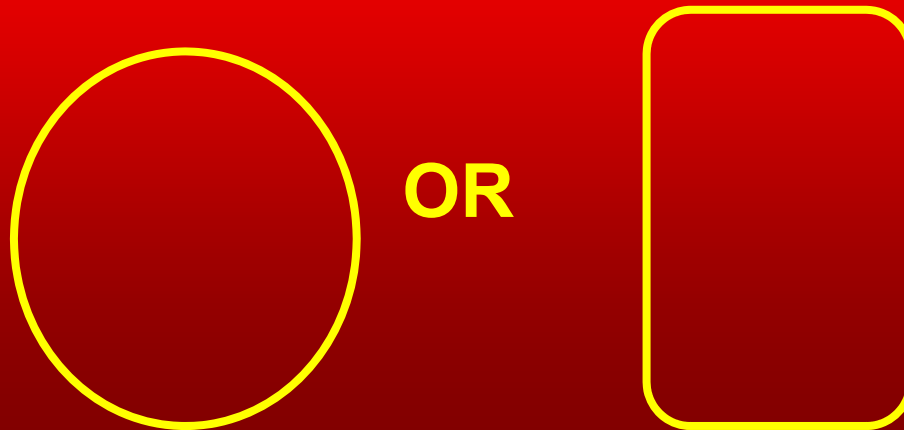
- Repositories of data in system
- Static data
- Data held for processing
- Name is a noun phrase



DFD-Symbols

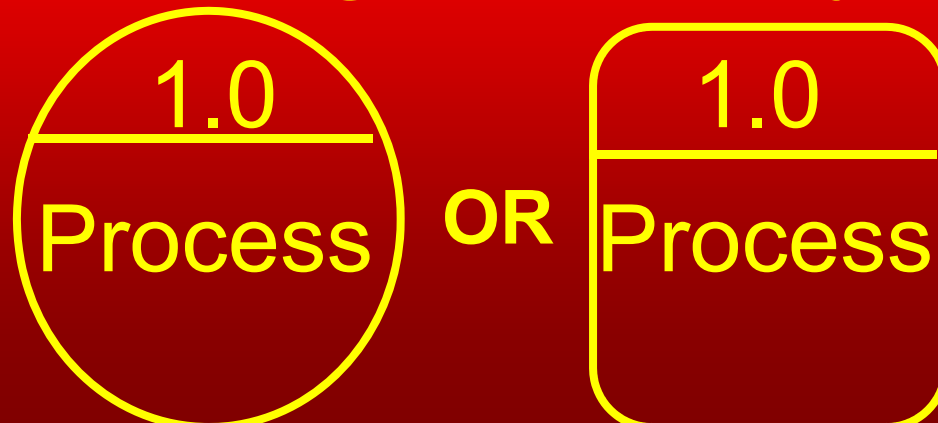
Process

transforms incoming data flow into
outgoing data flow



DFD-Process

- Numbered
- Name is verb/object phrase;
- Noun for high-level systems



DFD-Symbols

External Entities

- Sources/destinations for data
- Outside the system
- Name is Noun Phrase



DFD-Symbols

Collector

- Several data flows combine here
- No processing occurs here

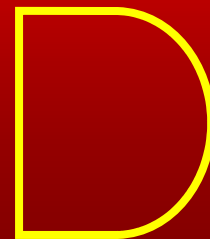


DFD-Symbols

Separator

➤ Several data flows split from here

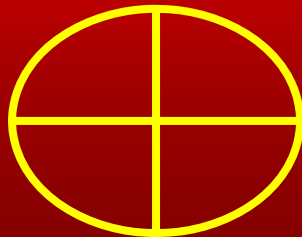
➤ No processing



DFD-Symbols

Ring-sum operator

- Shows two possible data flows
- Only one is followed



DFD-Symbols

AND operator

- Shows two data flows
- Both are followed



Types of DFD

- Context diagram
- Level 0 diagram
- Detailed diagram

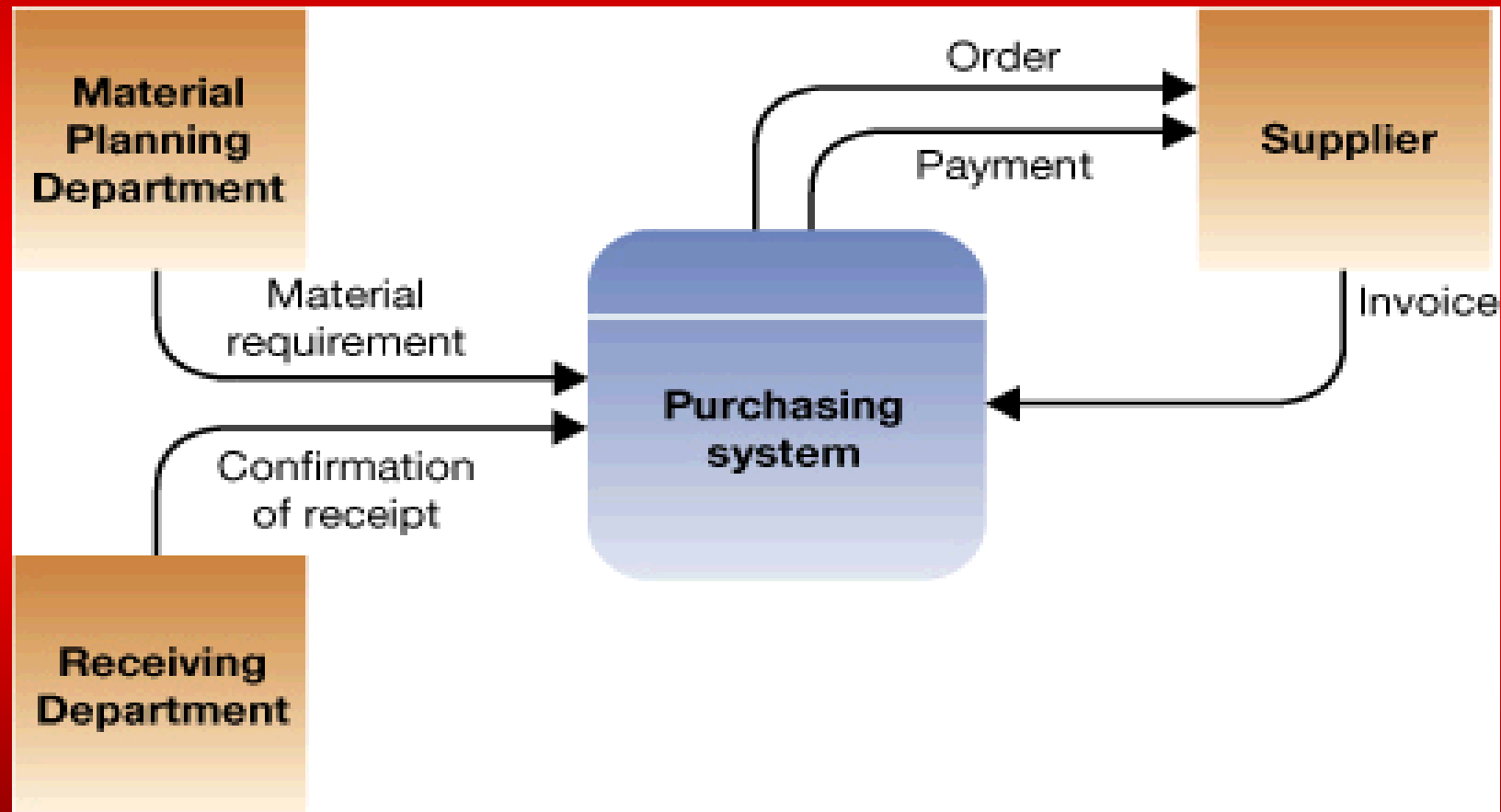
Context Diagram

- Consists of single
Process/System
- Represents the system/process
being analyzed
- Name: Usually a Noun phrase

Context Diagram

- Show Context Only
- Inputs/outputs
- External Entities
- No Data Stores
- No flows between external entities

Context Diagram



Context Diagram

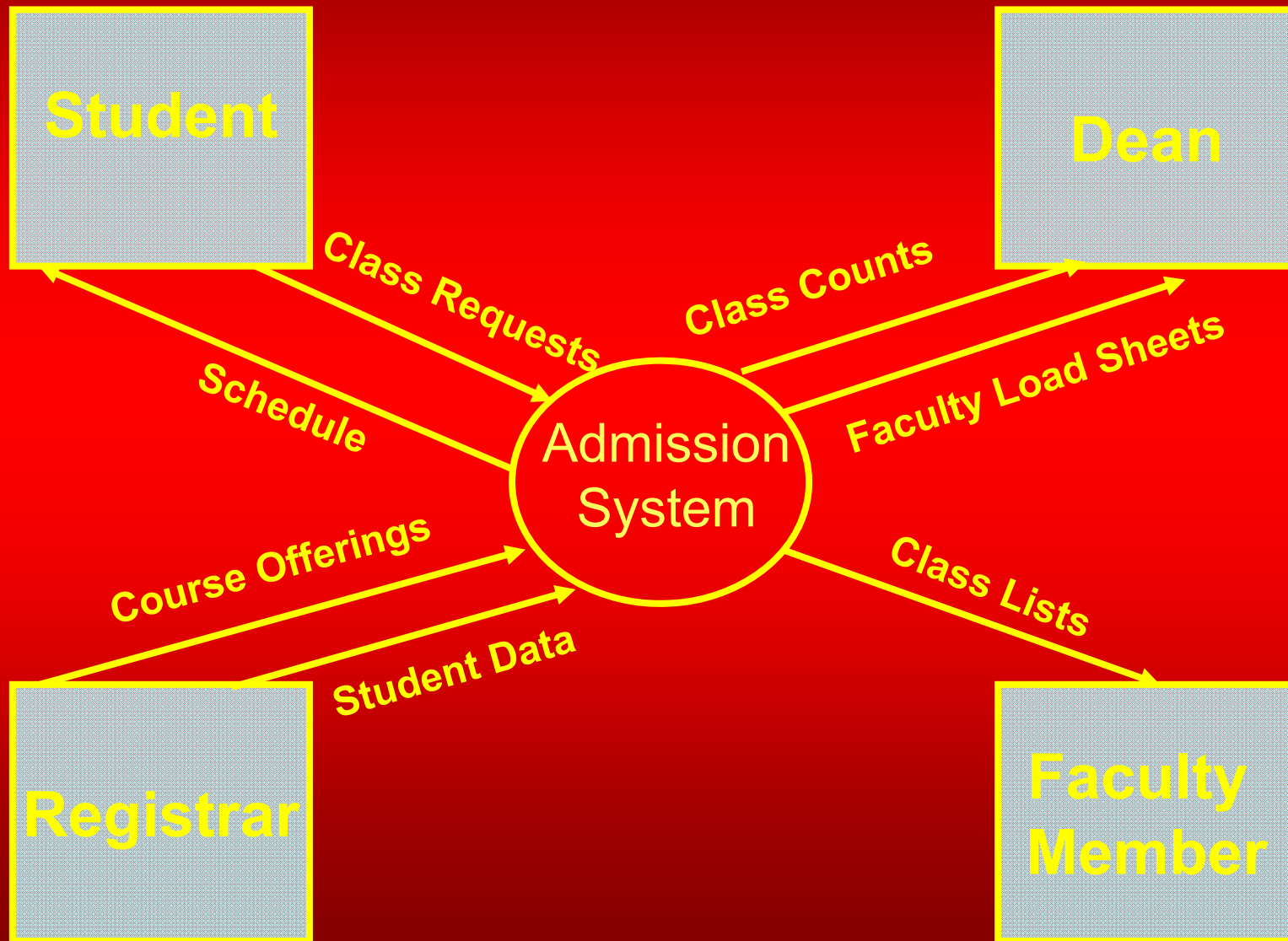
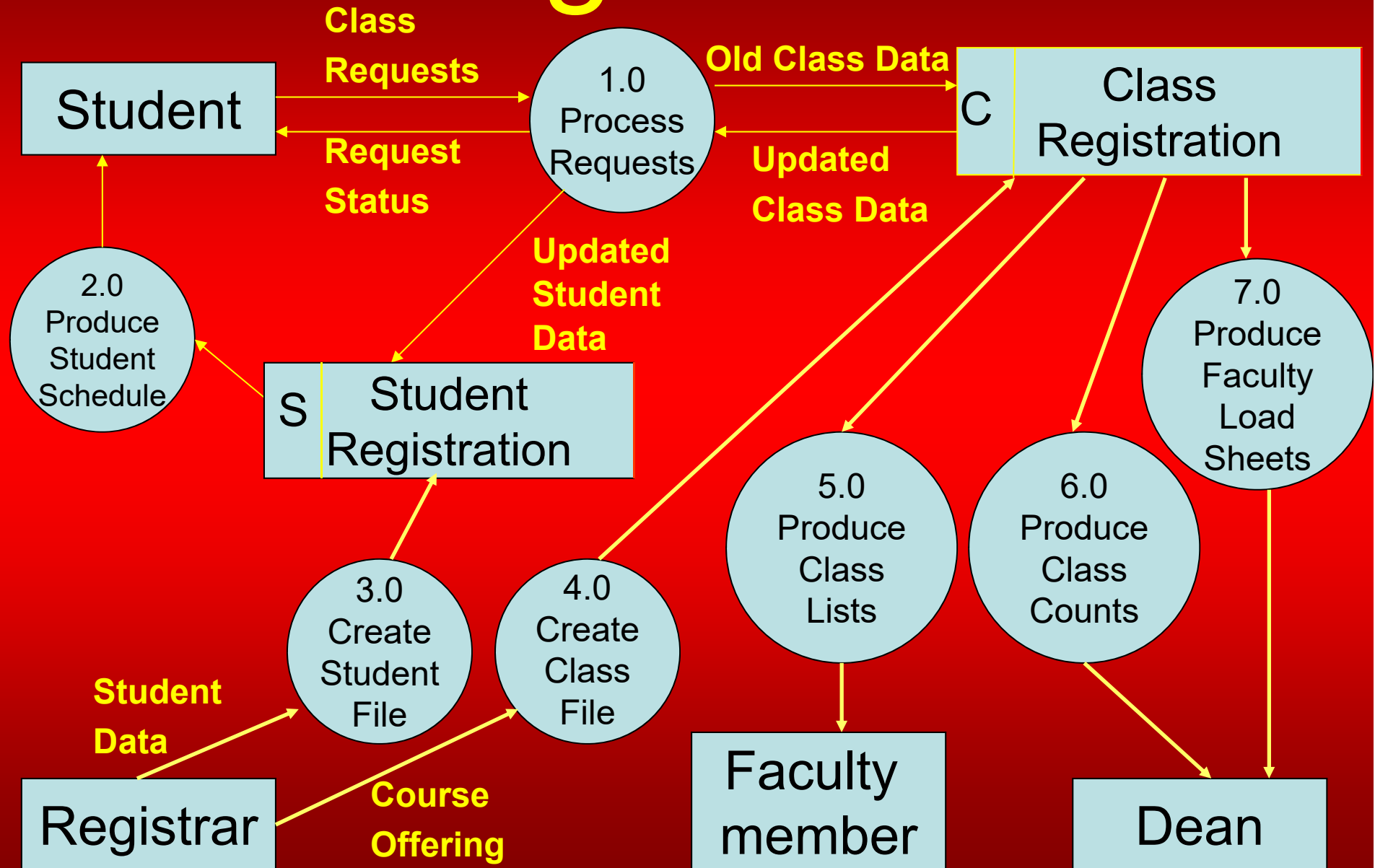
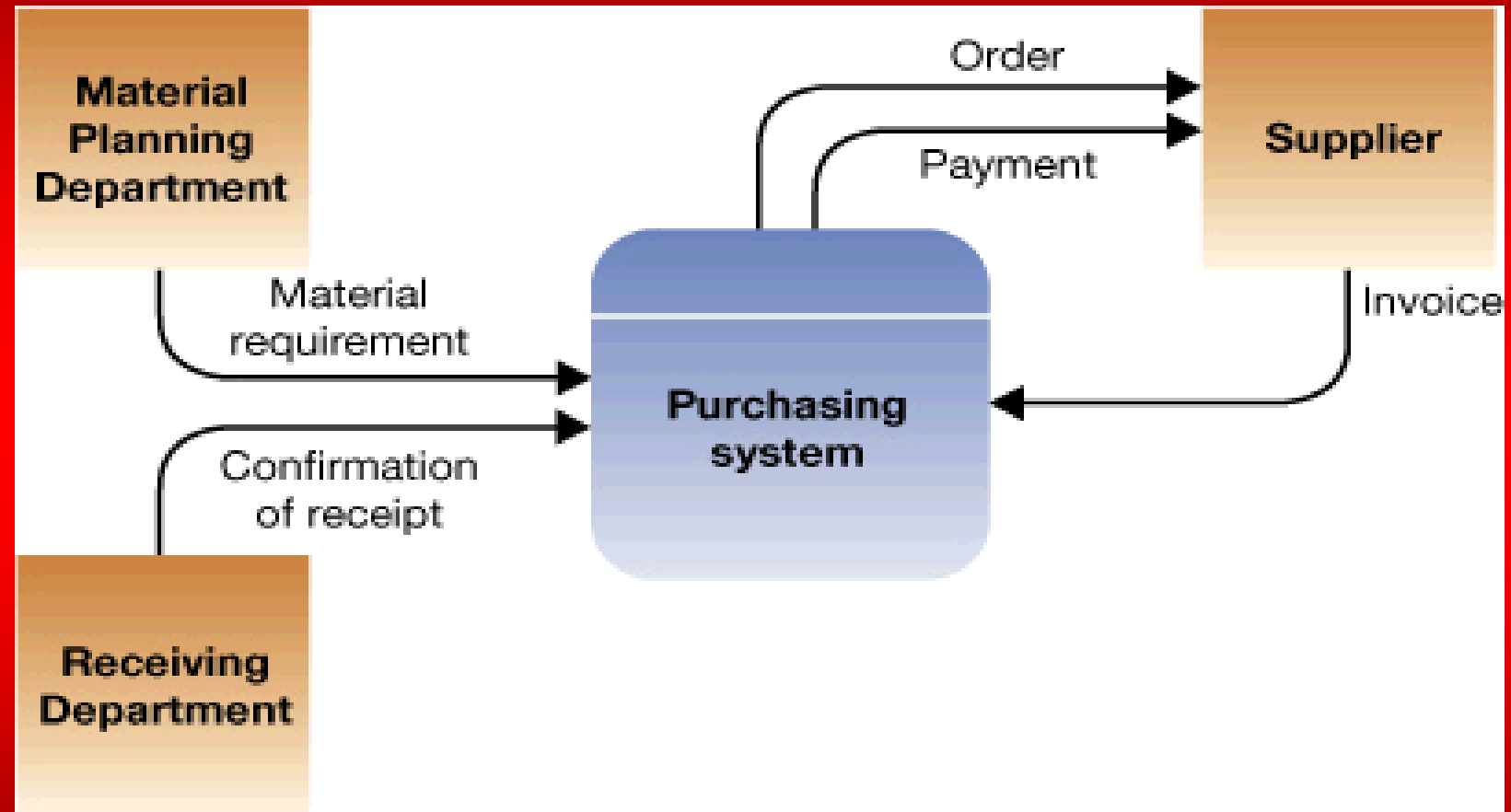


Diagram 0



That is all for lecture 5, however,
Due to time available discussed
Level 0 diagram in air, to be included
Later inshAllah



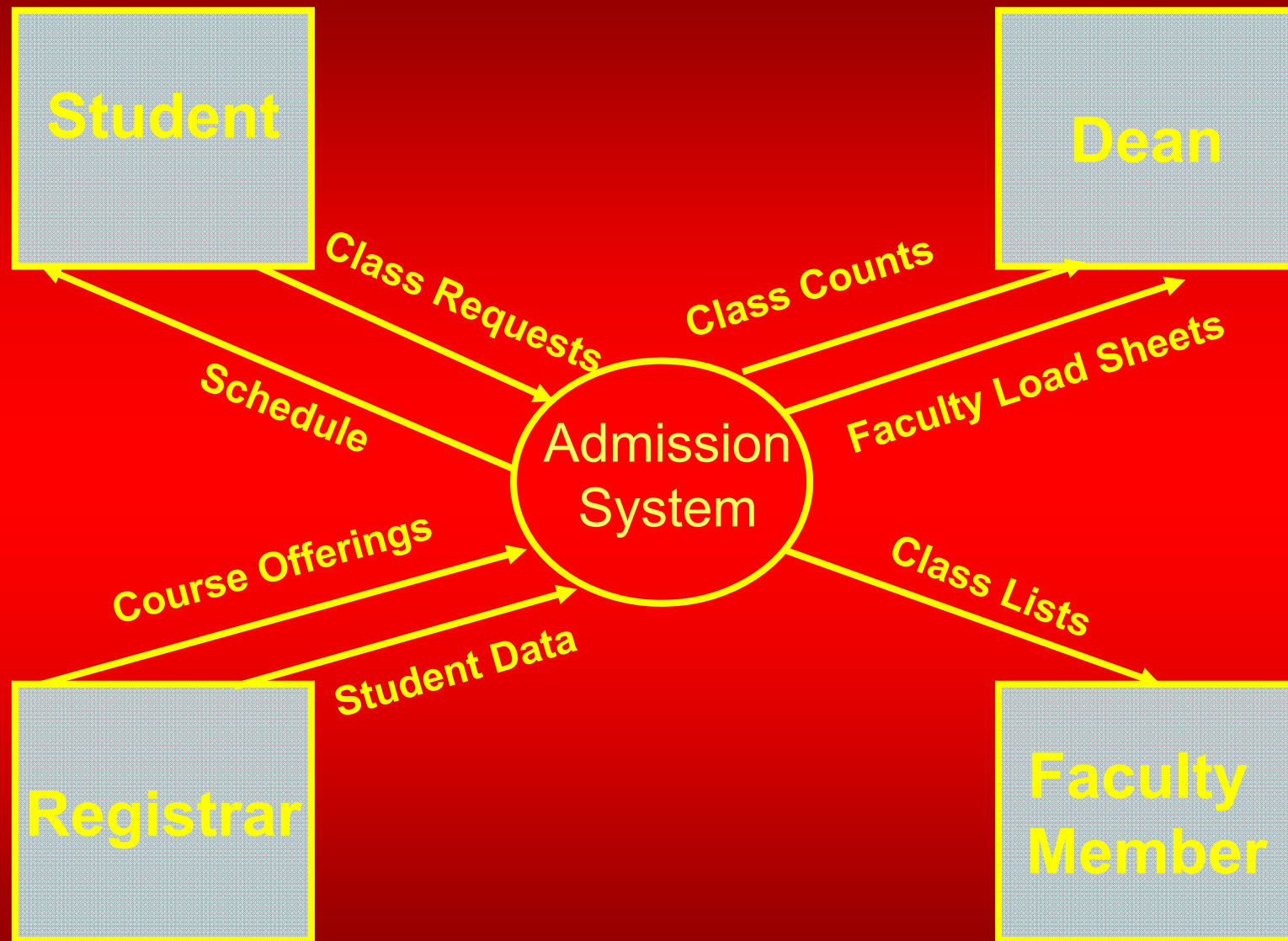
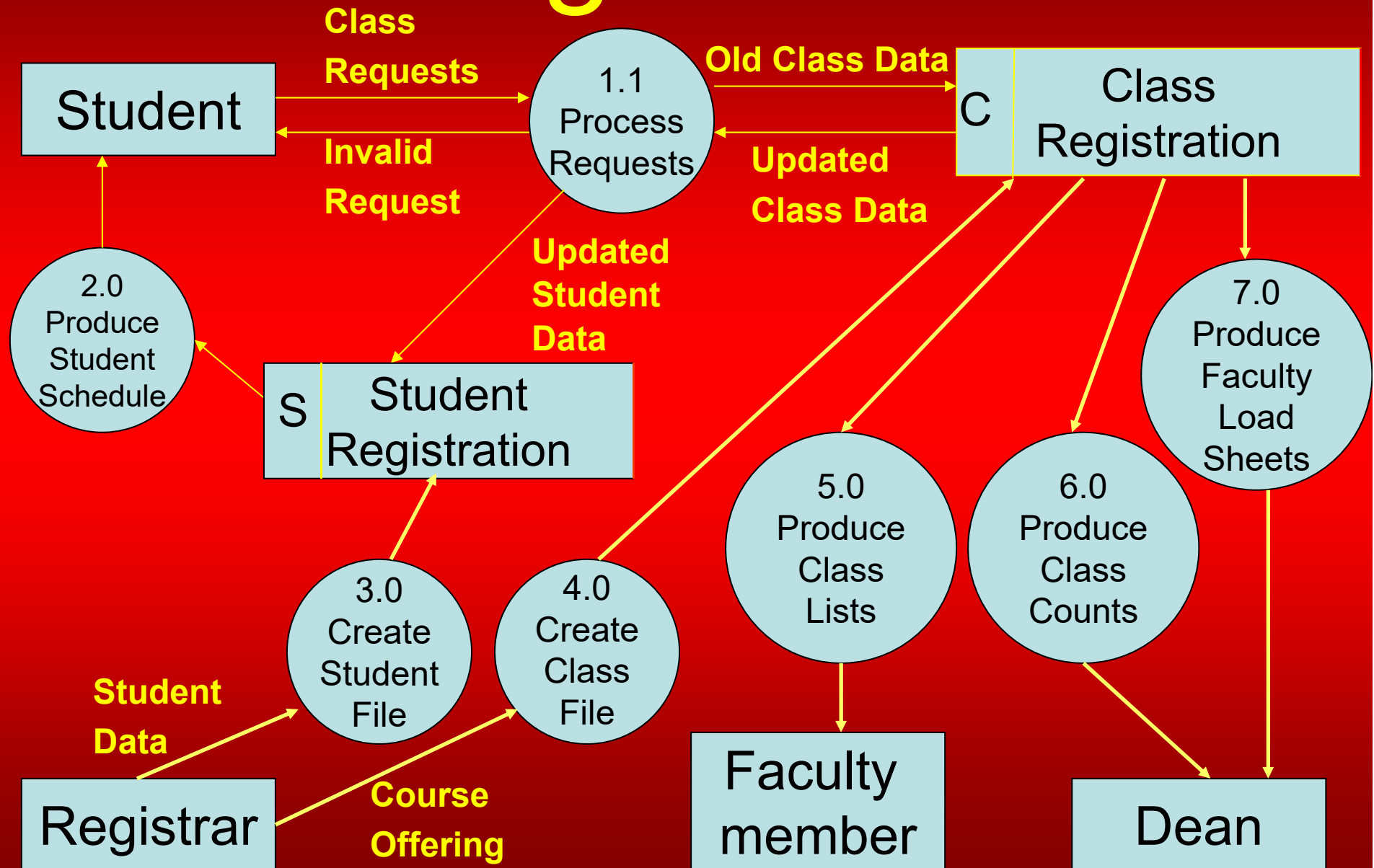
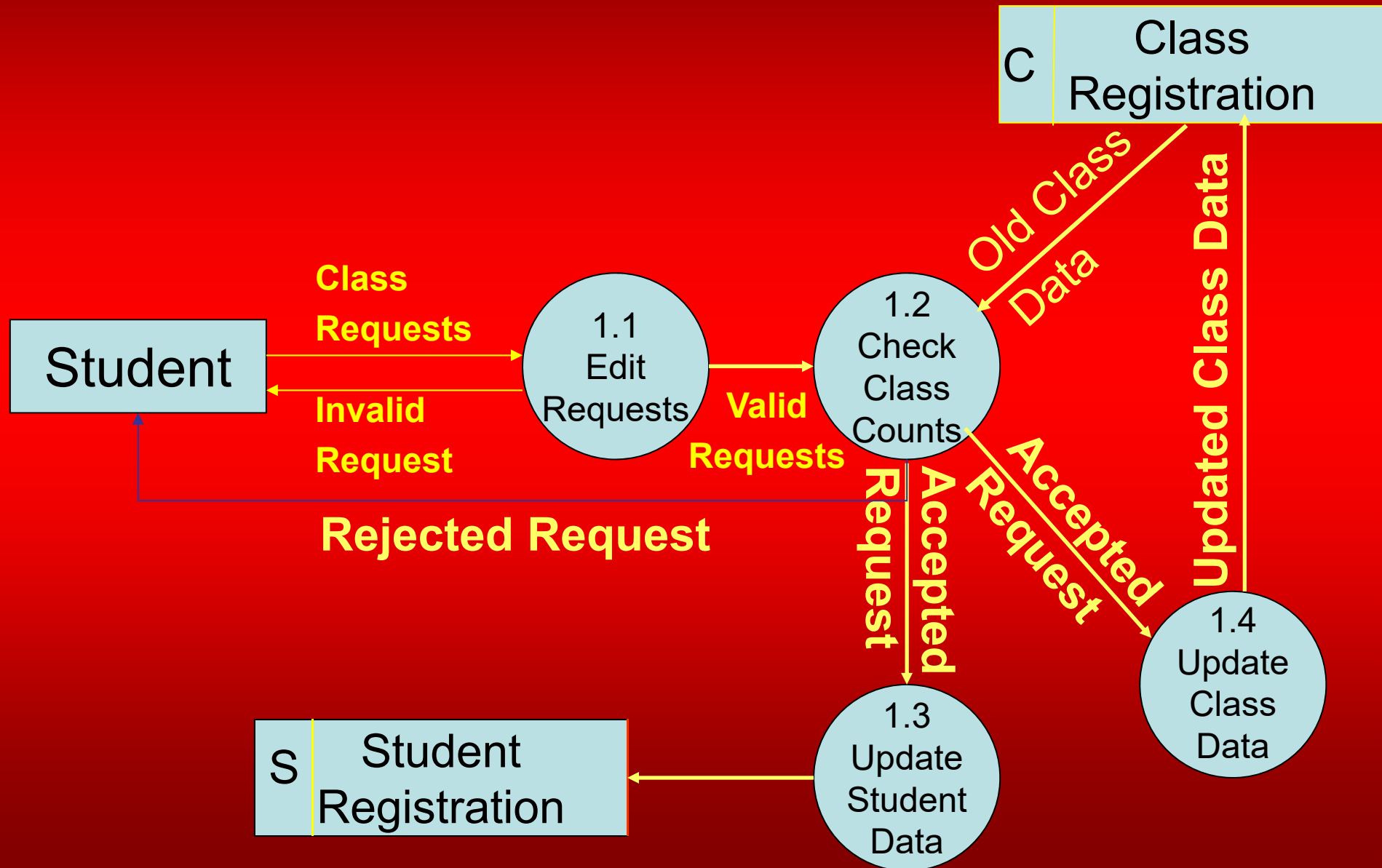


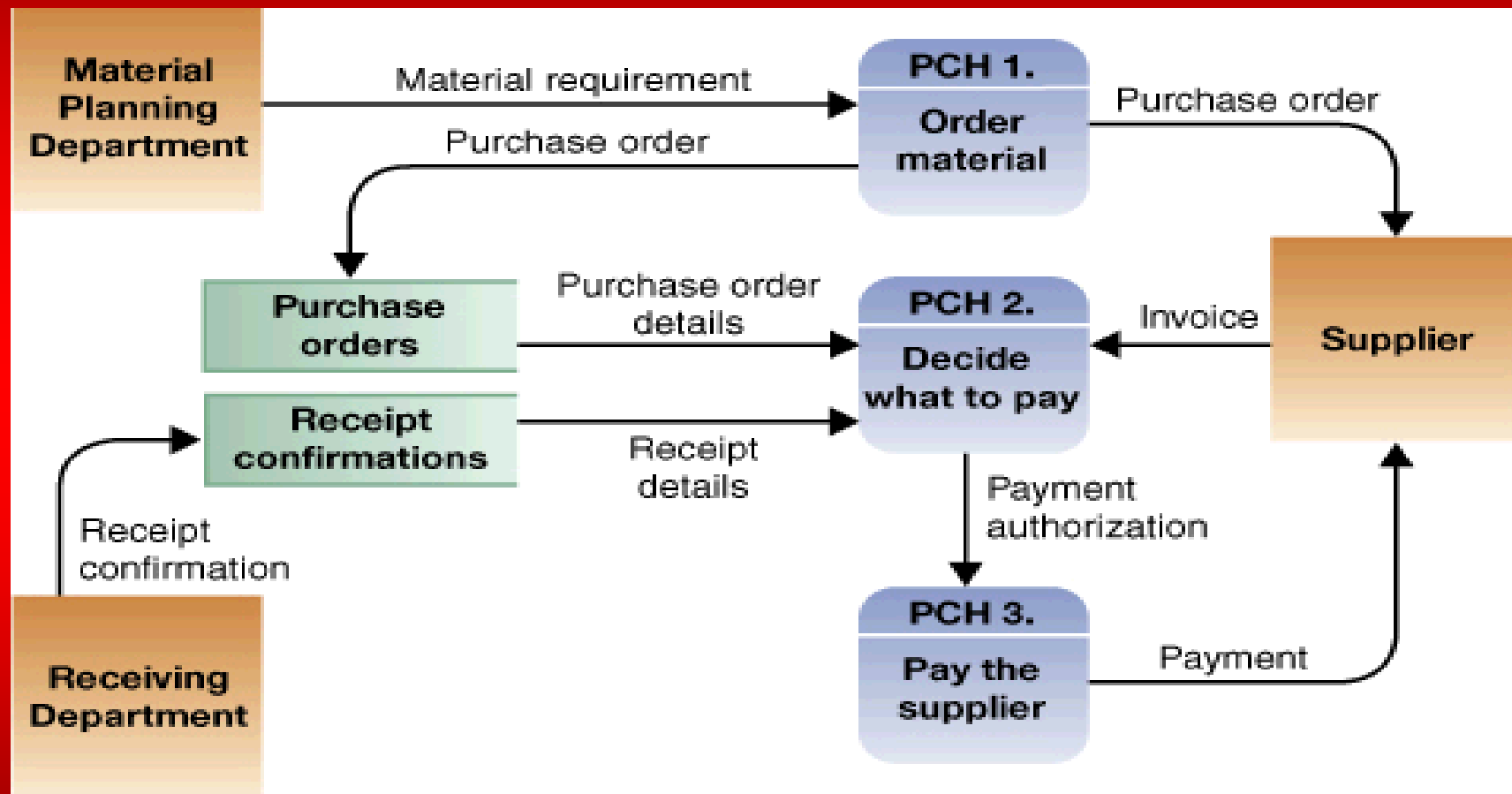


Diagram 0



Detailed Diagram





Overview

- Continue discussion on DFD
- Discussion on Design Phase
- Database Design and Data Models

Database Management Systems

Lecture - 5

Database Management Systems

Lecture - 6

In Previous Lecture

- We were discussing Context Diagram that

In Previous Lecture

- We were discussing Context Diagram that
- And also the level 0 Diagram which is

Detailed Diagrams

- Show processes in successive levels of detail
- shows major sub-processes

Detailed Diagrams

➤ Outline numbering

- 1.1, 1.2, ..., 1.n Level one, shows internal details for each major sub-process
- 1.1.1, 1.1.2,...1.1.n Level two

Detailed Diagrams

- Limit number of process to 7 to 9 per page/diagram for clarity
- All elements must be named

Detailed Diagrams

- All process have both inputs and outputs
 - Outputs must differ from inputs
- Child diagrams and parent process must have same input/output

Detailed Diagram

Data Dictionary

A repository of information that describes logical structure of database, or it contains the metadata

Data Dictionary

- May be integrated or freestanding
- Integrated is referred by DBMS
- Freestanding as a CASE tool

Cross Reference Matrix

- A Data Dictionary tool
- Used to link different things, like functions to entities, or requirements to attributes
- Can be manual even

Cross Reference Table

	Book Issued	Class List	Courses Offered	All Faculty
BooksToFac	✓			
BooksToStu	✓			
FacName		✓	✓	✓
FacAddress				✓
NoOfFac				✓
NoOfStu		✓		
StuName		✓		

Summary

- Analyze User Environment or
- Preliminary Study +
Requirements Analysis
- Use of CASE or manual tools

Database Design Phase

Database Design

- Design and Model mean the same
- Database Design represents logical structure of the database

Database Modeling

- Process of creating a logical representation of the structure of the database
- The most important task in database development

Points to Remember

- A database must mirror the real world. Only then can it answer questions about the real world!
- The emphasis of data modeling is on representing reality

Points to Remember

- Ideally it should be a represented graphically
- The goal is to identify the facts to be stored in the database
- Data modeling involves users and analysts

Data Model

- Created using a data model
- Data Model: is a set of tools or constructs that is used to construct a database design

Components of a DM

- Structures
- Manipulation language
- Integrity constraints
- Beep beep, caution

Significance of DM

- Facilitates and provides guidelines or rules in the DB design process
- Every DBMS is based on a DM

Types of Data Models

- Semantic: Entity-Relationship, Object-Oriented
- Record based: Hierarchical, Network, Relational

DB Design Types

- Conceptual: using SDM
- Logical: using DM of tool
- Physical: using the DBMS

Database Management System

Lecture - 6

Database Management System

Lecture - 7

Entity-Relationship Data Model

E-R Data Model

A semantic data model, used for the graphical representation of the conceptual database design

Major Components

- Entities
- Attributes
- Relationships

Entity

- Term used to mean three different meanings
 - Entity type
 - Entity instance
 - Entity set

Entity Type

- A name/label assigned to items/objects that exist in an environment and that have similar properties
- It could be person, place, event or even concept

Entity Type

- Distinguishable from other entity types on the basis of properties
- Identified through abstraction process
- Different from External Entity

Entity Instance & Set

- A particular object belonging to a particular entity type
- Entity Type: Employee
- Entity Instance: M. Sharif
- Entity Set: All employees

Types of Entity Types

Entity types can be classified into regular/strong/independent ETs or weak/dependent ETs

Weak Entity Types

An entity type whose instances cannot exist without being linked with instances of some other entity type, i.e., they cannot exist independently

Strong Entity Type

- A strong/regular entity type is the one whose instances can exist independently, i.e., without being linked to other instances
- Strong ETs have their own identity

Naming Entity Types

- Singular noun recommended
- Organization specific names
- Write in capitals
- Abbreviations can be used,
be consistent

Symbols

Regular Entity Type

NAME

Weak Entity Type

NAME

EMPLOYEE

DEPENDENTS

BOOK

BOOKCOPY

Attribute

An **attribute** of an entity type is a defining property or quality of the instances of that entity type. Entity instances of same entity type have the same attributes. (e.g. Student Identification, Student Name)

Domain of an Attribute

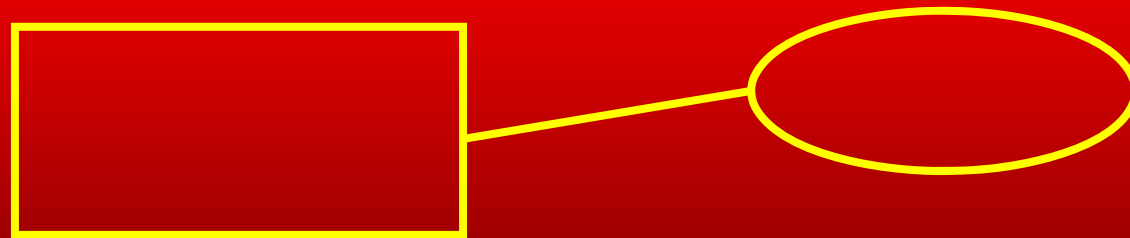
- Every attribute has a domain
- Set of possible values for an attribute
- The attributes in an entity set get the values from the same domain

Types of Attributes

- Single vs composite
- Single valued vs multi-valued
- Stored vs derived

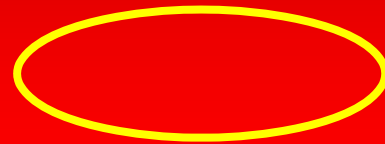
Symbols for Attributes

- Each represented as an oval, linked with an ET symbol

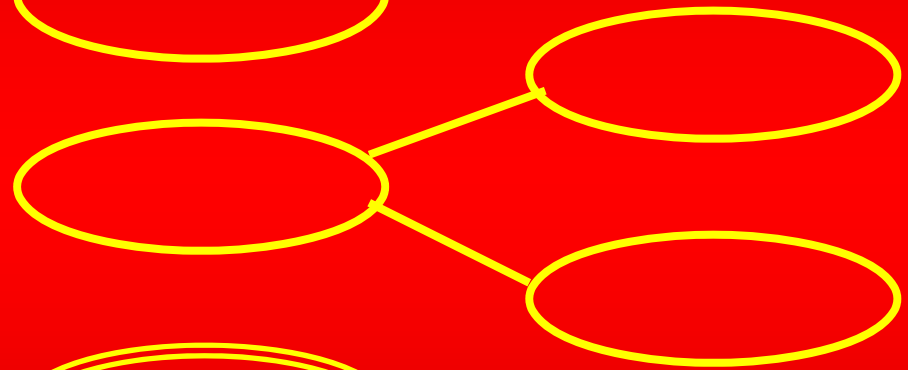


Symbols for Attributes

Simple



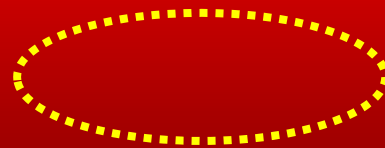
Composite



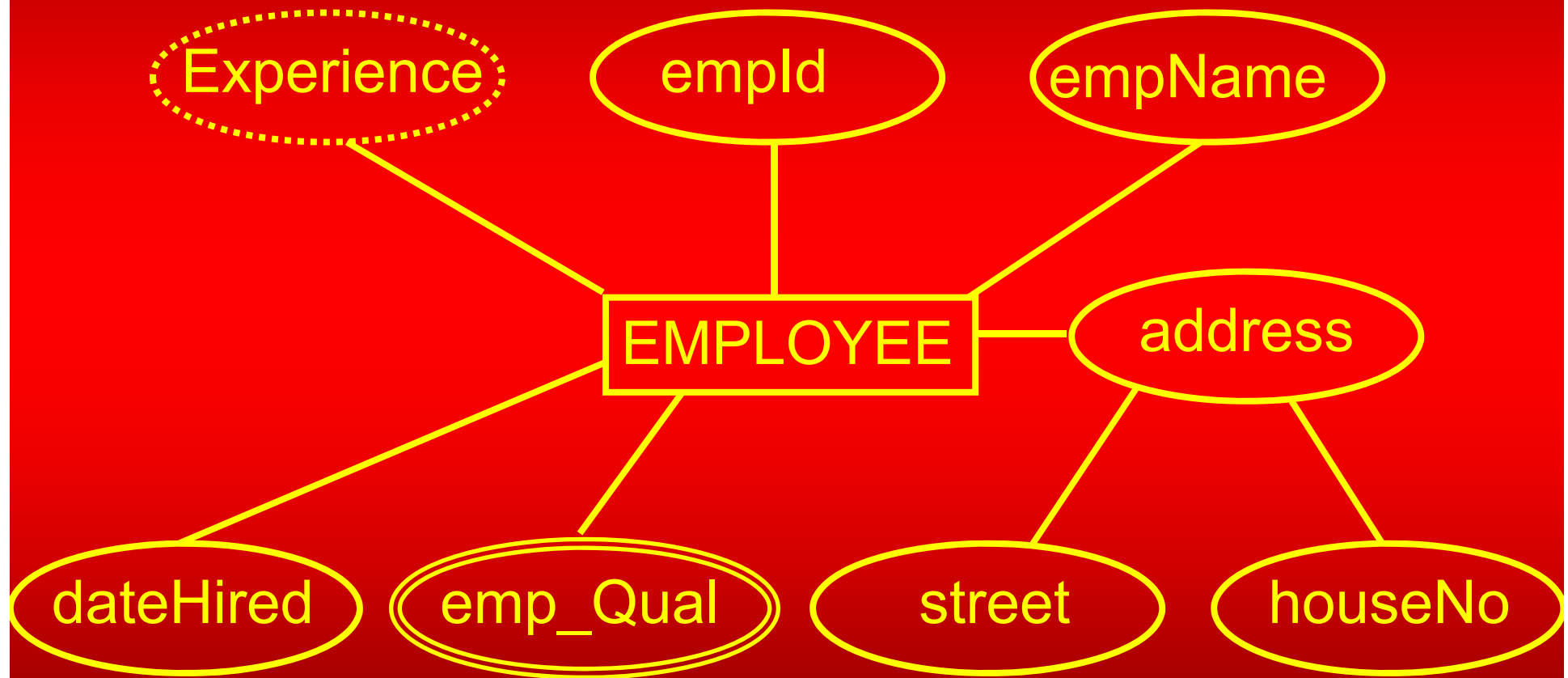
Multi-valued



Derived



Example



Thanks

Database Management System

Lecture - 7

Database Management System

Lecture - 8

Today's Overview

- Key and its different types
- Relationships in E-R Data Model
- Types of relationships

Key Attributes

- An attribute or set of attributes to identify an entity instance uniquely
- Types
 - Super key
 - Candidate key
 - Primary key
 - Secondary and Alternate keys

Example of Key

<u>StdId</u>	StdName	Address	ClName	CurSem
S1020	Suhail Dar	Mareer Hassan	MCS	4
S1038	Shoaib Baber	Model Town	BCS	3
S1015	Tahira Ejaz	Wah Cantt	MCS	2
S1018	Arif Mehmood	Satellite Town	BIT	4
S1025	Suhail Shah	Garhi Shahoo	BCS	6

Simple or Composite Key

- A key consisting of single attribute is called simple key, e.g., StudID, itemNo
- A key consisting of more than one attribute is known as **composite key**, like {Program_Code, Course_Code}

Composite Key Example

OFFERING

<u>ProgCode</u>	<u>CourseCode</u>	MarksAlloc	CrHrs
MCS	DS	100	3
MCS	DBS	100	3
MBA	DBS	100	3
BCS	NW	100	3

Super Key

- Definition same as of key
- For example, for EMPLOYEE and STUDENT entity types EmpID and StudID are the superkeys respectively.

Composite Key Example

<u>StdId</u>	<u>StdName</u>	Address	CIName	CurSem
S1020	Suhail Dar	Mareer Hassan	MCS	4
S1038	Shoaib Baber	Model Town	BCS	3
S1015	Tahira Ejaz	Wah Cantt	MCS	2
S1018	Arif Mehmood	Satellite Town	BIT	4
S1025	Suhail Shah	Garhi Shahoo	BCS	6

Super Key

Any set of attributes containing a super key is also a super key since it too uniquely identifies an entity e.g. {StudID, major}

Super Keys

<u>StdId</u>	StdName	Address	CIName	CurSem
S1020	Suhail Dar	Mareer Hassan	MCS	4
S1038	Shoaib Baber	Model Town	BCS	3
S1015	Tahira Ejaz	Wah Cantt	MCS	2
S1018	Arif Mehmood	Satellite Town	BIT	4
S1025	Suhail Shah	Garhi Shahoo	BCS	6

Candidate Key

A candidate key is the super key that does not contain extra attributes. It might have more than one attribute that uniquely identifies an entity. e.g {name, address}

Candidate Keys

A super key such that no proper subset of its attributes is itself a super key. e.g. {StudID, Major} is not a candidate key because it contains a subset, StudID, that is a super key

CK Example

<u>StdId</u>	StdName	Address	ClName	CurSem
S1020	Suhail Dar	Mareer Hassan	MCS	4
S1038	Shoaib Baber	Model Town	BCS	3
S1015	Tahira Ejaz	Wah Cantt	MCS	2
S1018	Arif Mehmood	Satellite Town	BIT	4
S1025	Suhail Shah	Garhi Shahoo	BCS	6

Primary Key

A primary key is the main/chosen candidate key from the possible set of candidate keys that is most suitable for entity identification.

Primary Key

- It may be a single attribute or a composite key.
- None of its attributes can have ***NULL*** values.

Primary Key

The other candidate keys called Alternate keys provide another method of accessing records.

Need for Key

- Need for unique identification and access
- Secondary Keys: We access on something not necessarily unique

Database Management System

Lecture - 8

Database Management System

Lecture - 9

Relationships

Relationships

- Relationships are the *connections and interactions* between the entities instances, e.g., Program and Student ETs are linked
- How to identify relationships

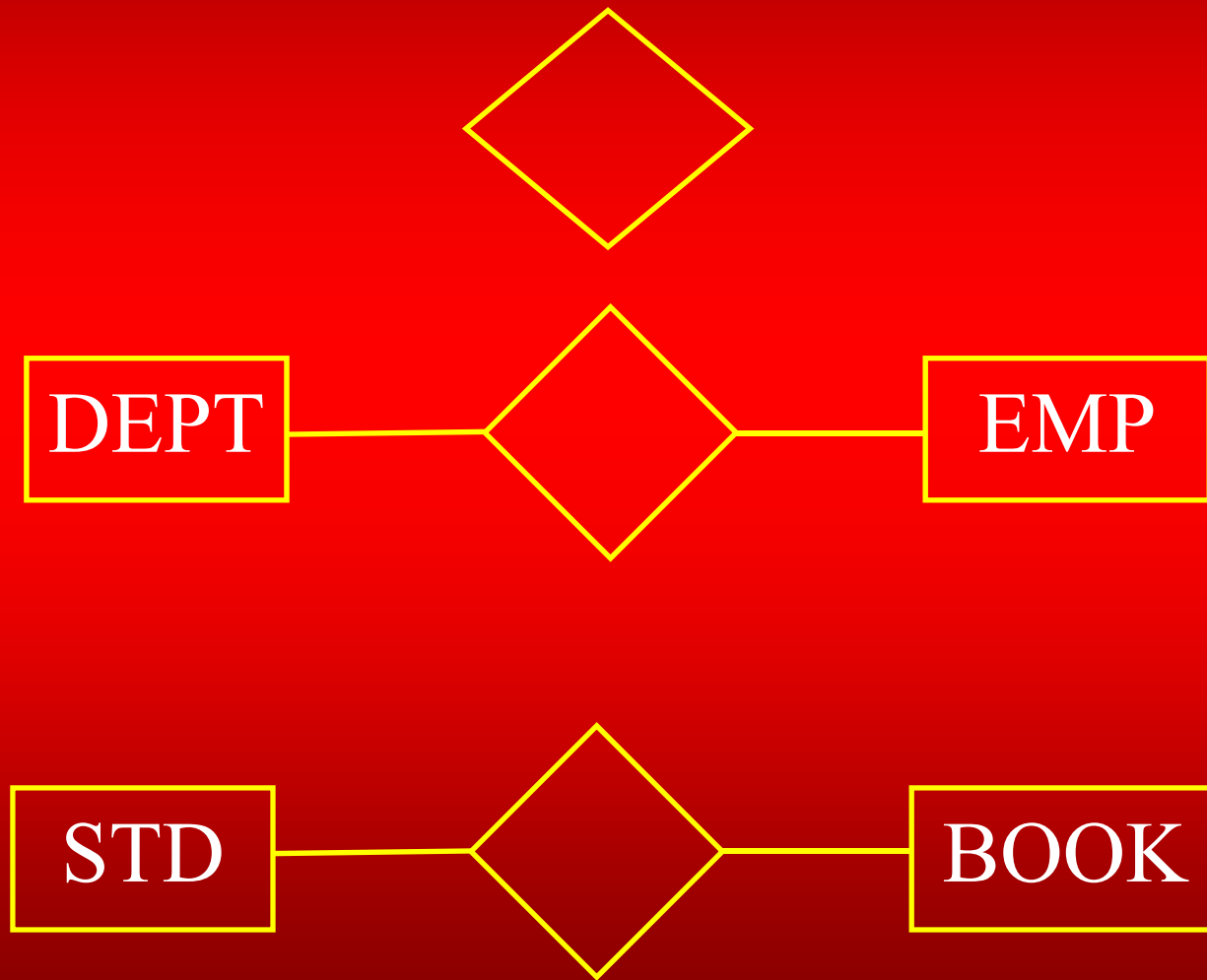
Naming Relationships

- Up to you
- If there is no proper name of the association in the system then participants' names or abbreviations are used

Naming

- STUDENT and CLASS have ENROLL relationship
- However, it can also be named as STD_CLS

Symbol for Relationships



Relationships

- Relationship type can be identified like an entity type
- A relationship type is an abstraction of a relationship

Relationships

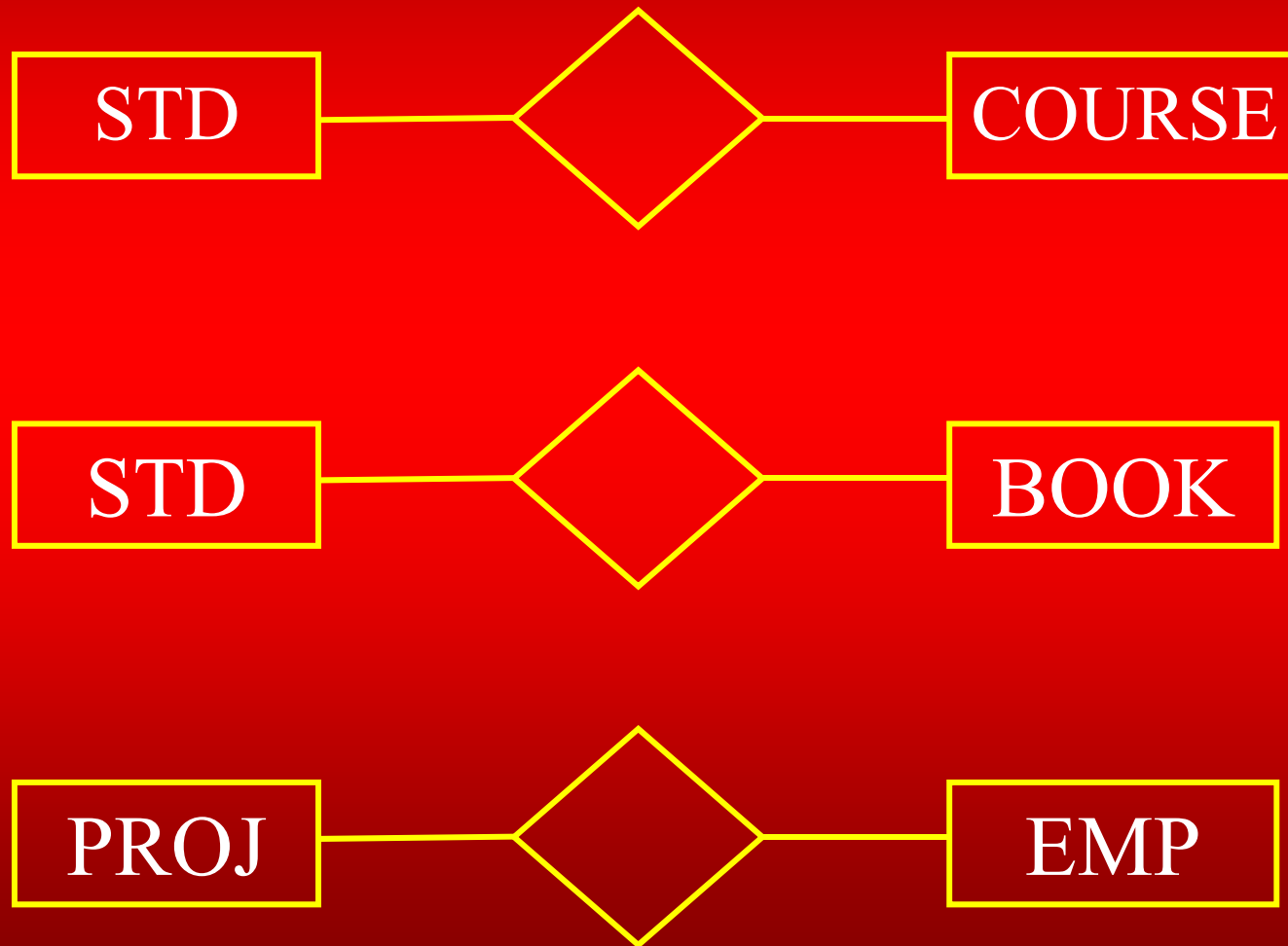
- Entities involved in a relationship are called its participants
- Types of the relationships can be established on the basis of participant ETs

Relationships Types

A Binary relationship is the one that links two entity types e.g.

STUDENT-CLASS

Binary Relationship Example



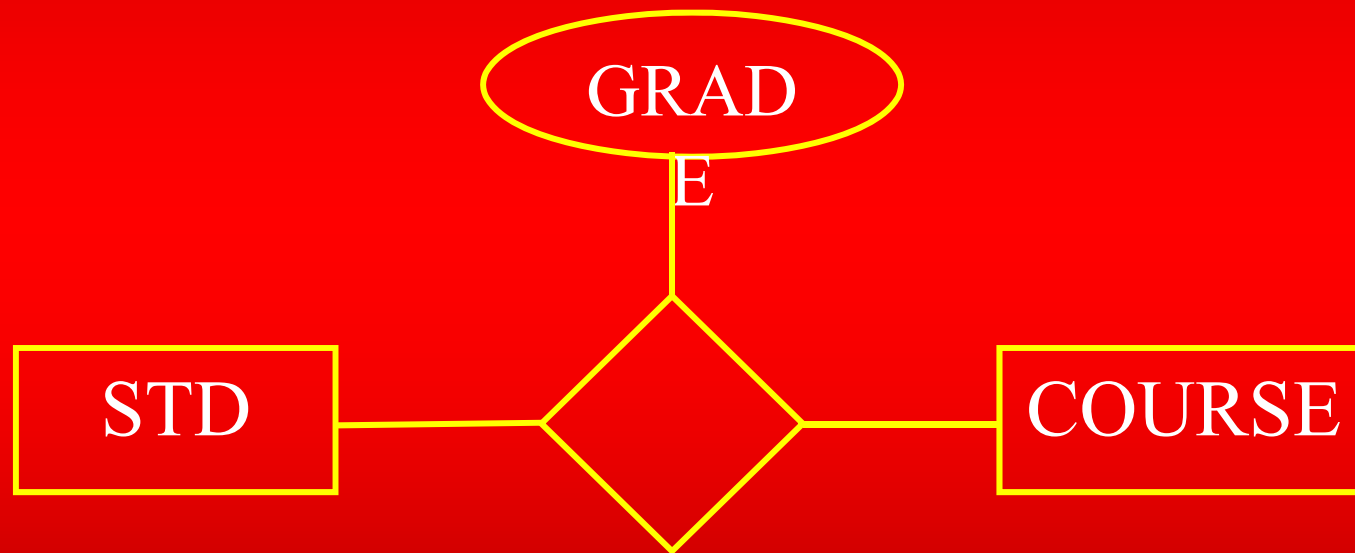
Binary Relationships

- May also have instances, that can be formally described in an ordered pair form
- $\{(S1001, OS), (S1020, DS), (S1002, DS), (S1058, NW)\}$

Attributes of the Rships

- The key
- The relationships can have their descriptive attributes
- Where to place

Attributes of Rships

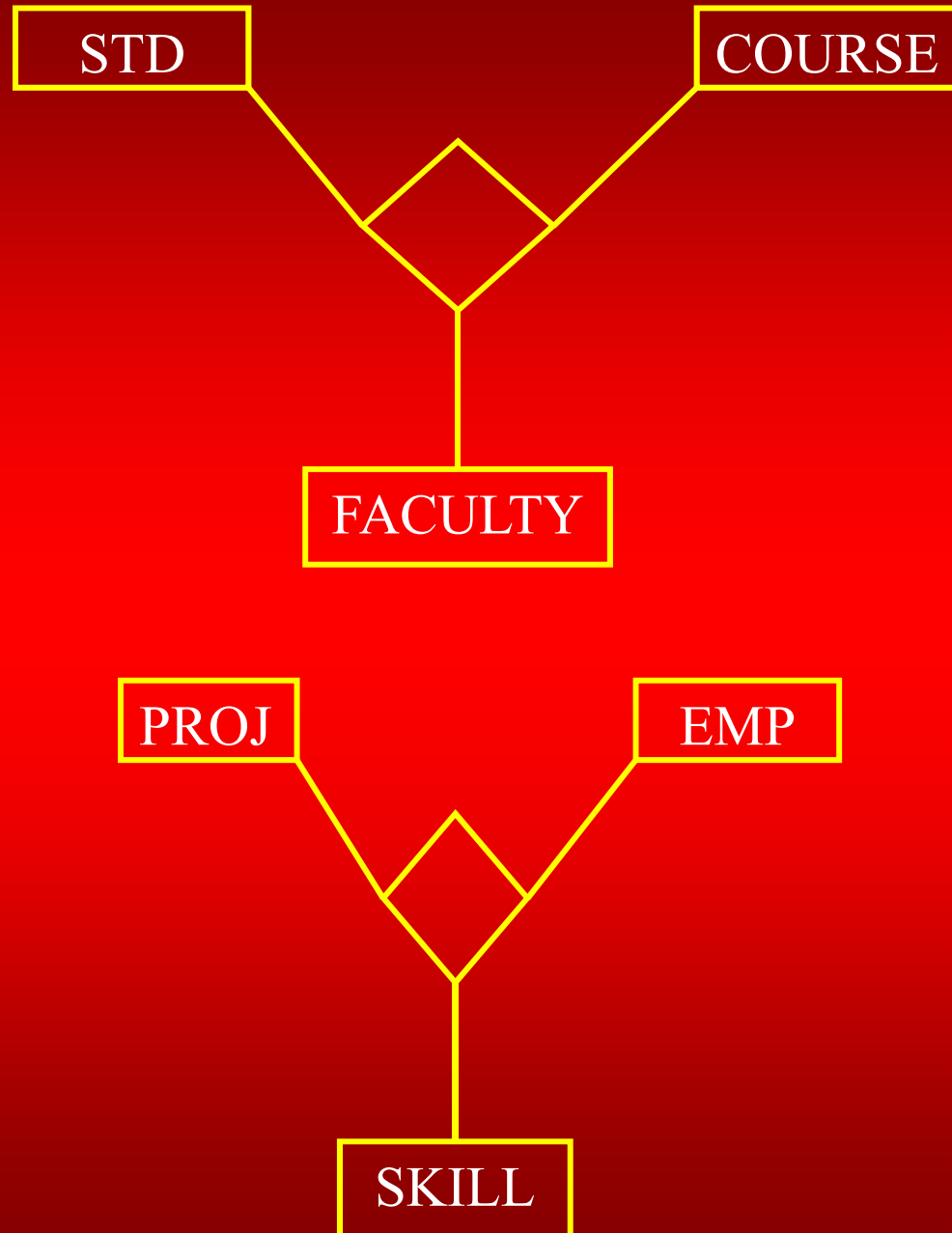


Ternary Relationships

➤ One that involves three entity types

➤ STUDENT-CLASS-FACULTY

Ternary Relationship Examples



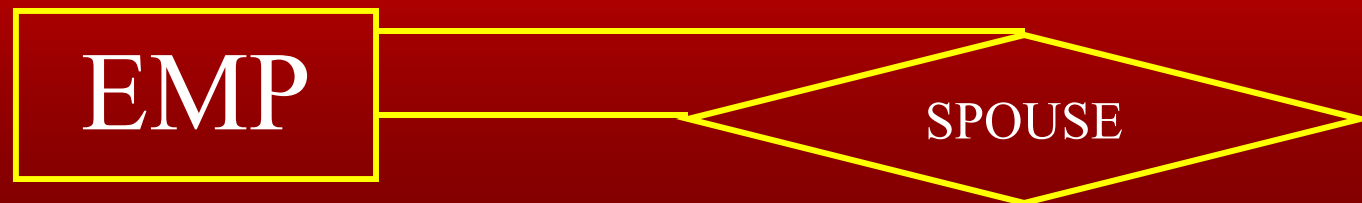
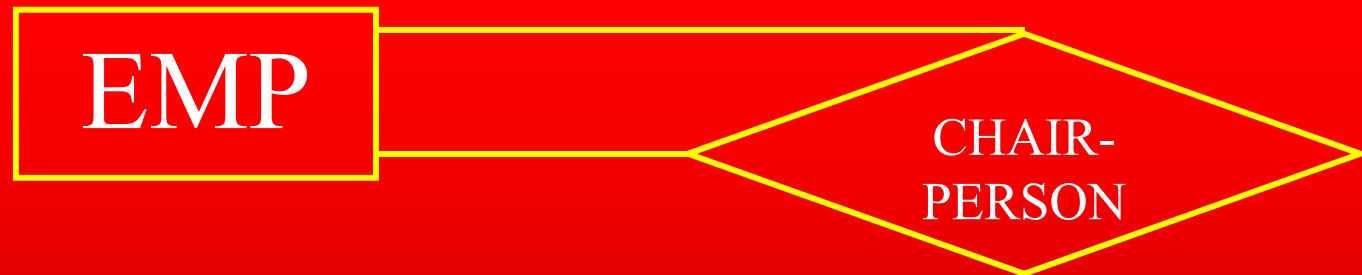
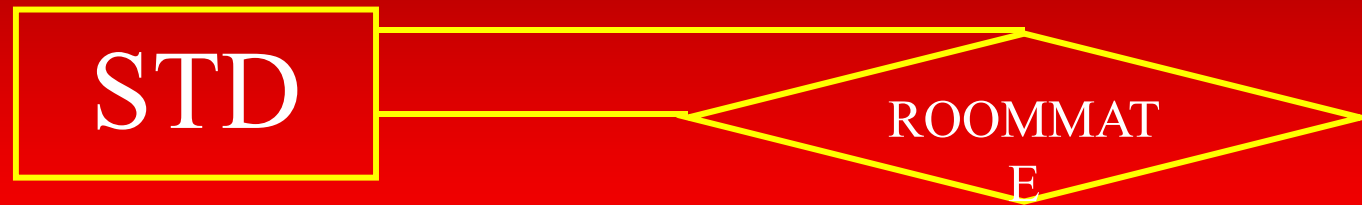
Ternary Relationships

- Instances in ordered triples
- Example {(S1013, MCS4, Adnan), (S1023, MCS3, Fasih)}

Unary Relationship

- An ET linked with itself, also called recursive relationship
- Example Roommate, where STUDENT is linked with STUDENT

Unary Relationship Examples



Cardinality of Rships

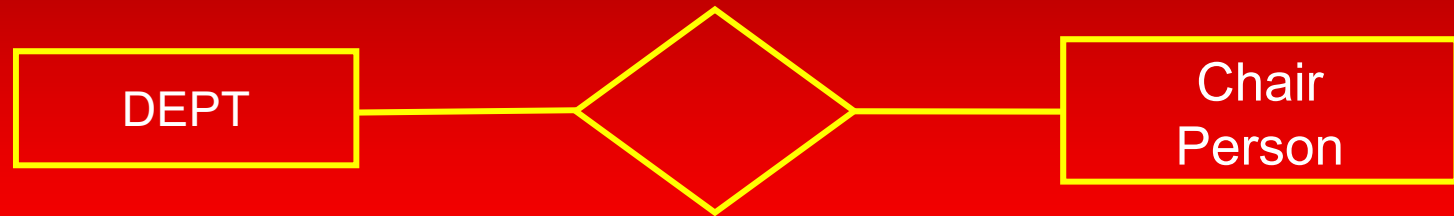
Number of instances of one entity type that can possibly be related to instances of other entity type

Types of Cardinalities

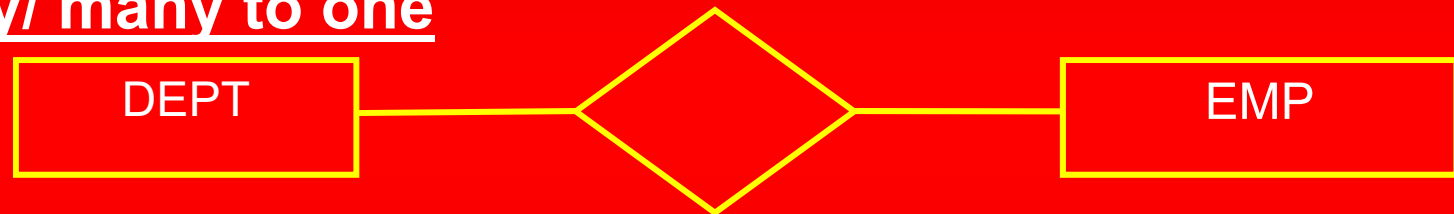
- One to one
- One to many
- Many to one
- Many to many

Types of Cardinalities

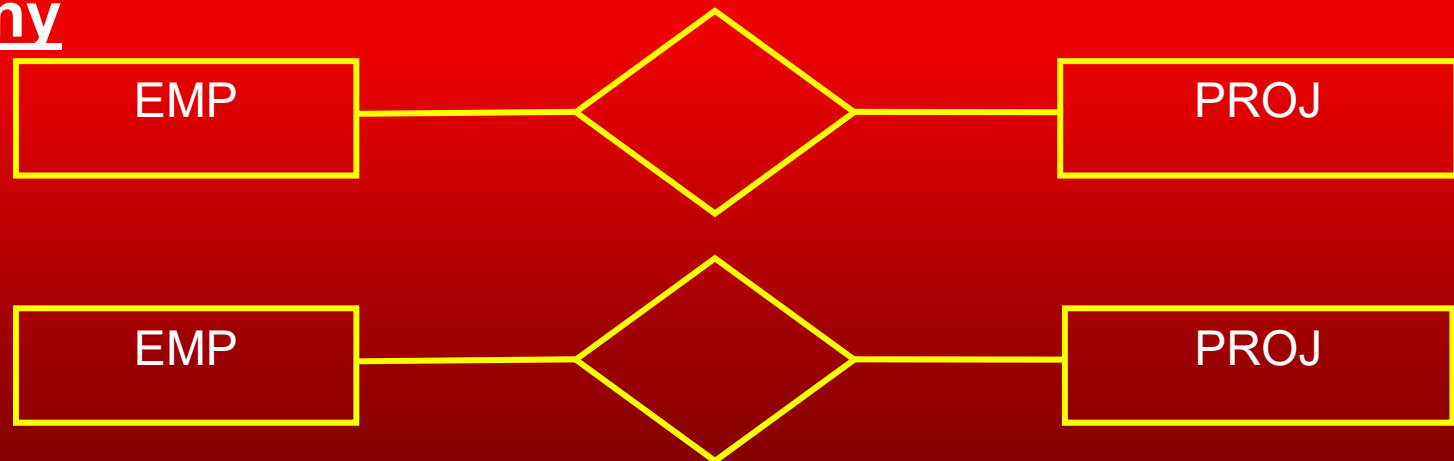
One to one



One to many/ many to one



Many to many



Database Management System

Lecture - 9

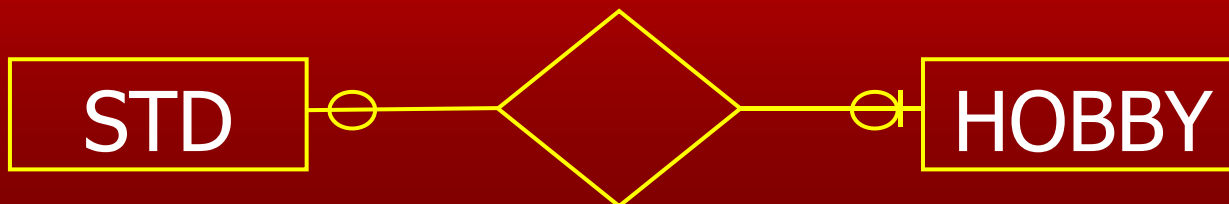
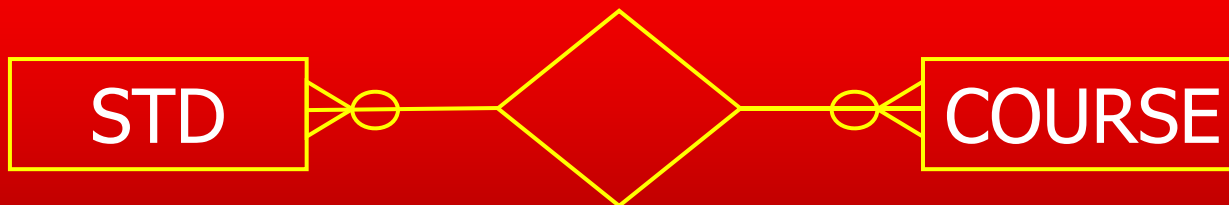
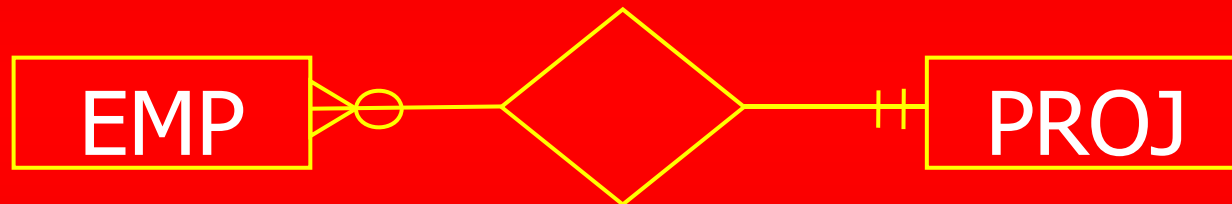
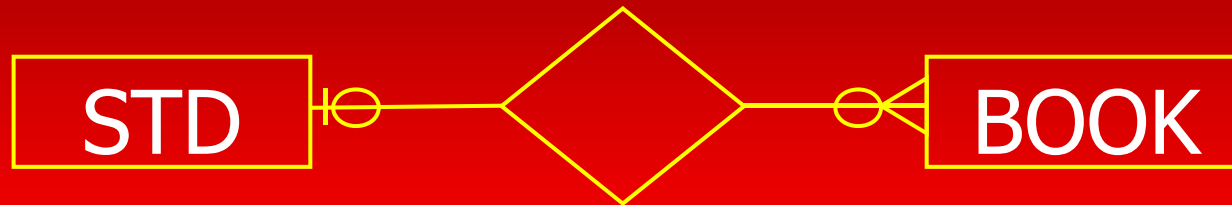
Database Management System

Lecture - 10

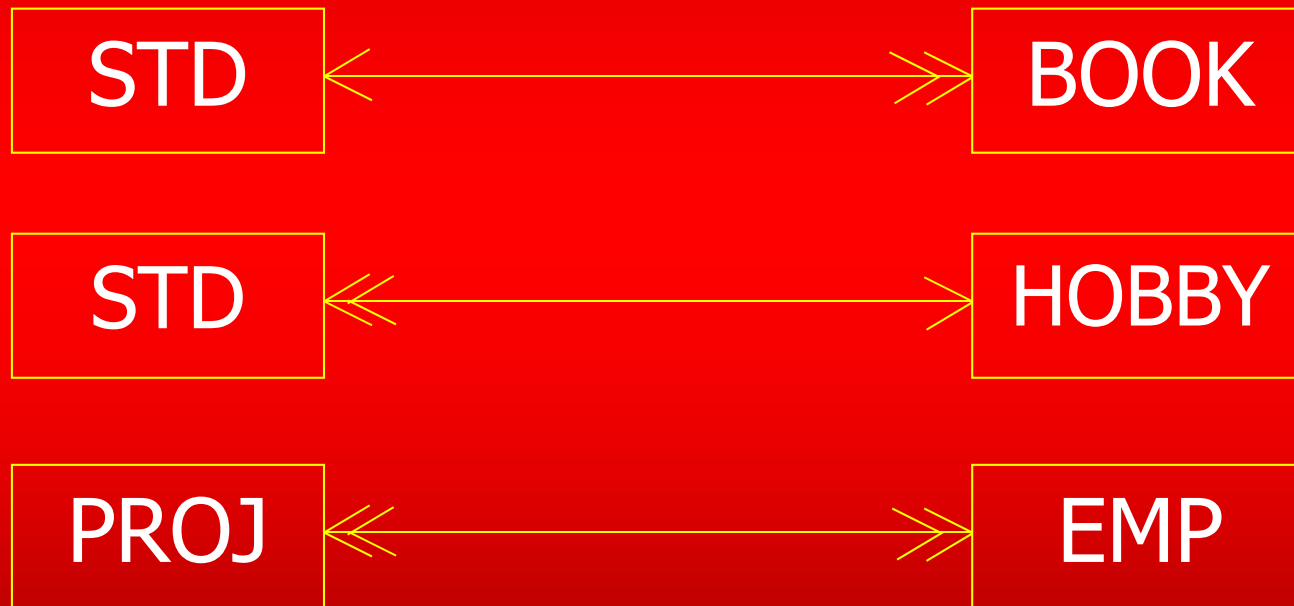
Minimum Cardinality

- Determines whether the link is compulsory or optional
- Important, since it effects the implementation

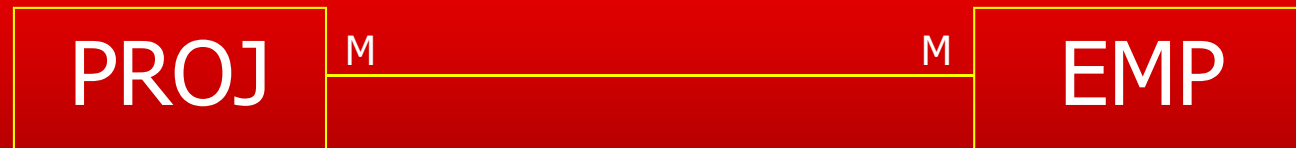
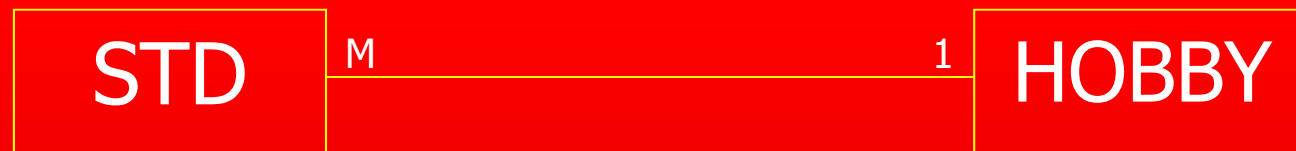
Cardinality Example



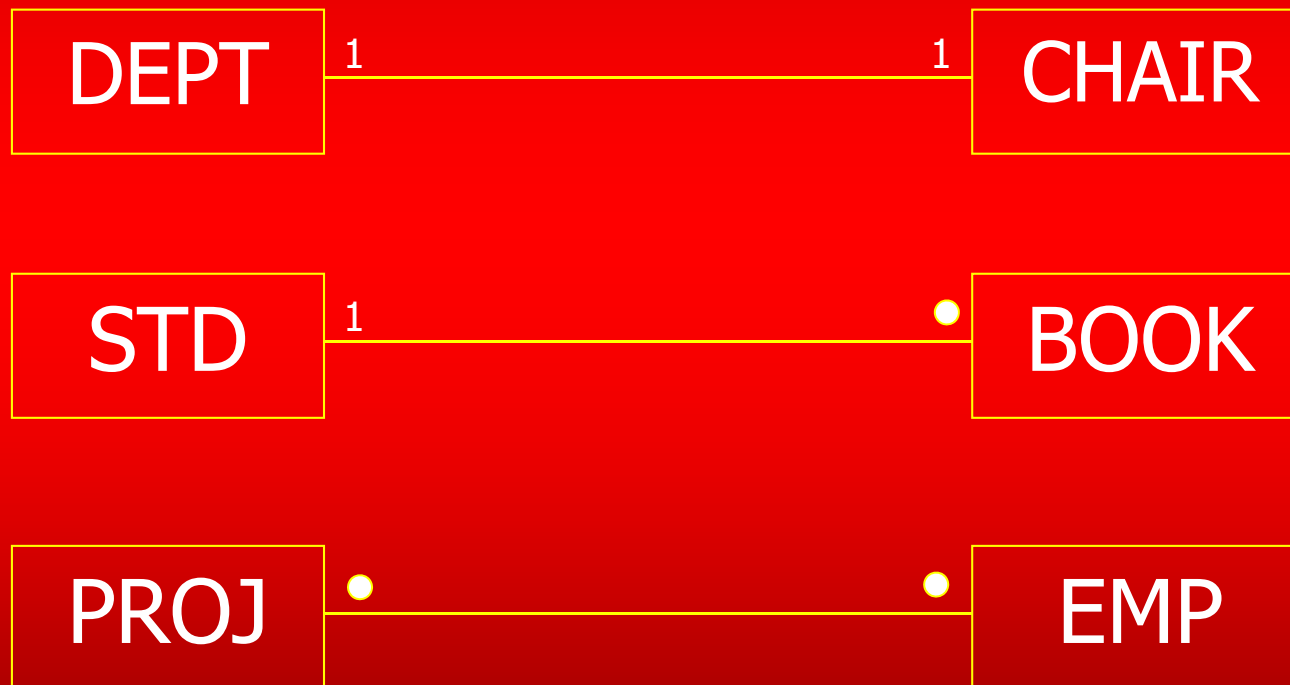
Other Notations



Other Notations



Other Notations



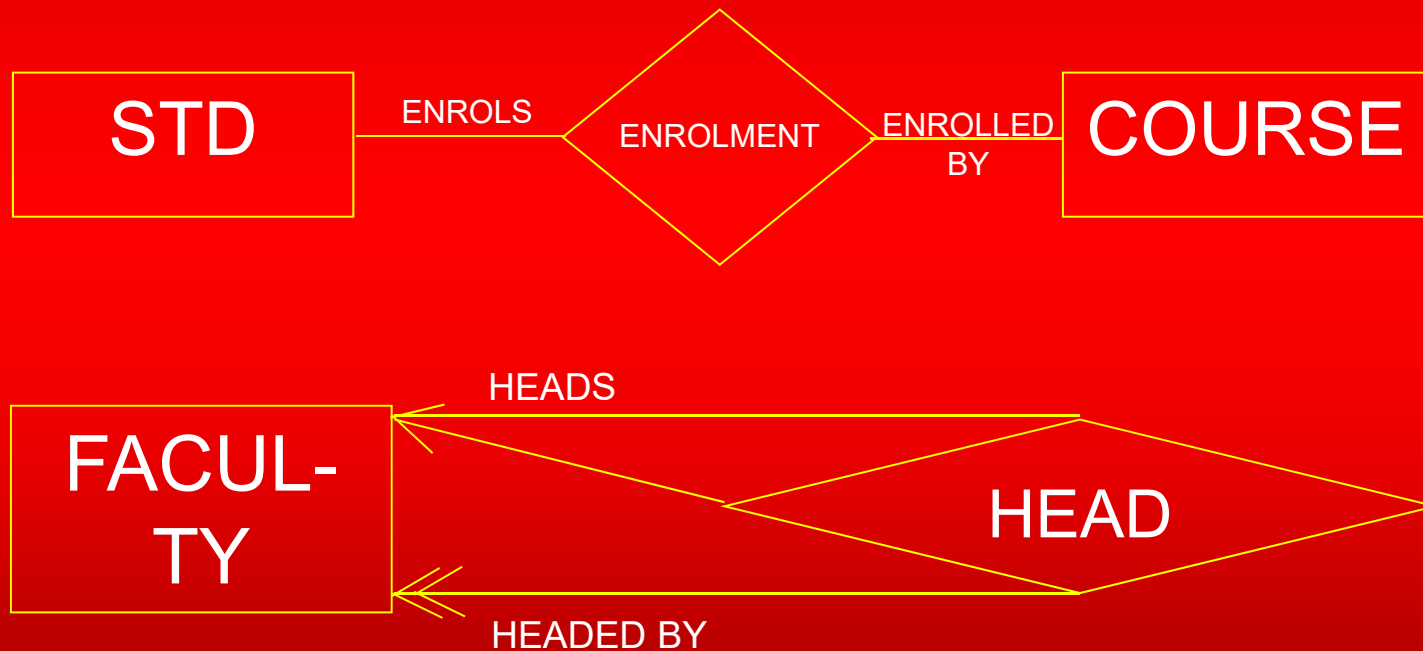
Roles in Relationships

- Determine the role ETs play in a relationship
- Most of the time is clear from the context, like in STD and COURSE relationship

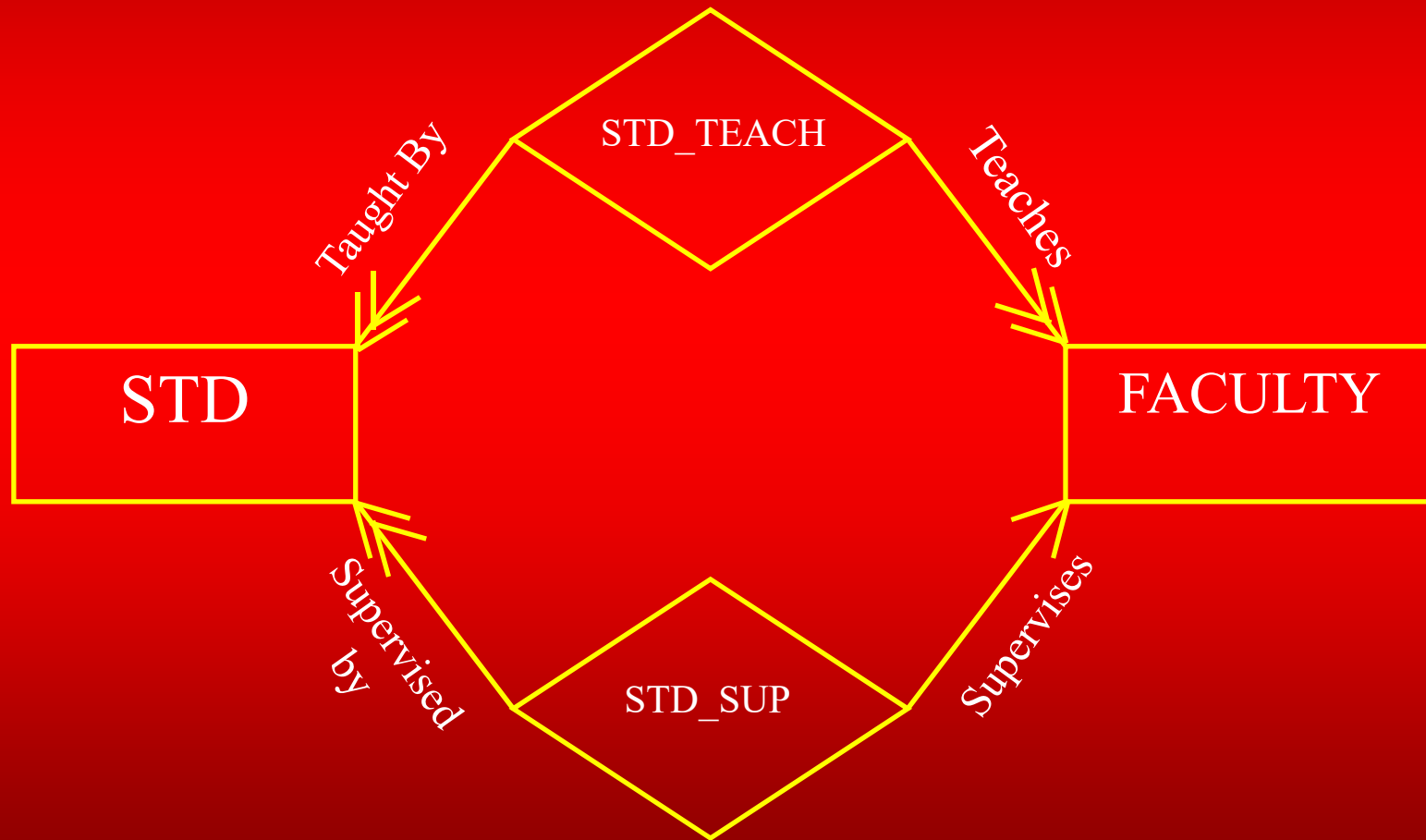
Roles in Relationships

- Two situation when they should be expressed explicitly
- A one to one relationship
- Two ETs having more than one relationship

Roles Examples



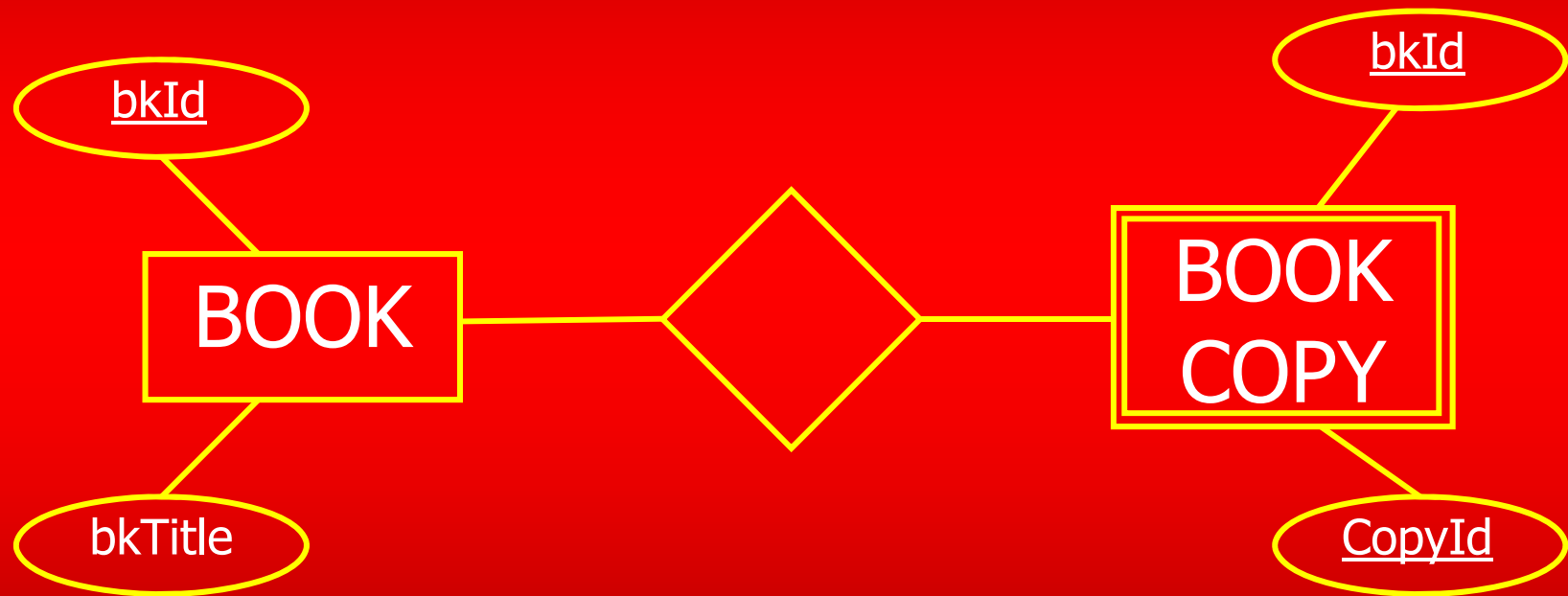
Roles Examples



Dependencies

- A type of constraint
- Existence dependency
- Identifier dependency
- Referential Dependency

Existence Dependency



Enhancements in E-R Data Model

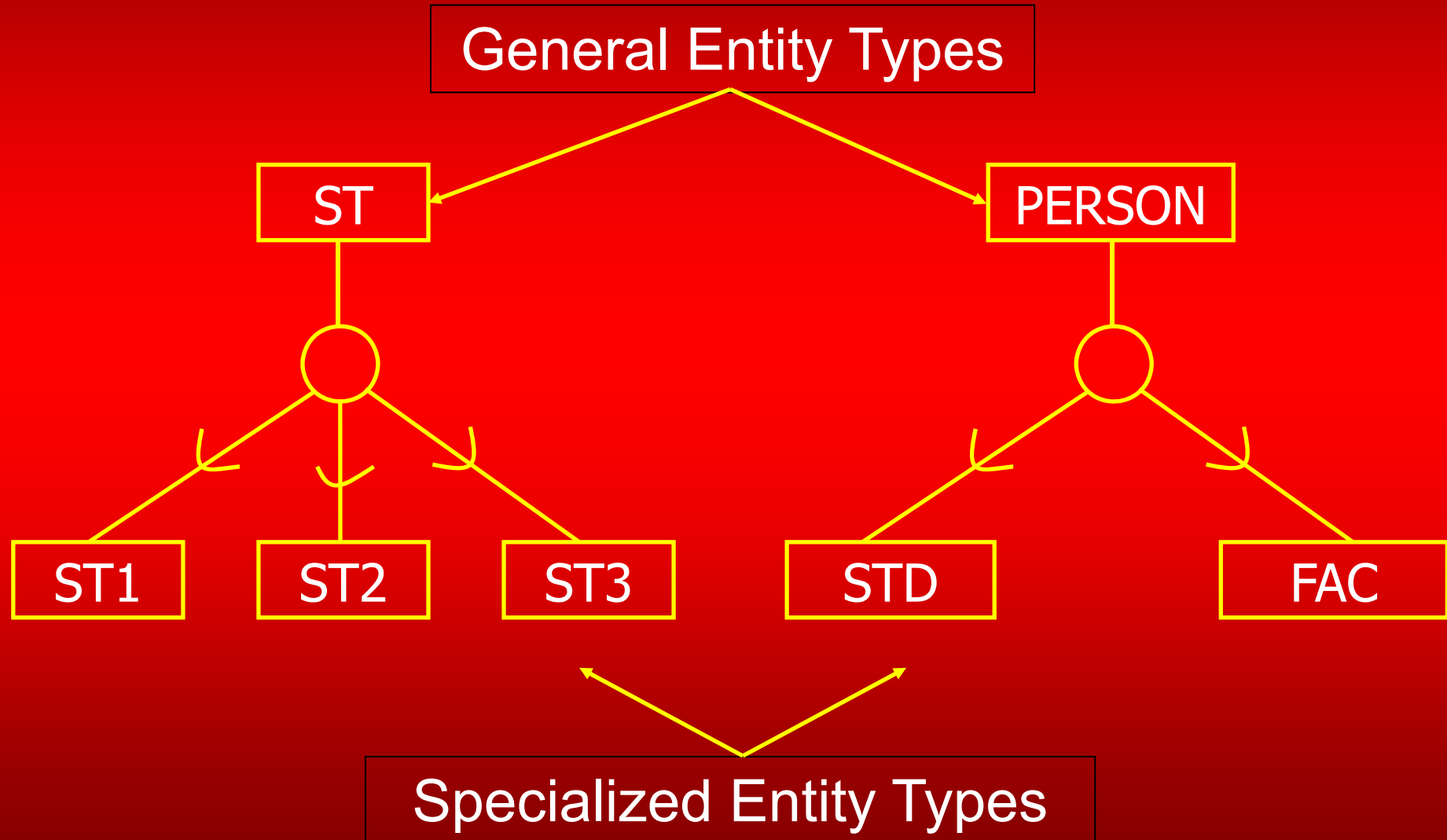
EE-R Data Model

- Different proposals
- Most common feature is representation of supertypes and subtypes
- A popular feature of Object Oriented paradigm

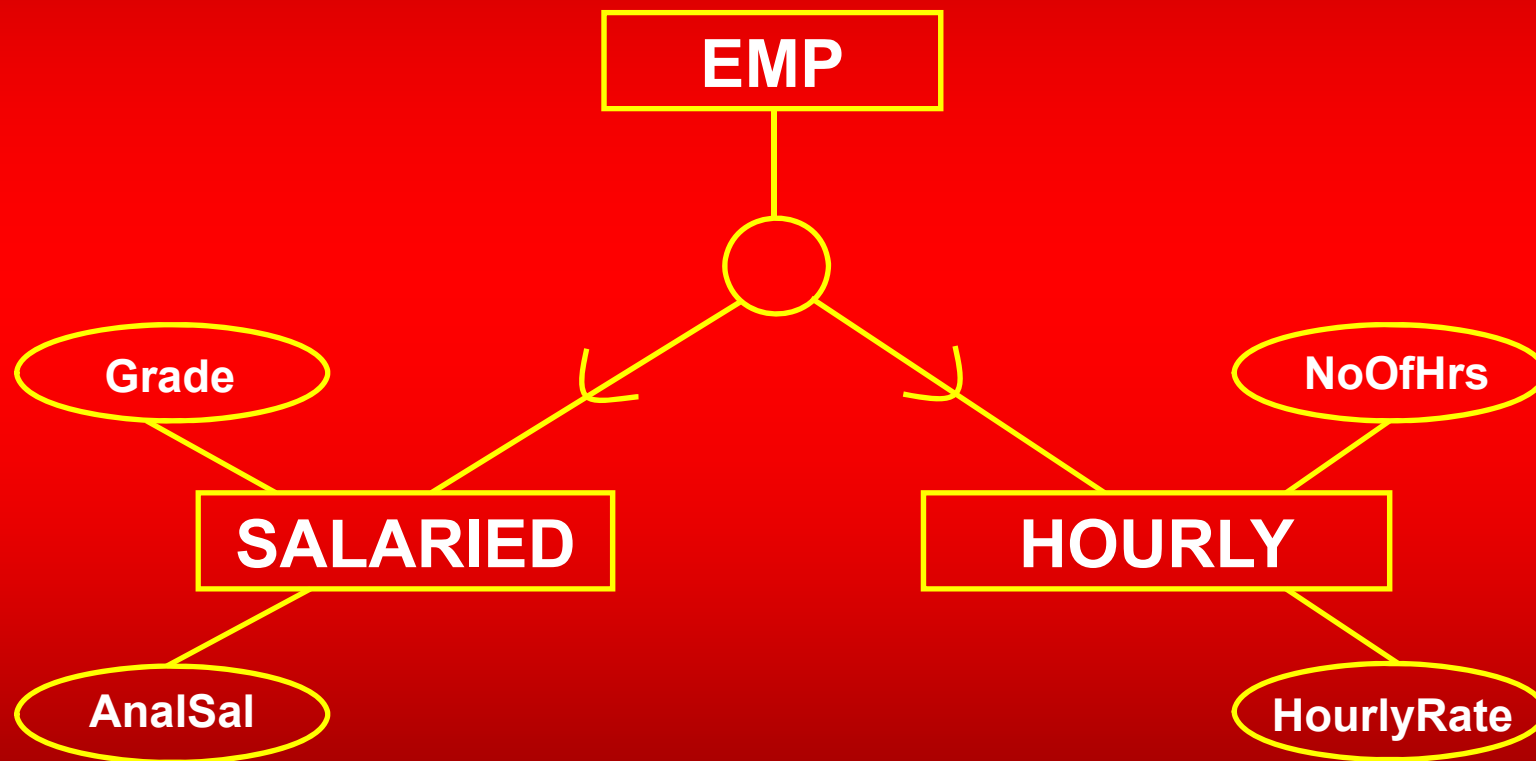
Super/Subtypes

- Also called generalization/specialization
- Supertype is called a General Entity type whereas subtypes are the specializations

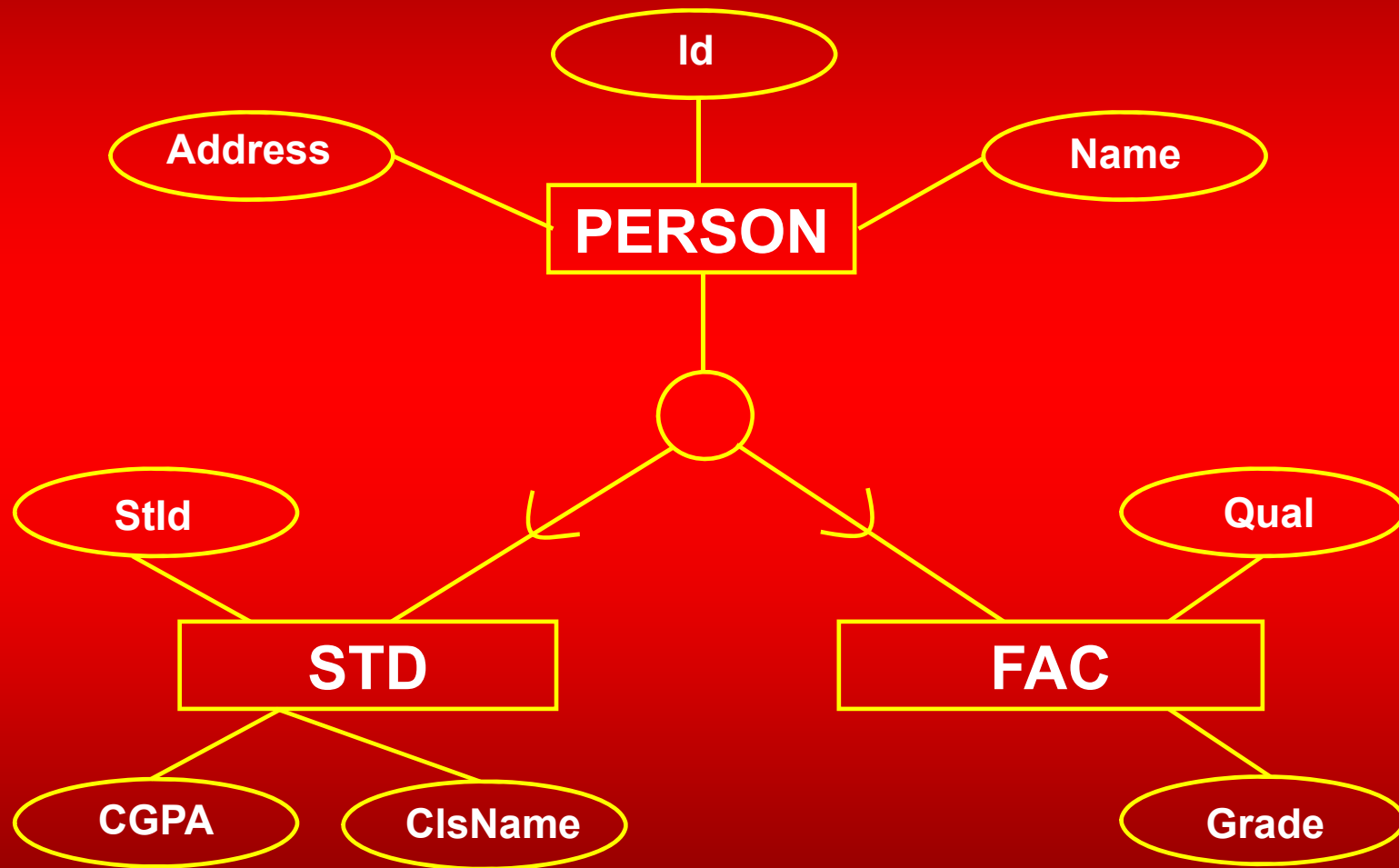
Super/Subtypes



Super/Subtypes



Super/Subtypes



Database Management System

Lecture - 10

Database Management System

Lecture - 11

Inheritance

- Gen/Spec relationship results inheritance between supertype and subtypes
- Subtypes inherit or get all the attributes of supertype

Super-Sub type Example

Super/Subtype Relationship

- Use/Advantage
- How to identify
 - General knowledge
 - Based on the attributes

Specifying Constraints

➤ Completeness constraint

- Total specialization rule
- Partial specialization

➤ Disjointness constraint

- Disjoint rule
- Overlap rule

Completeness Example 1

Completeness Example 2

Disjointness Example 1

Disjointness Example 2

Subtype Discriminator

- To determine the subtype of a supertype instance
- Place an attribute in the supertype whose value determines the subtype type

Disjoint Discriminator

Overlap Discriminator

Summary of the E-R Data Model

- A semantic data model
- Used for the conceptual database design
- Provides three main structures
- Entities, Relationships and attributes

- Entity reflects entity type, entity instance and entity set
- Entity types are classified into weak and regular entity types
- Regular ETs can exist independently and weak ETs can't

Symbols

Regular Entity Type



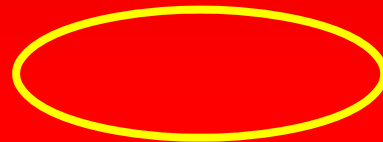
Weak Entity Type



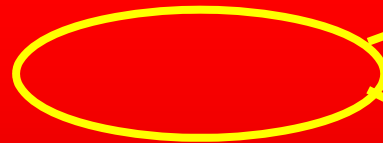
➤ Attribute represents a property or characteristic of an ET

➤ Attribute can be

Simple



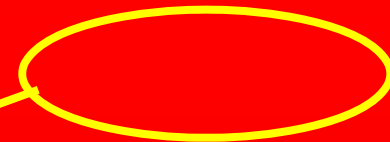
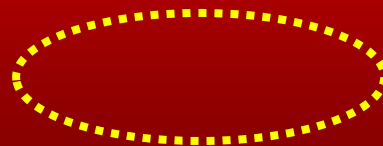
Composite



Multi-valued



Derived



➤ Keys are used for data access

➤ Keys types are

- Super key
- Candidate key
- Primary key
- Alternate key
- Secondary key

- Relationships are link or association between ETs
- Relationships can be unary, binary, ternary or n-ary
- Relationships are represented using diamonds linked with participant ETs

Unary, Binary and Ternary Rships

➤ Relationships have cardinalities

➤ Cardinalities can be

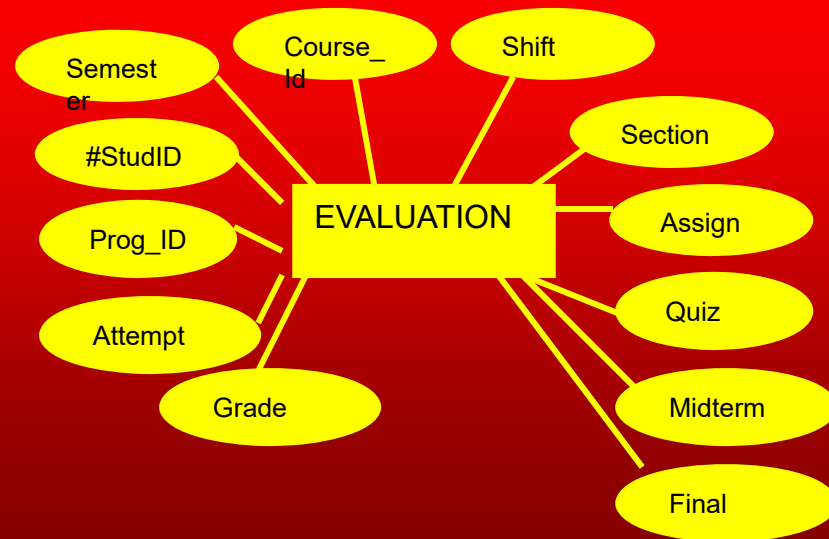
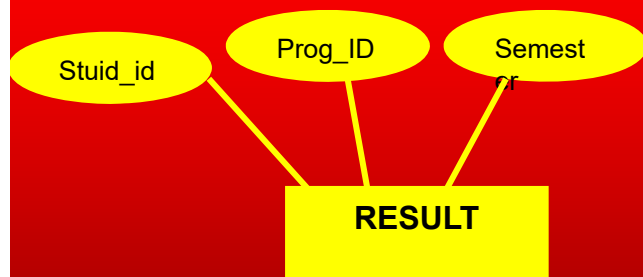
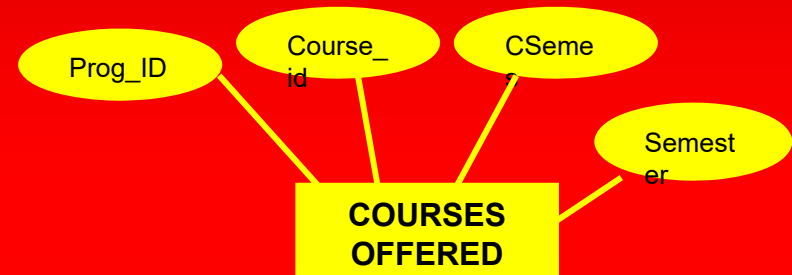
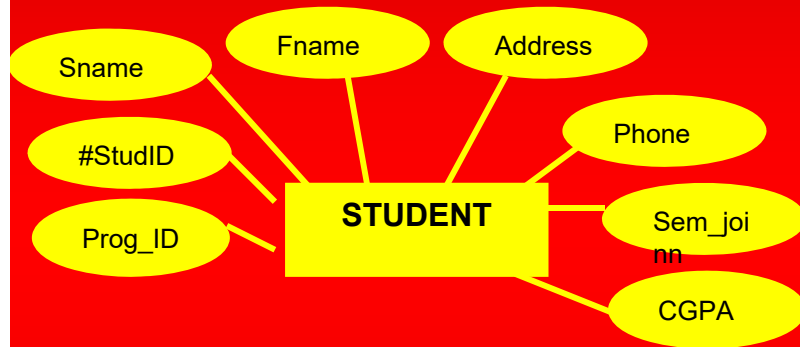
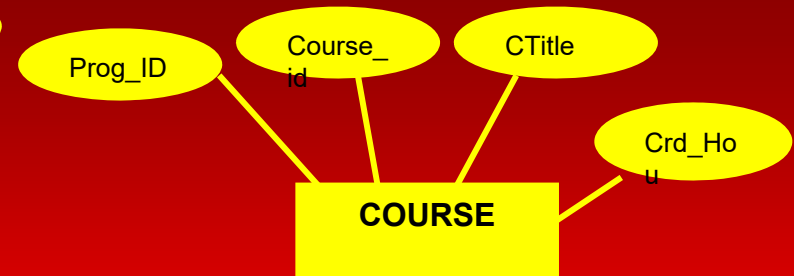
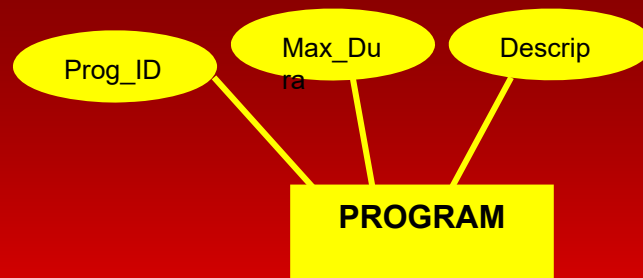
- One to one
- One to many
- Many to one
- Many to many

- Minimum and maximum cardinalities are also specified in the E-R data model
- Different notations are used in E-R data model to represent cardinalities

Cardinalities slide

- Enhanced E-R data model
adds the additional features in
the E-R data model
- Supertype/subtype relationship
also called gen/spec includes
inheritance





Database Management System

Lecture - 11

Database Management System

Lecture - 12

Practice Session

- Application of E-R data model
- We study an example scenario and apply the design phases we have studied so far
- Data flow diagram and E-R database design

Practice Session

- Lets study Examination System of an Educational Institution
- During initial study phase, we identified following major points

Preliminary Study

1. Students are enrolled in programmes/degrees
2. Programmes comprise of different courses
3. Different programmes are offered at the start of semester

Preliminary Study

4. Students register courses of different programmes at the start of every semester
5. After valid registration students attend the classes
6. Teachers submit results during and at the end of semester

Preliminary Study

7. Subjects' grade points are calculated on the basis of results submitted
8. Semester GPA is calculated on the basis of subjects' GPs

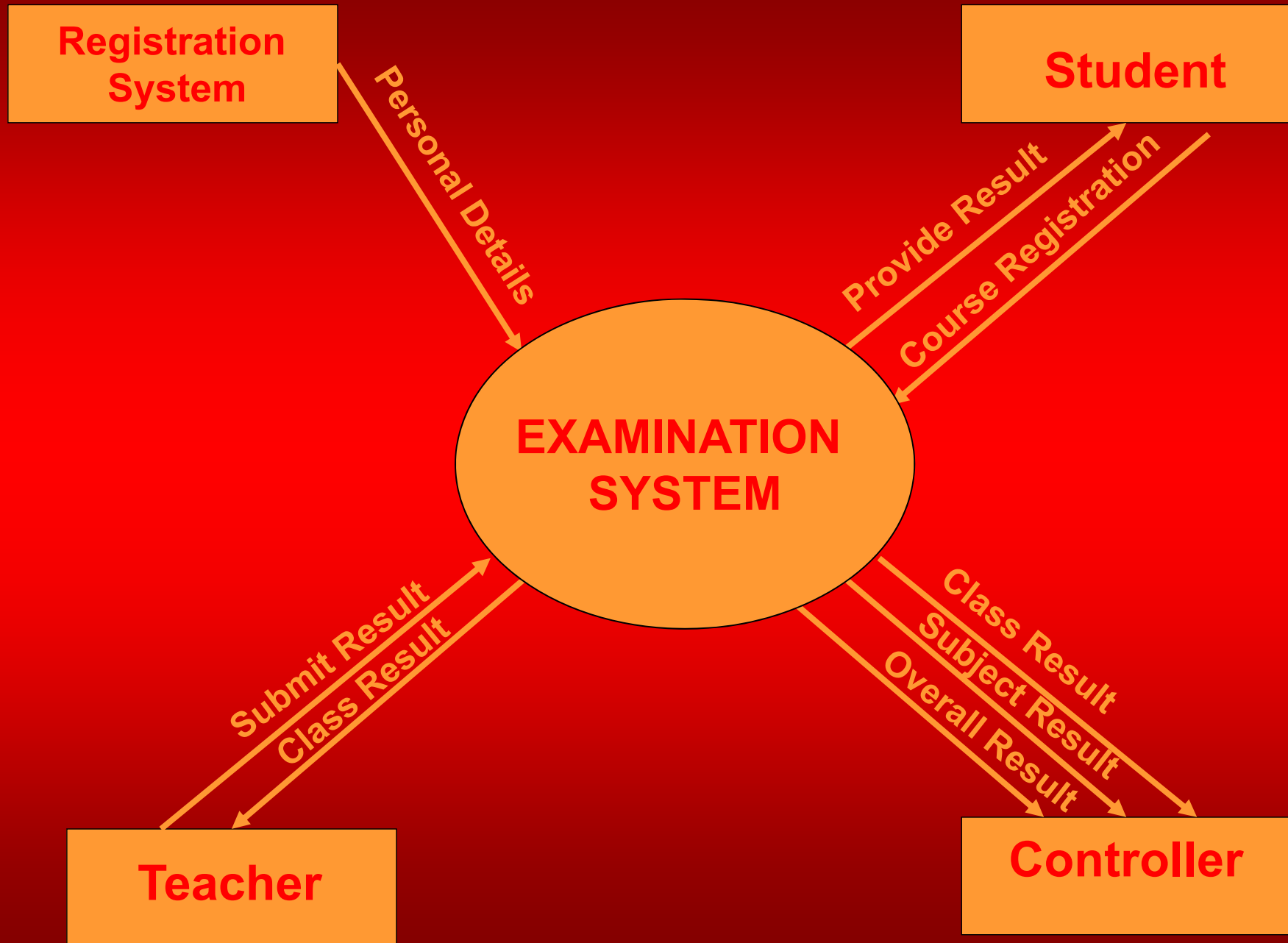
Preliminary Study

8. Cumulative GPA is calculated on the basis of semesters' GPAs
9. Courses, results, students data is stored in files

Outputs Required

- Teachers and controller need class list or attendance sheet, class result; subject and overall
- Students need transcripts, semester result card, subject result

Data Flow Diagram

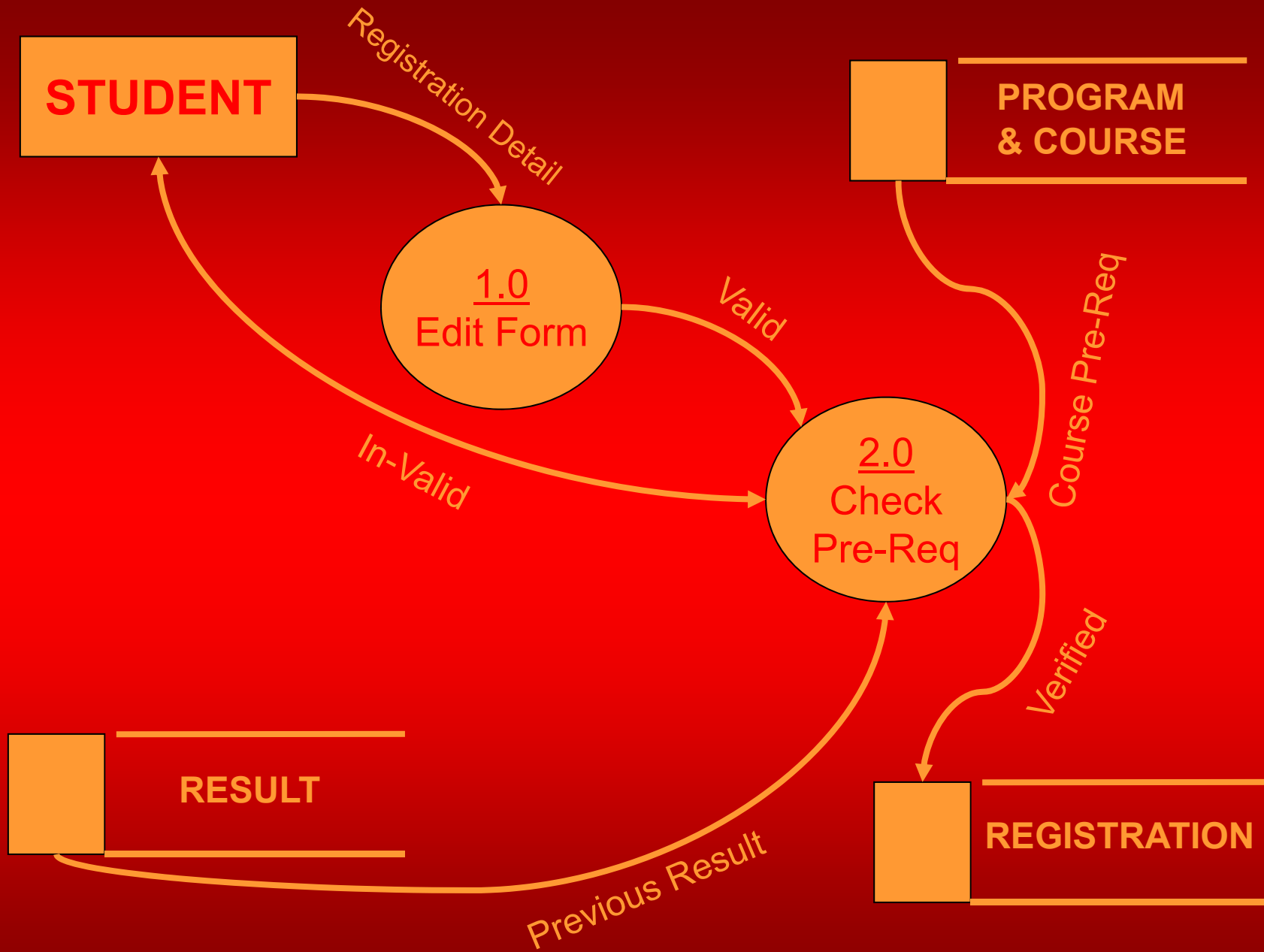


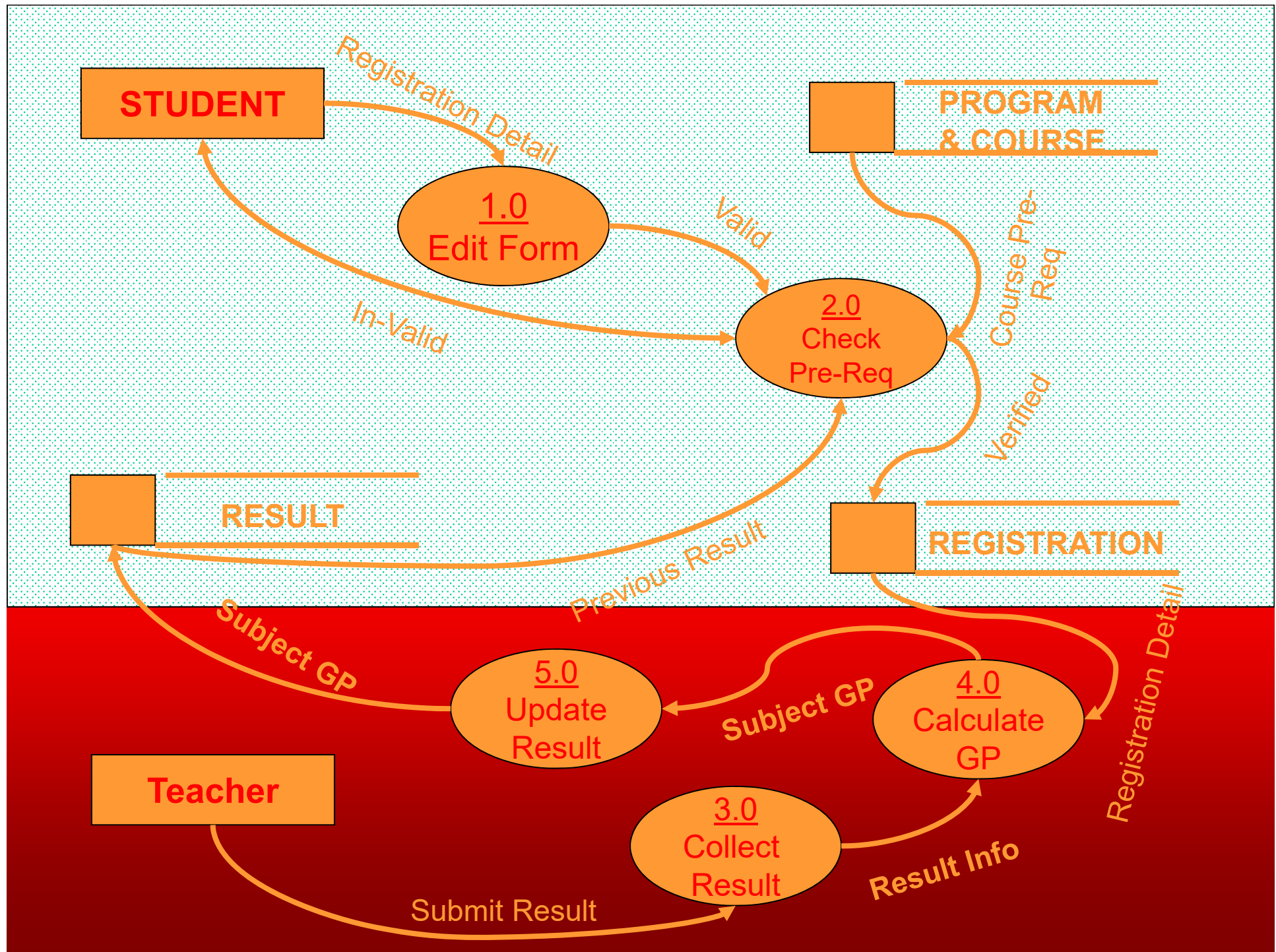
MAIN MODULES

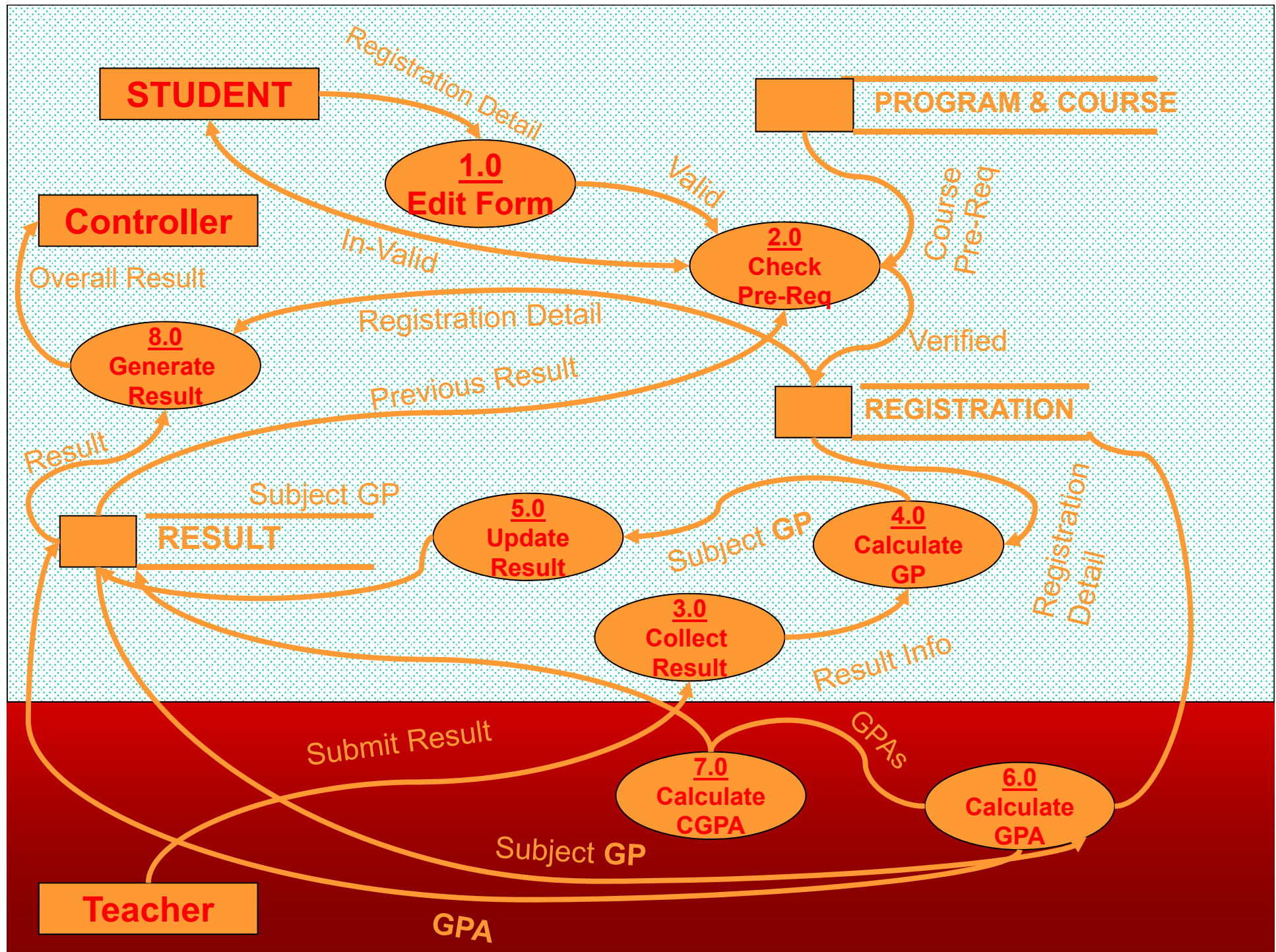
1. Subject registration

2. Result submission

3. Result calculation







Cross Reference Matrix

➤ Reports vs Items

Analysis Phase

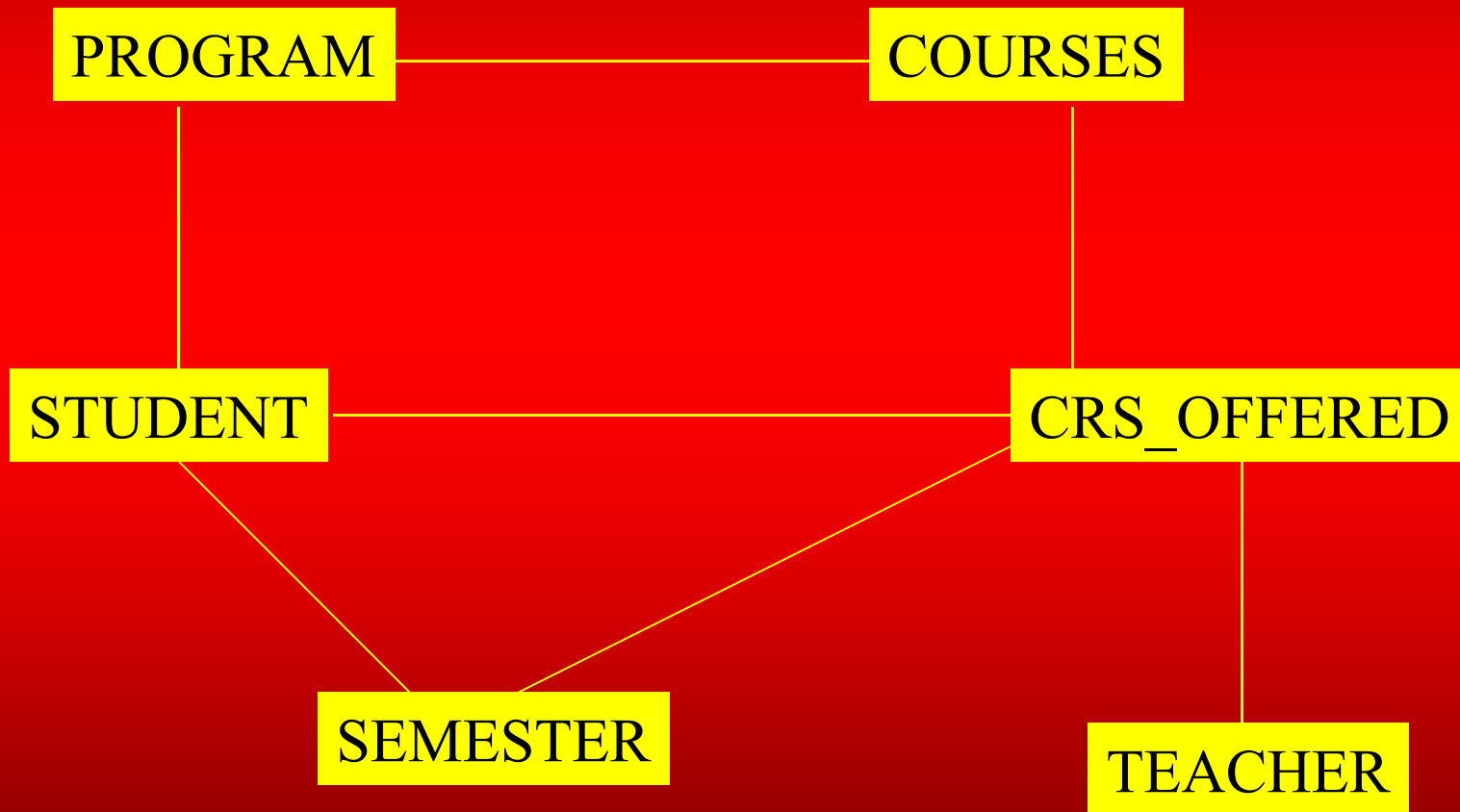
Candidate Entity Types

➤ From DFD

- Student
- Controller
- Courses
- Teachers
- Courses Offered
- Programmes
- Registration
- Results
- Semester

Support your ETs by
cross reference matrix

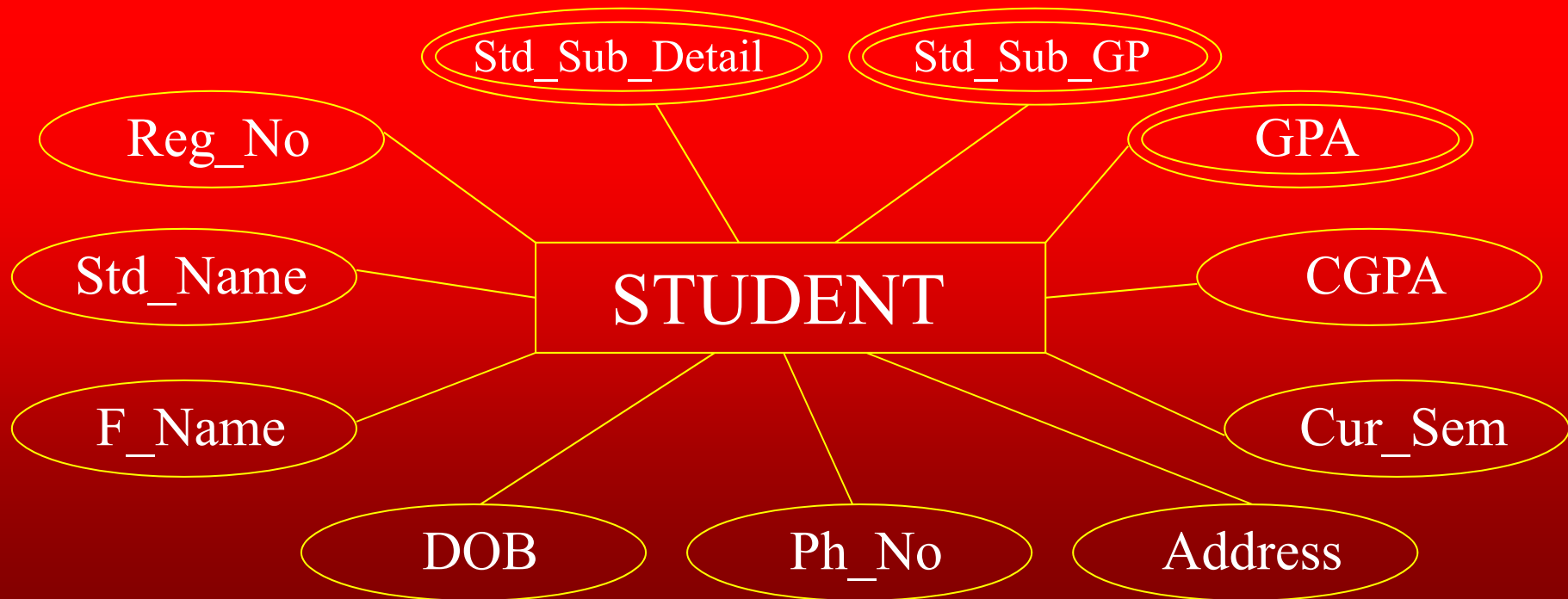
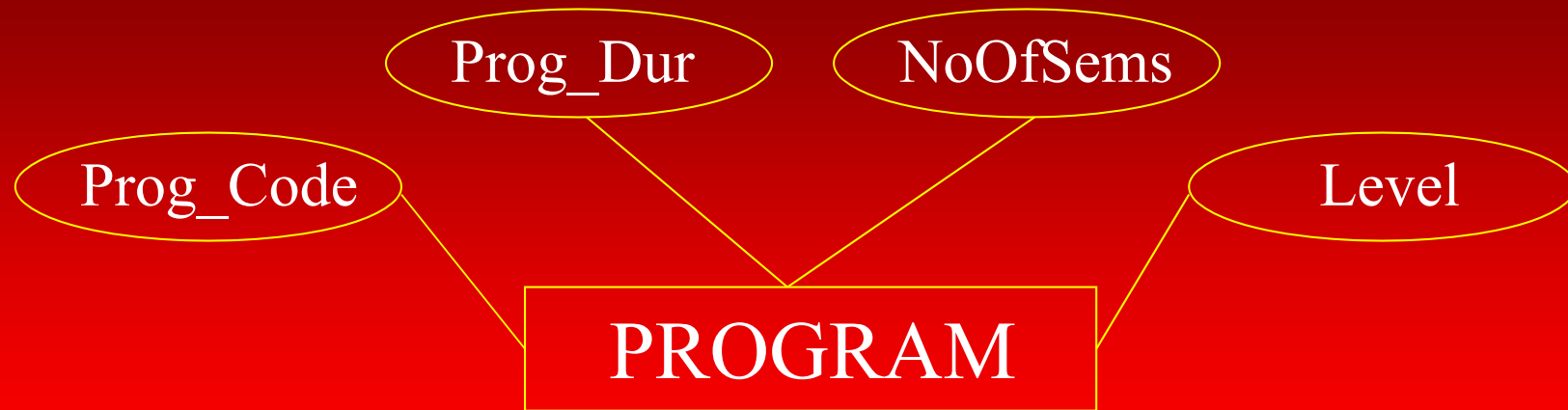
Candidate Entity Types

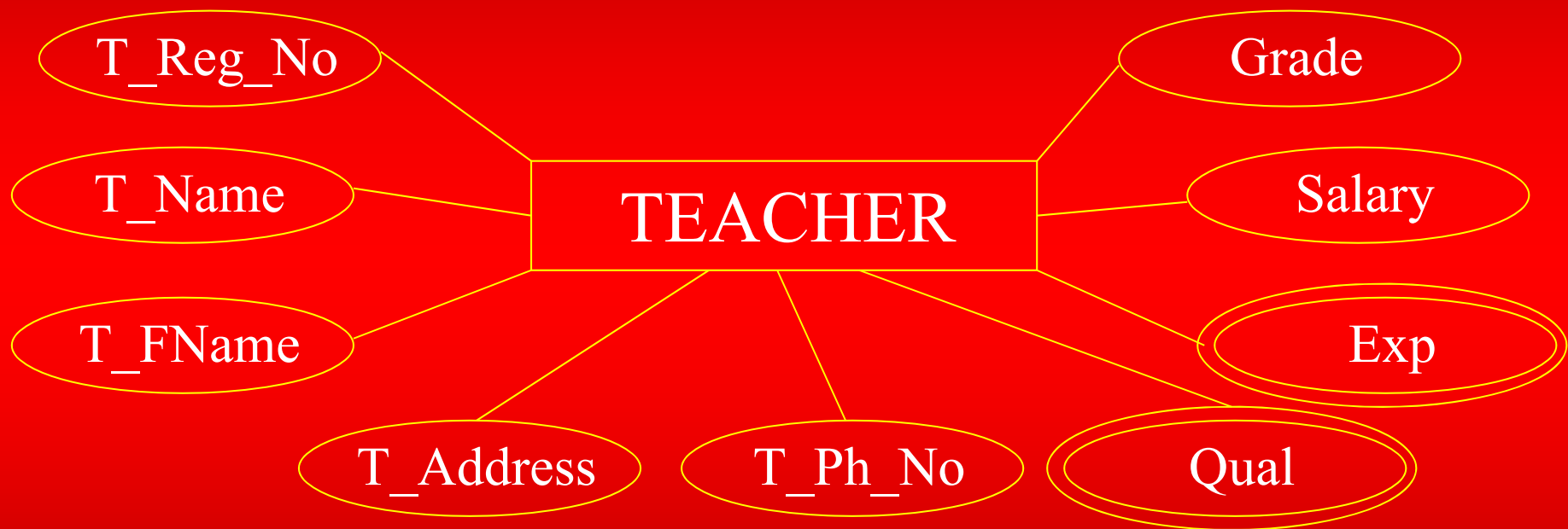


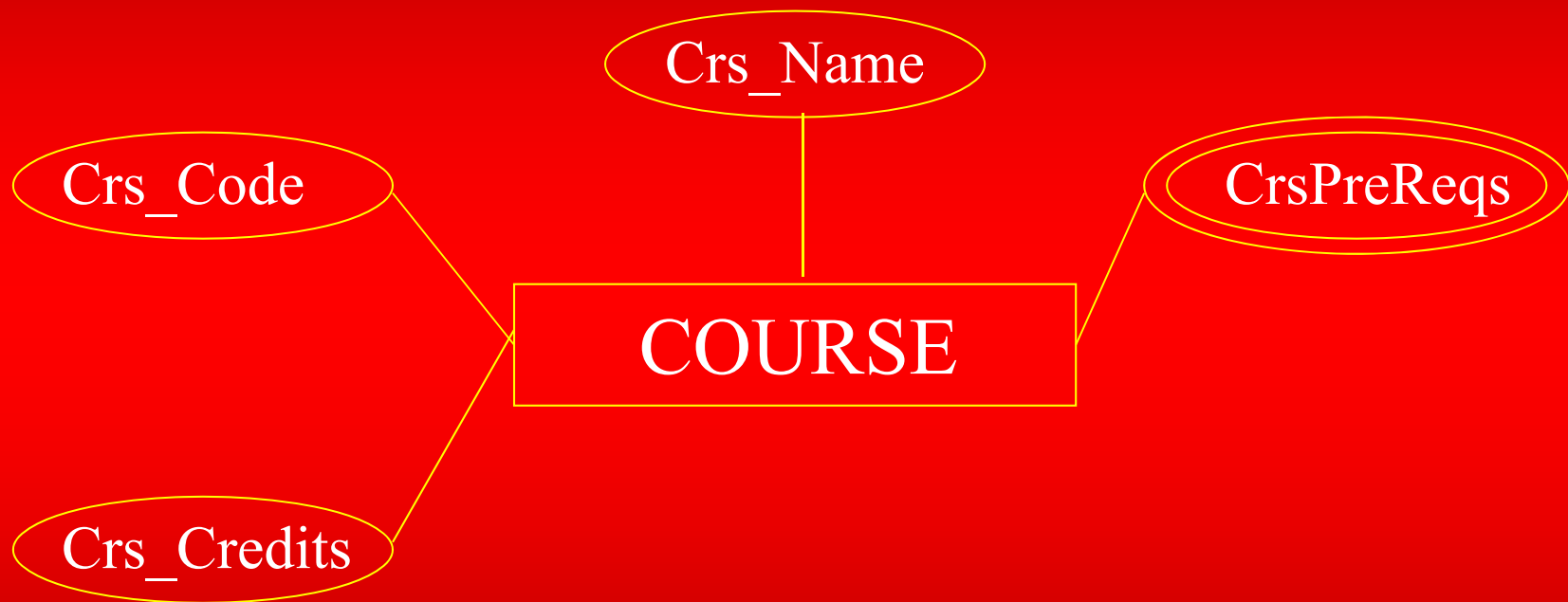
Database Management System

Lecture - 13

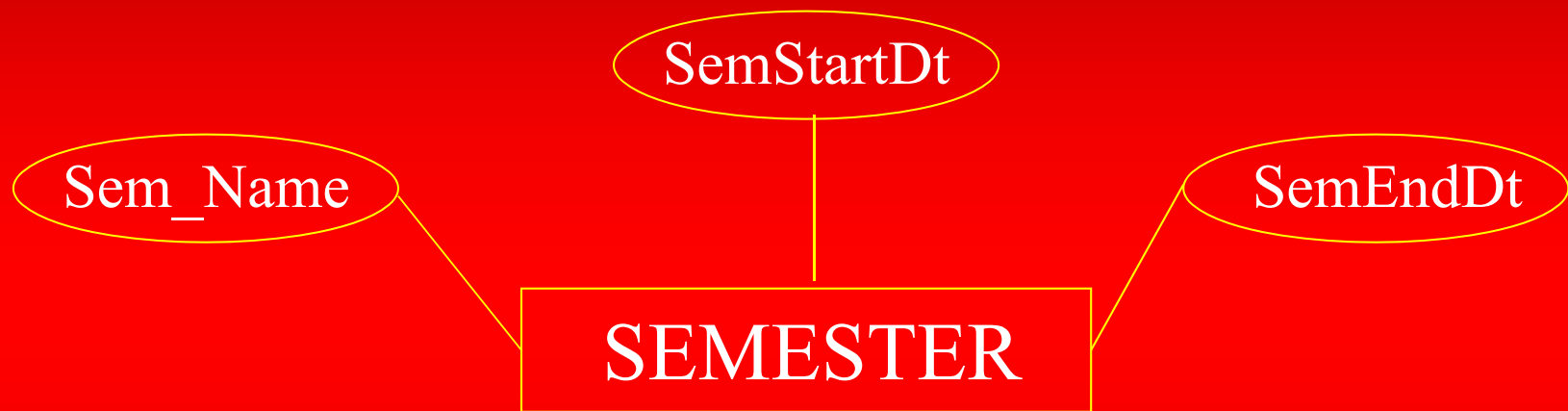
E-R Diagram

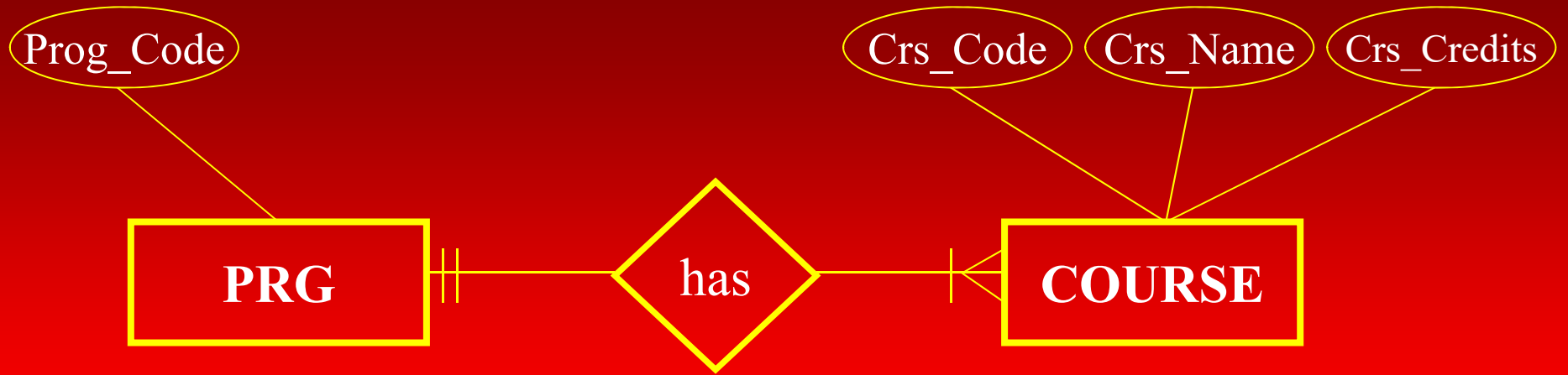


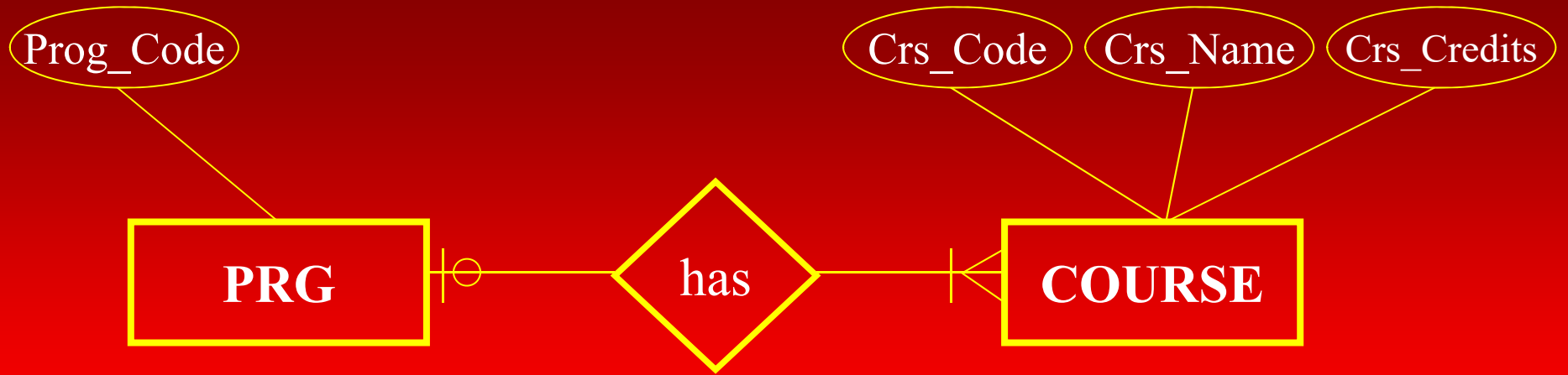


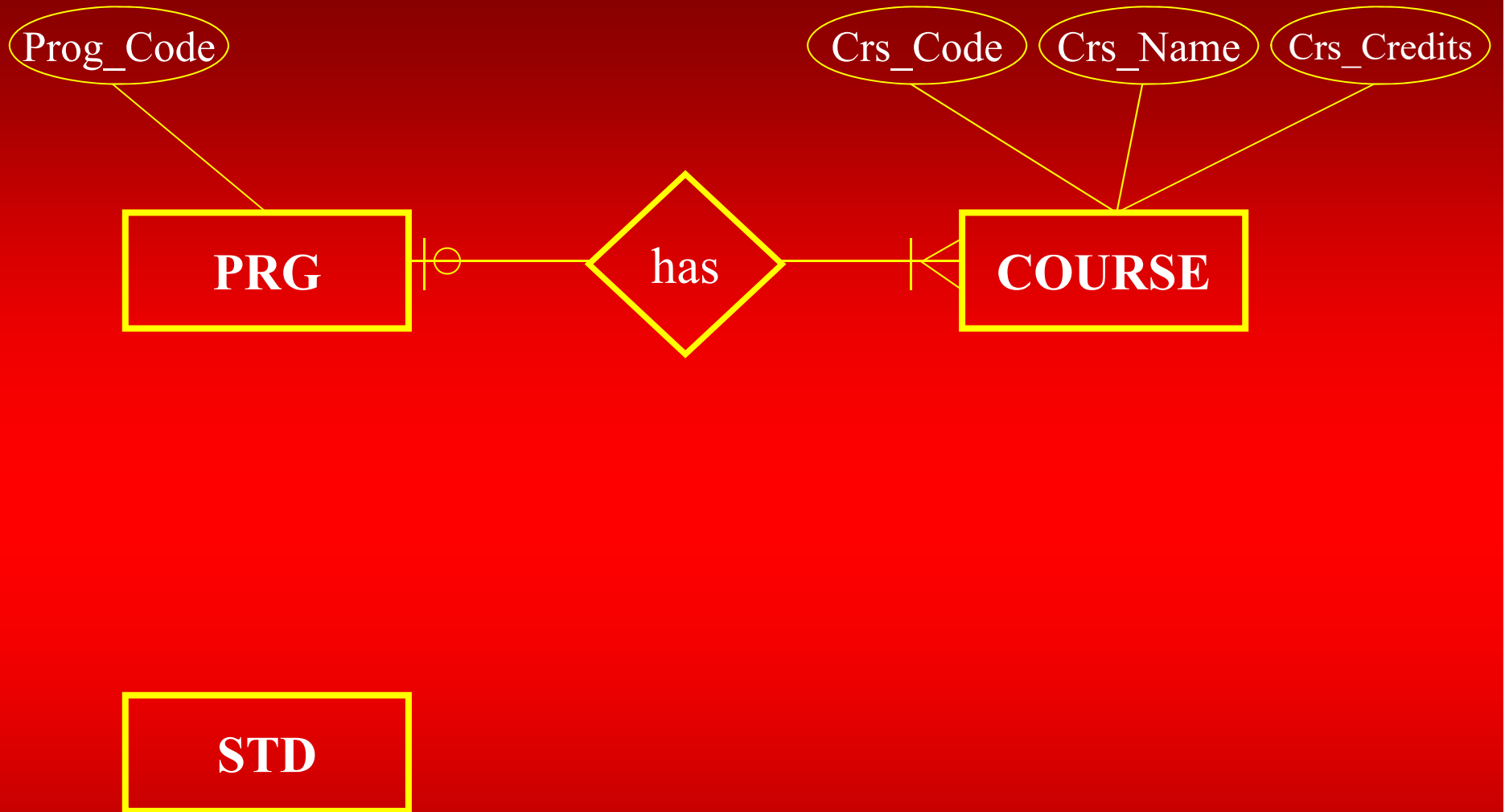


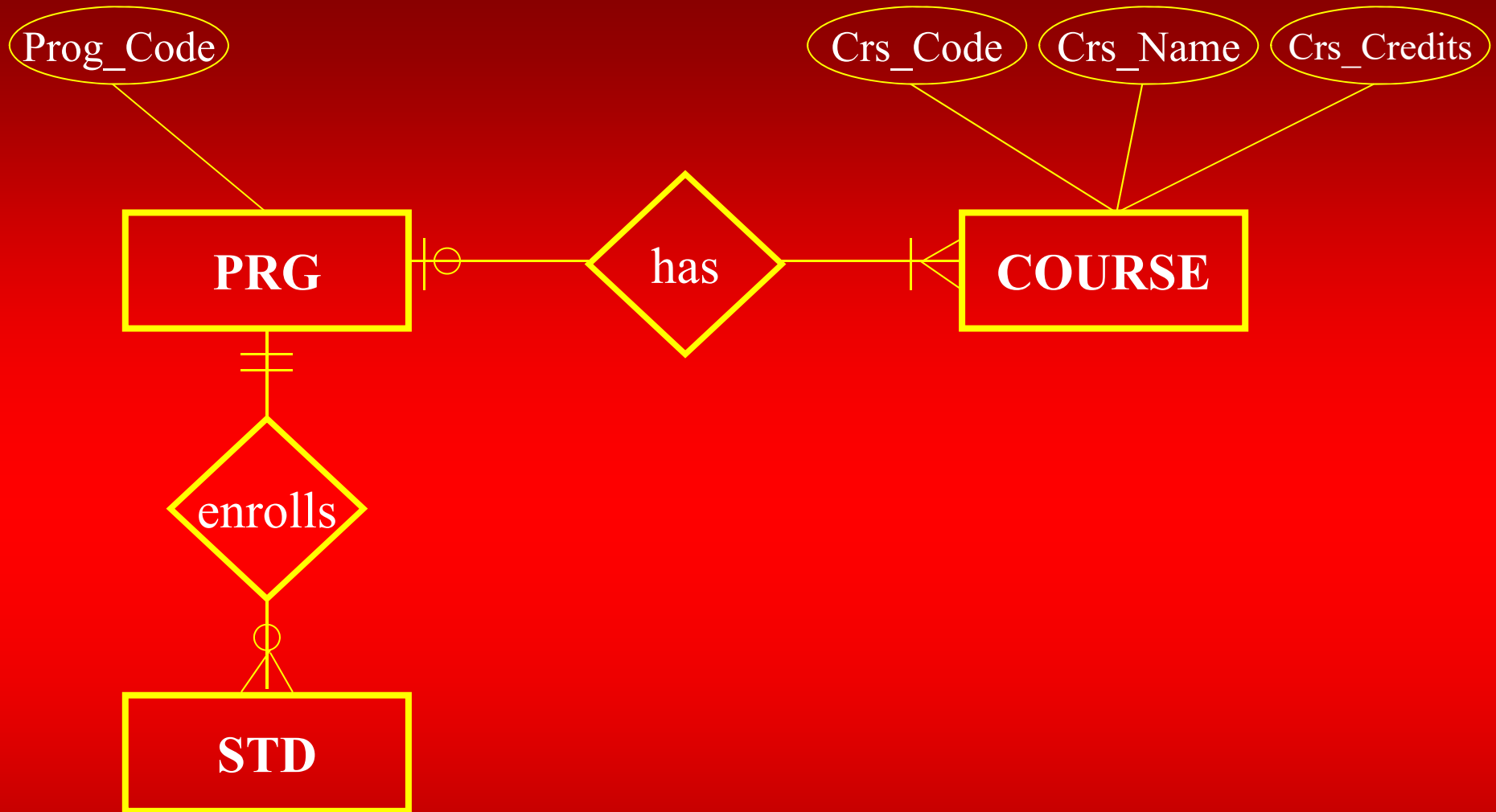
COURSE
OFFERED IN

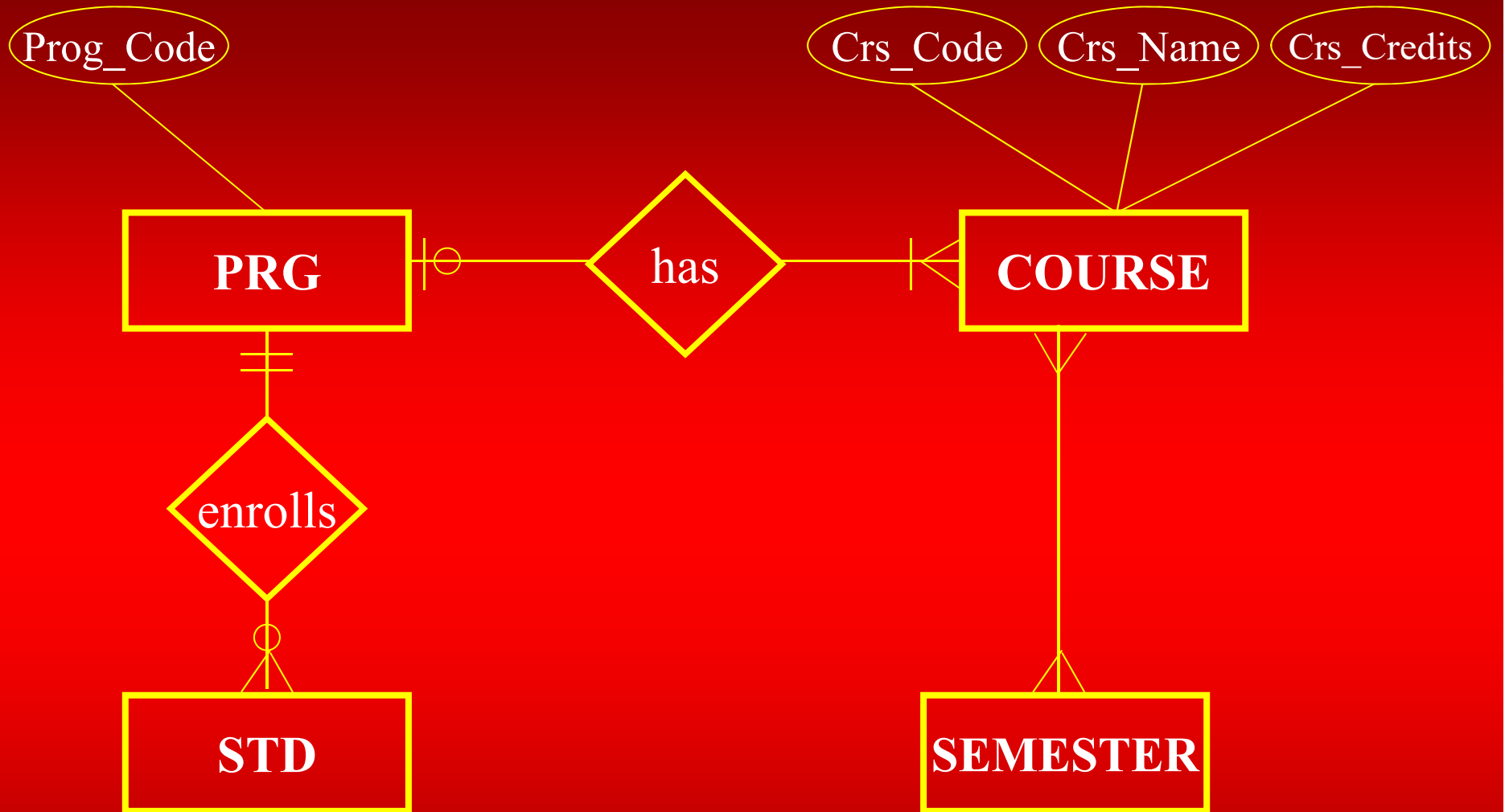


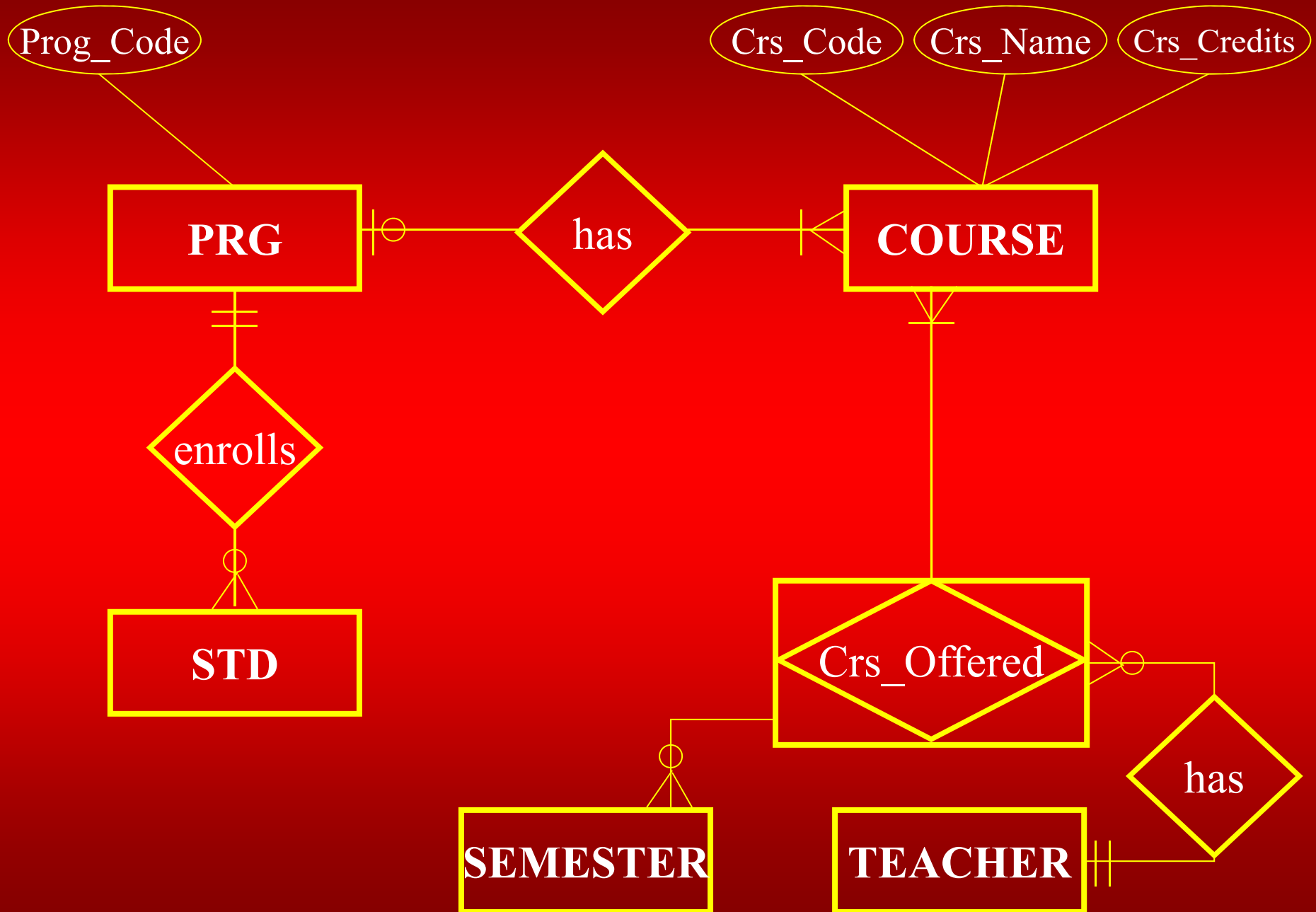


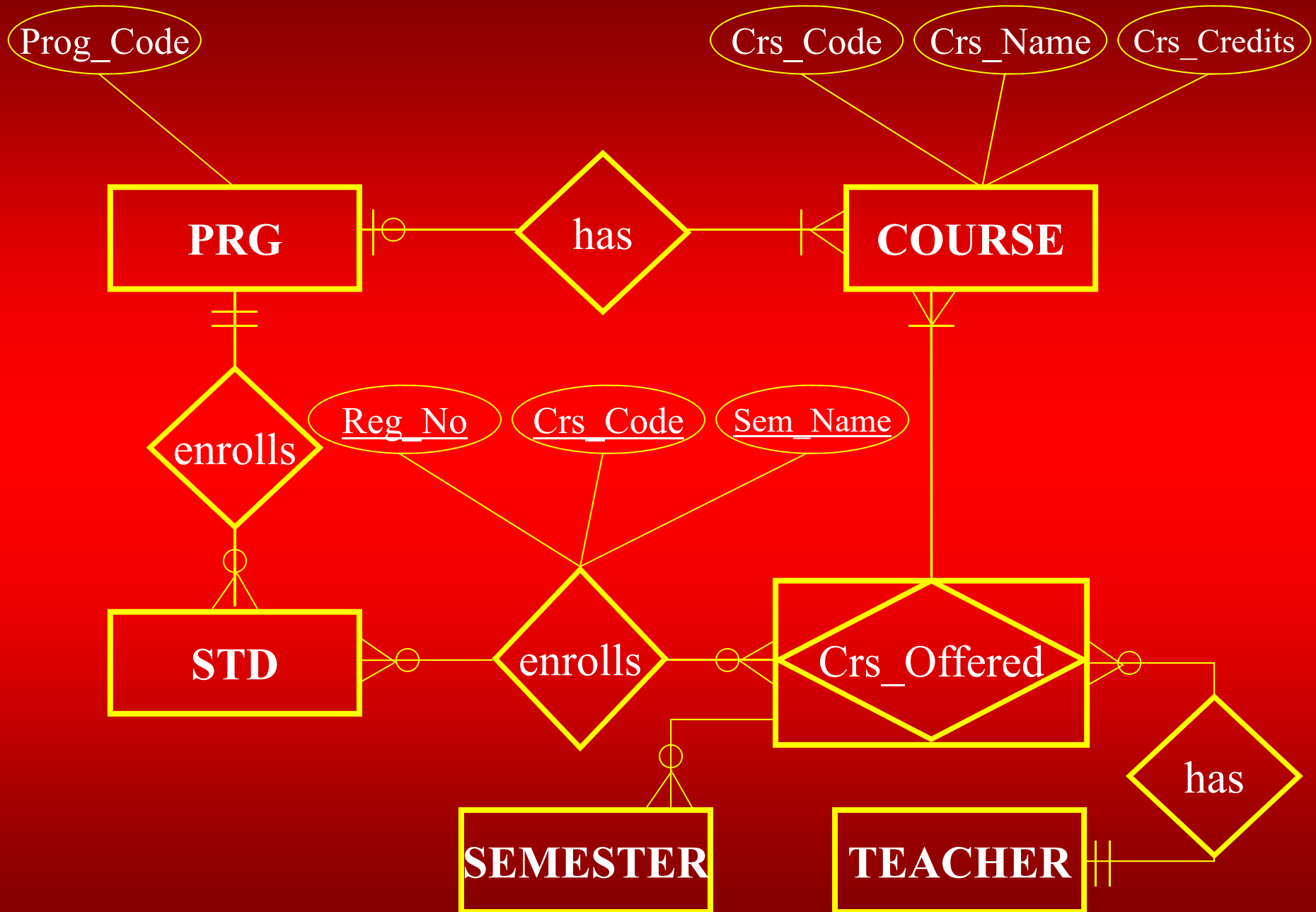


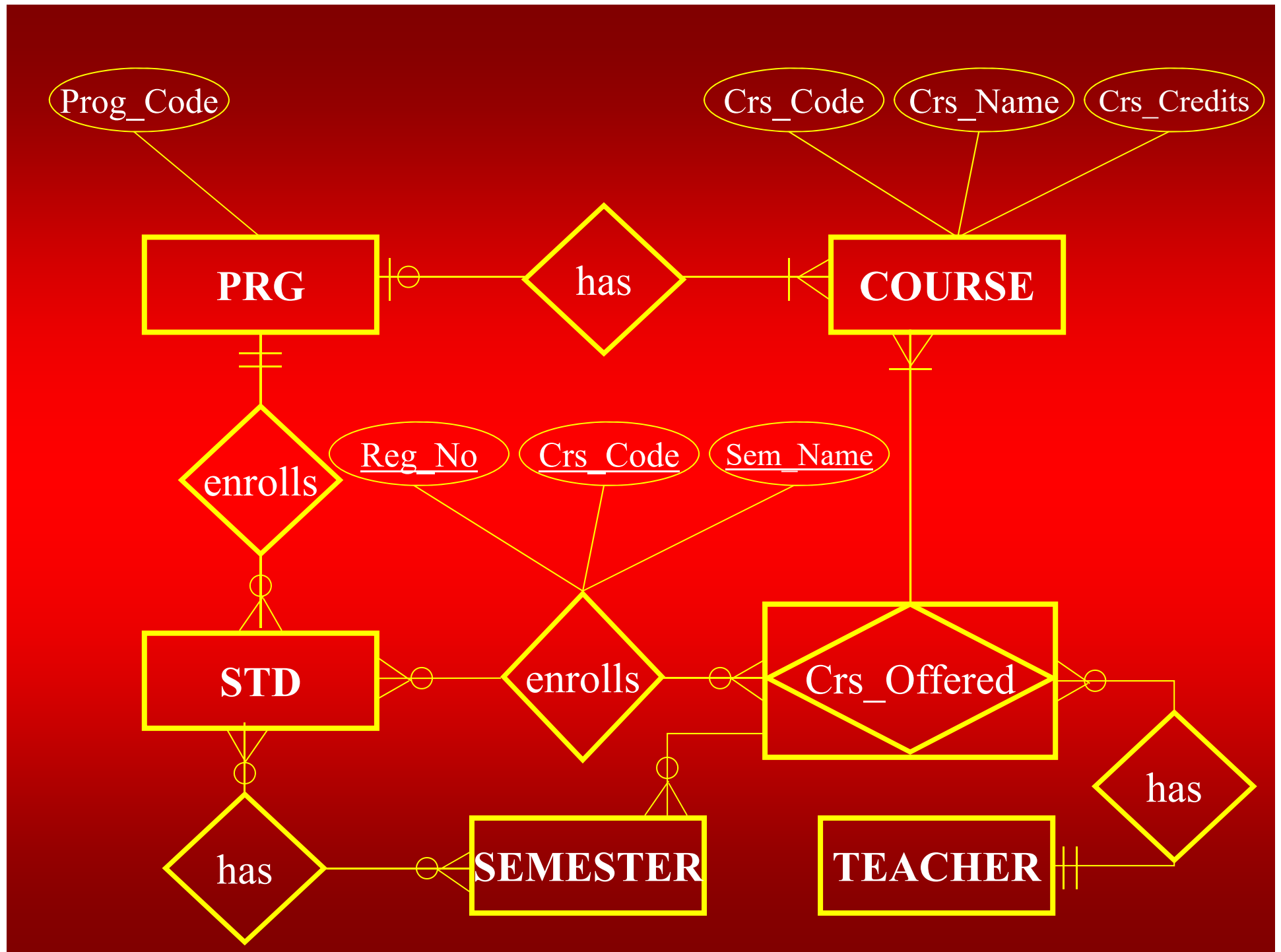


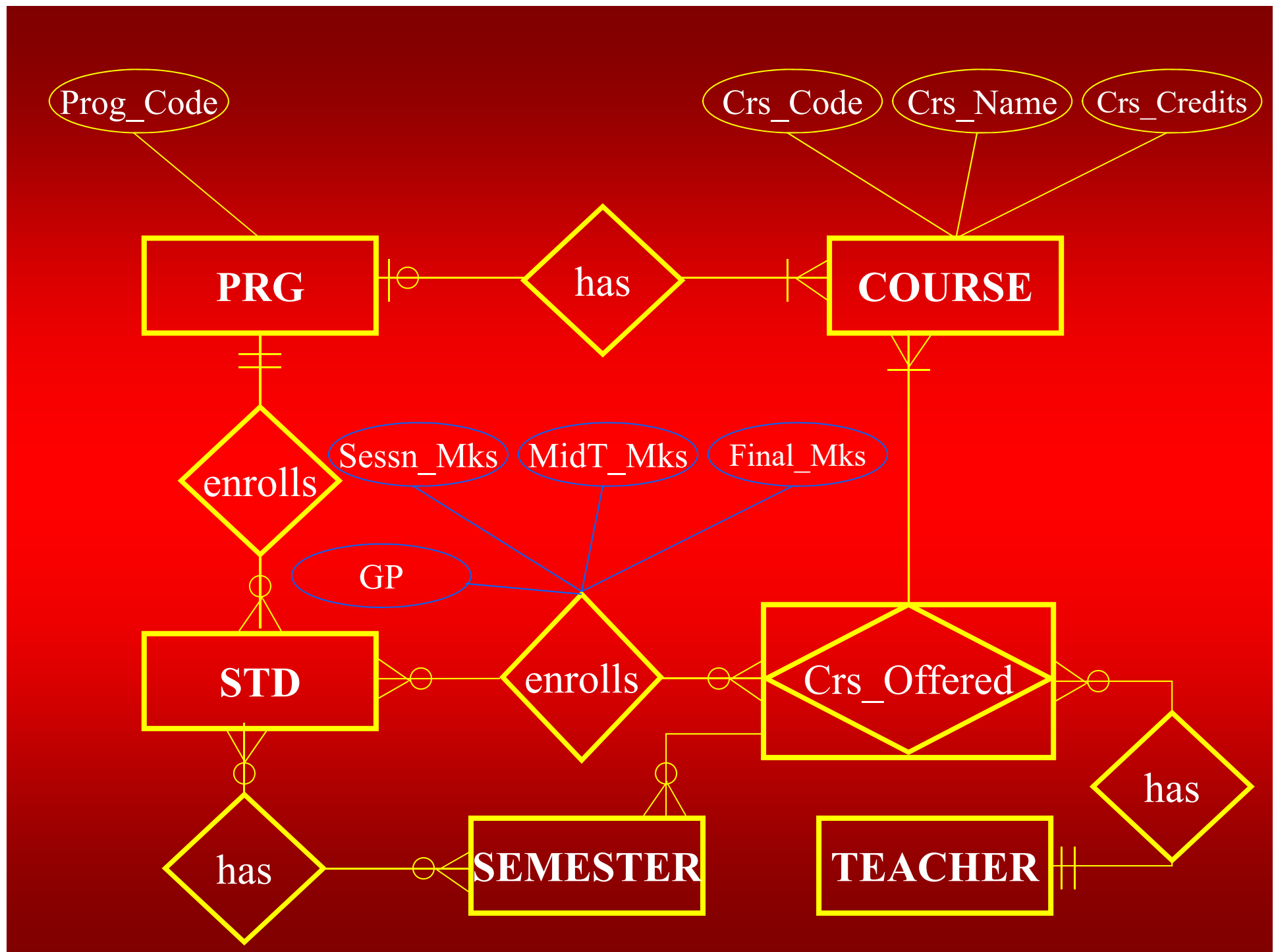


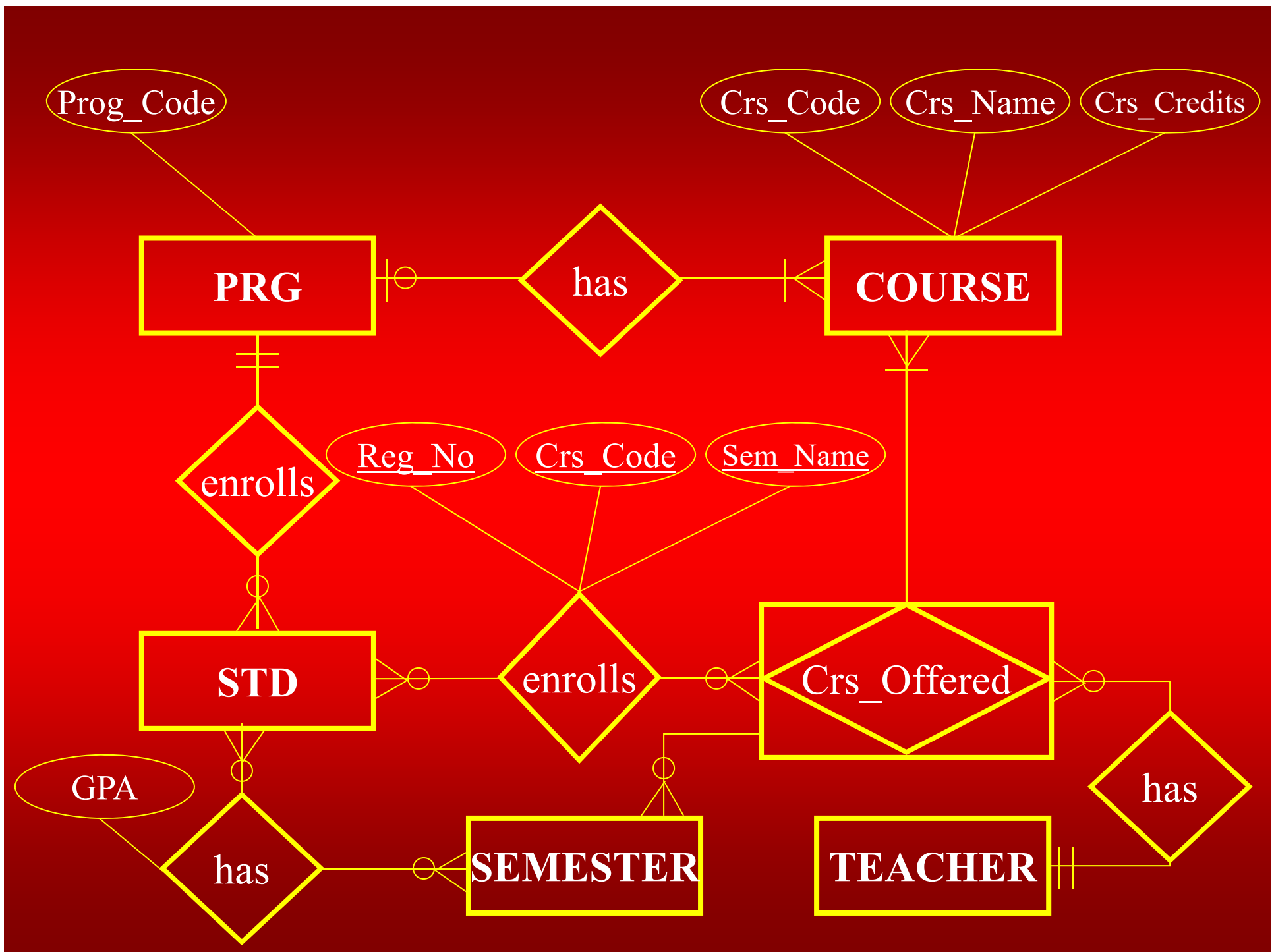


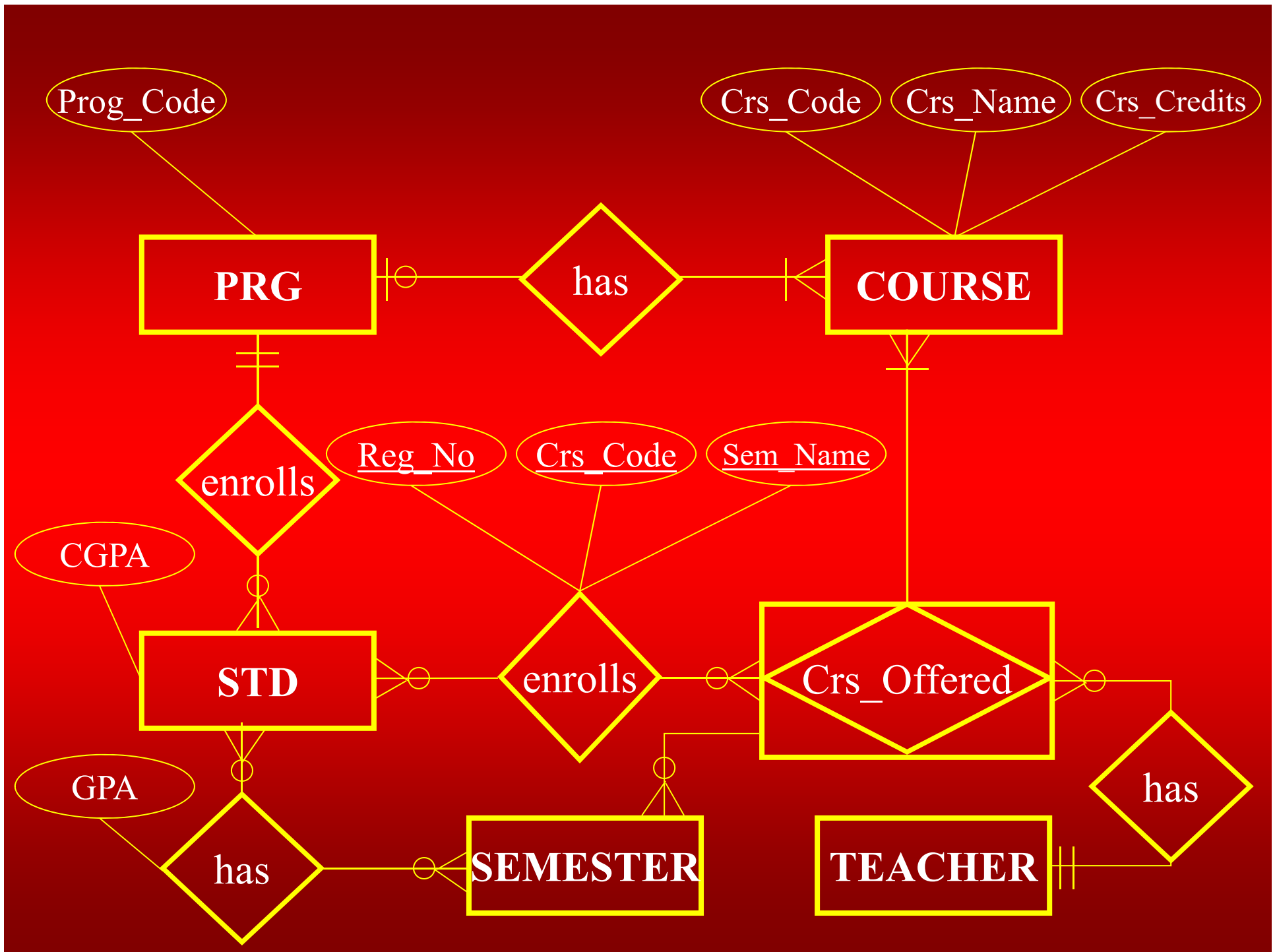












Database Management System

Lecture - 14

Logical Database Design

Introduction to Relational Data Model

Two Major Strengths

➤ Simplicity

➤ Strong Mathematical Foundation

Relational Data Model

- Presented by E. F. Codd in 1970, then of IBM
- Before Relational Data Model, two older data models were in use; Hierarchical, Network

Relational Data Model

- First DBMS built on Relational Data Model (RDM) was system R
- Another Relational DBMS built during those days was INGRES

Basic of RDM

- RDM used mainly for external, conceptual, and to some extent physical schema
- Separation of conceptual and physical levels makes manipulation much easier, contrary to previous data models

Basic of RDM

- RDM used for external, conceptual, and to some extent physical schema
- The basic structure is relation
- Both entities and relationships are modeled using tables/relations

Basics of RDM

- Relations physically represented as tables
- Table is a two dimensional representation of a relation
- Consists of rows and columns

Basics of RDM

- Columns represent attributes and rows represent records
- Rows, records and tuples all these terms are used interchangeably

Basic Properties of a Table

1. Each cell of a table contains atomic/single value
2. Each column has a distinct name; the name of the attribute it represents

Basic Properties of a Table

3. The values of the attributes come from the same domain
4. The order of the columns is immaterial

Basic Properties of a Table

5. The order of the rows is immaterial
6. Each row/tuple/record is distinct, no two rows can be same

A table

stID	stName	clName	doB	sex
S001	M. Suhail	MCS	12/6/84	M
S002	M. Shahid	BCS	3/9/86	M
S003	Naila S.	MCS	7/8/85	F
S004	Rubab A.	MBA	23/4/86	F
S005	Ehsan M.	BBA	22/7/88	M

Components of RDM

➤ As discussed in data model definition

1. Structure (relation/table)
2. Manipulation language (SQL)
3. Integrity constraints (Two)

Mathematical Relations

- Consider two sets
- $A = \{x, y\}$ $B = \{2, 4, 6\}$
- Cartesian product of these sets
- $A \times B = \{(x,2), (x,4), (x,6), (y,2), (y,4), (y,6)\}$

Mathematical Relations

➤ A relation is some subset of this

Cartesian product, For example,

➤ $R1 = \{(x,2), (y,2), (x,6), (x,4)\}$

➤ $R2 = \{(x,4), (y,6), (y,4)\}$

Mathematical Relations

The same notion of Cartesian product and relations can be applied to more than two sets, e.g. in case of three sets, we will have a relation of ordered triplets

Database Relations

➤ Thinking in some real world scenario

- Name = {Ali, Sana, Ahmed, Sara}
- Age = {15,16,17,18,.....,25}

Database Relations

Cartesian product of Name & Age

Name X Age = {(Ali,15), (Sana,15),
(Ahmed,15), (Sara,15),, (Ahmed,25),
(Sara,25)}

CLASS = {(Ali, 18), (Sana, 17), (Ali, 20),
(Ahmed, 19)}

Database Relations

- Let $A_1, A_2, A_3, \dots, A_n$ be some attributes and $D_1, D_2, D_3, \dots, D_n$ be their domains
- A relation scheme relates certain attributes with their domain in context of a relation

Relation Scheme

Can be represented as

$R = (A1:D1, A2:D2, \dots, An:Dn)$

STD = (stId:Text, stName: text,
stAdres:Text, doB:Date) OR

STD(stId, stName, stAdres, doB)

Database Relations

According to this scheme we can have a relation (instance of this scheme), like

STD={ (stId:S001, stName:Ali, stAdres: Lahore, doB:12/12/76), (stId:S003, stName:A. Rehman, stAdres: RWP, doB:2/12/77) }

Database Relations

STD={ (S001, Ali, Lahore, 12/12/76),
(S003, A. Rehman, RWP, 2/12/77) }

stId	stName	stAdres	doB
S001	Ali	Lahore	12/12/76
S002	A. Rehman	RWP	2/12/77

Database Management System

Lecture - 15

DB and Math Relations

- Properties of DB relations are similar to those of Mathematical relations, except
- The order of columns in Mathematical relation does matter

Degree and Cardinality

The number of rows in a relation is its cardinality and the number of columns is its degree

Relations Keys

- The concept of key and all different types of keys are applicable to the Relations
- Foreign Key: An attribute of a table B that is primary key in another table A

Foreign Key

Consider table EMP and DEPT

EMP (empId, empName, qual, depId)

DEPT (depId, depName, numEmp)

Integrity Constraints

- Two main types
- Entity integrity constraint
 - Primary key cannot have null value
- Referential integrity constraint
 - Value of Foreign key is either null or matches with a value in its home relation

Foreign Key Example

Significance of Constraints

- Constraints help to maintain the correctness, validity or integrity of the database
- Like null constraints, default value, domain constraint

RDM Components

- So far we have studied structure and integrity constraint component of the RDM
- Remaining; manipulation language will be discussed later

Designing Logical DB

- Logical DB design is obtained from conceptual DB design
- Generally involves transforming E-R data model to relational data model
- We have studied both DMs, now how to perform transformation

Transforming Rules

- Straightforward rules exist
- Can be performed manually as well as automatically
- Evaluate even if you use some tool, since multiple options exist

Mapping Entity Types

- Each regular entity type is transformed straightaway into a relation
- PK of the ET is declared as PK of relation
- Simple attributes of ET are included into the relation

Mapping Regular ET

Composite Attributes

- Since tables can contain only atomic values composite attributes need to be represented as a separate relation
- Quality becomes a limitation

Composite Attrib Example

Multi-valued Attributes

- An ET with a multi-valued attribute is transformed into two relations
- One contains the entity type and second the multi-valued attribute

Multi-valued Attribute

- The PK of the second relation is the PK of first relation and the attribute value itself
- Values are accessed through reference of the PK, that also serves as FK

MV Attrib Example

Mapping Weak ETs

- Identifier dependency: relation for weak entity type is created and the PK of the strong ET is used as a whole or part of the relation against weak ET

Identifier Dependency Example

Mapping Weak ETs

Referential Dependency: relation against weak ET has got its own PK, the link is established through FK, however, FK declared as not null

Ref. Dep. Example

Thanks

Database Management System

Lecture - 16

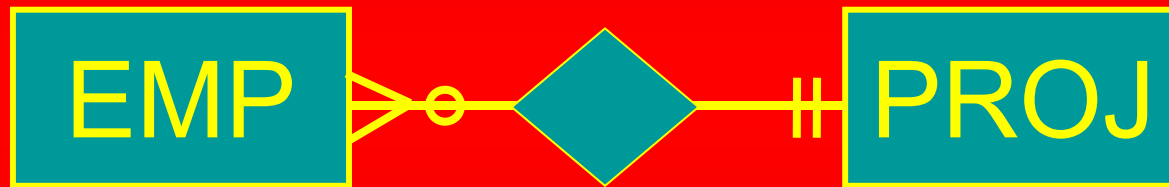
Mapping Relationships

- Relationships are mapped according to their degree and cardinalities
- Starting from binary relationships

Binary Relationships

- Cardinality: one to many
- The PK of the relation against many side ET contains the PK of the relation for ET on one side as FK

One to Many Example



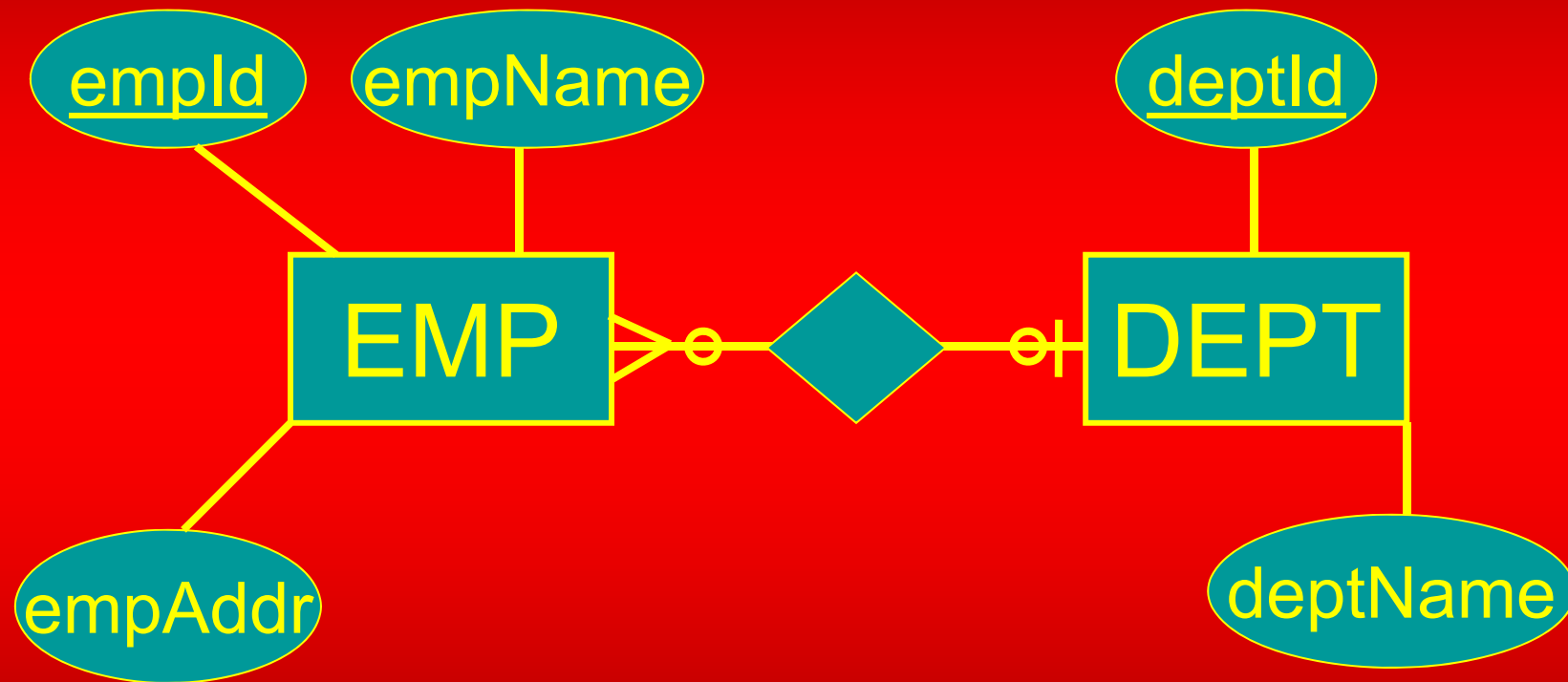
EMP(empId, empName, projId)

PROJ(projId, projName, projDur, projCost)

Minimum Cardinality

- Minimum cardinality on one side needs special attention
- If optional, define FK as normal
- If compulsory, make FK not null

Minimum Cardinality Example

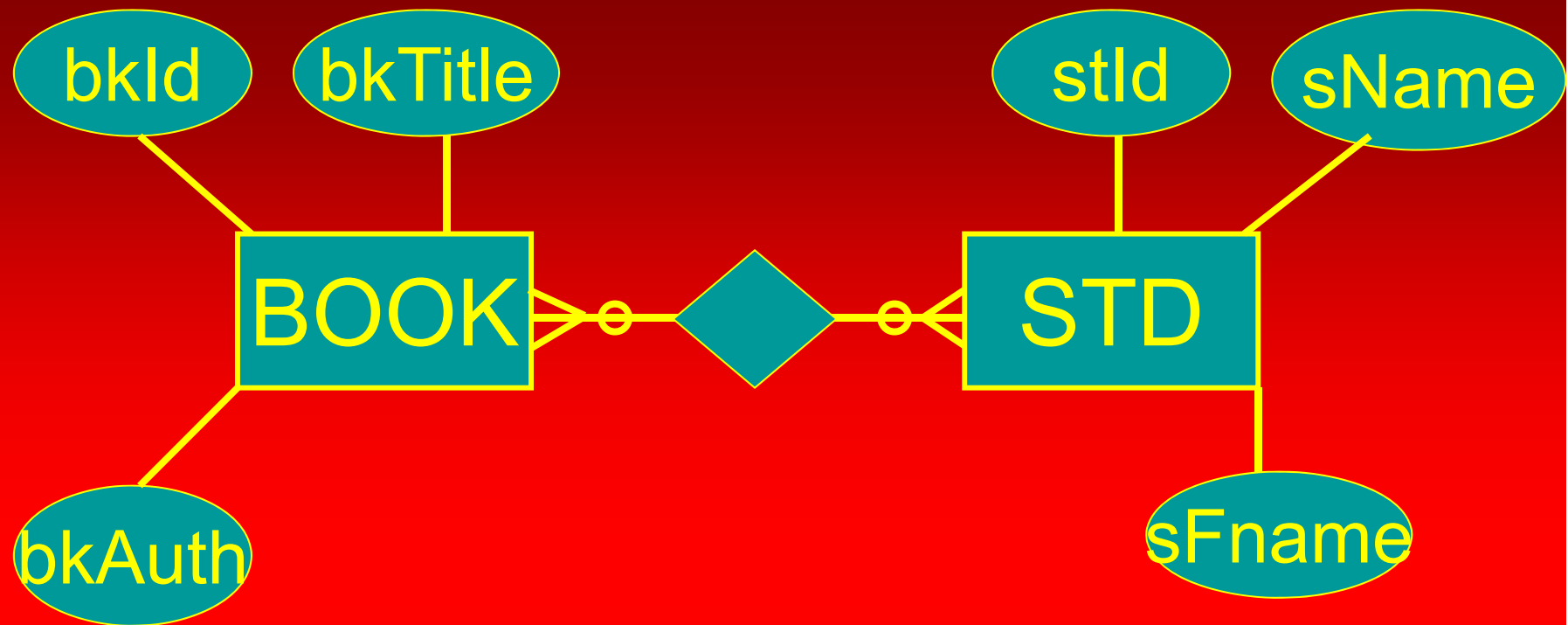


EMP(empId, empName, empAddr)
DEPT(deptId, deptName)

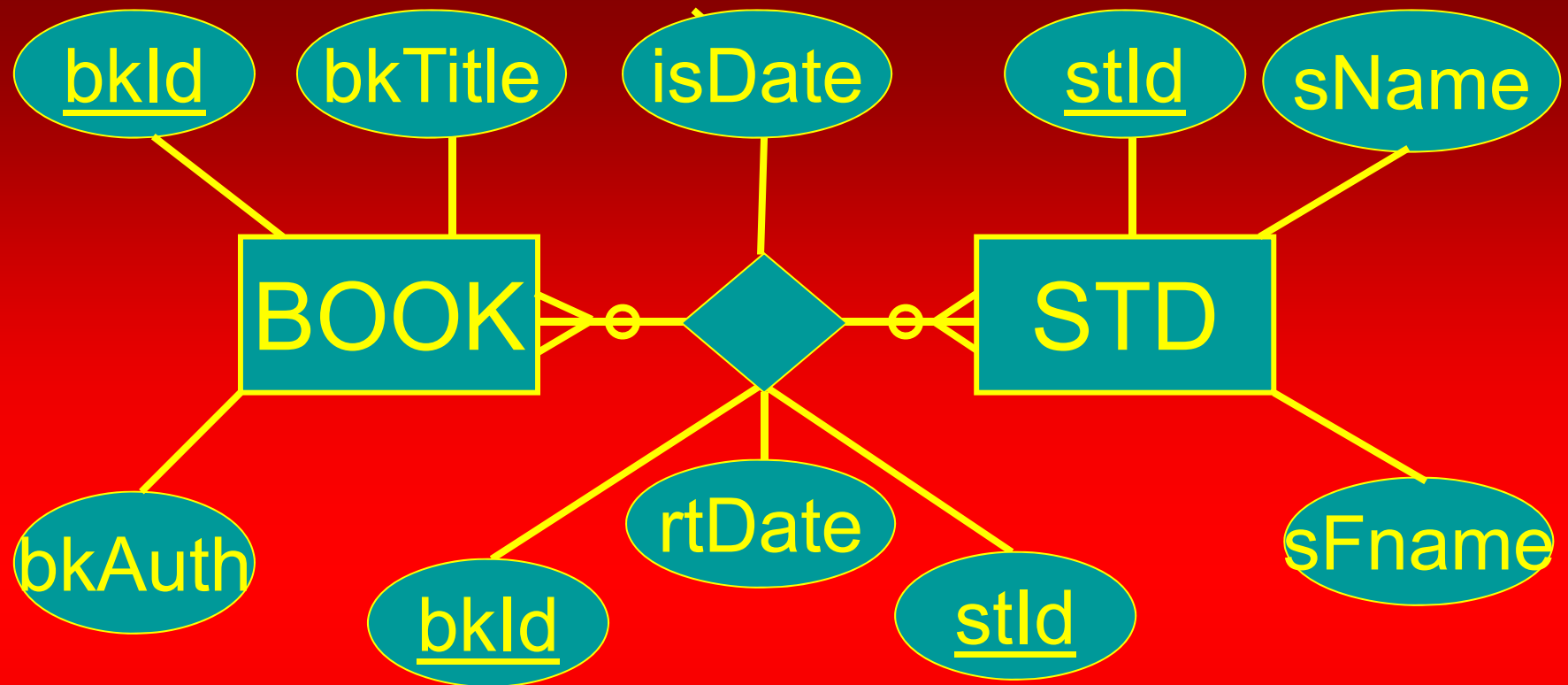
Many to Many Relationship

- A third table is created for the relationship, sometimes also called associative entity type
- Generally, the PKs of the participating ETs are used as PK of the third table

Many to Many Relationship Example



STD(stId, sName, sFname)
BOOK(bkId, bkTitle, bkAuth)
TARNS(bkId, stId)



STD(stId, sName, sFname)

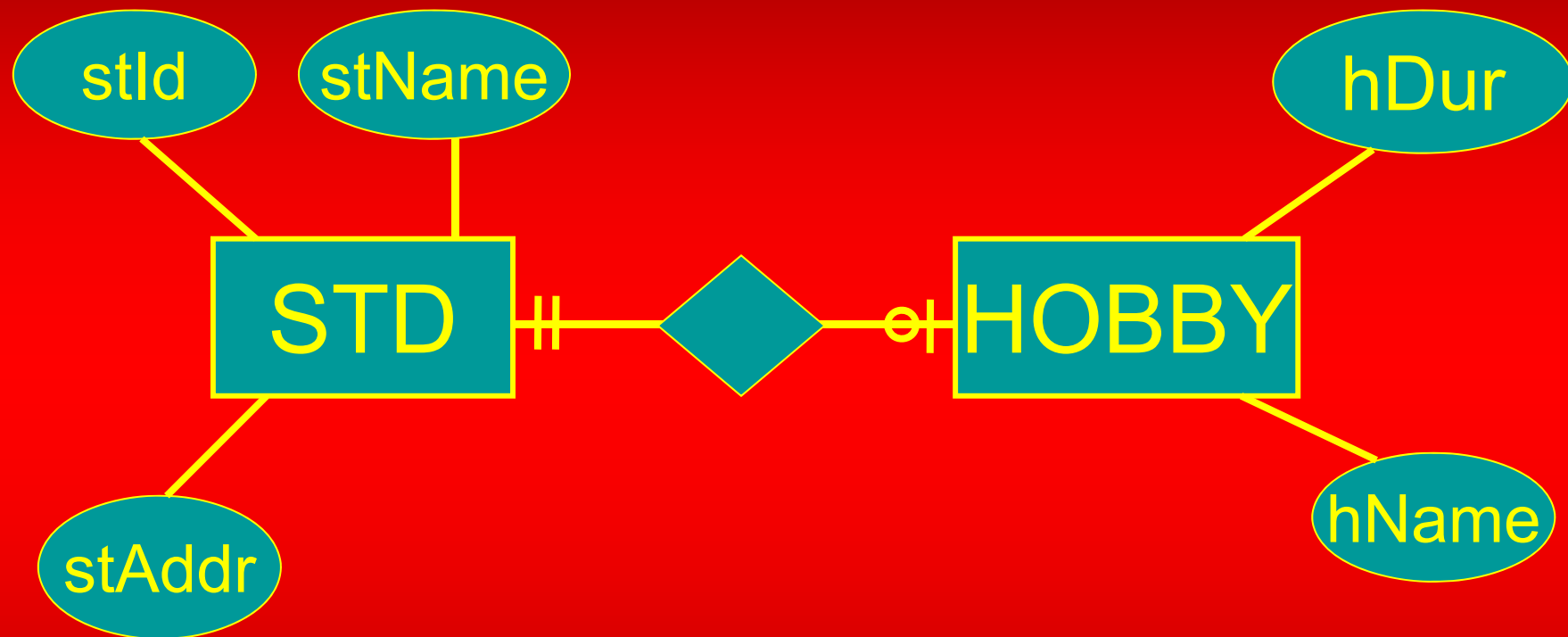
BOOK(bkId, bkTitle, bkAuth)

TARNS(bkId, stId, isDate, rtDate)

Binary one to one

- A special form of one to many relationship
- PK of one ET has to be included in the other as FK.
- Normally PK of compulsory side is included in the optional side

One to One Example

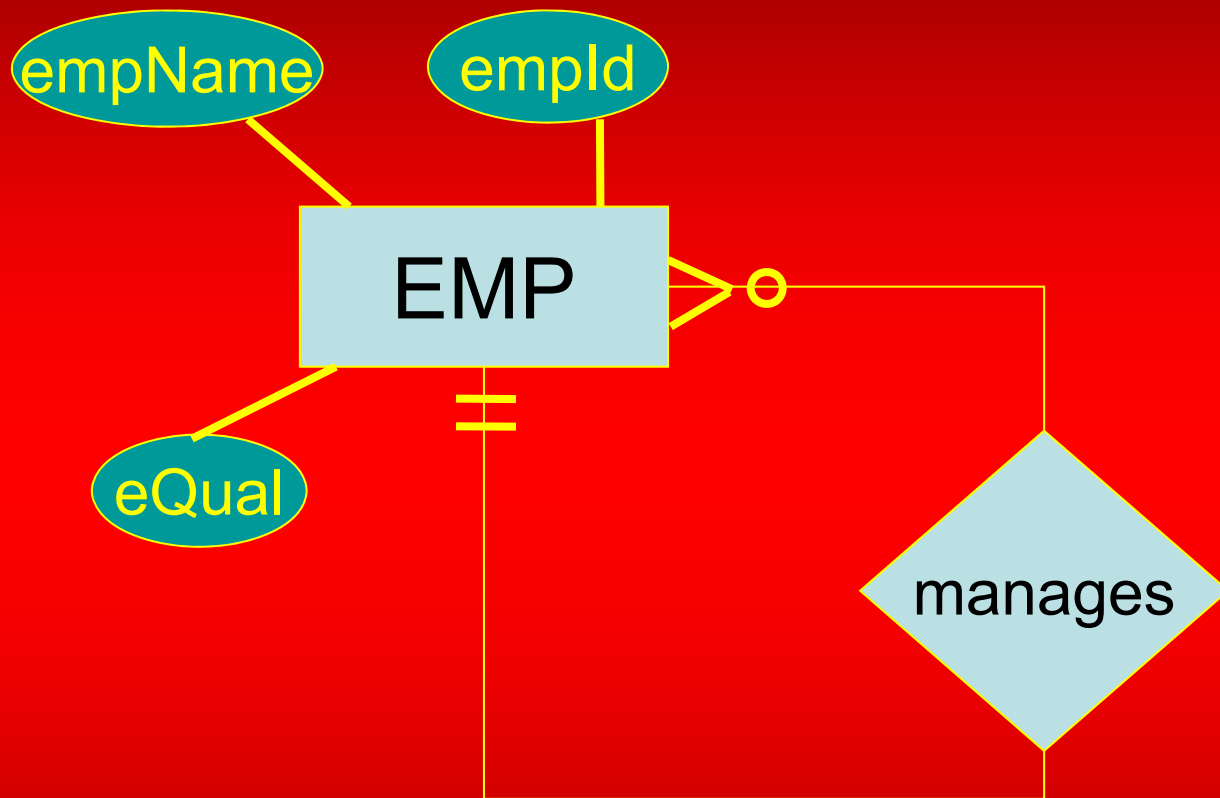


STD(stId, stName, stAddr)
HOBBY(hName, hDur, stId)

Unary Relationships

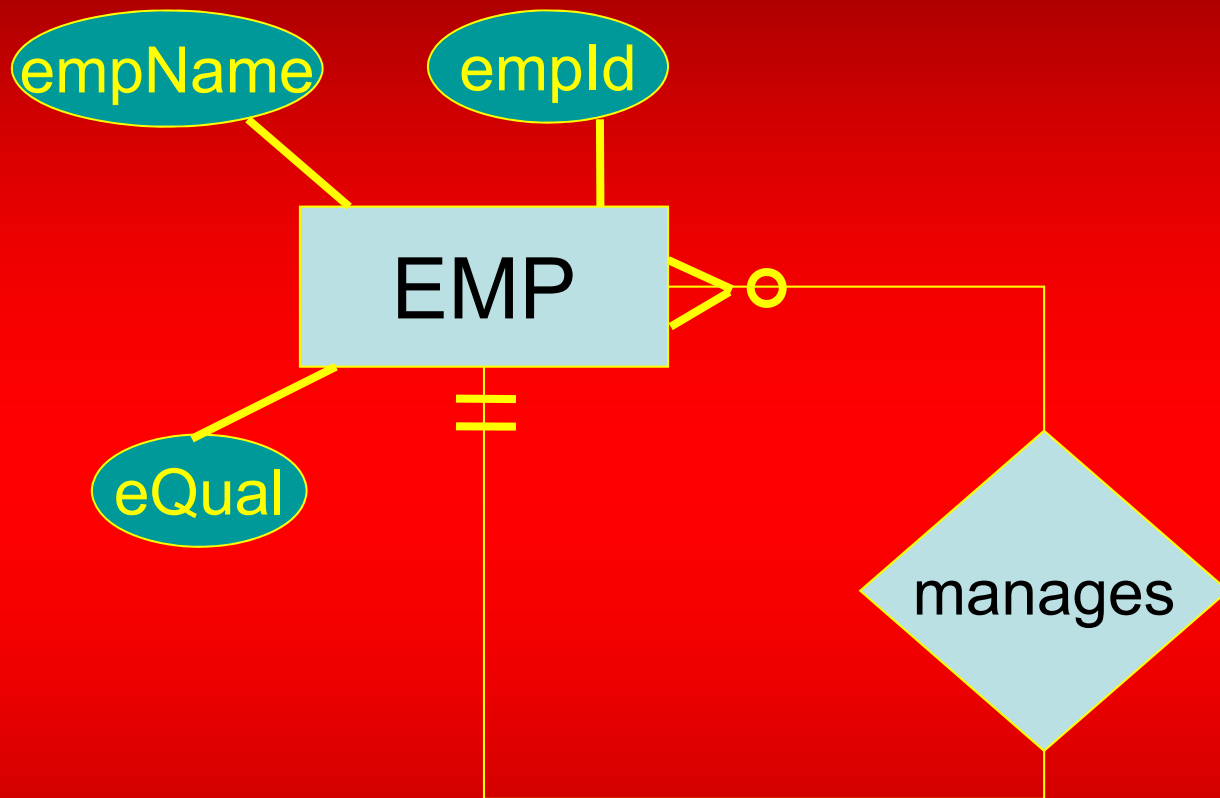
- Unary one to many introduces the PK of same ET as FK
- Many to many unary requires creation of another relation with a composite PK

Unary Example



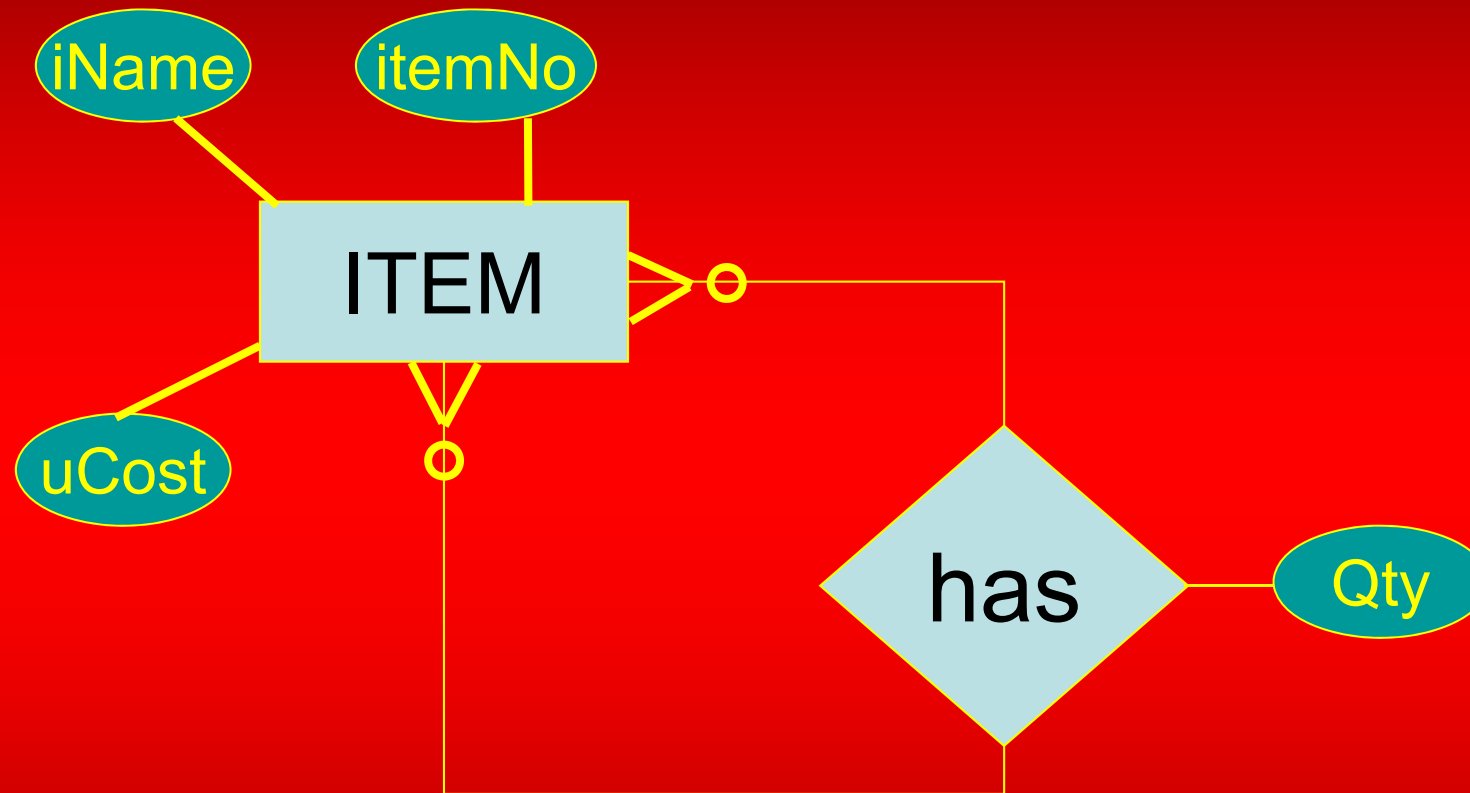
EMP(empId, empName, eQual, empId)

Unary Example



EMP(empId, empName, eQual, mgrId)

Unary Example



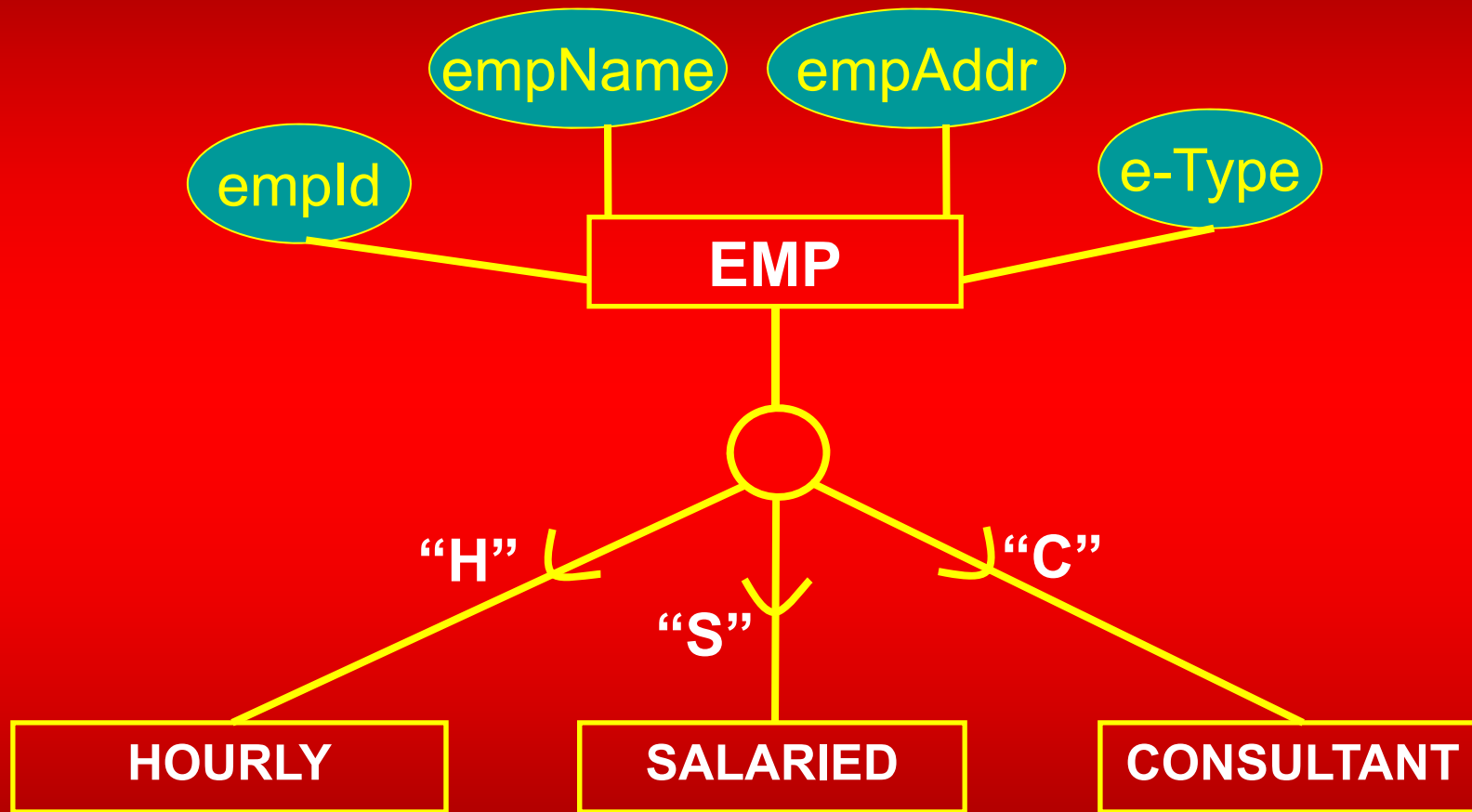
ITEM(itemNo, iName, uCost)

COMPONENT(itemNo, CompNo, Qty)

Super/Subtype Relationship

- Separate relations are created for each of the supertype and subtypes
- Assign relevant attributes
- Primary key of the supertype is included in all relations, that work for both the identity and the link

Super/Subtype example



Summary of Mapping E-R Diagram to Relational DM

DATA MANIPULATION LANGUAGES

Relational Algebra

- Relational algebra operations work on one or more relations to define another relation leaving the original intact.

Relational Algebra

- Both operands and results are relations, so output from one operation can become input to another operation.

Relational Algebra

- Allows expressions to be nested, just as in arithmetic. This property is called *closure*.

Relational Algebra

- 5 basic operations in relational algebra: Selection, Projection, Cartesian product, Union, and Set Difference.

Relational Algebra

- These perform most of the data retrieval operations needed.
- Also have Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations.

Database Management System

Lecture - 17

Five Basic Operators

➤Unary Operators

1.Select

2.Project

Five Basic Operators

➤ Binary Operators

3. Union
4. Set Difference
5. Cartesian Product

Select

➤ Selects tuples that satisfy a given predicate

➤ Notation is Greek symbol sigma:

$$\sigma_{PREDICATE}(RELATION)$$

Select

- To process a selection,
 - Look at each tuple
 - See if we have a match (based on the condition)

Select

- The Degree of the resulting relation is the same as the degree of the relation

$$| \sigma | = | r(R) |$$

- σ is Commutative

$$\sigma_{c1} (\sigma_{c2}(R)) = \sigma_{c2} (\sigma_{c1}(R))$$

Selection Example

stId	stName	stAdr	prName	curSem
S1020	Sohail Dar	H#14, F/8-4, Islamabad.	MCS	4
S1038	Shoaib Ali	H#23, G/9-1, Islamabad	BCS	3
S1015	Tahira Ejaz	H#99, Lala Rukh Wah.	MCS	5
S1018	Arif Zia	H#10, E-8, Islamabad.	BIT	5

Selection Example

1. $\sigma_{\text{Curr_Sem} > 3} (\text{STUDENT})$
2. $\sigma_{\text{Studid} = \text{'S1038'}} (\text{STUDENT})$

Selection

Example Cont'd)

FacID	Fname	Address	Salary	Rank
F2345	Usman	H#Car409	21000	lecturer
F3456	Tahir	H#Bac100	23000	Asso Prof
F4567	Ayesha	H#Car301	27000	Asso Prof
F5678	Samad	H#Irv342	32000	Professor

Selection

Example Cont'd)

1. $\sigma_{\text{Salary} > 27000}$ (FAC)
2. $\sigma_{(\text{Salary} > 26000) \text{ and } (\text{Rank} = \text{'Asso Prof'})}$ (FAC)

Projection

- Unary operation that returns its argument relation with certain attributes left out.
- Any duplicate rows are eliminated.

Projection

➤ Notation

$$\Pi_{a_1, a_2, \dots, a_k}(R)$$

where $a_1 \dots a_k$ are attribute names and R is a relation name

Projection

- Works on a single relation R and defines a relation that contains a vertical subset of R , extracting the values of specified attributes and eliminating duplicates.

Projection

- Reduces duplicate columns created by cross product.
- Creates a New relation

Projection Example

FacID	Fname	Dept	Salary	Rank
F2345	Usman	CSE	21000	lecturer
F3456	Tahir	CSE	23000	Asso Prof
F4567	Ayesha	ENG	27000	Asso Prof
F5678	Samad	MNG	32000	Professor

Projection Example

- π FacID, Salary (Faculty)
- π Fname, Rank (Faculty)
- π Facid, Salary, Rank (Faculty)

Union

- Assuming R & S are union compatible...
- Union: $R \cup S$ is the set of tuples in either R or S or both.

Union

➤ Since it is a set, there are no duplicate tuples

➤ Union is Commutative

$$R \cup S = S \cup R$$

Union Example

Course1

CID	ProgID	Cred_Hrs	CourseTitle
C2345	P1245	3	Operating Sytems
C3456	P1245	4	Database Systems
C4567	P9873	4	Financial Management
C5678	P9873	3	Money & Capital Market

Course2

CID	ProgID	Cred_Hrs	CourseTitle
C4567	P9873	4	Financial Management
C8944	P4567	4	Electronics

Union Example

Course 1 U Course 2

CID	ProgID	Cred_Hrs	CourseTitle
C2345	P1245	3	Operating Sytems
C3456	P1245	4	Database Systems
C4567	P9873	4	Financial Management
C5678	P9873	3	Money & Capital Market
C8944	P4567	4	Electronics

Intersection (\cap)

- Assuming that R & S are union compatible
- Intersection: $R \cap S$ is the set of tuples in both R and S

Intersection (\cap)

➤ Intersection is Commutative

$$R \cap S = S \cap R$$

Intersection (\cap)

Example

Course 1 \cap Course 2

CID	ProgID	Cred_Hrs	CourseTitle
C4567	P9873	4	Financial Management

Difference (-)

- Difference: $R - S$ is the set of tuples that appear in R but do not appear in S .

Difference (-) Example

Course1 – Course2

CID	ProgID	Cred_Hrs	CourseTitle
C2345	P1245	3	Operating Sytems
C3456	P1245	4	Database Systems
C5678	P9873	3	Money & Capital Market

Cartesian Product (X)

- Sets do not have to be union compatible.

$$R(A_1, A_2, \dots, A_N)$$

$$X$$

$$S(B_1, B_2, \dots, B_M) \text{ is}$$

Cartesian Product (X)

$Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$

- If R has C tuples and S has D tuples, the result is $C * D$ tuples.

Cartesian Product (X)

- Also called “cross product”
- Note that union, intersection and cross product are commutative and associative

Cartesian Product (X)

Example

Course X Registration

Course

CID	CourseTitle
C3456	Database Systems
C4567	Financial Management
C5678	Money & Capital Market

Registration

SID	StudName
S101	Ali Tahir
S103	Farah Hasan

Cartesian Product (X)

Example

CID	CourseTitle	SID	StudName
C3456	Database Systems	S101	Ali Tahir
C4567	Financial Management	S101	AliTahr
C5678	Money & Capital Market	S101	Ali Tahir
C3456	Database Systems	S103	Farah Hasan
C4567	Financial Management	S103	Farah Hasan
C5678	Money & Capital Market	S103	Farah Hasan

Join Operation

Lecture 17 ends here...

Database Management System

Lecture - 18

Types of Join

- A special form of cross product of two tables
- Different types
- Theta, Equi, Natural, Semi, Outer Joins are different types

Theta Join

We apply the condition through select on one relation and then only those rows are used in the cross product with the second relation

$$R \bowtie_{\theta} S$$

Theta Join Example

FACULTY

facId	facName	dept	salary	rank
F2345	Usman	CSE	21000	lecturer
F3456	Tahir	CSE	23000	Asso Prof
F4567	Ayesha	ENG	27000	Asso Prof
F5678	Samad	MNG	32000	Professor

COURSE

crCode	crTitle	fId
C3456	Database Systems	F2345
C4567	Financial Management	
C5678	Money & Capital Market	F4567
C3425	Introduction to Accounting	F2345

$(\sigma_{\text{rank} = \text{'Asso Prof'}}(\text{FACULTY})) \times \text{COURSE}$

$(\sigma_{\text{rank} = \text{'Asso Prof'}}(\text{FACULTY})) \times \text{COURSE}$

facId	facName	dept	salary	Rank	crCode	crTitle	fId
F3456	Tahir	CSE	23000	Asso Prof	C3456	Database Systems	F2345
F3456	Tahir	CSE	23000	Asso Prof	C4567	Financial Management	
F3456	Tahir	CSE	23000	Asso Prof	C5678	Money & Capital Market	F4567
F3456	Tahir	CSE	23000	Asso Prof	C3425	Introduction to Accounting	F2345
F4567	Ayesha	ENG	27000	Asso Prof	C3456	Database Systems	F2345
F4567	Ayesha	ENG	27000	Asso Prof	C4567	Financial Management	
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market	F4567
F4567	Ayesha	ENG	27000	Asso Prof	C3425	Introduction to Accounting	F2345

Equijoin

- Rows are joined on the basis of values of a common attribute between the two relations
- Rows having the same value in the common attribute are joined

Equijoin

- Common attributes appears twice in the output
- Common attribute with the same name is qualified with the relation name in the output

Equijoin Example

FACULTY

facId	facName	dept	salary	rank
F2345	Usman	CSE	21000	lecturer
F3456	Tahir	CSE	23000	Asso Prof
F4567	Ayesha	ENG	27000	Asso Prof
F5678	Samad	MNG	32000	Professor

COURSE

crCode	crTitle	fId
C3456	Database Systems	F2345
C4567	Financial Management	
C5678	Money & Capital Market	F4567
C3425	Introduction to Accounting	F2345

FACULTY ⋈ **COURSE**
FACULTY.facId = COURSE.fId

Equijoin Example

FACULTY  **FACULTY.facId = COURSE.fId** **COURSE**

facId	facName	dept	salary	Rank	crCode	crTitle	fId
F2345	Usman	CSE	21000	lecturer	C3456	Database Systems	F2345
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market	F4567
F2345	Usman	CSE	21000	lecturer	C3425	Introduction to Accounting	F2345

Equijoin Example

FACULTY  FACULTY.facId = COURSE.facId COURSE

FACULTY. facId	acName	dept	salary	Rank	crCode	crTitle	COURSE. facId
F2345	Usman	CSE	21000	lecturer	C3456	Database Systems	F2345
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market	F4567
F2345	Usman	CSE	21000	lecturer	C3425	Introduction to Accounting	F2345

Natural Join

- Also called simply the join, most general form of join
- Same as equijoin with common column appearing once

Natural Join Example

FACULTY

facId	facName	dept	salary	rank
F2345	Usman	CSE	21000	lecturer
F3456	Tahir	CSE	23000	Asso Prof
F4567	Ayesha	ENG	27000	Asso Prof
F5678	Samad	MNG	32000	Professor

COURSE

crCode	crTitle	facId
C3456	Database Systems	F2345
C4567	Financial Management	
C5678	Money & Capital Market	F4567
C3425	Introduction to Accounting	F2345

FACULTY ⋈ **COURSE**
FACULTY.facId, COURSE.facId

Natural Join Example

FACULTY ⋈_{FACULTY.facId, COURSE.facId} COURSE

facId	facName	dept	salary	Rank	crCode	crTitle
F2345	Usman	CSE	21000	lecturer	C3456	Database Systems
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market
F2345	Usman	CSE	21000	lecturer	C3425	Introduction to Accounting

Types Of Joins

- Left Outer Join
- Right Outer Join
- Outer Join
- Semijoin

Left Outer Join

- Keep all of the tuples from the “left” relation
- Join with the right relation
- Pad the non-matching tuples with nulls

Left Outer Join Example

FACULTY LEFT JOIN COURSE

facId	facName	dept	salary	rank	crCode	crTitle	facId
F2345	Usman	CSE	21000	lecturer	C3456	Database Systems	F2345
F2345	Usman	CSE	21000	lecturer	C3425	Intro. To Accounting	
F3456	Tahir	CSE	23000	Asso Prof			
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market	F4567
F5678	Samad	MNG	32000	Professor			

Right Outer Join

Same as the left, but keep tuples
from the “right” relation

Right Outer Join Example

FACULTY RIGHT OUTER JOIN COURSE

facId	facName	dept	salary	rank	crCode	crTitle	facId
F2345	Usman	CSE	21000	lecturer	C3456	Database Systems	F2345
					C4567	Financial Management	
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market	F4567
F2345	Usman	CSE	21000	lecturer	C3425	Intro. To Accounting	F2345

Outer Join

Same as left, but keep all tuples
from both relations

Outer Join Example

FACULTY OUTER JOIN COURSE

facId	facName	dept	salary	rank	crCode	crTitle	facId
F2345	Usman	CSE	21000	lecturer	C3456	Database Systems	F2345
F2345	Usman	CSE	21000	lecturer	C3425	Intro. To Accounting	F2345
F4567	Ayesha	ENG	27000	Asso Prof	C5678	Money & Capital Market	F4567
F3456	Tahir	CSE	23000	Asso Prof			
F5678	Samad	MNG	32000	Professor			
					C4567	Financial Management	

Semijoin

- First take the natural join of two tables
- Then take the projection on the attributes of first table

FACULTY \bowtie COURSE

Semijoin Example

FACULTY \bowtie COURSE

facId	facName	dept	salary	rank
F2345	Usman	CSE	21000	lecturer
F4567	Ayesha	ENG	27000	Asso Prof

Relational Calculus

Relational Calculus

- Relational calculus is a non-procedural formal data manipulation language.
- Use simply specifies what data should be retrieved.

Relational Calculus

- Comes in two forms
 1. Tuple- Oriented Relational Calculus
 2. Domain-Oreinted Relation Calculus.

Tuple-Oriented Relational Calculus

- Finding relational tuple s for which a predicate is true.
- A tuple variable is a variable that takes on only the tuples of some relations or relations as its RANGE of value.

Tuple-Oriented Relational Calculus

- Specify the range of tuple variable by statement

RANGE OF S IS STUDENT

Tuple-Oriented Relational Calculus

- S is a tuple variable and
STUDENT is the range.
- S represent the tuple of
STUDENT

Tuple-Oriented Relational Calculus

➤ Express as

$$\{ S \mid P(S) \}$$

➤ Find the set of all tuples S such that $P(S)$ is true, where P implies the predicate condition.

Tuple-Oriented Relational Calculus

➤ RANGE of R is STUDENT

$\{ R \mid R.Credits > 50 \}$

➤ Find the studid, stuName, major, credits etc of all student having more than 50 credits.

Domain-Oriented Relational Calculus

- Use variables that take values from domain instead of tuple of relations.

Domain-Oriented Relational Calculus

- If $P(x_1, x_2, \dots, x_n)$ stands for predicate with variables x_1, x_2, \dots, x_n then

Domain-Oriented Relational Calculus

$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$

- which means, the set of all domain variables x_1, x_2, \dots, x_n for which predicate $p(x_1, x_2, \dots, x_n)$ is true.

Domain-Oriented Relational Calculus

- we test for membership condition, to determine whether values belong to a relation.

NORMALIZATION

Normalization

- A step by step process to produce more efficient and accurate database design
- Purpose is to produce an anomaly free design

Anomalies

- An inconsistent, incomplete or incorrect state of database
- Four types of anomalies are of concern here; Redundancy, insertion, deletion and updation

Normalization

- A strongly recommended step
- Normalized design makes the maintenance of database easier
- Normalization applied on each table of a DB design

Normalization

- Performed after the logical database design
- Informally also performed during conceptual DB design

Normalization Process

- Different forms or levels of normalization
- Called first, second, third and so on forms
- Each form has got certain conditions
- If a table fulfils the condition(s) for a normal form then the table is in that normal form

Normalized DB Design

- Process is applied on each table
- The minimum form in which all tables are in is called the normal form of the entire database
- Objective is to place the DB in highest form of normalization

Database Management System

Lecture - 18

Database Management System

Lecture - 19

Functional Dependency

- Normalization is based on functional dependencies (FDs)
- A type of relationship between attributes of a relation

Functional Dependency

Definition: If A and B are attributes of a relation R, then B is functionally dependent on A if each value of A in R is associated with exactly one value of B; written as $A \longrightarrow B$

Functionally Dependency

- It does not mean that A derives B, although it may be the case sometime
- Means that if we know value of A then we can precisely determine a unique value of B

Functional Dependency

➤ Attribute of set of attributes on the left side are called determinant and on the right are called dependents

➤ Like R (a, b, c, d, e)

$a \longrightarrow b, c, d$

$d \longrightarrow d, e$

FD Example

STD(stId, stName, stAdr, prName, credits)

stId \longrightarrow stName, stAdr, prName, credits

prName \longrightarrow credits

FD Example from Table

stId	stName	stAdr	prName	prCrdts
S1020	Sohail Dar	I-8 Islamabad	MCS	64
S1038	Shoaib Ali	G-6 Islamabad	BCS	132
S1015	Tahira Ejaz	L Rukh Wah	MCS	64
S1015	Tahira Ejaz	L Rukh Wah	MCS	132
S1018	Arif Zia	E-8, Islamabad.	BIT	134

FDs and Keys

- We can determine the keys of a table seeing its FDs
- The determinant of an FD that determines all attributes of that table is the super key

FDs and Keys

- A minimal super key is the candidate key, so if a determinant of an FD determines all attributes of that relation then it is definitely a super key, and

FDs and Keys

- If there is no other FD where a subset of this determinant/SK is a super key, then it is a candidate key
- So FDs help to identify keys, how

FDs and Keys

EMP(eId, eName, eAdr, eDept, prId, prSal)

eId \rightarrow eName, eDept, eAdr

eId, prId \rightarrow prSal

STD(stId, stName, prName, adr, nic, cgpa)

stId \rightarrow stName, prName, adr, nic, cgpa

nic \rightarrow stName, prName, adr, stId, cgpa

Inference Rules

- Called inference axioms or armstrong axioms
- These are rules that establish certain FDs from a given set of FDs
- These rules are sound

Reflexivity

➤ If B is a subset of A then $A \longrightarrow B$,
it also implies that $A \longrightarrow A$ always
hold, that is

$\text{stName}, \text{stAdr} \longrightarrow \text{stName}$

Or $\text{stName} \longrightarrow \text{stName}$

Augmentation

➤ If we have $A \longrightarrow B$ then

$AC \longrightarrow BC$ that is

if $stId \longrightarrow stName$ then

$stId, stAdr \longrightarrow stName, stAdr$

Transitivity

➤ If $A \longrightarrow B$ and $B \longrightarrow C$ then $A \longrightarrow C$

that is

If $stId \longrightarrow prName$ and $prName \longrightarrow credits$

Then

$stId \longrightarrow credits$

Additivity or Union

➤ If $A \longrightarrow B$ and $A \longrightarrow C$ then $A \longrightarrow BC$

if $\text{empId} \longrightarrow \text{eName}$ and $\text{empId} \longrightarrow \text{qual}$

Then we can write it as

$\text{empId} \longrightarrow \text{eName}, \text{qual}$

Projectivity or Decomposition

➤ If $A \longrightarrow BC$ then $A \longrightarrow B$ and $A \longrightarrow C$

if $\text{empId} \longrightarrow \text{eName}, \text{qual}$

Then we can write it as

$\text{empId} \longrightarrow \text{eName}$ and $\text{empId} \longrightarrow \text{qual}$

Pseudotransitivity

➤ If $A \longrightarrow B$ and $CB \longrightarrow D$ then $AC \longrightarrow D$

if $stId \longrightarrow stName$ and

$stName, fName \longrightarrow stAdr$

Then we can write it as

$stId, fName \longrightarrow stAdr$

Normal Forms

First Normal Form

- A relation is in first normal form iff every attribute in every tuple contains an atomic value
- There is no multivalued (repeating group) in the relation

First Normal Form

- One of the basic properties of relation
- Multiple values create problems in performing different operations, like, select or join
- Remember the treatment of multivalued attributes in the mapping process

First Normal Form

STD(stId, stName, stAdr, prName, bkId)

stId	stName	stAdr	prName	bkId
S1020	Sohail Dar	I-8 Islamabad	MCS	B00129
S1038	Shoaib Ali	G-6 Islamabad	BCS	B00327
S1015	Tahira Ejaz	L Rukh Wah	MCS	B08945, B06352
S1018	Arif Zia	E-8, Islamabad.	BIT	B08474

First Normal Form

stId	stName	stAdr	prName	bkId
S1020	Sohail Dar	I-8 Islamabad	MCS	B00129
S1038	Shoaib Ali	G-6 Islamabad	BCS	B00327
S1015	Tahira Ejaz	L Rukh Wah	MCS	B08945
S1015	Tahira Ejaz	L Rukh Wah	MCS	B06352
S1018	Arif Zia	E-8, Islamabad.	BIT	B08474

Mind the key please

Second Normal Form

Full functional dependency: an attribute B is fully functionally dependent on A if the B can be determined by whole of A not by any proper subset of A

Full Functional Dependence

Consider the relation

CLASS(crId, stId, stName, fId, room, grade)

$\text{crId, stId} \rightarrow \text{stName, fId, room, grade}$

$\text{stId} \rightarrow \text{stName}$

$\text{crId} \rightarrow \text{fId, room}$

Database Management System

Lecture - 19

Database Management System

Lecture - 20

Second Normal Form

- A relation is in 2nd normal form iff it is in the first normal form and all non key attributes are fully functionally dependent on key, that is, there is no partial dependency

Second Normal Form

CLASS(crId, stId, stName, fld, room, grade)

crId, stId \rightarrow stName, fld, room, grade

stId \rightarrow stName

crId \rightarrow fld, room

Anomalies

- Redundancy
- Insertion Anomaly
- Deletion Anomaly
- Updation Aomaly

Anomalies

crId	stId	stName	flId	room	grade
C3456	S1020	Suhail Dar	F2345	104	B
C5678	S1020	Suhail Dar	F4567	106	
C3456	S1038	Shoaib Ali	F2345	104	A
C5678	S1015	Tahira Ejaz	F4567	106	B

Second Normal Form

Relation is decomposed based on the FDs

CLASS(crId, stId, stName, fld, room, grade)

$\text{crId}, \text{stId} \rightarrow \text{stName}, \text{fld}, \text{room}, \text{grade}$

$\text{stId} \rightarrow \text{stName}$

$\text{crId} \rightarrow \text{fld}, \text{room}$

STD(stId, stName)

COURSE(crId, fld, room)

CLASS(crId, stId, grade)

Second Normal Form

- Each of these tables is in second normal form
- Free of anomalies due to partial dependency

Third Normal Form

A table is in third normal form (3NF) iff it is in 2NF and there is no transitive dependency, that is, no non-key attribute is dependent on another non-key attribute

Transitive Dependency

STD(stId, stName, stAdr, prName, prCrdts)

stId \rightarrow stName, stAdr, prName, prCrdts

prName \rightarrow prCrdts

Anomalies

stId	stName	stAdr	prName	prCrdts
S1020	Sohail Dar	I-8 Islamabad	MCS	64
S1038	Shoaib Ali	G-6 Islamabad	BCS	132
S1015	Tahira Ejaz	L Rukh Wah	MCS	64
S1015	Tahira Ejaz	L Rukh Wah	MCS	64
S1018	Arif Zia	E-8, Islamabad.	BIT	134

Third Normal Form

- STD(stId, stName, stAdr, prName, prCrdts)
- stId \rightarrow stName, stAdr, prName, prCrdts
- prName \rightarrow prCrdts
- STD (stId, stName, stAdr, prName)
- PROGRAM (prName, prCrdts)

3NF Relations

- Each of the table is in 3NF
- Free of all anomalies

Boyce-Codd Normal Form

- A general form of 3NF
- Every relation in BCNF is in 3NF
vice-versa is not always true
- 3NF is checked in steps, BCNF
checked directly

BCNF

- A table is in BCNF if every determinant is a candidate key
- Situation when table in 3NF is not in BCNF
- A non-key determines a part of the composite primary key

BCNF

FACULTY(fId, dept, office, rank, dateHired)

fId, dept \rightarrow office, rank, dateHired

office \rightarrow dept

- Table is in 3NF, not in BCNF since the office is not a candidate key
- Generates multiple overlapping candidate keys so we have fId, office \rightarrow dept, rank, dateHired

BCNF

➤ We decompose the table again to bring it into BCNF

FACULTY (fId, dept, office, rank, dateHired)

FACULTY(fId, office, rank, dateHired)

OFFICE(office, dept)

Higher Normal Forms

- After BCNF fourth, fifth and domain-key normal forms exist
- 4NF deals with multivalued dependency, fifth deals with possible lossless decompositions, DKNF reduces further chances of any possible inconsistency

Database Management System

Lecture - 20

Database Management System

Lecture - 21

Normalization Summary

- A step by step process to make DB design more efficient and accurate
- A strongly recommended activity performed after the logical DB design phase

Normalization Summary

- Un-normalized relations are more prone to errors or inconsistencies
- Normalization is based on the FDs
- FDs are not created rather identified by the designer/analyst

Normalization Summary

- Normalization forms exist up to 6NF, however, for most of the situations 3NF is sufficient
- Performed through Analysis or Synthesis process

Normalization Example

- Identify FDs
- Apply on the relevant tables; see if any normalization requirement is being violated, that is, causing some anomaly

Normalization Example

[illegible]

Different Data as mentioned in the book...

Some Facts

1. Each project has a unique name,
but names of employees and
managers are not unique
2. Each project has one manager,
whose name is stored in PROJMGR

PROJNAME \longrightarrow PROJMGR

3. Many employees may be assigned to work on each project, and an employee may be assigned to more than one project. HOURS tells the number of hours per week that a particular employee is assigned to work on a particular project

PROJNAME, EMPID \longrightarrow HOURS

4. Budget stores the amount budgeted for a project,
and STARTDATE gives the starting date for a project

PROJNAME → PROJMgr, BUDGET, STARTDATE

5. Salary gives the annual salary of an employee

EMPID → SALARY

6. EMPMGR gives the name of the employee's manager, who is not the same as the project manager

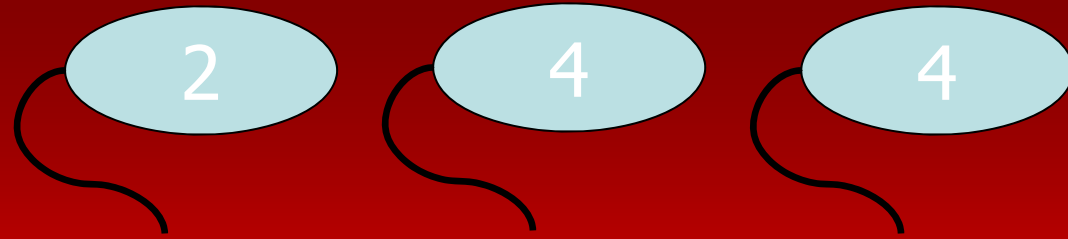
EMPID → EMPMGR

7. EMPDEPT gives the employee's department. Department names are unique. The employee's manager is the manager of the employee's department

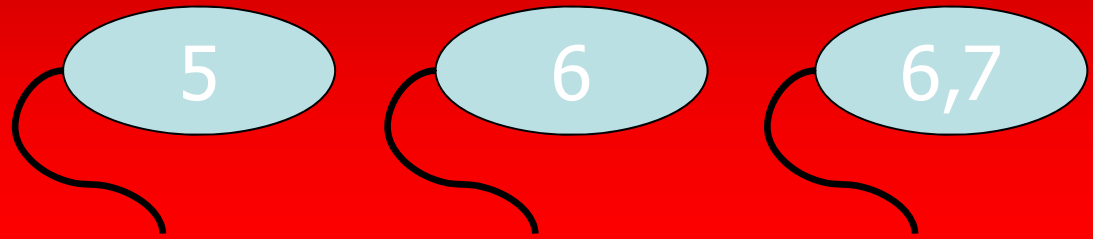
EMPDEPT → EMPMGR

8. RATING gives the employee's rating for a particular project. The project manager assigns the rating at the end of the employee's work on that project

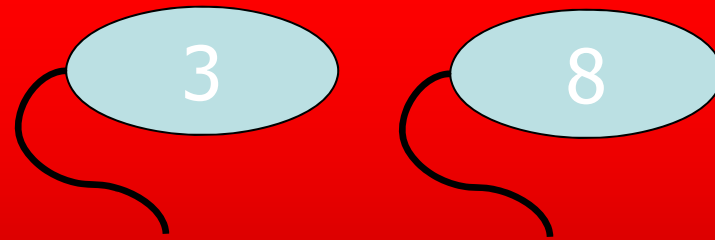
PROJNAME, EMPID \longrightarrow RATING



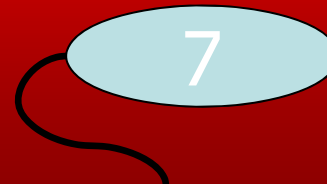
PROJNAME → PROJMGR, BUDGET, STARTDATE



EMPID → EMPNAME, SALARY, EMPMGR, EMPDEPT



PROJNAME, EMPID → HOURS, RATING



EMPDEPT → EMPMGR

Original relation:

WORK \longrightarrow (PROJNAME, PROJMGR, EMPID, HOURS, EMPNAME, BUDGET, STARTDATE, SALARY, EMPPMGR, EMPDEPT, RATING)

New relations:

PROJ (PROJNAME, PROJMGR, BUDGET, STARTDATE)

EMP (EMPID, EMPNAME, SALARY, EMPPMGR, EMPDEPT)

WORK (PROJNAME, EMPID, HOURS, RATING)

PROJ (PROJNAME, PROJMgr, BUDGET, STARTDATE)

EMP (EMPID, EMPNAME, SALARY, EMPMgr, EMPDEPT)

WORK (PROJNAME, EMPID, HOURS, RATING)

PROJ (PROJNAME, PROJMgr, BUDGET, STARTDATE)

EMP (EMPID, EMPNAME, SALARY, EMPDEPT)

DEPT (EMPDEPT, EMPMgr)

WORK (PROJNAME, EMPID, HOURS, RATING)

Checking for BCNF

PROJ (PROJNAME, PROJMgr, BUDGET, STARTDATE)

PROJNAME → PROJMgr, BUDGET, STARTDATE

EMP (EMPID, EMPNAME, SALARY, EMPDEPT)

EMPID \longrightarrow EMPNAME, SALARY, EMPMGR, EMPDEPT

WORK (PROJNAME, EMPID, HOURS, RATING)

PROJNAME, EMPID \longrightarrow HOURS, RATING

DEPT (EMPDEPT, EMPMGR)

EMPDEPT \longrightarrow EMPMGR

Physical Database Design

Objective

- Basic goal is data processing efficiency
- Transforms logical DB design into technical specifications for storing and retrieving data
- Does not include practically implementing the design however tool specific decisions are involved

Inputs Required

- Normalized relations
- Definitions of each attribute
- Descriptions of data usage
- Requirements for response time, data security, backup etc.
- Tool to be used

Decisions Involved

1. Choosing data types
2. Grouping attributes (although normalized)
3. Deciding file organizations
4. Selecting structures
5. Preparing strategies for efficient access

Database Management System

Lecture - 21

Database Management System

Lecture - 22

Data Volume and Usage Analysis

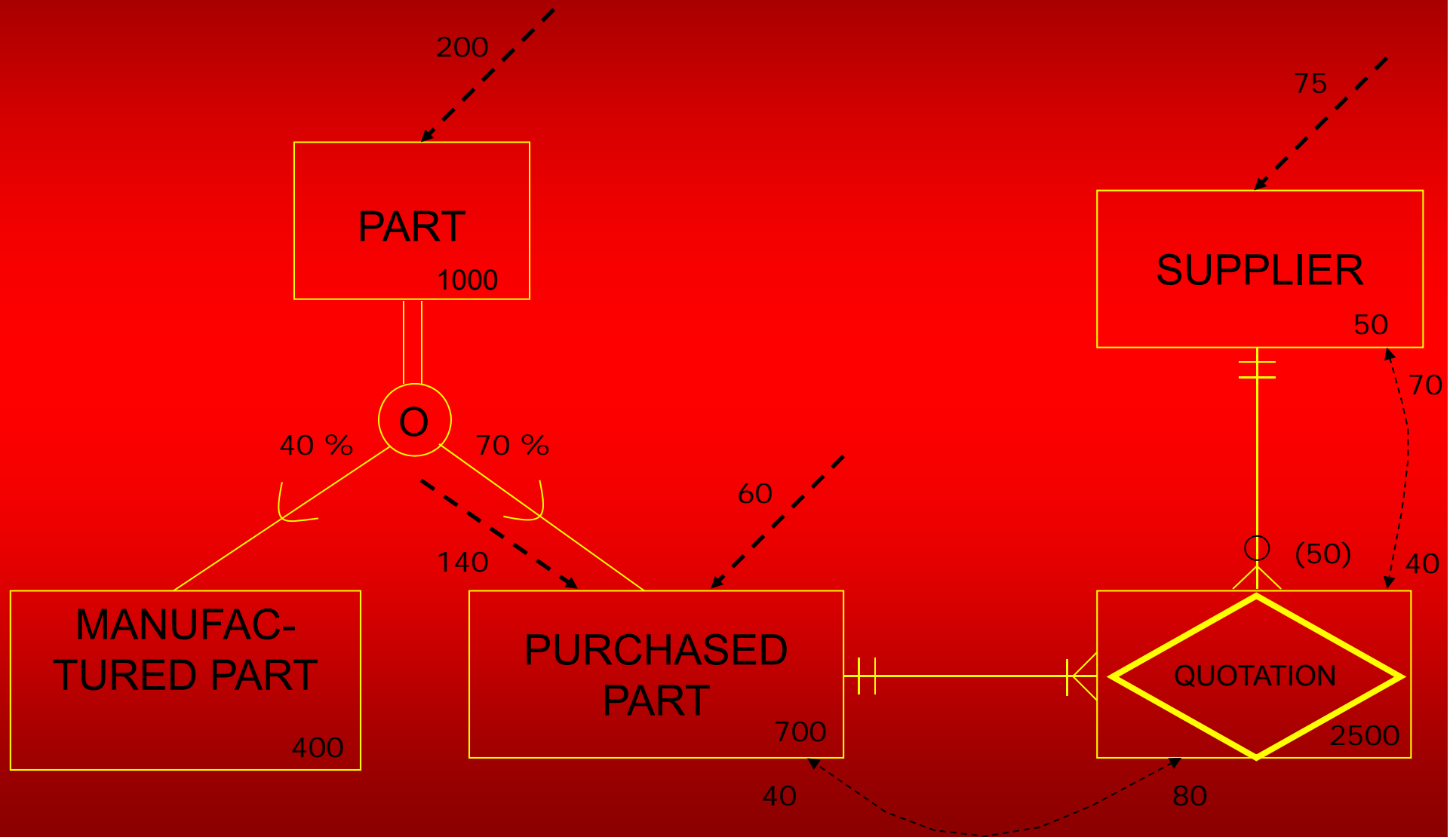
- Statistics about the size and usage of data plays critical role in data processing efficiency
- Final step of logical DB design or first step in physical DB design

Data Volume and Usage Analysis

- Statistics collected during analysis phase from the users of the system
- May not be accurate; give the tentative and relevant figures

Data Volume and Usage Analysis

Composite Usage Map



Designing Fields

- Field is smallest unit of application data; corresponds to a simple attribute
- Involves different decisions about fields

Choosing Data Type

- Data type is defined as set of values along with the operations that can be performed on them
- Precisely depends on the particular DBMS

Choosing Data Types

➤ Involves four objectives

- Minimize storage space
- Represent all possible values
- Improve data integrity
- Support all data manipulation

Coding Techniques

- Values of the attributes with small domains can be replaced by codes
- Codes can be stored in lookup table or can be hard coded, example

Coding Example

STUDENT

stId	stName	hobby
S1020	Sohail Dar	Reading
S1038	Shoaib Ali	Gardening
S1015	Tahira Ejaz	Reading
S1015	Tahira Ejaz	Movies
S1018	Arif Zia	Reading

Coding Example

STUDENT

stId	stName	hobby
S1020	Sohail Dar	R
S1038	Shoaib Ali	G
S1015	Tahira Ejaz	R
S1015	Tahira Ejaz	M
S1018	Arif Zia	R

HOBBY

code	Hobby
R	Reading
G	Gardening
M	Movies

Controlling Data Integrity

- Concerns the possible values that a field can assume
- First such control is enforced by the data type
- Some others are...

Controlling Data Integrity

- Default value
- Range control
- Null values
- Referential integrity
- Handling missing data

Database Management System

Lecture - 22

Database Management System

Lecture - 23

Physical Records and Denormalization

- In logical DB design we group things logically related, accessed through same PK
- In physical DB design fields are grouped as they are stored physically and accessed by DBMS

Denormalization

- In some cases for the efficiency purposes we have to denormalize the relations
- Process of transforming normalized relations into unnormalized physical record specifications

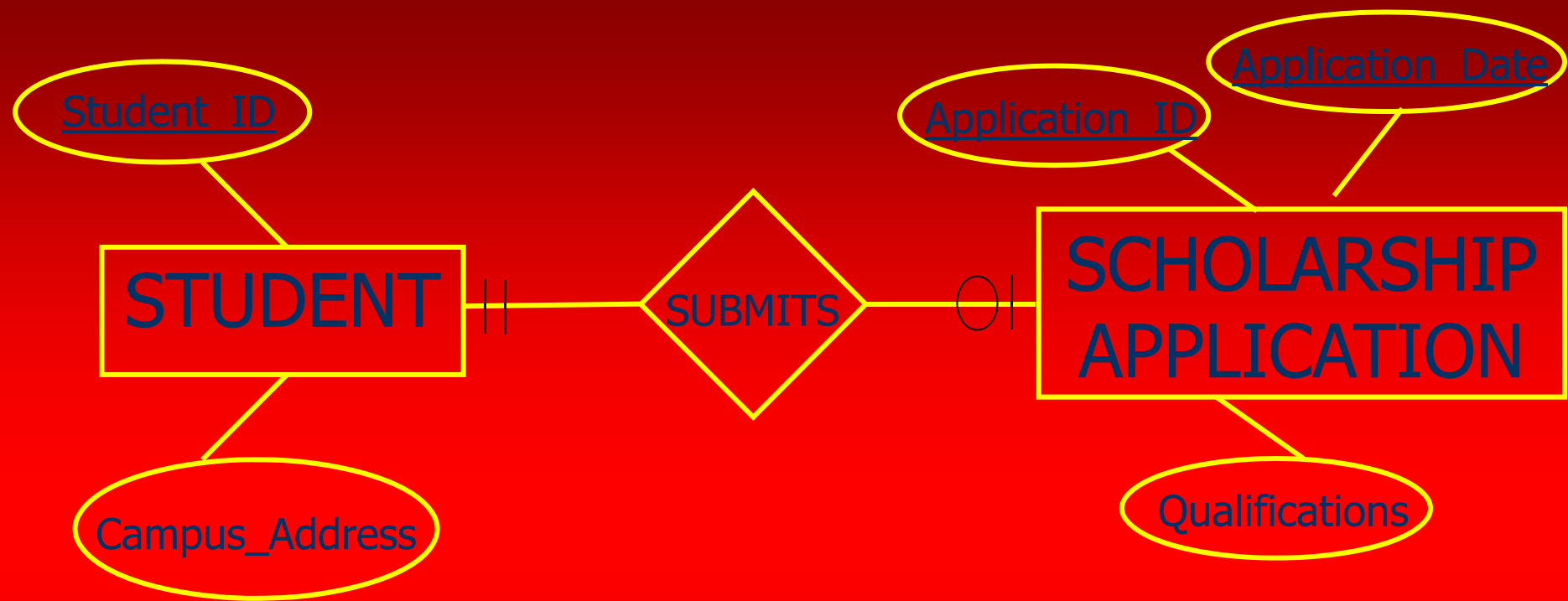
Denormalization

- In general, may decompose one logical relations into separate physical records, combine some, or do both
- See the situation, it is an option that you adopt purely for efficiency

Denormalization Situation 1

- Merge two ET into one with one to one relationship
- Even if one of the ETs is optional, so joining can lead to wastage of storage, however if two accessed together very frequently then merging might be a wise decision

One to One Relationship



STUDENT

<u>Student_ID</u>	Campus_Address
-------------------	----------------

APPLICATION

<u>Application_ID</u>	<u>Application_Date</u>	Qualifications	<u>Student_ID</u>
-----------------------	-------------------------	----------------	-------------------

Normalized Relations:

STUDENT

<u>Student_ID</u>	Campus_Address
-------------------	----------------

APPLICATION

<u>Application_ID</u>	Application_Date	Qualifications	<u>Student_ID</u>
-----------------------	------------------	----------------	-------------------



Denormalized Relations:

STUDENT

<u>Student_ID</u>	Campus_Address	Application_Date	Qualifications
-------------------	----------------	------------------	----------------

Denormalization Situation 2

- Many to many binary relationships mapped to three relations
- Queries needing data from two participating ETs need joining of three relations, that is expensive

Denormalization Situation 2

- The relation created against relationship is merged with one of the relations created against participating ETs
- Reduces one join

Many to many relationship

- EMP(empId, eName, pjId, sal)
- PROJ(pjId, pjName)
- WORK(empId, pjId, dtHired, sal)

Many to many relationship

- EMP(empId, eName, pjId, sal)
- PROJ(empId, pjId, pjName, dtHired, sal)

Denormalization Situation 3

- Reference data: one to many situation when the ET on one side does not participate in any other relationship, then many side ET is appended with reference data rather than the FK

One to Many Relationships

➤ Student – Hobby(slide)

Partitioning

- Denormalization leads to merging different relations, whereas partitioning splits same relation into two
- Two types are possible horizontal and vertical

Horizontal Partitioning

- Table is split on the basis of rows
- The idea is that it is more efficient to process a table with small number of rows rather than a large table

Horizontal Partitioning

- Also helps in the maintenance of tables, like security, authorization, backup
- Even can be placed on different disks to reduce disk contention

Horizontal Partitioning

Some of the horizontal partitioning types:

- Range partitioning
- Hash partitioning
- List partitioning

Database Management System

Lecture - 23

Database Management System

Lecture - 24

Vertical Partitioning

- Same table is split into different physical records; depending on the nature of accesses
- PK is repeated in all vertical partitions of a table to get the original table

Vertical Partitioning

- STD(stId, sName, sAdr, sPhone, cgpa, prName, school, mtMrks, mtSubs, clgName, intMarks, intSubs, dClg, bMarks, bSubs)
- STD(stId, sName, sAdr, sPhone, cgpa, prName)
- STDACD(sId, school, mtMrks, mtSubs, clgName, intMarks, intSubs, dClg, bMarks, bSubs)

Replication

- Final form of denormalization
- Data duplicated
- Increases the access speed and failure damage
- Entire table or part of table can be replicated

Replication Example

- Generally adopted where updation is not very frequent
- In PROJ-EMP we replicate the data with both the PROJ and EMP tables so on the cost of extra storage both tables have the related data

Clustering Files

- Clustering means to place records from different tables to place in adjacent physical locations, called clusters
- Increases efficiency since related records are placed close to each other

Clustering Files

- Clustering is also suitable to relatively static situations
- Define cluster, define the key of the cluster, include the tables into the cluster while creating associating the key with it

Other Related Issues

➤ To be discussed later during
storage concepts discussion

- File structures
- Indexes
- RAID

Summary of Physical DB Design

Structured Query Language (SQL)

Introduction

- Also pronounced as “Sequel”
- A de-facto standard for relational DBMS
- Standard accepted by bodies like ANSI and ISO

Introduction

- First commercial DBMS that support SQL in 1979 was Oracle
- Another form of query language is Query-by-Example (QBE) supported by PC-based DBMSs

History

- Relational data model introduced by Codd in 1970
- A relational DBMS project started by IBM was system R that included sequel as manipulation language

History

- First commercial DBMS launched was Oracle in 1979
- Other early DBMSs DB2, INGRES
- ANSI and ISO defined first standard of SQL in 1986 known as SQL-86 that was further extended to SQL-89

History

- Later two more standards known as SQL-92 and SQL-99 were defined
- Almost all of the current DBMSs support SQL-92 and many support SQL-99 partially or completely
- SQL today is supported in all sort of machines

Benefits of Standard SQL

- Reduced training cost
- Application portability
- Application longevity
- Reduced dependence on a single vendor
- Cross-system communication

SQL

- SQL is used for any type of interaction with the database through DBMS
- Right from creating database, tables, views, users
- Insertion, updation, deletion of data in the database

SQL and DBMSs

- Implementation of SQL always includes the flavor of the particular DBMS, like in names of data types
- Proprietary SQLs exist, like T-SQL and PL-SQL

MS SQL Server

- The DBMS for our course is Microsoft's SQL Server 2000 desktop edition
- Two main tools Query Analyzer and Enterprise Manager; both can be used
- For SQL practice we will use QA

Terminology of SS

- Instance of SQL Server
- Instance contains different objects like databases, security, DTS etc.
- Then databases object of SS contains multiple databases, each database contains multiple objects like tables, views, users, roles etc.

Working in SS

- After installing SS, we will create database using create command
- After this we will create tables and other objects within this database

Database Management System

Lecture - 24

Database Management System

Lecture - 25

Rules of the Format

➤ Writing general format

- – Reserved words in capital
- User-defined identifiers in lowercase
- Valid identifiers
- Optionals in []
- Curly braces means required items
- | means choices
- [,.....n] means n items separated by comma

Rules of the Format

➤ Writing general format

- Reserved words in capital

-  – User-defined identifiers in lowercase

- Valid identifiers

- Optionals in []

- Curly braces means required items

- | means choices

- [,.....n] means n items separated by comma

Rules of the Format

➤ Writing general format

- Reserved words in capital
- User-defined identifiers in lowercase
- – Valid identifiers
- Optionals in []
- Curly braces means required items
- | means choices
- [,.....n] means n items separated by comma

Rules of the Format

➤ Writing general format

- Reserved words in capital
- User-defined identifiers in lowercase
- Valid identifiers
- – Optionals in []
- Curly braces means required items
- | means choices
- [,.....n] means n items separated by comma

Rules of the Format

➤ Writing general format

- Reserved words in capital
- User-defined identifiers in lowercase
- Valid identifiers
- Optionals in []
- – Curly braces means required items
- | means choices
- [,.....n] means n items separated by comma

Rules of the Format

➤ Writing general format

- Reserved words in capital
- User-defined identifiers in lowercase
- Valid identifiers
- Optionals in []
- Curly braces means required items
- – | means choices
- [,.....n] means n items separated by comma

Rules of the Format

➤ Writing general format

- Reserved words in capital
- User-defined identifiers in lowercase
- Valid identifiers
- Optionals in []
- Curly braces means required items
- | means choices
- – [,.....n] means n items separated by comma

Example

- SELECT [ALL|DISTINCT]
{*|select_list} FROM
{table|view[,...n]}
- Select * from std

Data Types in SS

➤ Integer

- Bit (0 or 1)
- Tinyint (0 through 255)
- Smallint -32768 to 32767
- Int -2^{31} to $2^{31}-1$ (10 digits)
- Bigint -2^{63} to $2^{63}-1$ (19 digits)

➤ Decimal or numeric

Data Types in SS

- Text: handles textual data
 - char: by default 30 characters, max 8000
 - varchar: variable length text, max 8000
 - Text: variable length automatically
 - nchar, nvarchar, ntext
- Money: handles monetary data
 - smallmoney: 6 digits, 4 decimal
 - money: 15 digits, 4 decimal

Data Type in SS

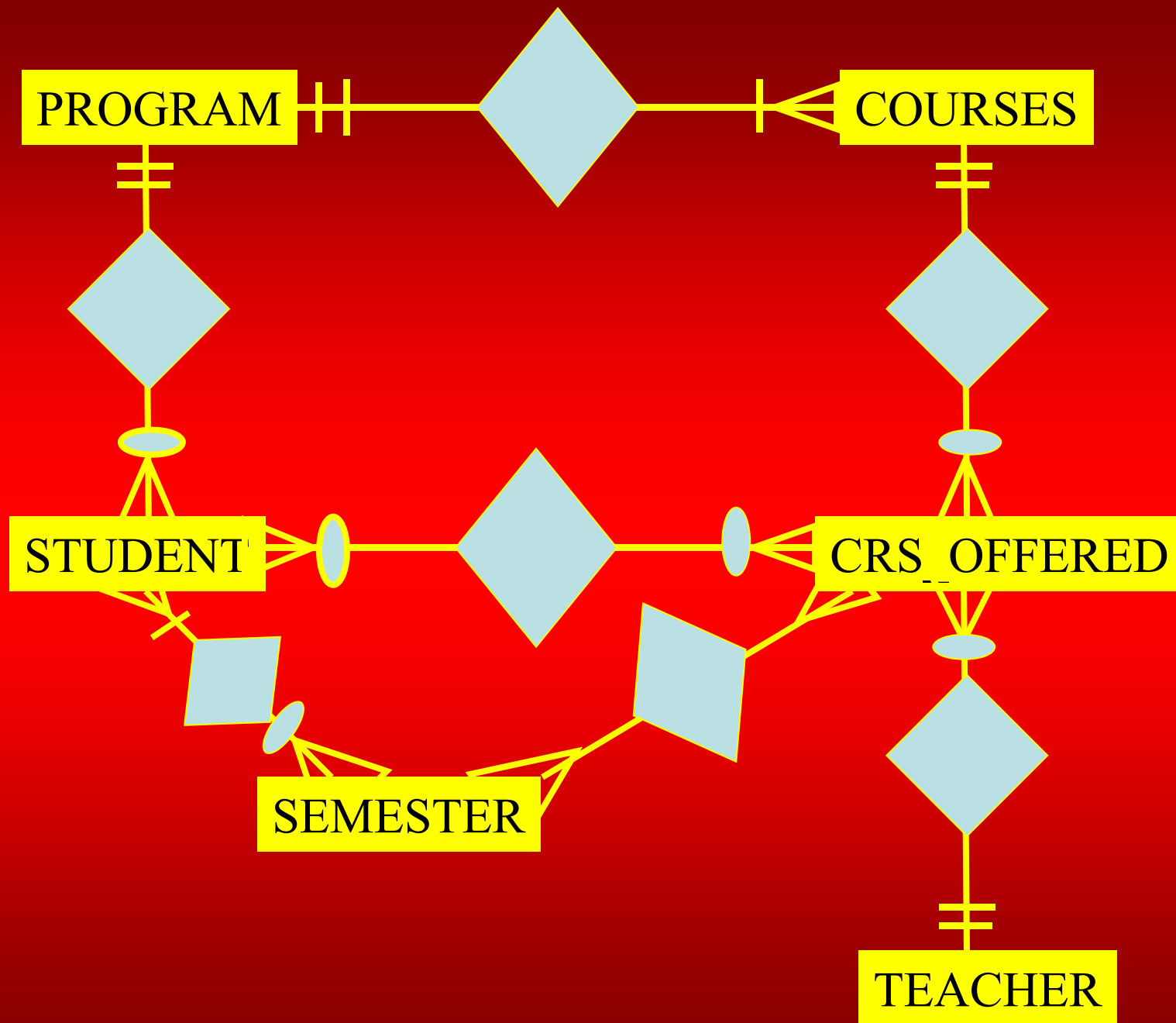
➤ Floating point

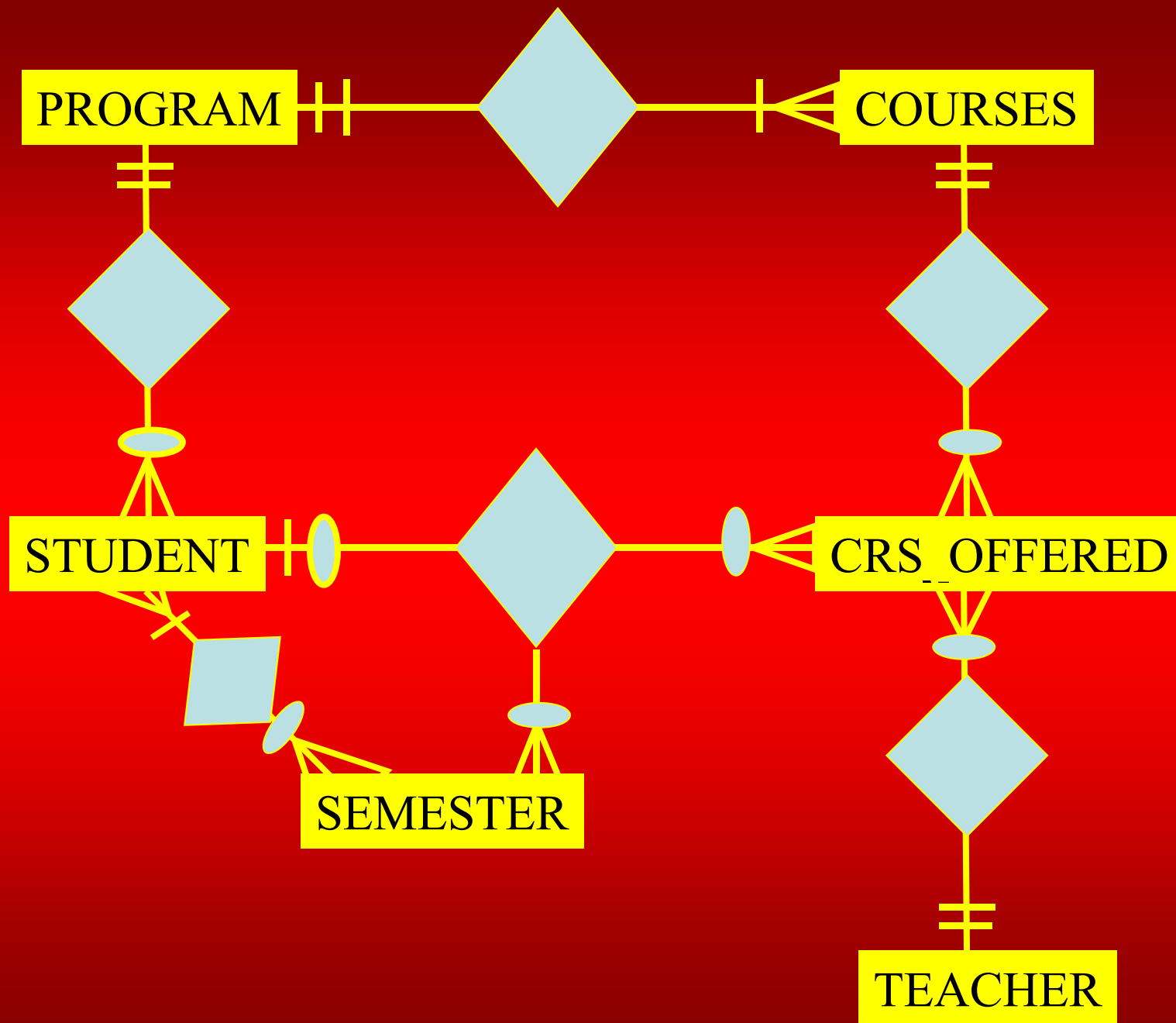
- Float
- Real

➤ Date

- Smalldatetime
- datetime

Example Database





PROGRAM (prName, totSem, prCredits)
COURSE (crCode, crName, crCredits, prName)
SEMESTER (semName, stDate, endDate)
CROFRD (crCode, semName, facId)
FACULTY (facId, fName, fQual, fSal, rank)
STUDENT (stId, stName, stFName, stAdres,
stPhone, prName, curSem, cgpa)
ENROLL (stId, crCode, semName, mTerm,
sMrks, fMrks, totMrks, grade, gp)
SEM_RES (stId, semName, totCrs, totCrdts,
totGP, gpa)

SQL Commands Types

- SQL commands are categorized in three broad classes
- Data definition language (DDL)
- Data manipulation language (DML)
- Data control language (DCL)

Data Manipulation Language

- DML commands are used to create, alter or drop tables or other database objects
- DBMSs provide graphic interface for creating tables

Database Management System

Lecture - 26

DDL

➤ Create

➤ Alter

➤ Drop

Create Command

➤ Creating database

CREATE DATABASE db_name

Create database exam

➤ Next step is to create tables

➤ Two approaches:

- Through SQL Create command
- Through Enterprise Manager

Create Table Command

- Create table command is used to:
 - Create a table
 - Define attributes of the table with data types
 - Define different constraints on attributes, like primary and foreign keys, check constraint, not null, default value etc.

Format of Create Table

➤ CREATE TABLE

```
[ database_name.[ owner ] . | owner. ] table_name  
( { < column_definition >  
    | column_name AS computed_column_expression  
    | < table_constraint >  
    }  
    | [ { PRIMARY KEY | UNIQUE } [ ,...n ] ]  
)
```

Chevrons Explained

➤ $\langle \text{column_definition} \rangle ::= \{ \text{column_name data_type} \}$
[DEFAULT *constant_expression*]
[$\langle \text{column_constraint} \rangle$] [...*n*]

Chevrons Explained

➤ `< column_constraint > ::= [CONSTRAINT constraint_name]`
 `{ [NULL | NOT NULL]`
 `| [{ PRIMARY KEY | UNIQUE }`
 `]`
 `| [[FOREIGN KEY]`
 `REFERENCES ref_table [(ref_column)]`
 `[ON DELETE { CASCADE | NO ACTION }]`
 `[ON UPDATE { CASCADE | NO ACTION }]`
 `]`
 `| CHECK(logical_expression)`
 `}`

Create Table Command

➤ CREATE TABLE Program (
 prName char(4),
 totSem tinyint,
 prCredits smallint)

Create Table Command

```
➤ CREATE TABLE Student  
  (stId char(5),  
   stName char(25),  
   stFName char(25),  
   stAdres text,  
   stPhone char(10),  
   prName char(4)  
   curSem smallint,  
   cgpa real)
```

Create Table with Constraint

```
➤ CREATE TABLE Student (  
    stId char(5) constraint ST_PK primary key  
        constraint ST_CK check (stId like  
            'S[0-9][0-9][0-9][0-9]'),  
    stName char(25) not null,  
    stFName char(25),  
    stAdres text,  
    stPhone char(10),  
    prName char(4),  
    curSem smallint default 1,  
    cgpa real)
```


stId char(5) constraint ST_PK primary key

stId char(5) primary key

Create Table with Constraint

```
➤ CREATE TABLE Student  
  (stId char(5) primary key  
    check (stId like 'S[0-9][0-9][0-9][0-9]'),  
  stName char(25) not null,  
  stFName char(25),  
  stAdres text,  
  stPhone char(10),  
  prName char(4),  
  curSem tinyint default 1,  
  cgpa real)
```

Create Table with Constraint

```
➤ CREATE TABLE semester (  
    semName char(5) primary key,  
    stDate smalldatetime,  
    endDate smalldatetime,  
    constraint ck_date_val check (datediff(week,  
    stDate, endDate) between 14 and 19))
```

Database Management System

Lecture - 26

Database Management System

Lecture - 27

Alter Table Statement

- Purpose is to make changes in the definition of a table already created through Create statement
- Can add, drop attributes or constraints, activate or deactivate constraints; the format is

➤ ALTER TABLE *table*

```
{ [ ALTER COLUMN column_name
  { new_data_type [ ( precision [ , scale ] ) ]
    [ NULL | NOT NULL ]
  ]
| ADD
  { [ < column_definition > ]
    | column_name AS computed_column_expression
  } [ ,...n ]
| DROP
  { [ CONSTRAINT ] constraint_name
    | COLUMN column } [ ,...n ]
| { CHECK | NOCHECK } CONSTRAINT
  { ALL | constraint_name [ ,...n ] }
}
```

Alter Table Command

- ALTER TABLE Student
add constraint fk_st_pr
foreign key (prName) references
Program (prName)
- ALTER TABLE program
add constraint ck_totSem
check (totSem < 9)

Removing or Changing Attribute

➤ ALTER TABLE student

ALTER COLUMN stFName char(20)

➤ Alter table student

drop column curSem

➤ Alter table student

drop constraint ck_st_pr

Removing Rows and Tables

- TRUNCATE TABLE table_name
- Truncate table class
- Delete can also be used
- DROP TABLE table_name

Data Manipulation Language

➤ Insert

➤ Select

➤ Update

Insert Statement

➤ To insert records into tables

➤ INSERT [INTO] *table*

{ [(*column_list*)]

{ VALUES

({ DEFAULT | NULL | *expression* } [,...*n*])

}

}

| DEFAULT VALUES

Database Management System

Lecture - 27

Database Management System

Lecture - 28

Insert

➤ COURSE (crCode, crName, crCredits, prName)

```
INSERT INTO course VALUES ('CS-211',  
    'Operating Systems', 4, 'MCS')
```

```
INSERT INTO course (crCode, crName)  
    VALUES ('CS-316', Database Systems')
```

```
INSERT INTO course ('MG-103', 'Intro to  
    Management', NULL, NULL)
```

stId	stName	prName	cgpa
S1020	Sohail Dar	MCS	2.8
S1038	Shoaib Ali	BCS	2.78
S1015	Tahira Ejaz	MCS	3.2
S1034	Sadia Zia	BIT	
S1018	Arif Zia	BIT	3.0

prName	totSem	prCrdts
BCS	8	134
BIT	8	132
MBA	4	65
MCS	4	64

STUDENT

PROGRAM

ENROLL

crCode	crTitle	crCrdts	prName
CS-616	Intro to Database Systems	4	MCS
MG-314	Money & Capital Market	3	BIT
CS-516	Data Structures and Algos	4	MCS
MG-105	Introduction to Accounting	3	MBA

COURSE

stId	crCode	semName	mTerm	sMrks	fMrks	totMrks	grade	gPoint
S1020	CS-616	F04	25	12.5	35	72.5	B-	2.7
S1018	MG-314	F04	26.5	10.5	39	76	B	3.1
S1015	CS-616	F04	30	10	40	80	B+	3.5
S1015	CS-516	F04	32	12	42	86	A-	4.1

Select Statement

- Maximum used command in DML
- Used not only to select certain rows but also the columns
- Also used for different forms of product, that is, different joins

Select

➤ Selecting rows from one or more tables

➤ `SELECT {*|col_name[,...n]}`
`FROM table_name`

Select

Q: Get the data about students

```
SELECT * FROM student
```

	stId	stName	stFName	stAdres	stPhone	prName	curSem	cgpa
1	S0123	Amjad	Hussain	8-SD Lahore	234322	MCS	1	NULL
2	S1012	Amjad	Rehan	I8 Ibd	5456754	MCS	2	2.3
3	S1015	Tahira Ejaz	Nek Muhammad	AJ Road	4323456	MCS	3	3.2
4	S1018	Arif Zia	Zia Khan	GM Rawal...	4356488	BIT	2	2.8
5	S1020	Suhai Dar	Nek Muhammad	I-8 Isla...	5523240	MCS	2	3.2
6	S1021	M. Ali	M. Saad	JT Lahore	544325	MBA	2	0.0
7	S1034	Sadia Zia	NULL	NULL	NULL	BIT	1	NULL
8	S1038	Shoaib Ali	Rahmat Ali	G-6 Isla...	5343240	BCS	2	3.2

Select

Q: Give the names of the students
with the program name

```
SELECT stName, prName  
FROM student
```

	stName	prName
1	Amjad	MCS
2	Amjad	MCS
3	Tahira Ejaz	MCS
4	Arif Zia	BIT
5	Suhai Dar	MCS
6	M. Ali	MBA
7	Sadia Zia	BIT
8	Shoaib Ali	BCS

Attribute Alias

- Another Name “Urf” for an attribute

```
SELECT {*|col_name [[AS] alias]  
[, ...n]} FROM tab_name
```

```
SELECT stName as 'Student Name',  
prName 'Program' FROM Student
```

	Student Name	Program
1	Amjad	MCS
2	Amjad	MCS
3	Tahira Ejaz	MCS
4	Arif Zia	BIT
5	Suhai Dar	MCS
6	M. Ali	MBA
7	Sadia Zia	BIT
8	Shoaib Ali	BCS

Expression in Select

- In the column list we can also give the expression; value of the expression is computed and displayed

Expression Example

Q: Display the total sessional marks of each student obtained in each subject

Select stId, crCode, mTerm +
sMrks 'Total out of 50' from
enroll

	stId	crCode	Total out of 50
1	S1015	CS-516	44
2	S1015	CS-616	40
3	S1018	MG-314	36
4	S1020	CS-616	37

Expression Example

Q: List the names of the students and the program they are enrolled in, in the format “std studies in prg”

```
SELECT stName + ' studies in ' +  
prName FROM student
```

1	Amjad	studies in MCS
2	Amjad	studies in MCS
3	Tahira Ejaz	studies in MCS
4	Arif Zia	studies in BIT
5	Suhai Dar	studies in MCS
6	M. Ali	studies in MBA
7	Sadia Zia	studies in BIT
8	Shoaib Ali	studies in BCS

Select Distinct

- The DISTINCT keyword is used to return only distinct (different) values

Select Distinct

- Just add a DISTINCT keyword to the SELECT statement
- `SELECT DISTINCT
column_name(s) FROM table_name`

Select Distinct

Q: Get the program names in which students are enrolled

```
SELECT prName FROM Student
```


	programs
1	MCS
2	MCS
3	MCS
4	BIT
5	MCS
6	MBA
7	BIT
8	BCS

Select Distinct

Q: Get the program names in which students are enrolled

```
SELECT prName FROM Student
```

```
SELECT DISTINCT prName FROM  
Student
```

	programs		programs
1	BCS	1	MCS
2	BIT	2	MCS
3	MBA	3	MCS
4	MCS	4	BIT
		5	MCS
		6	MBA
		7	BIT
		8	BCS

Placing the Checks on Rows

- Limit rows or to select certain rows, like,
- Programs of certain length, credits; students with particular names, programs, age, places etc.

The WHERE Clause

- We used names to limit columns, but rows cannot be named due to the dynamicity
- We limit the rows using conditions

The WHERE Clause

Conditions are defined on the values of one or more attributes from one or more tables and placed in the WHERE clause

Database Management System

Lecture - 28

Database Management System

Lecture - 29

Where: Format

➤ SELECT [ALL|DISTINCT]

{*|column_list [alias][,....n]}

FROM table_name

[WHERE <search_condition>]

```

< search_condition > ::=
{   [ NOT ] < predicate > | ( < search_condition > ) }
    [ { AND | OR } [ NOT ] { < predicate > |
        ( < search_condition > ) } ]
}   [ ,...n ]
< predicate > ::=
{   expression { = | < > | != | > | > = | ! > | < | < = | ! < }
        expression
    | string_expression [ NOT ] LIKE string_expression
    | expression [ NOT ] BETWEEN expression AND
        expression
    | expression IS [ NOT ] NULL
    | expression [ NOT ] IN ( subquery | expression [ ,...n ] )
    | expression { = | < > | != | > | > = | ! > | < | < = | ! < }
        { ALL | SOME | ANY } ( subquery )
    | EXISTS ( subquery )
}

```

Where Example

Q: Display all courses of the MCS
program

Select crCode, crName, prName
from course

where prName = 'MCS'

	crCode	crName	prName
1	CS-504	Analysis of Algorithm	MCS
2	CS-511	Operating System	MCS
3	CS-516	Data Structures and Algos	MCS
4	CS-616	Intro to Database Systems	MCS
5	MT-305	Linear Algebra	MCS

	crName	(No column name)	crCode	(No column name)	prName	
1	Analysis of Algorithm	whose code is	CS-504	is offered in	MCS	
2	Operating System	whose code is	CS-511	is offered in	MCS	
3	Data Structures and Algos	whose code is	CS-516	is offered in	MCS	
4	Intro to Database Systems	whose code is	CS-616	is offered in	MCS	
5	Linear Algebra	whose code is	MT-305	is offered in	MCS	

Not Operator

Inverses the predicate's value

Q: List the course names offered to
programs other than MCS

```
SELECT crCode, crName, prName  
FROM course  
WHERE not (prName = 'MCS')
```

	crCode	crName	prName
1	CS-105	Intro to Programming	BCS
2	CS-216	Data Structures	BCS
3	CS-316	Database Systems	BCS
4	MG-105	Intro to Accounting	EBA
5	MG-314	Money & Capital Mark	BIT
6	MG-505	Intro to Accounting	MBA

```
SELECT crCode, crName, prName  
FROM course  
WHERE (prName != 'MCS')
```


The BETWEEN Operator

Checks the value in a range

Q: List the IDs of the students with course codes having marks between 70 and 80

```
SELECT stId, crCode, totMrks
```

```
From ENROLL
```

```
WHERE totMrks between 70 and 80
```

	stId	crCode	totMrks
1	S1015	CS-616	80
2	S1018	MG-314	75
3	S1020	CS-616	76

The IN Operator

➤ Checks in a list of values

Q: Give the course names of MCS and BCS programs

```
SELECT crName, prName
```

```
From course
```

```
Where prName in ('MCS', 'BCS')
```

	crName	prName
1	Intro to Programming	BCS
2	Data Structures	BCS
3	Database Systems	BCS
4	Analysis of Algorithm	MCS
5	Operating System	MCS
6	Data Structures and Algos	MCS
7	Intro to Database Systems	MCS
8	Linear Algebra	MCS

```
SELECT crName, prName
```

```
From course
```

```
Where (prName = 'MCS') OR  
(prName = 'BCS')
```

Like Operator

Q: Display the names and credits of
CS programs

```
SELECT crName, crCrdts, prName  
FROM course  
WHERE prName like '%CS'
```

	crName	crCrdts	prName
1	Intro to Programming	4	BCS
2	Data Structures	4	BCS
3	Database Systems	4	BCS
4	Analysis of Algorithm	4	MCS
5	Operating System	4	MCS
6	Data Structures and Algos	4	MCS
7	Intro to Database Systems	3	MCS
8	Linear Algebra	3	MCS

“Order By” Clause

- Sorts the rows in a particular order

SELECT *select_list*

FROM *table_source*

[WHERE *search_condition*]

[ORDER BY *order_expression* [ASC | DESC]

[.....n]

]

Database Management System

Lecture - 29

Database Management System

Lecture - 30

Order By Example

Q: Display the students' data in the ascending order of names

```
SELECT * from STUDENT  
ORDER BY stName
```

	stId	stName	stFName	stAdres	stPhone	prName	curSem	cgpa
1	S0123	Amjad	Hussain	8-SD Lahore	234322	MCS	1	NULL
2	S1012	Amjad	Rehan	I8 Ibd	5456754	MCS	2	2.3
3	S1018	Arif Zia	Zia Khan	GM Rawalpindi	4356488	BIT	2	2.8
4	S1021	M. Ali	M. Saad	JT Lahore	544325	MBA	2	2.8
5	S1034	Sadia Zia	NULL	NULL	NULL	BIT	1	NULL
6	S1038	Shoaib Ali	Rahmat Ali	G-6 Islamabad	5343240	BCS	2	3.2
7	S1020	Suhai Dar	Nek Muhammad	I-8 Islamabad	5523240	MCS	2	3.2
8	S1015	Tahira Ejaz	Nek Muhammad	AJ Road	4323456	MCS	3	3.2

Practice Query

Display the name and cgpa of students for all those students who are in second or above semester in descending order of names

Functions in SQL

- Built-in functions are pre-written programs to perform specific tasks
- Accept certain arguments and return the result

Categories of Functions

- Depending on the arguments and the return value, categorized
 - Mathematical (ABS, ROUND, SIN, SQRT)
 - String (LOWER, UPPER, SUBSTRING, LEN)
 - Date (DATEDIFF, DATEPART, GETDATE())
 - System (USER, DATALENGTH, HOST_NAME)
 - Conversion (CAST, CONVERT)

Using Functions

```
SELECT upper(stName), lower(stFName),  
       stAdres, len(convert(char, stAdres)),  
FROM student
```


	(No column name)	(No column name)	stAdres	Adress Length
1	AMJAD	hussain	8-SD Lahore	11
2	AMJAD	rehan	I8 Ibd	6
3	TAHIRA EJAZ	nek muhammad	AJ Road	7
4	ARIF ZIA	zia khan	GM Rawalpindi	13
5	SUHAI DAR	nek muhammad	I-8 Islamabad	13
6	M. ALI	m. saad	JT Lahore	9
7	SADIA ZIA	NULL	NULL	NULL
8	SHOAIB ALI	rahmat ali	G-6 Islamabad	13

Aggregate Functions

- Operate on a set of rows and return a single value, like, AVG, SUM, MAX, MIN, STDEV
- Attribute list cannot contain other attributes if an aggregate function is being used

Aggregate Function Example

- `SELECT avg(cgpa) as 'Average CGPA', max(cgpa) as 'Maximum CGPA' from student`
- Output is.....

	Average CGPA	Maximum CGPA
1	2.9166666666666665	3.2

Aggregate Function Example

- `SELECT avg(cgpa) as 'Average CGPA', max(cgpa) as 'Maximum CGPA' from student`
- `SELECT convert(decimal(5,2), avg(cgpa)) as 'Average CGPA', max(cgpa) as 'Maximum CGPA' from student`

	Average CGPA	Maximum CGPA
1	2.9166666666666665	3.2



	Average CGPA	Maximum CGPA
1	2.92	3.2

Group By Clause

- `SELECT stName, avg(cgpa) as 'Average CGPA', max(cgpa) as 'Maximum CGPA' from student`
- `SELECT prName, max(cgpa) as 'Max CGPA', min(cgpa) as 'Min CGPA' FROM student GROUP BY prName`

	prName	cgpa
1	MCS	NULL
2	MCS	2.3
3	MCS	3.2
4	BIT	2.8
5	MCS	3.2
6	MBA	2.8
7	BIT	NULL
8	BCS	3.2

	prName	Max CGPA	Min CGPA
1	BCS	3.2	3.2
2	BIT	2.8	2.8
3	MBA	2.8	2.8
4	MCS	3.2	2.3

HAVING Clause

- We can restrict groups by using having clause; groups satisfying having condition will be selected

HAVING Clause

```
SELECT select_list  
[ INTO new_table ]  
FROM table_source  
[ WHERE search_condition ]  
[ GROUP BY group_by_expression ]  
[ HAVING search_condition ]  
[ ORDER BY order_expression [ ASC |  
DESC ] ]
```

HAVING Example

```
SELECT prName, min(cgpa), max(cgpa)  
FROM student  
GROUP BY prName  
HAVING max(cgpa) > 3
```

	prName	(No column name)	(No column name)
1	BCS	3.2	3.2
2	MCS	2.3	3.25

Where and having can be combined

Accessing Multiple Tables

- Cartesian Product
- Inner join
- Outer Join
- Full outer join
- Semi Join
- Natural Join

Cartesian Product

- No specific command; Select is used
- Simply give the names of the tables involved; Cartesian product will be produced

Cartesian Product

- Produces $m \times n$ rows
- Select * from program, course

	prName	totSem	prCredits	crCode	crName	crCrdts	prName	
1	BBA	8	130	CS-105	Intro to Programming	4	BCS	
2	BBA	8	130	CS-216	Data Structures	4	BCS	
3	BBA	8	130	CS-316	Database Systems	4	BCS	
4	BBA	8	130	CS-504	Analysis of Algorithm	4	MCS	
5	BBA	8	130	CS-511	Operating System	4	MCS	
6	BBA	8	130	CS-516	Data Structures and Algos	4	MCS	
7	BBA	8	130	CS-616	Intro to Database Systems	3	MCS	
8	BBA	8	130	MG-103	Intro to Management	NULL	NULL	
9	BBA	8	130	MG-105	Intro to Accounting	3	BBA	
10	BBA	8	130	MG-314	Money & Capital Mark	3	BIT	
11	BBA	8	130	MG-505	Intro to Accounting	3	MBA	
12	BBA	8	130	MT-305	Linear Algebra	3	MCS	
13	BCS	8	134	CS-105	Intro to Programming	4	BCS	
14	BCS	8	134	CS-216	Data Structures	4	BCS	
15	BCS	8	134	CS-316	Database Systems	4	BCS	
16	BCS	8	134	CS-504	Analysis of Algorithm	4	MCS	
17	BCS	8	134	CS-511	Operating System	4	MCS	

Cartesian Product

- Certain columns can be selected, same column name needs to be qualified
- Similarly can be applied to more than one tables, and even can be applied on the same table

```
SELECT * from Student, class, program
```

Database Management System

Lecture - 30

Database Management System

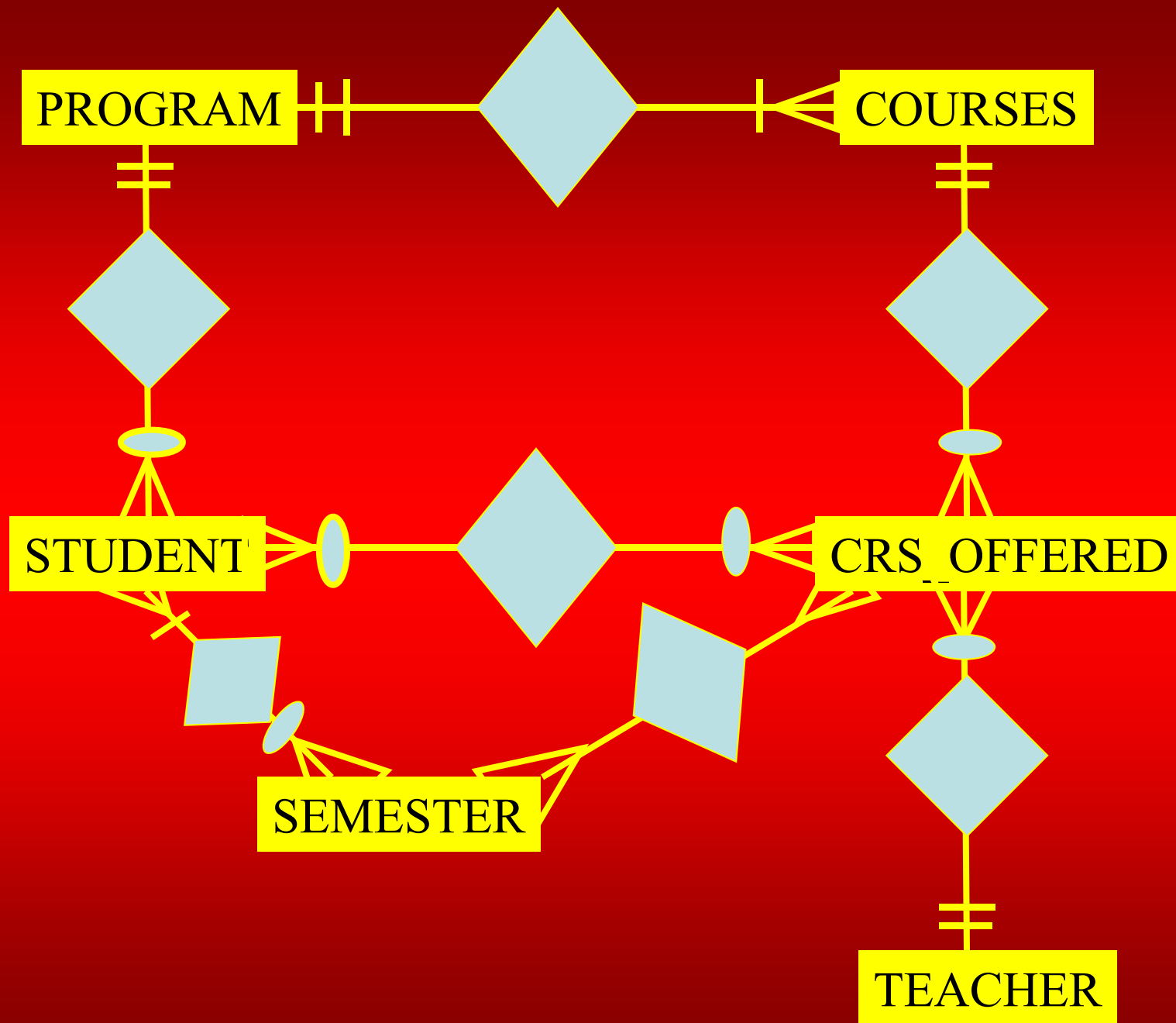
Lecture - 31

Inner join

- Only those rows from both tables are merged that have same value for the common attribute; equijoin
- Implemented by different methods

Inner join

- Common attributes need not to have same name, but must have same domain
- Applied generally between tables having referential integrity between them



	crCode	crName	crCrds	prName
1	CS-105	Intro to Programmin...	4	BCS
2	CS-216	Data Structures	4	BCS
3	CS-316	Database Systems	4	BCS
4	CS-504	Analysis of Algorit...	4	MCS
5	CS-511	Operating System	4	MCS
6	CS-516	Data Structures and...	4	MCS
7	CS-616	Intro to Database S...	3	MCS
8	MG-103	Intro to Management...	NULL	NULL
9	MG-105	Intro to Accounting...	3	BBA
10	MG-314	Money & Capital Mar...	3	BIT
11	MG-505	Intro to Accounting...	3	MBA
12	MT-305	Linear Algebra	3	MCS

	prName	totSem	prCredits
1	BBA	8	130
2	BCS	8	134
3	BIT	8	132
4	MEA	4	64
5	MCS	4	64

Inner Join

➤ SELECT *

FROM course INNER JOIN program

ON course.prName = program.prName

➤ Select * FROM

Course c inner join program p

ON c.prName = p.prName

	crCode	crName	crCrdts	prName	prName	totSem	prCredits
1	CS-105	Intro to Programming	4	BCS	BCS	8	134
2	CS-216	Data Structures	4	BCS	BCS	8	134
3	CS-316	Database Systems	4	BCS	BCS	8	134
4	CS-504	Analysis of Algorithm	4	MCS	MCS	4	64
5	CS-511	Operating System	4	MCS	MCS	4	64
6	CS-516	Data Structures and Algos	4	MCS	MCS	4	64
7	CS-616	Intro to Database Systems	3	MCS	MCS	4	64
8	MG-105	Intro to Accounting	3	BBA	BBA	8	130
9	MG-314	Money & Capital Mark	3	BIT	BIT	8	132
10	MG-505	Intro to Accounting	3	MBA	MBA	4	64
11	MT-305	Linear Algebra	3	MCS	MCS	4	64

Inner Join

➤ Can also be performed using the where clause, like

```
SELECT * FROM course, program
```

```
WHERE
```

```
    course.prName = program.prName
```

Outer Join

- Inner join plus the missing rows from one or more tables
- Left, Right and Full Outer Joins

Outer Joins

- Right Outer Join: Inner join plus rows from the non-matching rows from right table
- Left outer join performs the same thing but missing rows of the left side table

Outer Joins

- A Left Outer Join B = B Right Outer Join A
- Missing values are replaced with NULLs
- Full Outer Join: Inner join plus the non-matching rows from both tables

Outer Join Examples

- Select * from course c LEFT OUTER JOIN program p on c.prName = p.prName
- Select * from program p RIGHT OUTER JOIN course c on c.prName = p.prName

	crCode	crName	crCrdts	prName	prName	totSem	prCredits
1	CS-105	Intro to Programming	4	BCS	BCS	8	134
2	CS-216	Data Structures	4	BCS	BCS	8	134
3	CS-316	Database Systems	4	BCS	BCS	8	134
4	CS-504	Analysis of Algorithm	4	MCS	MCS	4	64
5	CS-511	Operating System	4	MCS	MCS	4	64
6	CS-516	Data Structures and Algos	4	MCS	MCS	4	64
7	CS-616	Intro to Database Systems	3	MCS	MCS	4	64
8	MG-103	Intro to Management	NULL	NULL	NULL	NULL	NULL
9	MG-105	Intro to Accounting	3	BBA	BBA	8	130
10	MG-314	Money & Capital Mark	3	BIT	BIT	8	132
11	MG-505	Intro to Accounting	3	MBA	MBA	4	64
12	MT-305	Linear Algebra	3	MCS	MCS	4	64

Outer Join Examples

- Select * from program p LEFT
OUTER JOIN course c on p.prName =
c.prName
- Select * from course c RIGHT
OUTER JOIN program p on c.prName
= p.prName

	crCode	crName	crCrds	prName	prName	totSem	prCredits
1	MG-105	Intro to Accounting...	3	BBA	BBA	8	130
2	CS-105	Intro to Programmin...	4	BCS	BCS	8	134
3	CS-216	Data Structures	4	BCS	BCS	8	134
4	CS-316	Database Systems	4	BCS	BCS	8	134
5	MG-314	Money & Capital Mar...	3	BIT	BIT	8	132
6	MG-505	Intro to Accounting...	3	MBA	MBA	4	64
7	CS-504	Analysis of Algorit...	4	MCS	MCS	4	64
8	CS-511	Operating System	4	MCS	MCS	4	64
9	CS-516	Data Structures and...	4	MCS	MCS	4	64
10	CS-616	Intro to Database S...	3	MCS	MCS	4	64
11	MT-305	Linear Algebra	3	MCS	MCS	4	64
12	NULL	NULL	NULL	NULL	MIT	4	62

Full Outer Join

```
SELECT * FROM program p
FULL OUTER JOIN
course c
ON p.prName = c.prName
```

	prName	totSem	prCredits	crCode	crName	crCrds	prName
1	NULL	NULL	NULL	MG-103	Intro to Management	NULL	NULL
2	BBA	8	130	MG-105	Intro to Accounting	3	BBA
3	BCS	8	134	CS-105	Intro to Programming	4	BCS
4	BCS	8	134	CS-216	Data Structures	4	BCS
5	BCS	8	134	CS-316	Database Systems	4	BCS
6	BIT	8	132	MG-314	Money & Capital Mark	3	BIT
7	MBA	4	64	MG-505	Intro to Accounting	3	MBA
8	MCS	4	64	MT-305	Linear Algebra	3	MCS
9	MCS	4	64	CS-504	Analysis of Algorithm	4	MCS
10	MCS	4	64	CS-616	Intro to Database Systems	3	MCS
11	MCS	4	64	CS-511	Operating System	4	MCS
12	MCS	4	64	CS-516	Data Structures and Algos	4	MCS
13	MIT	4	62	NULL	NULL	NULL	NULL

Semi Join

- First inner join and then projected on the attributes of one table
- Advantage: tells the rows involved in join
- No operator available as such, but can be implemented by `select_list`

Semi Join Example

```
SELECT distinct p.prName, totsem, prCredits  
FROM program p inner JOIN  course c  
ON p.prName = c.prName
```

	prName	totsem	prCredits
1	BBA	8	130
2	BCS	8	134
3	BIT	8	132
4	MBA	4	64
5	MCS	4	64

Self Join

- Applied on the same table having referential integrity constraint implemented onto itself

```
ALTER TABLE student
```

```
ADD cr char(5)
```

```
REFERENCES student(stId)
```

	stId	stname	prName	cr
1	s0123	Amjad	MCS	NULL
2	s1012	Amjad	MCS	s0123
3	s1015	Tahira Ejaz	MCS	s0123
4	s1018	Arif Zia	BIT	NULL
5	s1020	Suhai Dar	MCS	s0123
6	s1021	M. Ali	MBA	NULL
7	s1034	Sadia Zia	BIT	NULL
8	s1038	Shoaib Ali	BCS	s0123

Self Join

```
SELECT a.stId, a.stName, b.stId, b.stName  
FROM student a, student b  
WHERE (a.cr = b.stId)
```

	stId	stName	stId	stName
1	s1012	Amjad	s0123	Amjad
2	s1015	Tahira Ejaz	s0123	Amjad
3	s1020	Suhai Dar	s0123	Amjad
4	s1038	Shoaib Ali	s0123	Amjad

Subqueries

- A query within a query
- Useful when condition has to be applied against an unknown value
- Get the names of those students who have more cgpa than that of maximum of BCS students

Subqueries

- Can be used anywhere where an expression is allowed
- Given in parentheses
- Generally in where
- Can be nested

Subqueries

- Careful about operator; depends whether subquery returns single or multiple values

```
SELECT * from student where cgpa >  
(select max(cgpa) from student  
where prName = 'BCS')
```

	stId	stName	stFName	stAdres	stPhone	prName	curSem	cgpa	cr
1	S1015	Tahira Ejaz	Nek Muhammad	AJ Road	4323456	MCS	3	3.25	\$0123

Subqueries

```
SELECT * from student
WHERE cgpa =
  ANY (select max(cgpa)
        FROM student
        GROUP BY prName)
```


Wrap up DML

➤ That is it that we have to cover about DML in this course, and the golden rule

“Practice makes even the students perfect”

DCL

D.I.Y

Subqueries, Summary of DML and DCL need to be discussed in detail in handouts,

Database Management System

Lecture - 31

Database Management System

Lecture - 32

Application Programs

Application Programs

- Programs written to perform different requirement posed by the users/organization
- The concern of the developer
- Performed in parallel or after DBD
- Tool selection is critical BUT

General Activities

- Data input programmes
- Editing
- Display
- Processing related to activities
- Reports

User Interface

- Almost all activities through UI
- A gateway to DB world for end users
- Main concern of users in an application; crucial

User Interfaces

- Text based
- Graphical User Interface (GUI)
(Most Common, generally called
Forms)

Text Based UI

Adding a Record	-----	1
Deleting a Record	-----	2
Enrollment	-----	3
Result Calculation	-----	4
Exit	-----	5

Forms

➤ Browser based

- HTML, scripting languages, Front Page

➤ Non-Browser/Simple

- Visual Basic, Developer, MS Access

➤ Form tools support simple built-in functionality

Graphical User Interface

prName

totSem

prCredits

STUDENT

	stId	stName	stFName	stAdres	
▶	S1020	Suhal Dar	Loving	I-8 Islamabad	55232
	S1038	Shoaib Ali	Rahmat Ali	G-6 Islamabad	53432
	S1040	Ahmad Ali	Ali Hussain		
	S1042	Ahmad Ali	Ali Hassan		
*					

Record: of 4

	stId	crCode	semName	mTerm	
▶	S1020	CS-516	F04	26	
	S1020	CS-616	F04	25	
*	S1020				

Record: of 2

User Friendly Interface

- Beginners (What and How)
- Intermediate (What)
- Experts (Both)

Tips

- User friendly
- Make users in charge
- Don't test users' memory
- Be consistent

Tips

- Should be processes based rather than the data structure based
- Document structure (single page vs multiple pages)

Entities and Relationships

- Simple entity on a page

- Hierarchies of entities

Simple Entity

ADD STUDENT FORM

stId

stName

stFName

stAdres

stPhone

curSem

prName

cgpa

Add Record

Delete Record

Save Record



Command33



Hierarchies of Entities










prName

totSem








prCredits

STUDENT

	stId	stName	stFName	stAdres	
▶	S1020	Suhal Dar	Loving	I-8 Islamabad	55232
	S1038	Shoaib Ali	Rahmat Ali	G-6 Islamabad	53432
	S1040	Ahmad Ali	Ali Hussain		
	S1042	Ahmad Ali	Ali Hassan		
*					

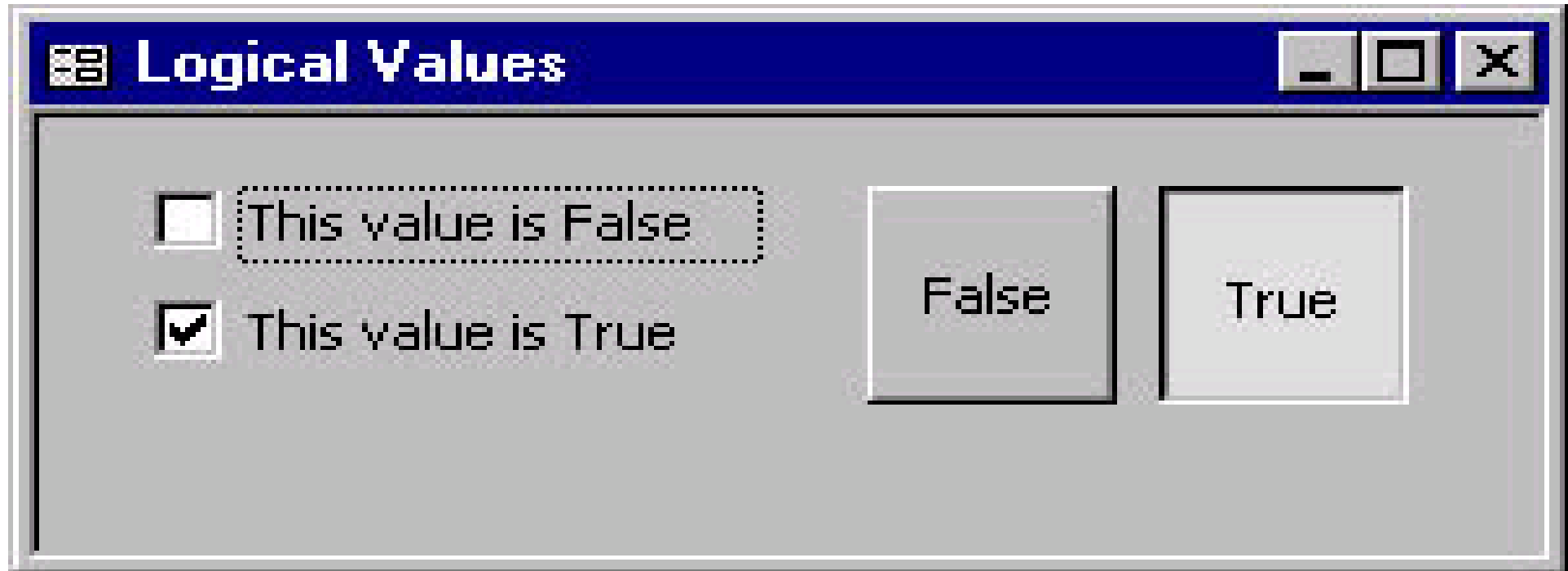
Record:        of 4  

	stId	crCode	semName	mTerm	
▶	S1020	CS-516	F04	26	
	S1020	CS-616	F04	25	
*	S1020				

Record:        of 2  

Windows Controls

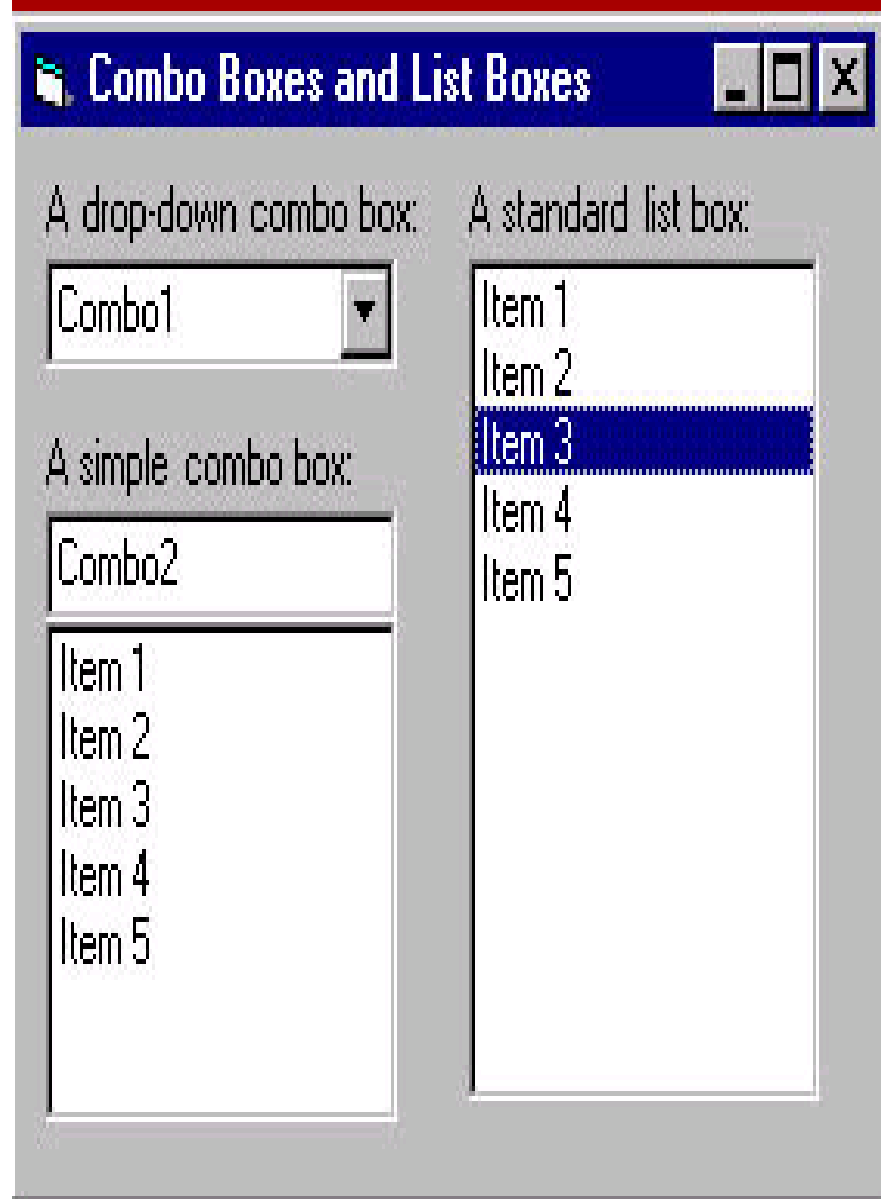
- So many of them
- Logical values



Windows Controls

- So many of them
- Logical values
- Set of values

Windows Controls



Set of Values

Table Wizard

Which of the sample tables listed below do you want to use to create your table?

After selecting a table category, choose the sample table and sample fields you want to include in your new table. Your table can include fields from more than one sample table. If you're not sure about a field, go ahead and include it. It's easy to delete a field later.

☒ Business

☐ Personal

Sample Tables:

Mailing List

Contacts

Customers

Employees

Products

Orders

Sample Fields:

MailingListID

Prefix

FirstName

MiddleName

LastName

Suffix

Nickname

Title

OrganizationName

Address

Fields in my new table:

MailingListID

FirstName

MiddleName

LastName

Rename Field...

Cancel

< Back

Next >

Finish

Numbers, Dates and Text

➤ Normally text boxes

Date/Time Properties [?] [X]

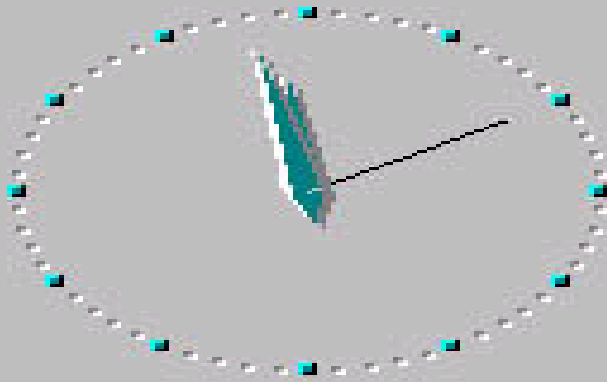
Date & Time | Time Zone

Date

April 1999

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Time



11:58:10 AM

Current time zone: W. Europe Daylight Time

OK Cancel Apply

Maintaining Data Integrity

- Basic rules
- Business rules

Database Management System

Lecture - 32

Database Management System

Lecture - 34

Data Storage Concepts

Classification of Physical Storage Media

- Media are classified according to three characteristics
 - Speed of access
 - Cost per unit of data
 - Reliability

Classification of Physical Storage Media

- We can also differentiate storage as either
 - volatile storage
 - non-volatile storage

Physical Storage Media

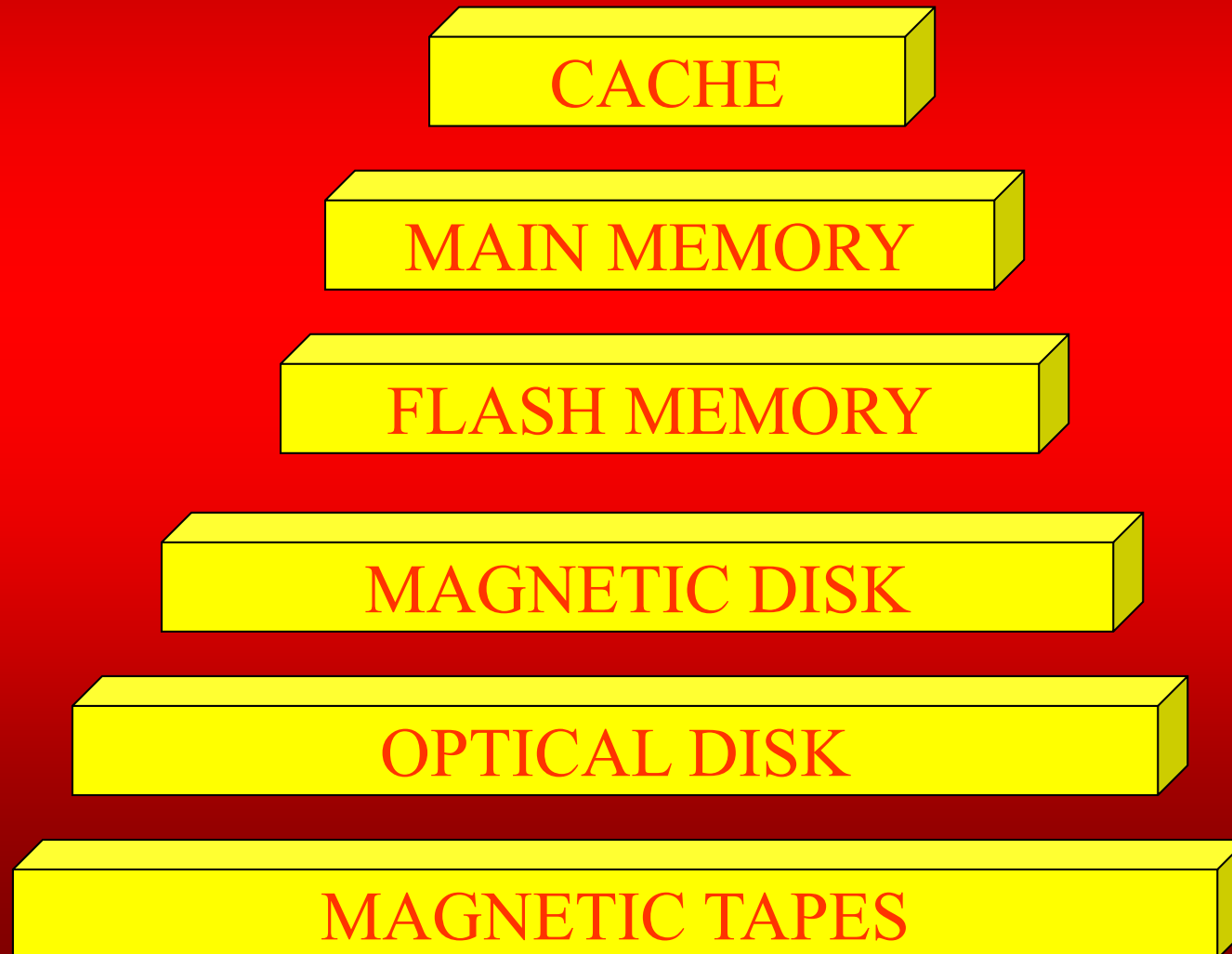
➤ Typical media available are:

- Cache
- Main memory
- Flash Memory

Physical Storage Media

- Magnetic disk
- Optical storage (CD or DVD)
- Tape storage

Memory Hierarchy



RAID – Redundant Array of Inexpensive Disks

- Many disk that look as a single disk to OS but have better *performance* and better *reliability*.

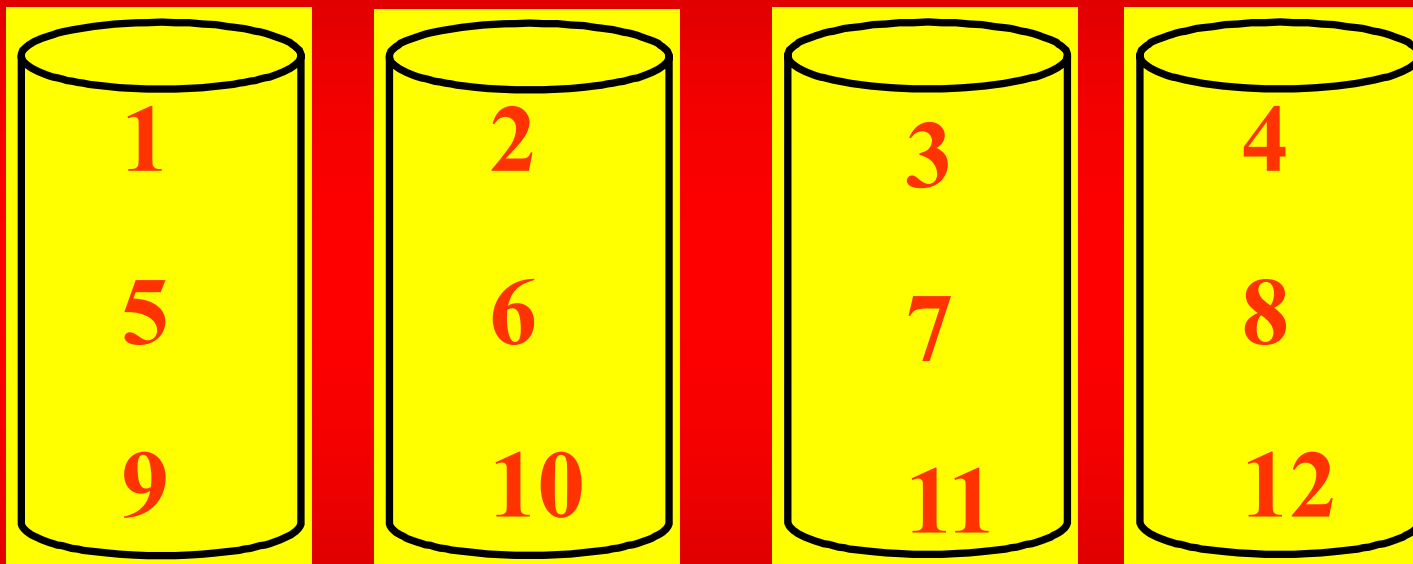
RAID

- RAID have the property that the data are distributed over the drives to allow parallel operations.

RAID 0

- Simple Striping
- Virtual single disk is divided up into strips of k sectors each.
- RAID 0 writes consecutive stripes over the drives in round robin fashion.

RAID 0



Note: This example is a basic virtual drive where each element depicted as a disk is a physical disk

RAID 0

- Controller will break any disk command into k commands (one for each of the disks) and will read/write in parallel.

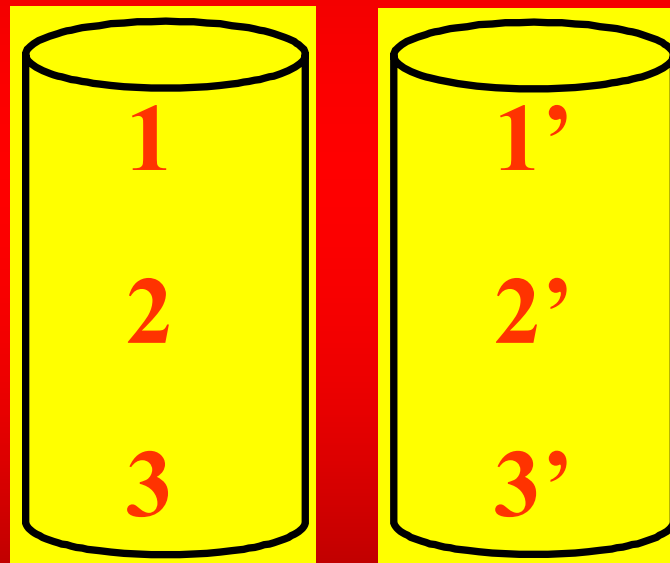
RAID 1

- It duplicates all the disks.
- On write, every stripe is written twice. On a read, either copy can be used.

RAID 1

- *Fault tolerance* is excellent: if a drive crashes, the copy from the duplicated disk is simply used instead.

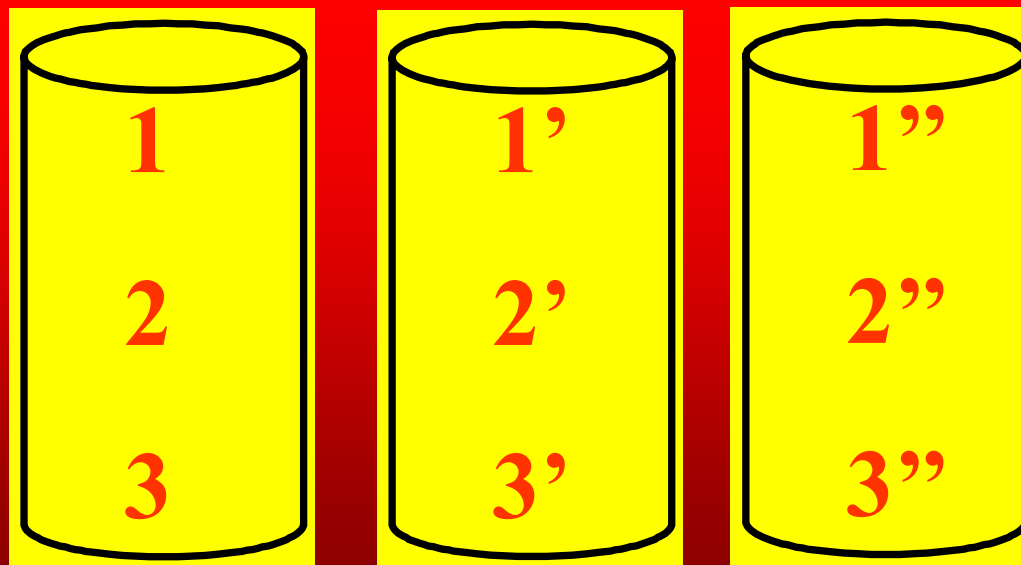
RAID 1 -Simple Mirroring



RAID 1

Multi-Mirroring

➤ Three copies of all data



RAID 2-3

- For reliability simple parity check code is used.
- Parity bit is stored on separate disk.

RAID 3

- Can correct single error.
- If a drive crashes, the controller just consider all its bits as 0.

RAID 3

- If no parity error, its bit is considered 0. If parity error, its bit is considered 1.
- Huge load of the parity disk.

RAID 4

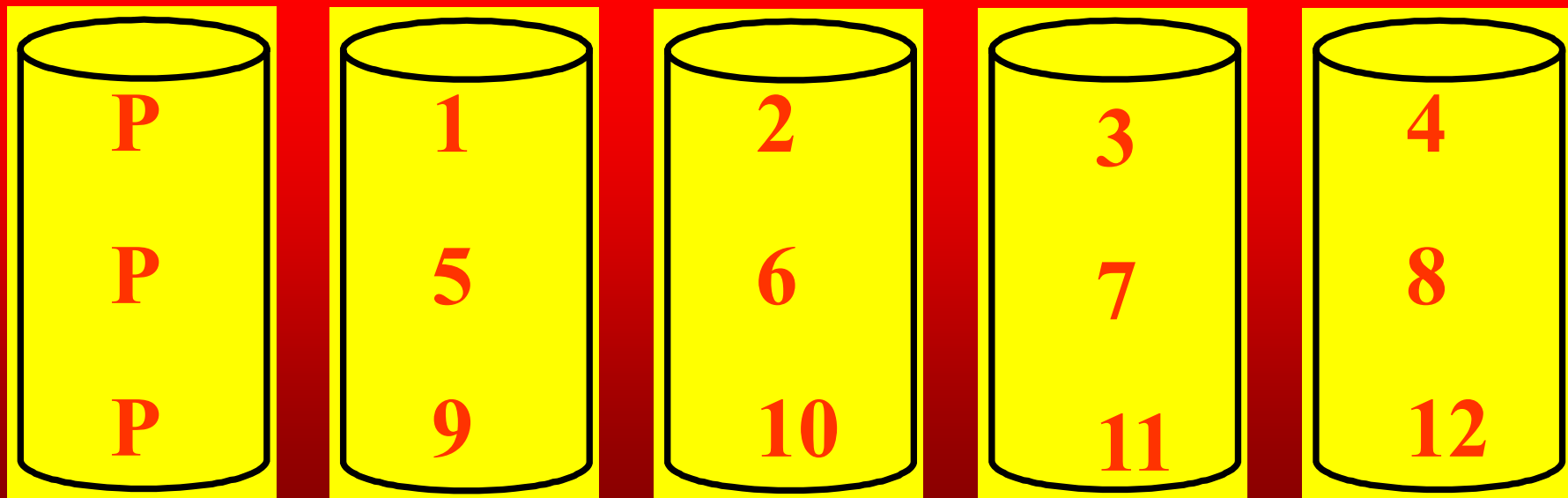
- RAID 4 works with stripe again.
- Stripe-to-stripe parity written onto a extra drive.

RAID 4

- If each stripe is k bytes long, all the strips are EXORed together, resulting in a parity strip stored on parity disk

RAID 3 or RAID 4 – Non-Rotating Parity 0

➤ Fixed parity on the first element of the virtual drive



RAID 5

- Eliminates the bottleneck of the heavy load on the parity drive by distributing the parity bits uniformly over all drives.

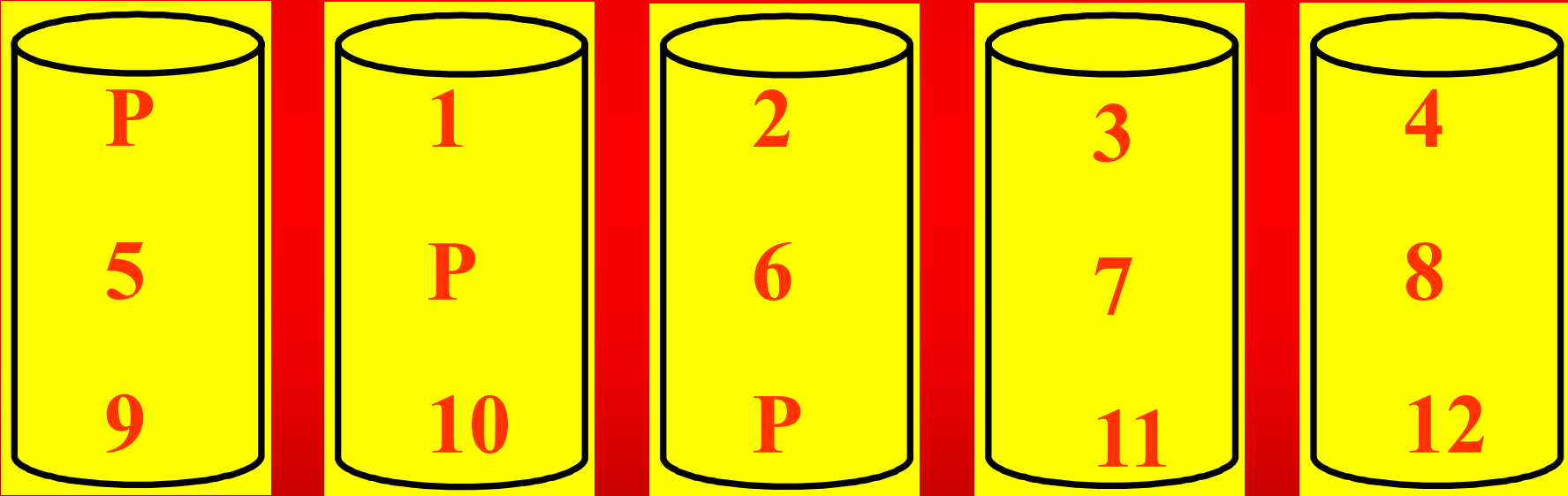
RAID 5

- Rotating Parity 0 with Data

Restart

- Parity starts on the first element of the virtual drive

RAID 5



File Organization

- The manner data records are stored and retrieved on physical devices
- The technique used to find and retrieve store records are called *access methods*.

Sequential File Organization

- Records are arranged on storage devices in some sequence based on the value of some field, called **sequence field**.

Sequential File Organization

- Sequence field is often the key field that identifies the record.
- Simple, easy to understand and manage, best for providing sequential access.

Sequential File Organization

- Not feasible for direct or random access, inserting/deleting a record in/from the middle of the sequence involves cumbersome record searches and rewriting of the file.

Sequential File Organization- not shown

STUDENT

stId	stName	prName	cgpa
S1020	Sohail Dar	MCS	2.8
S1014	Shoaib Ali	BCS	2.78
S1028	Tahira Ejaz	MCS	3.2
S1034
S1048

Database Management System

Lecture - 34

Database Management System

Lecture - 35

File Organizations

- Sequential files provide access only in a particular sequence
- That does not suit many applications since it involves too much time
- Some mechanism for direct access is required

Direct Access FO

- Indexed Sequential
- Direct File Organization

Indexed Sequential File

- As the name suggests, records are stored in a sequence (generally w.r.t. PK)
- But index is also maintained to provide direct access

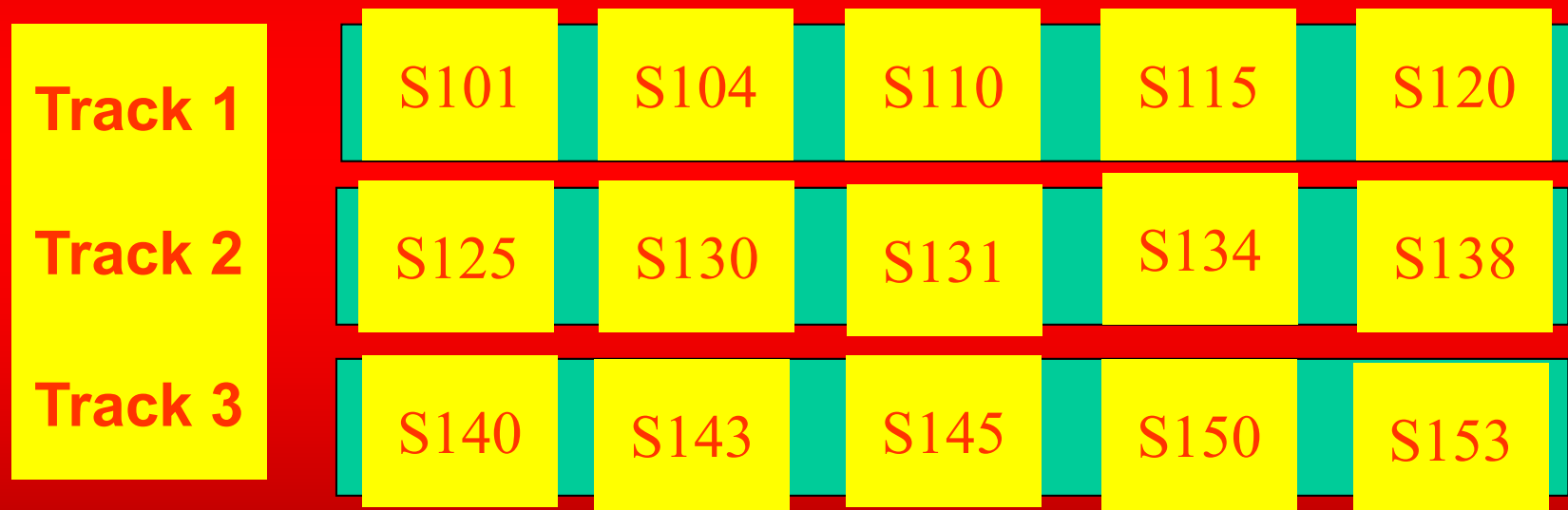
Indexed Sequential File

- Index is a special file, sort of a table, that lists the key value for each record and the address of the record
- Address may be the track no.

Indexed Sequential File

- In case of direct access requirement, index is accessed rather than the file itself
- Location of the record is obtained from index and then the record is accessed from that location

Indexed Sequential File



Indexed Sequential File

➤ Index File

StudID	Track
S101	1
S104	1
S110	1
S115	1
S120	1
S125	2
S130	2
S131	...

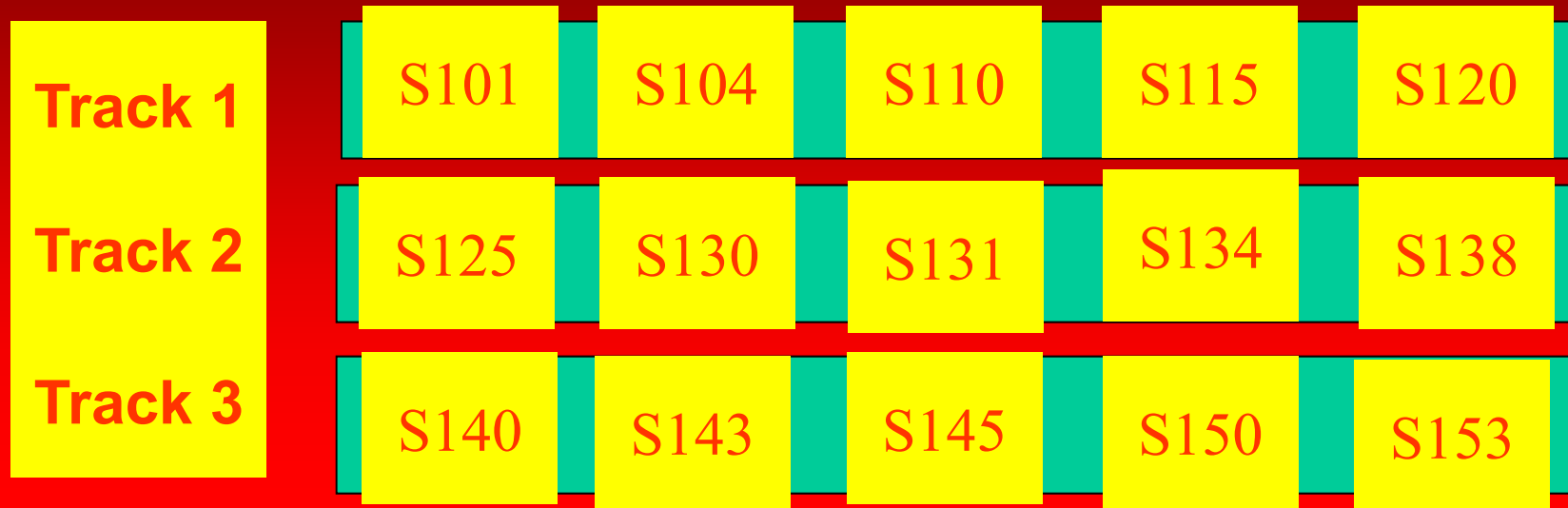
Dense Key Index

- An index that stores entry for each record
- One entry for each record is wastage, since within the track search has to be sequential

Nondense Index

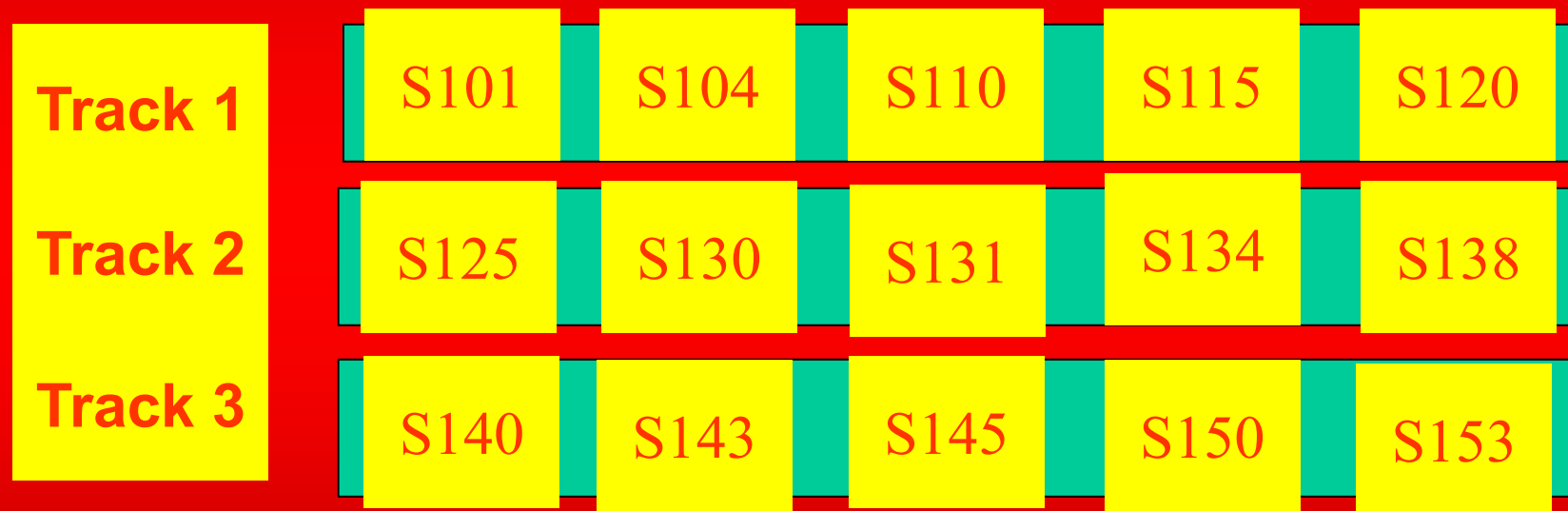
- Rather than entry for every record, entry for the record with highest key value in the track is maintained
- Index size reduces, search also gets more efficient
- Pur Kaisay?

Non-Dense Index



HK on Track	Track
S120	1
S138	2
S153	3
....	...

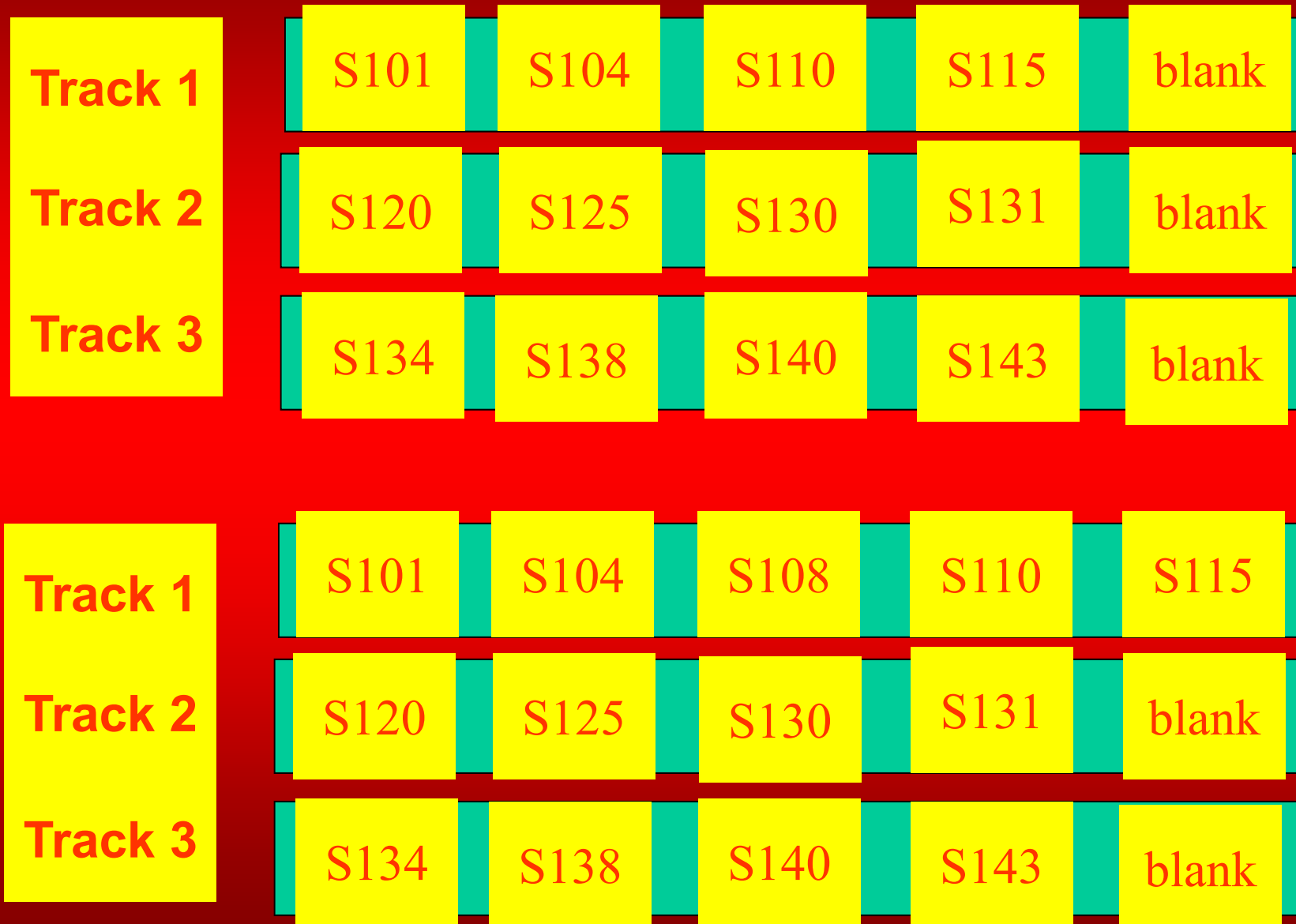
Overflow in Tracks



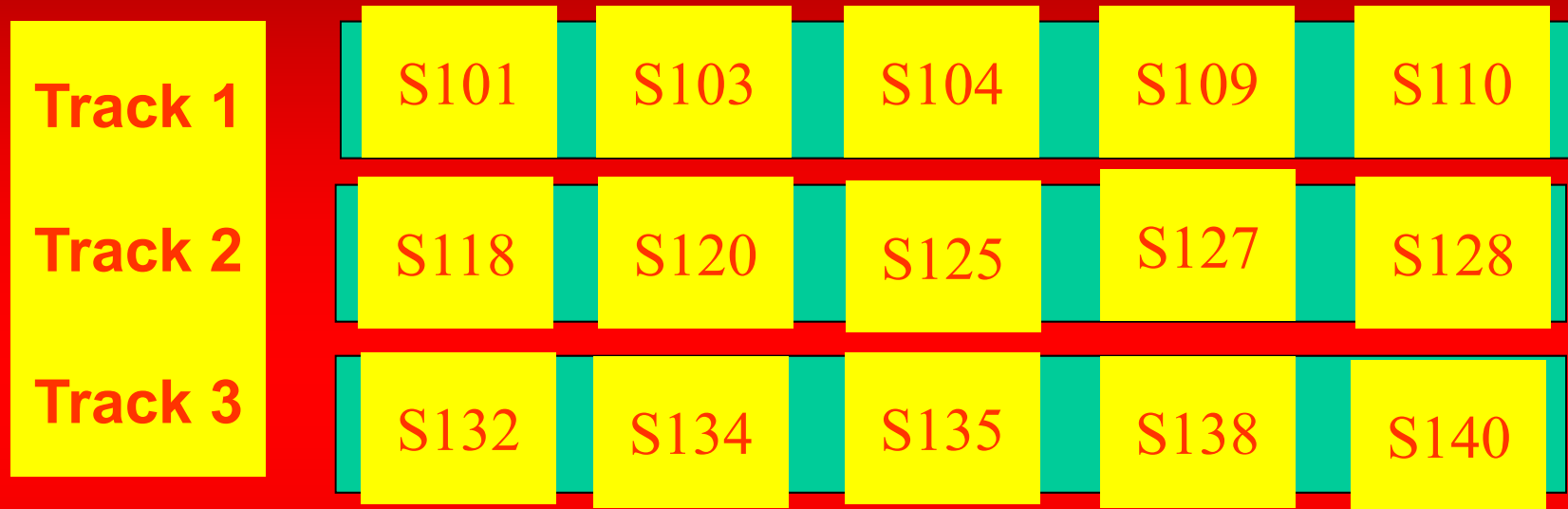
Overflow in Tracks

Track 1	S101	S104	S110	S115	blank
Track 2	S120	S125	S130	S131	blank
Track 3	S134	S138	S140	S143	blank

Overflow in Tracks



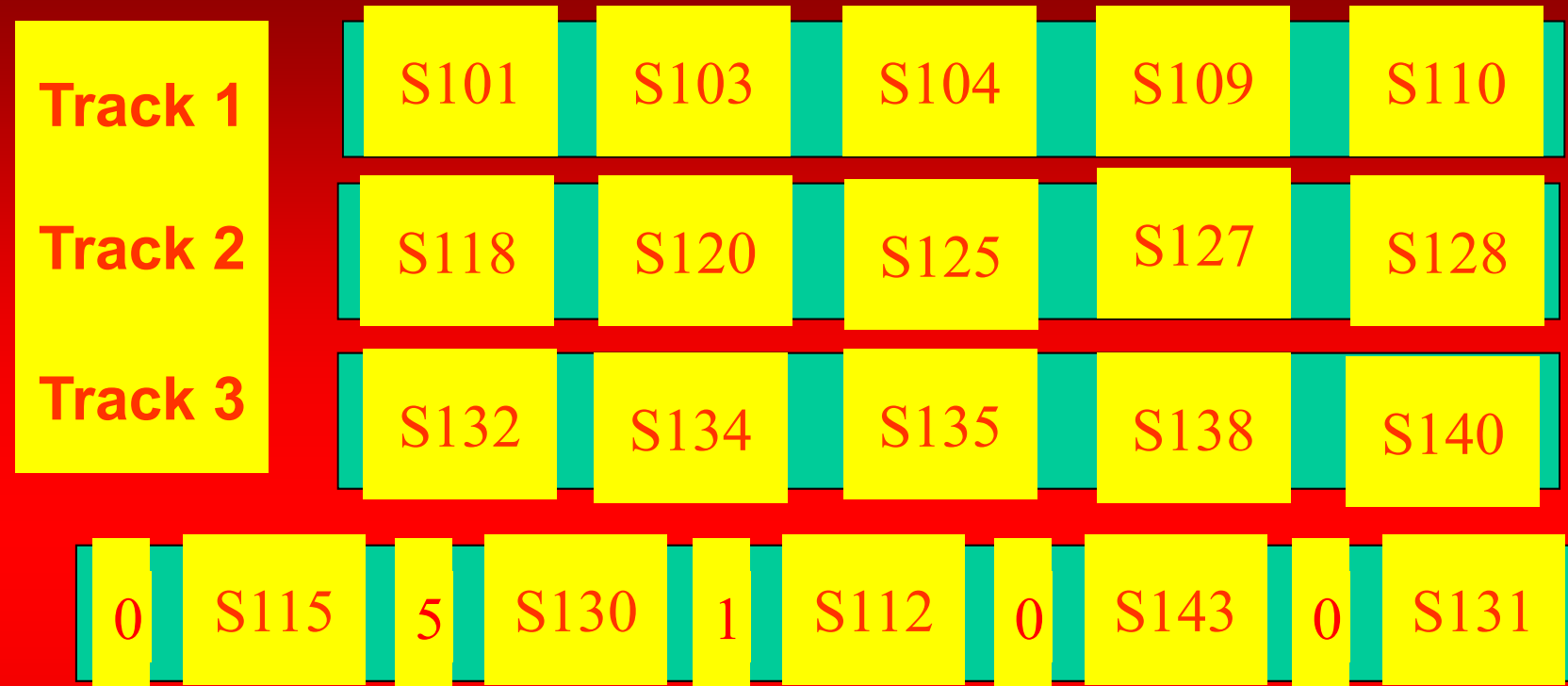
Overflow Area



Overflow Area



Extended Index



Track	HK in Track	HK in Overflow	Pointer to First Overflow
1	S110	S115	Address of S112
2	S128	S131	Address of S130
3	S140	S143	Address of S143
.....			

Cylinder Index(not shown)

- Searching among track indexes can further be improved by introducing the Cylinder Index
- Entry contains the highest key in the cylinder, saves time and effort

Indexed Sequential: Summarized (not shown)

- Records stored in sequence, index maintained
- Dense and nondense types of indexes
- Track overflow and file overflow areas
- Cylinder index increases efficiency

Database Management System

Lecture - 35

Database Management System

Lecture - 37

Introduction

- Any subset of the fields of a relation can be the search key for an index on the relation.
- *Search key* is not the same as *key* (e.g. doesn't have to be unique ID).

Introduction

An index contains a collection of *data entries*, and supports efficient retrieval of all records with a given search key value k .

Introduction

Typically, index also contains
auxiliary information that directs
searches to the desired data entries

Introduction

- Can have multiple (different) indexes per file.
 - e.g. file sorted by *stid*, with a hash index on *cgpa* and a B+tree index on *stname*.

Introduction

- Indexes on primary key and on attribute(s) in the unique constraint are automatically created
- We can still create more

Index Classification

- What selections does it support?
- Representation of data entries in index
 - what kind of info is the index actually storing?

Index Classification

➤ 3 alternatives here

- Clustered vs. Un-clustered Indexes
- Single Key vs. Composite Indexes
- Tree-based, inverted files, pointers

Creating Index

```
CREATE [ UNIQUE ]  
      [ CLUSTERED | NONCLUSTERED ]  
INDEX index_name  
ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
```

Create Index Example

```
CREATE UNIQUE INDEX
```

```
pr_prName
```

```
ON program(prName)
```

Create Index Example

- Can also be created on composite attributes

```
CREATE UNIQUE INDEX
```

```
St_Name ON
```

```
Student(stName ASC, stFName DESC)
```

Properties of Indexes

- Indexes can be defined even when there is no data in the table
- Existing values are checked on execution of this command

Properties of Indexes

- Support selections of form
 - field <operator> constant
- Support equality selections
 - Either “tree” or “hash”
indexes help here.

Properties of Indexes

- Support Range selections
(operator is one among $<$, $>$,
 $<=$, $>=$, BETWEEN)
- “Hash” indexes don’t work
for these

Secondary Key Index

- Users often need to access data on the basis of non-key or non-unique attribute; secondary key
- Like student name, program name, students enrolled in a particular program

Secondary Key Index

- Records are stored on the basis of key attribute; three possibilities
 - Sequential search
 - Sorted on the basis of name
 - Sort for command execution
- Problematic

Secondary Key Index

- Index on secondary key is solution
- Data remains in original order, however secondary key index maintains ordered sequence of records, and provides direct access

Secondary Key Index

- Three implementation approaches
 - Inverted files or inversions
 - Linked lists
 - B+ Trees

Database Management System

Lecture - 37

Database Management System

Lecture - 38

Inverted Files

- The attribute on which we want direct access, inverted file is created on that field
- File is said to be inverted on that field
- More than one inversions can be created, even on all attributes

	stId	stName	stFname	stAdres
1	S0123	Amjad	Hussain	8-SD Lahore
2	S1012	Amjad	Rehan	I8 Ibd
3	S1015	Tahira Ejaz	Nek Muhammad	AJ Road
4	S1018	Arif Zia	Zia Khan	GM Rawalpindi
5	S1020	Suhai Dar	Nek Muhammad	I-8 Islamabad
6	S1021	M. Ali	M. Saad	JT Lahore
7	S1034	Sadia Zia	NULL	NULL
8	S1038	Shoaib Ali	Rahmat Ali	G-6 Islamabad

Inversion On stName	Amjad	1
	Amjad	2
	Arif Zia	4
	M. Ali	6
	Sadia Zia	7
	Shoaib Ali	8
	Suhail Dar	5
	Tahira Ejaz	3

Inversion On stAdres	NULL	7
	8-SD Lahore	1
	AJ Road	3
	G-6 Islamabad	8
	GM Rawalpindi	4
	I-8 Islamabad	5
	I8 Ibd	2
	JT Lahore	6

Inverted Files

- Inverted file contains the values of attribute in sorted order and the relative address of the record
- DBMS searches appropriate record no and hands over to access method to retrieve the desired record

Linked Lists

- AKA Pointer Chain
- Needs an extra field in the physical record to store the link
- Relative addresses are used

	stname	stFname	stadres	prname	cgpa	pointer
1	Amjad	Hussain	8-SD Lahore	MCS	NULL	2
2	Amjad	Rehan	I8 Ibd	MCS	2.3	4
3	Tahira Ejaz	Nek Muhammad	AJ Road	MCS	3.25	0
4	Arif Zia	Zia Khan	GM Rawalpindi	BIT	2.8	6
5	Suhai Dar	Nek Muhammad	I-8 Islamabad	MCS	3.2	3
6	M. Ali	M. Saad	JT Lahore	MBA	2.8	7
7	Sadia Zia	NULL	NULL	BIT	NULL	8
8	Shoaib Ali	Rahmat Ali	G-6 Islamabad	BCS	3.2	5

Linked Lists

- Head pointer needs to be stored
- Pointers for multiple fields can be created
- Circular pointers
- Double linked lists

Tree Index

- Index is maintained in a hierarchical arrangements of nodes
- Each node stores key values and pointers; nodes are connected through links

Tree Index

- There is a special node called the root node
- Each node has a parent node (but the root) and has one or more children (but the leaf nodes)
- Different types of trees exist

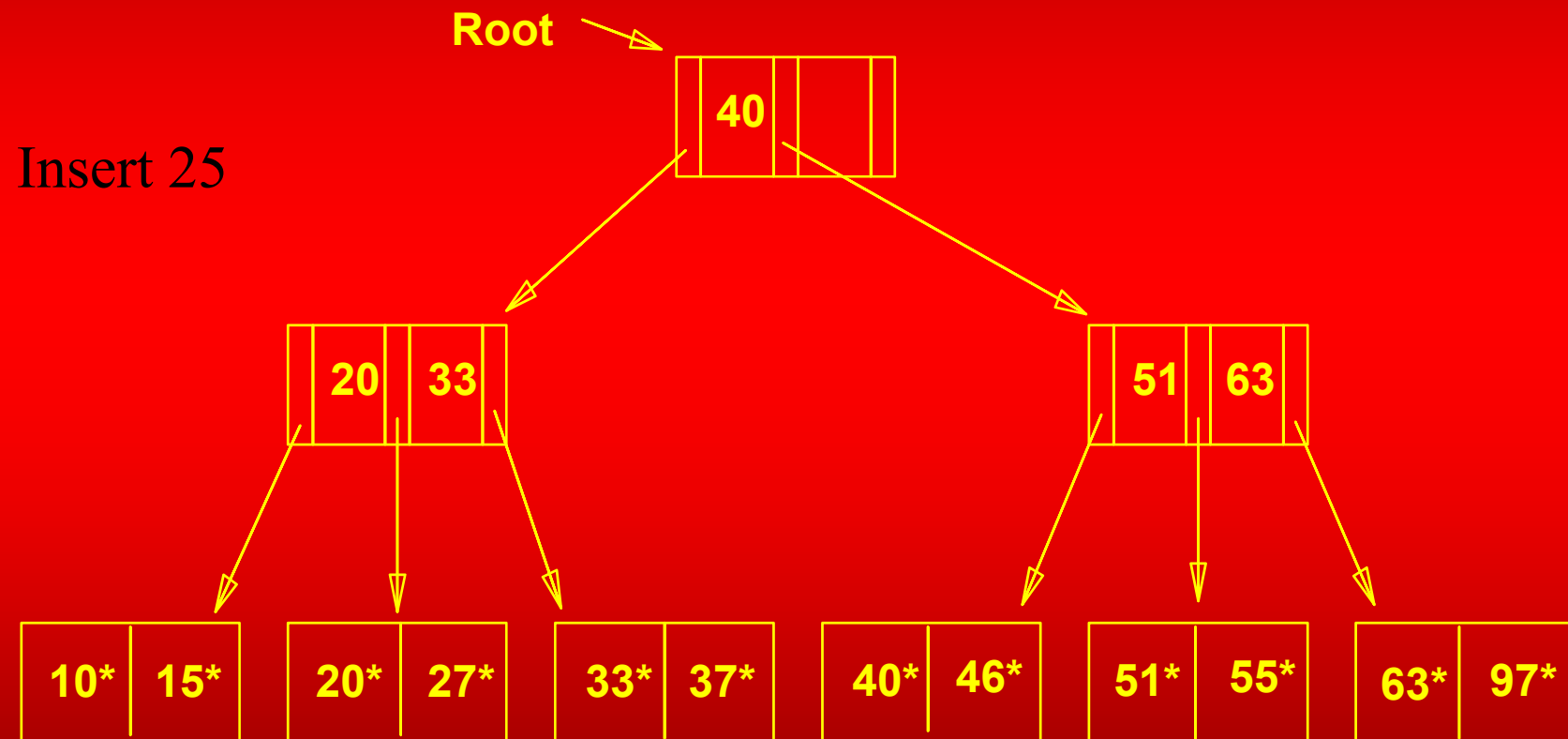
B+ Trees

- Number of pointers in a node is one more than the number of keys
- Number of pointers called order of tree
- Distance of all leaf nodes from the root is same

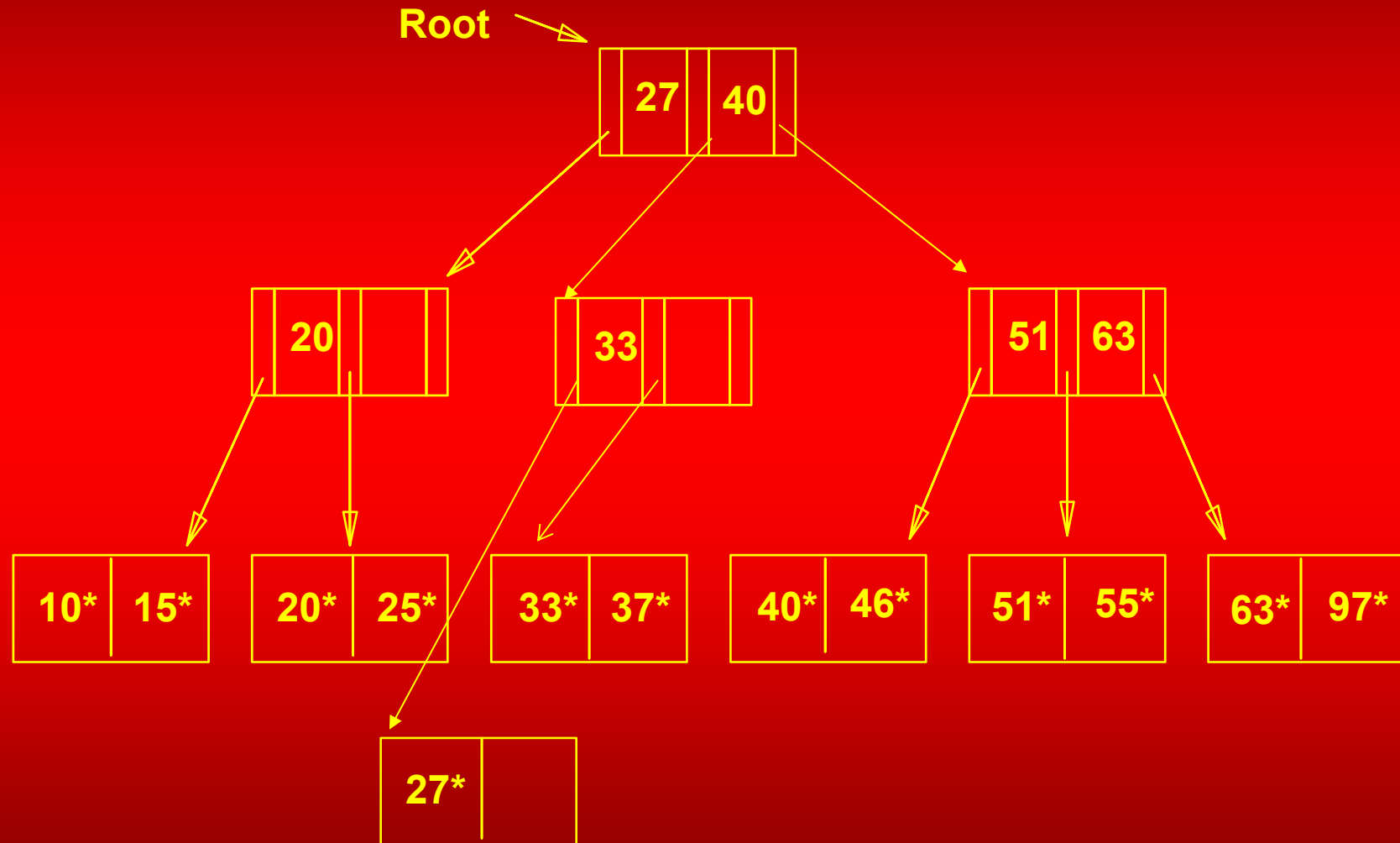
Example Tree Index

- *Index entries:* <search key value, page id> they direct search for data entries *in leaves*.
- Example where each node can hold 2 entries.

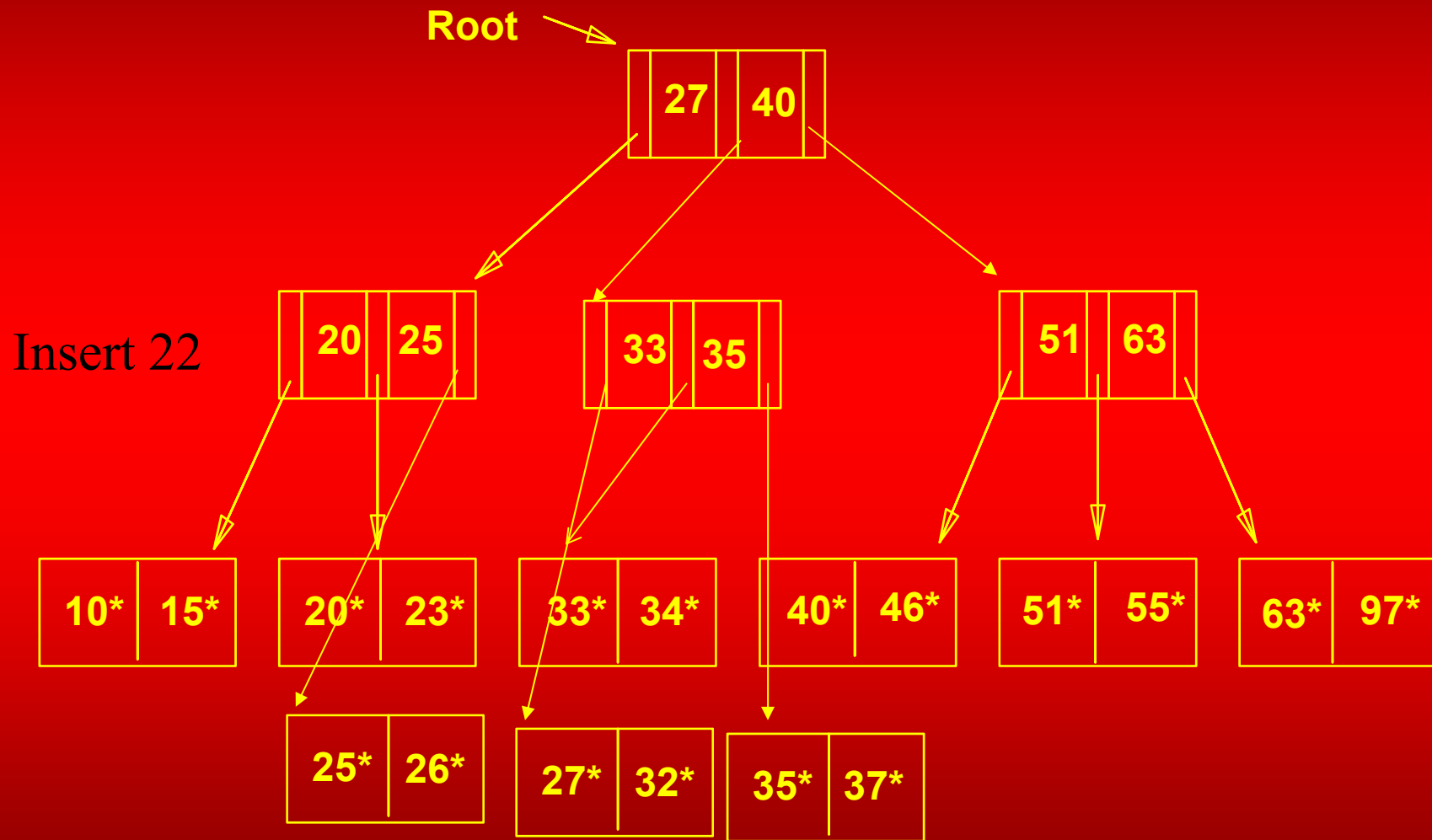
Example Tree Index

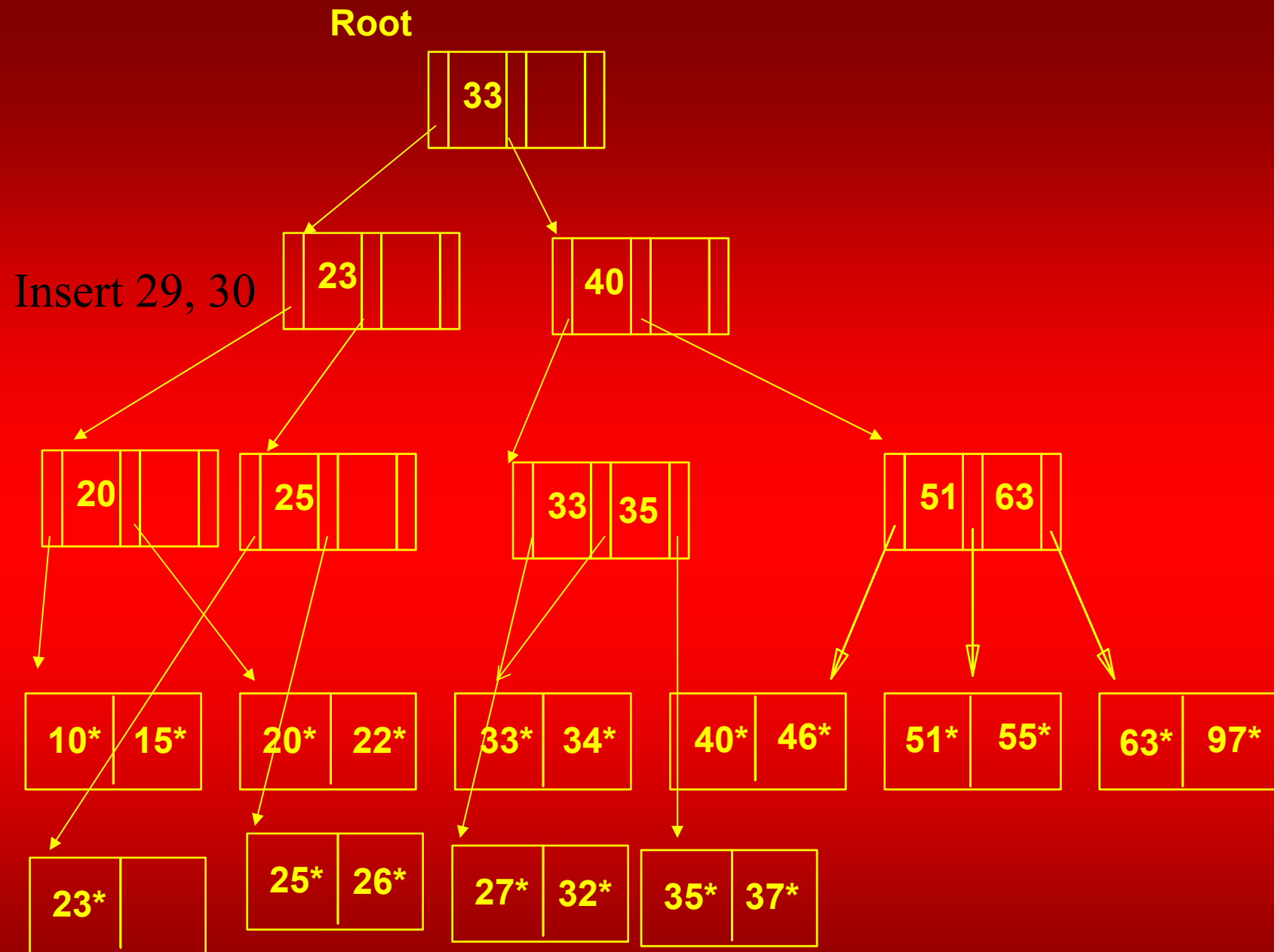


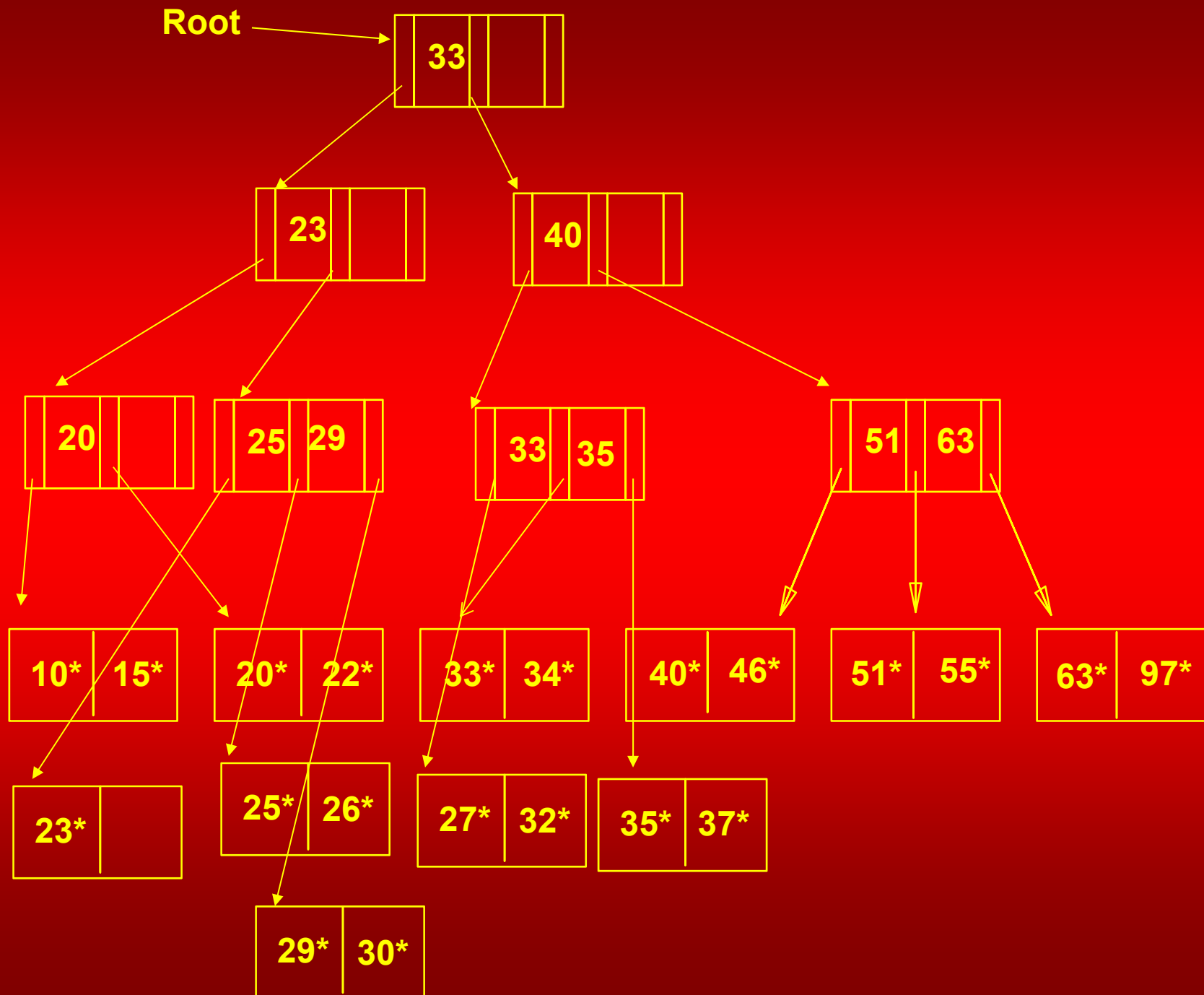
Example Tree Index



Example Tree Index







Alternatives for Data Entry k^* in Index

- What is actually stored in the leaves of the index for key value “k”?

Three Alternatives

1. Actual data record(s) with key value **k**
2. $\{ \langle \mathbf{k}, \text{rid of matching data record} \rangle \}$
3. $\langle \mathbf{k}, \text{list of rids of matching data records} \rangle$

Alternative 1

- Actual data record (with key value **k**)
 - If this is used, index structure is a file organization for data records (like Heap files or sorted files).

Alternative 1

- At most one index on a given collection of data records can use Alternative 1.

Alternative 1

- This alternative saves pointer lookups but can be expensive to maintain with insertions and deletions.

Alternative 2 & 3

➤ Alternative 2

{<**k**, rid of matching data record>}

➤ Alternative 3

<**k**, list of rids of matching data
records>

Alternative 2 & 3

- Easier to maintain than Alt 1.
- If more than one index is required on a given file, at most one index can use Alternative 1
- Rest must use Alternatives 2 or 3.

Alternative 2 & 3

- Alternative 3 more compact than Alternative 2, but leads to *variable sized data* entries even if search keys are of fixed length.

Alternative 2 & 3

- Even worse, for large rid lists the data entry would have to span multiple blocks!

Database Management System

Lecture - 38

Database Management System

Lecture - 39

Index Classification

➤ *Clustered vs. unclustered:*

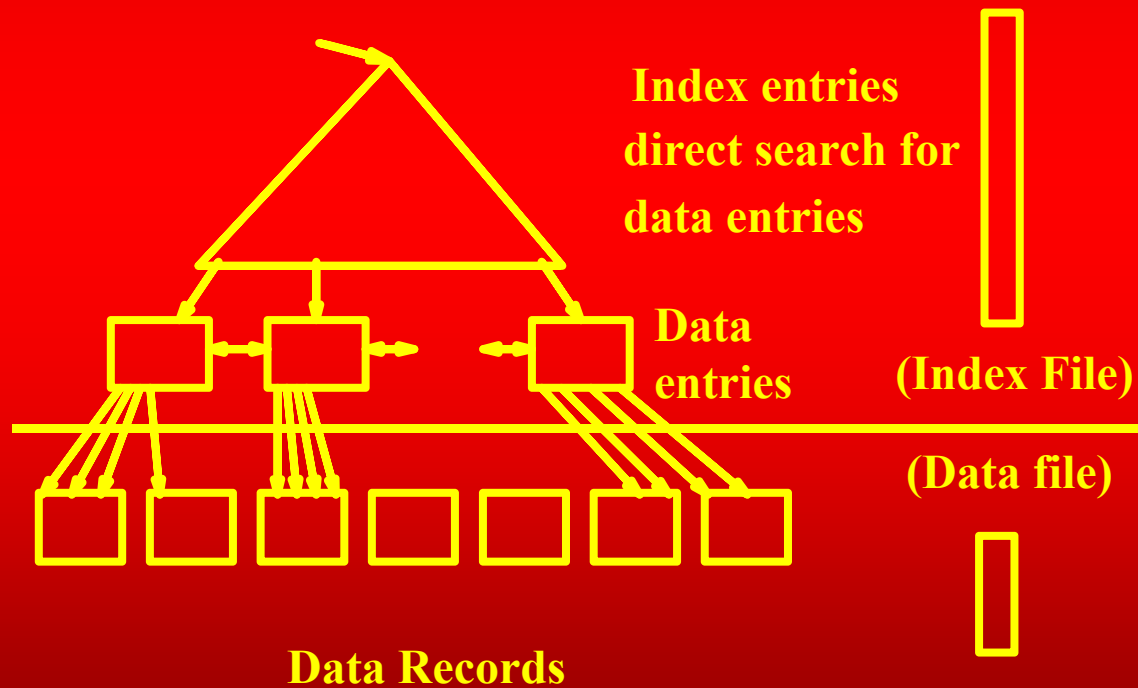
- If order of data records is the same as order of index data entries, then called *clustered index*.

Index Classification

- A file can be clustered on at most one search key.
- Cost of retrieving data records through index varies *greatly* based on whether index is clustered or not!

Clustered vs. Unclustered Index

CLUSTERED



UNCLUSTERED



When to Use Indexes

- Useful on large tables
- Required on PK
- Useful for attributes appearing in where, order by and group by clauses

When to use

- Significant variety in values
- Check the limit of indexes with your DBMS
- Make your decision very much justifiable; indexes introduce overhead

Indexes Summary

- Useful tool to speedup the data access
- Can be unique or non-unique
- Can be implemented through different techniques
- Be careful; they involve overhead

Views

Definition

- A view is defined to combine certain data from one or more tables for different reasons
- A view is like a window through which we can see data from one or more tables

Why Views?

➤ Security

- Show the data to a users' group that is necessary/required for them
- Give only necessary authorizations
- No concern with rest of the data

Why Views?

➤ Efficiency

- Part of a query that is frequently used (a subquery), define it as a view
- The view definition will be stored and will be executed anytime anywhere view is referred

Why Views?

- Join columns from multiple tables so that they look like a single table
- Aggregate information instead of supplying details

Characteristics of Views

- Not exactly the external views of 3-level schema architecture
- Major Types
 - Dynamic
 - Materialized
 - Partitioned
 - Simple/complex

Dynamic Views

- Data is not stored for the views
- Definition is stored in the schema
- Executed every time view is referred

Defining Dynamic Views

```
CREATE VIEW [ < database_name > .  
            ] [ < owner > . ] view_name [ ( column  
            [ ,...n ] ) ]  
[ WITH ENCRYPTION |  
SCHEMABINDING [ ,...n ] ]  
AS  
select_statement  
[ WITH CHECK OPTION ]
```

Database Management System

Lecture - 39

Database Management System

Lecture - 40

Dynamic Views Example

- `CREATE VIEW st_view1 AS`
 (select stName, stFname,
 prName from student
 WHERE prName = 'MCS')
- View can be referred in SQL
statements like tables

SELECT * FROM st_view1

	stName	stFname	prName
1	Amjad	Hussain	MCS
2	Amjad	Rehan	MCS
3	Tahira Ejaz	Nek Muhammad	MCS
4	Suhal Dar	Loving	MCS

With Check Option

```
➤ CREATE VIEW st_view2 AS  
  (select stName, stFname,  
   prName from student  
   WHERE prName = 'BCS')  
WITH CHECK OPTION
```

SELECT * FROM ST_VIEW1

	stName	stFname	prName
1	Amjad	Hussain	MCS
2	Amjad	Rehan	MCS
3	Tahira Ejaz	Nek Muhammad	MCS
4	Suhal Dar	Loving	MCS

Update ST_VIEW1 set prName = 'BCS'
Where stFname = 'Loving'

Select * from ST_VIEW1

	stName	stFname	prName
1	Amjad	Hussain	MCS
2	Amjad	Rehan	MCS
3	Tahira Ejaz	Nek Muhammad	MCS

SELECT * FROM ST_VIEW2

	stName	stFname	prName
1	Suhal Dar	Loving	BCS
2	Shoaib Ali	Rahmat Ali	BCS

Update ST_VIEW2 set prName = 'MCS'
Where stFname = 'Loving'

Server: Msg 550, Level 16, State 1, Line 1

The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that
The statement has been terminated.

Characteristics of Views

➤ Different attribute names

```
CREATE VIEW st_view3 (name,  
abbaG, pata) as (select stname,  
stfname, stadres from student)
```


SELECT * FROM ST_VIEW3

	name	abbaG	pata
1	Amjad	Hussain	8-SD Lahore
2	Amjad	Rehan	I8 Ibd
3	Tahira Ejaz	Nek Muhammad	AJ Road
4	Arif Zia	Zia Khan	GM Rawalpindi
5	Suhal Dar	Loving	I-8 Islamabad
6	M. Ali	M. Saad	JT Lahore
7	Sadia Zia	NULL	NULL
8	Shoaib Ali	Rahmat Ali	G-6 Islamabad

Characteristics of Views

- Computed attributes
- Nesting of views

```
CREATE VIEW enr_view AS (select *  
from enroll)
```

```
CREATE VIEW enr_view1 as (select  
stId, crcode, smrks, mterm, smrks +  
mterm sessional from enr_view)
```

Select * from enr_view1

	stId	crcode	smrks	mterm	sessional
1	S1015	CS-516	12	32	44
2	S1015	CS-616	10	30	40
3	S1018	MG-314	10	26	36
4	S1020	CS-516	13	26	39
5	S1020	CS-616	12	25	37

Data from Multiple Tables

- View can include data from multiple tables
- Through product, or any form of join
- Generally Join

```
CREATE VIEW st_pr_view AS (select  
stName, stFname,  
student.prName, prcredits  
FROM student, program WHERE  
student.prname = program.prname)  
  
SELEC * FROM st_pr_view
```

	stName	stFname	prName	prcredits
1	Amjad	Hussain	MCS	64
2	Amjad	Rehan	MCS	64
3	Tahira Ejaz	Nek Muhammad	MCS	64
4	Arif Zia	Zia Khan	BIT	132
5	Suhal Dar	Loving	BCS	134
6	M. Ali	M. Saad	MBA	64
7	Sadia Zia	NULL	BIT	132
8	Shoaib Ali	Rahmat Ali	BCS	134

```
CREATE VIEW st_cr_enr_view1  
(a1, a2, a3, a4, a5, a6, a7)
```

```
AS (select stName , crname, mterm,  
smrks, fmrks, mterm+smrks, totmrks,  
FROM student, course, enroll  
WHERE student.stid = enroll.stid and  
course.crcode = enroll.crcode)
```

SELECT * FROM st_cr_enr_view1

	a1	a2	a3	a4	a5	a6	a7
1	Tahira Ejaz	Data Structures and Algos	32	12	42	44	86
2	Tahira Ejaz	Intro to Database Systems	30	10	40	40	80
3	Arif Zia	Money & Capital Mark	26	10	39	36	75
4	Suhal Dar	Data Structures and Algos	26	13	40	39	79
5	Suhal Dar	Intro to Database Systems	25	12	39	37	76

Updating Data

- If single table, updation piece of cake (with check option)
- We can even insert data, BUT
 - Not through computed attribute
 - Cannot miss “Not NULL” attributes
 - One of the multiple tables
 - Not in case of aggregate functions

- CREATE VIEW st_view1 as
(SELECT stId, stName, stFname,
prName from student where prName
= 'MCS')
- INSERT INTO st_view1
values ('S1042', 'Ahmad Ali',
'Ali Hussain', 'BCS')

	stId	stName	stFName	stAdres	stPhone	prName	curSem
1	S0123	Amjad	Hussain	8-SD Lahore	234322	MCS	1
2	S1012	Amjad	Rehan	I8 Ibd	5456754	MCS	2
3	S1015	Tahira Ejaz	Nek Muhammad	AJ Road	4323456	MCS	3
4	S1018	Arif Zia	Zia Khan	GM Rawalpindi	4356488	BIT	2
5	S1020	Suhal Dar	Loving	I-8 Islamabad	5523240	BCS	2
6	S1021	M. Ali	M. Saad	JT Lahore	544325	MBA	2
7	S1034	Sadia Zia	NULL	NULL	NULL	BIT	1
8	S1038	Shoaib Ali	Rahmat Ali	G-6 Islamabad	5343240	BCS	2
9	S1040	Ahmad Ali	Ali Hussain	NULL	NULL	BCS	1

- ALTER VIEW st_view1 as
(SELECT stId, stName, stFname,
prName from student where prName
= 'MCS') WITH CHECK OPTION
- INSERT INTO st_view1
values ('S1044', 'Ali Raza',
'M. Raza ', 'BCS')

Server: Msg 550, Level 16, State 1, Line 1

The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that

The statement has been terminated.

Database Management System

Lecture - 40

Database Management System

Lecture - 41

Updating Multiple tables

- One at a time
- CREATE VIEW st_pr_view1 (a1, a2, a3, a4) AS (select stId, stName, program.prName, prcredits from student, program WHERE student.prName = program.prName)

insert into st_pr_view1 (a3, a4) values
('MSE', 110)

Select * from program

	prName	totSem	prCredits
1	BBA	8	130
2	BCE	8	134
3	BCS	8	134
4	BIT	8	132
5	MBA	4	64
6	MCS	4	64
7	MIT	4	62
8	MSE	NULL	110

insert into st_pr_view1 (a1, a2) values
(‘S1043’, ‘Bilal Masood’)

SELECT * from student

	stId	stName	stFName	stAdres	stPhone	prName	curSem
1	S0123	Amjad	Hussain	8-SD Lahore	234322	MCS	1
2	S1012	Amjad	Rehan	I8 Ibd	5456754	MCS	2
3	S1015	Tahira Ejaz	Nek Muhammad	AJ Road	4323456	MCS	3
4	S1018	Arif Zia	Zia Khan	GM Rawalpindi	4356488	BIT	2
5	S1020	Suhal Dar	Loving	I-8 Islamabad	5523240	BCS	2
6	S1021	M. Ali	M. Saad	JT Lahore	544325	MBA	2
7	S1034	Sadia Zia	NULL	NULL	NULL	BIT	1
8	S1038	Shoaib Ali	Rahmat Ali	G-6 Islamabad	5343240	BCS	2
9	S1040	Ahmad Ali	Ali Hussain	NULL	NULL	BCS	1
10	S1042	Ahmad Ali	Ali Hussain	NULL	NULL	BCS	1
11	S1044	Bilal Moasood	NULL	NULL	NULL	NULL	1

Interesting Thingi

Select * from st_pr_view1

	a1	a2	a3	a4	
1	s0123	Amjad	MCS	64	
2	s1012	Amjad	MCS	64	
3	s1015	Tahira Ejaz	MCS	64	
4	s1018	Arif Zia	BIT	132	
5	s1020	Suhal Dar	BCS	134	
6	s1021	M. Ali	MBA	64	
7	s1034	Sadia Zia	BIT	132	
8	s1038	Shoaib Ali	BCS	134	
9	s1040	Ahmad Ali	BCS	134	
10	s1042	Ahmad Ali	BCS	134	

Aggregate functions

➤ CREATE VIEW st_view3

(a1, a2) AS

(select prName, avg(cgpa) from
student group by prName)

➤ Select * from st_view3

	a1	a2
1	NULL	NULL
2	BCS	3.20000000476837158
3	BIT	2.79999999523162842
4	MBA	2.79999999523162842
5	MCS	2.7749999761581421

Materialized Views

- Views are virtual tables
- Query executed every time
- For complex queries involving large number of join rows and aggregate functions
- Problematic



Materialized Views

- Solution is materialized views also called indexed views created through clustered index
- Creating a clustered index on a view stores the result set built at the time the index is created.

Materialized Views

An indexed view also automatically reflects modifications made to the data in the base tables after the index is created, the same way an index created on a base table does.

Materialized Views

Create indexes only on views where the improved speed in retrieving results outweighs the increased overhead of making modifications.

Materialized views

```
alter view st_view1 with  
schemabinding as (select  
stName, stFname, prName from  
dbo.student where prName =  
'MCS')
```

Materialized Views

```
create unique clustered index  
stdview_ind1 on st_view1  
(stfname)
```

Partitioned Views

- Lets not discuss it
- Idea is, data lying at multiple places and combined in a view
- Same table partitioned horizontally based on some condition

Simple or Complex Views

- One form of classification
- View on single table is simple, while involving multiple tables is called complex view
- There is lot more on views

Transaction Management

Transaction Management

➤ Comprises

- Database Recovery
- Concurrency Control

Transaction

- The base of Transaction Management (TM)
- A logical unit of work performed on the database
- May involve one or more operations

Database Management System

Lecture - 41

Database Management System

Lecture - 42

Transaction

- Generally reflects the activities in the real world system
- One transaction in our exam system may be computing gpa of a student, Operations involved.....
- More complex transactions

Consistent State of DB

- Consistent state of the database
 - Duplicated data does not conflict
 - DB properly reflect the real-world system, i.e. follows the business rules
- A transaction should transform database from one consistent state to another consistent state, may be inconsistent temp

Transaction Termination

- Two ways a transaction terminates or ends
- It may Commit or Abort
- If executed successfully, brings the database in a new consistent state, said to be committed

Transaction Termination

- If otherwise, database has to be brought in prior consistent state
- Such transaction is rolled back or undone
- A committed transaction cannot be undone; compensating transaction

Transaction Boundaries

- Means being and end of Tr
- Generally responsibility of programmer
- Otherwise approach varies from DBMS to DBMS

ACID Properties

- Atomicity: all or none
- Consistency: maintains consistency of the DB
- Isolation: Tr should be protected from the effect of other Trs
- Durability: changes by Tr are permanent

Database Updates

- Updates are first made in the buffers in RAM, then transferred to database
- There may be some delay between the two

Database Recovery

- If a crash occurs during this delay then the database is in in consistent state
- On restart DBMS has to identify it and recover DB into a consistent state

Reasons of Failure

- Natural disaster
- Sabotage
- Carelessness
- Disk crash
- System software errors
- Generally, improper shut down of DBMS

Recovery Techniques

- System crashes, buffers are lost but the disk copy is safe
- Need to determine the transactions that need to redone or undone

Log File

- Tool used for database recovery
- Transaction Record
 - $\langle T, \text{starts} \rangle$
 - $\langle T, \text{commits} \rangle$ or $\langle T, \text{aborts} \rangle$
 - Data entries?

Transaction Operations

Read (X)

$X = X + 5$

Write (X)

$Y = Y * 3$

Write (Y)

Commit

➤ Operations of DBMS
concern

➤ Concern of Recovery
Manager

➤ So log file contains
entries only for write
operations

Deferred Updates

- Called incremental log with deferred updates
- Updates are not performed until commit is encountered
- Data entries in the log file of the form $\langle T, X, c \rangle$; simplified

Log File Entries

Read (X)	Supposing	<T, starts>
$X = X + 5$		
Write (X)		<T, X, 55>
$Y = Y * 15$	$X = 50$	
		<T, Y, 30>
Write (Y)		
Commit	$Y = 10$	<T, commit>

Database Management System

Lecture - 42

Database Management System

Lecture - 43

Write Sequence

- For a write operation entry is made in log file in RAM
- On commit, first,
 - Database buffer is updated
 - Log file is moved to disk log file
- Then write is performed from DB buffer is moved to disk

Recovery Sequence

- If system crashes
- On restart, recovery manager examines the log file
- Transactions with begin and commit to be redone
- No action for begin and aborted

Checkpoint Record

- Repetition, no harm
- How far in the log file should be redone
- Checkpoint record is inserted in log file at certain intervals

At Checkpoint

- Modified DB buffers to disk
- All log records from buff to disk
- Writing a checkpoint record to log; log record mentions all active transactions at the time

Recovery Routine

- Check last check point
- Any transaction committed before checkpoint, no action
- On or started after and commit: redone
- On and abort, or on and no end: no action

Summary of Deferred Updates

- Writes are deferred until commit for transaction is found
- For recovery purpose, log file is maintained
- Log file helps to redo the actions that may be lost due to system crash
- Log file also contains checkpoint records

Immediate Updates

- Incremental log with immediate updates
- DB buffers are updated immediately, and files updated when convenient

Write Sequence

- Start Tr record in log, $\langle T, \text{start} \rangle$
- On write operation, write a log record $\langle T, X, \text{old val}, \text{new val} \rangle$
- Move log record to file
- Update the DB when convenient
- On commit write $\langle T, \text{commit} \rangle$ in log file

Recovery Sequence

- Trs with commit to redo, by copying new values
- Trs, with abort or no end undo, by copying the old value
- Repetition doesn't matter
- Checkpoint record can be used

Summary of DB Recovery

- An improper shut down may damage database consistency
- Has to be detected and database to be brought in an consistent state

- Log file contains records of all updates
- Updates may be deferred or immediate, record entries vary
- Transactions are redone or undone depending on situation
- Checkpoints help in efficient database recovery

Concurrency Control

Concurrent Access

- Sharing, one of the basic objective
- Multiple users accessing simultaneously is preferred
- Operations of more than one transactions can be performed simultaneously; interleaved transactions

Concurrent Access

- Two factors allow concurrent access
 - Processors being very fast
 - I/O controller can handle I/O
- Problems due to interleaving

Concurrency Control?

➤ Three problems due to interleaved transaction

- Lost Update
- Uncommitted Update
- Inconsistent Analysis

Lost Update Problem

TIME	TA	TB	BAL
t_1	Read (BAL)		1000
t_2	Read (BAL)	1000
t_3	BAL = BAL - 50	1000
t_4	Write (BAL)		950
t_5	BAL = BAL + 10	950
t_6	Write (BAL)	1010

Database Management System

Lecture - 43

Database Management System

Lecture - 44

Uncommitted Update Problem

TIME	TA	TB	BAL
t_1	Read (BAL)	1000
t_2	BAL=BAL+100	1000
t_3	Write (BAL)	1100
t_4	Read (BAL)	1100
t_5	BAL=BAL*1.5	1100
t_6	Write (BAL)	2650
t_7	ROLLBACK	?

Inconsistent Analysis

- One transaction operates many records, meanwhile another modifies some of them effecting result of first
- Suppose Tr-A computes interest on all accounts balances and other moves balance from one account to another

TIME	TA	TB
t_1	Read (BAL_A) (5000)
t_2	$INT = BAS_A * .05$
t_3	$TOT = TOT + INT$
t_4
t_5	Read (BAL_A)
t_6	$BAL_A = BAL_A - 1000$
t_7	Write (BAL_A)
t_8	Read (BAL_E) (5000)
t_9	$BAL_E = BAL_E + 1000$
t_{10}	Write (BAL_E)
t_{11}	Read (BAL_E) (6000)
t_{12}	$INT = BAS_E * .05$
t_{13}	$TOT = TOT + INT$

Serial Execution

- An execution where transaction are executed in a serial
- Schedule or History
- For two transactions two serial executions or schedules are possible

Serial Execution

TA, TB	BAL	TB, TA	BAL
Read (BAL) _A	50	Read (BAL) _B	50
(BAL = BAL - 10) _A	50	(BAL=BAL * 2) _B	50
Write (BAL) _A	40	Write (BAL) _B	100
Read (BAL) _B	40	Read (BAL) _A	100
(BAL=BAL * 2) _B	80	(BAL = BAL - 10) _A	100
Write (BAL) _B	80	Write (BAL) _A	90

Serial Execution

- Different serial schedules may result in a different final DB state, BUT
- Serial execution is guaranteed to leave DB in a consistent state
- None of the three concurrent access problems

Lesson?

- Serial schedule is guaranteed correct, should we adopt it?
- NNNNNNNNo, that will be inefficient, under utilization of the resources
- Disliked by users

Interleaved Schedule

- We would prefer concurrent execution, multiple users accessing database at the same time
- An interleaved schedule consists of operations from different transactions

Schedule

TA = {Read(X), Write(X), commit}

TB = {Read(Y), Read(X), Write(Y), commit}

S₁ = {R_A(X), W_A(X), C_A, R_B(Y), R_B(X), W_B(Y), C_B}

S₂ = {R_B(Y), R_B(X), W_B(Y), C_B, R_A(X), W_A(X), C_A}

S₃ = {R_B(Y), R_B(X), R_A(X), W_B(Y), C_B, W_A(X), C_A}

S₄ = {R_A(X), R_B(Y), R_B(X), W_A(X), C_A, W_B(Y), C_B}

Interleaved Schedule

- Interleaved schedule provides concurrent execution BUT
- Three problems of concurrent access may be encountered

Real Problem

- Two transactions
 - Writing different variables
 - Reading same variable
 - Accessing same variable and one of them writing
- Conflicting operations
- Conflicting transactions

Serializability

- An interleaved schedule is serializable if the final state of the schedule is equivalent to a serial schedule OR
- A schedule where conflicting operations are in a serial order

Serializability

➤ Two major approaches

- Locking
- Timestamping

Locking

- An object is locked before it performs any operation
- Two types of operations, two types of locks
- Shared and exclusive

Lock Compatibility

TA holds	TB requests	
	Shared	Exclusive
Shared	YES	NO
Exclusive	NO	NO

Database Management System

Lecture - 44

Database Management System

Lecture - 45

Locking Idea

- Before performing an operation a transaction applies for a S/E lock on an object
- If object free or lock compatible then lock granted otherwise transaction will wait

Deadlock

- Deadlock is a situation when two transactions are waiting for each other to release a lock
- The transaction involved in deadlock keep on waiting unless deadlock is over

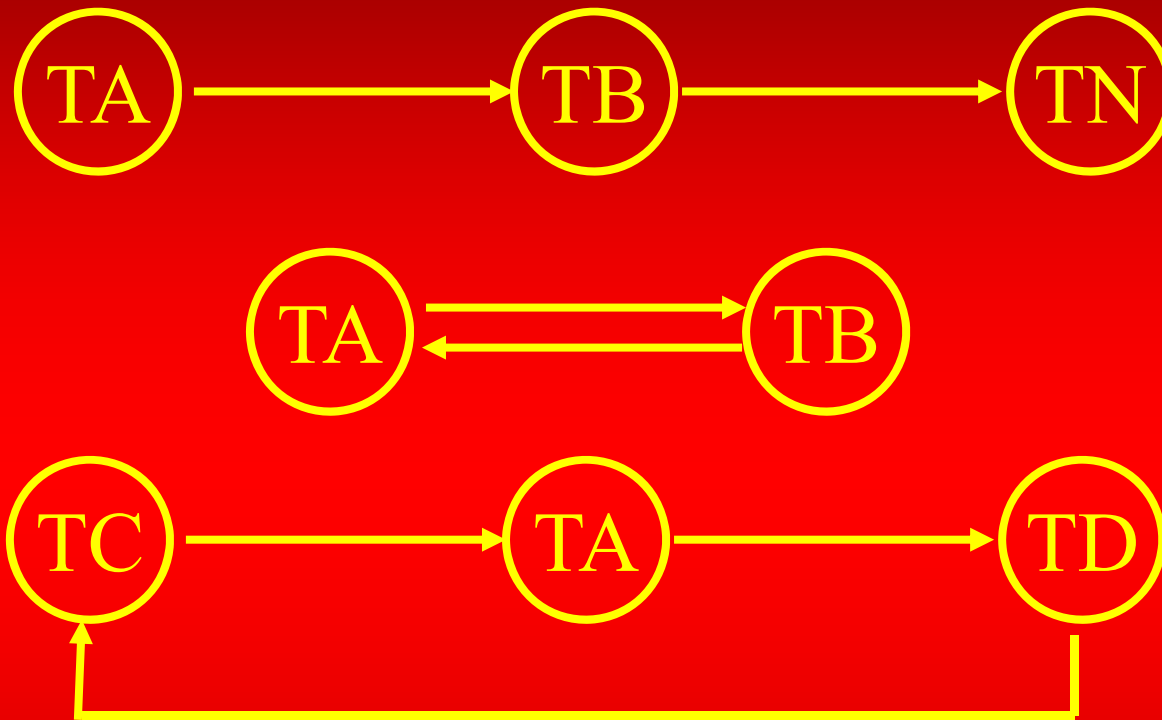
Deadlock

TIME	TA	TB
t_1	XLock (X)
t_2	XLock (Y)
t_3	Request Y
t_4	Wait	Request X
t_5	Wait	Wait
t_6	Wait	Wait

Deadlock Handling

- Deadlock prevention
- Deadlock detection and resolution
- Prevention is not always possible
- Deadlock is detected by wait-for graph

Wait-for Graph



Victim Transaction

Two-Phase Locking

- 2PL ensures serializability but generates deadlock
- The locks are granted and released in two phases, the growing and shrinking phase

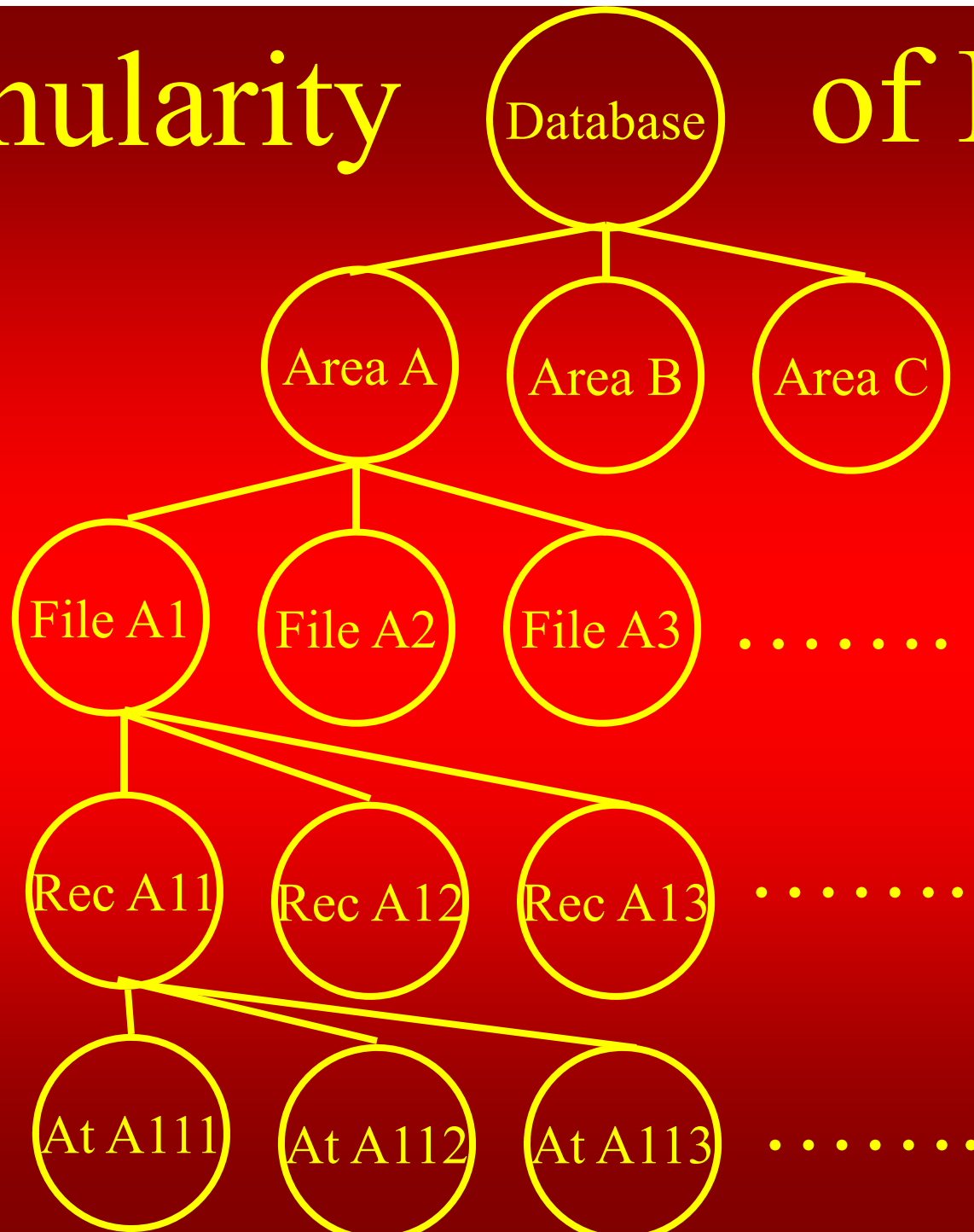
Two-Phase Locking

- A transaction must acquire a lock before doing any operation
- Locks are only granted in growing phase, once a transaction enters in shrinking phase no locks can acquire

Levels of Locking

- Lock can be applied on attribute, record, file, group of files or even entire database
- Granularity of locks
- Finer the granularity more concurrency but more overhead

Granularity of Locks



Granularity of Locks

- When a lock at lower level is applied, compatibility is checked upward
- How will a lock at higher level know about a lock at lower level
- Individual checking is not feasible

Intension Locks

- When a lock is applied on a node, an intension shared (IS) or intension exclusive (IE) lock is applied on all nodes till the root
- An intension lock at a node indicates a lock at the lower level

Intension Locks

- While applying for a lock on a node, compatibility with the intension lock is also checked
- Share intension lock (SIX)
- Conflicts with locks conflicting with S or IX; only one SIX

Extended Comp Matrix

S IS X IX SIX

S	Y	Y	N	N	N
IS	Y	Y	N	Y	Y
X	N	N	N	N	N
IX	N	Y	N	Y	N
SIX	N	Y	N	N	N

Locking Summary

Timestamping

- No locks are used
- Guarantees consistency and no deadlocks, however, certain transactions are cancelled
- Based on the concept of timestamp

Timestamp

- An identifier that determines the relative order of the transactions
- Greater timestamp younger the transaction
- The transactions' operations are allowed in the timestamp order

Timestamp

- Timestamps are also allocated to data items as well
- A read timestamp and a write timestamp to determine the last transaction that read or wrote

Two Problems

- When a transaction wants to read an item that has been updated by a younger transaction
- A transaction wants to write an item that has been read or written by a younger transaction

Versions of Data Items

- First problem is met by maintaining multiple versions of data items

Value	read-timestamp	write-timestamp
742	6	8

Versions of Data Items

	Value	read-timestamp	write-timestamp
V1	742	6	6
V2	1096	9	9

Transaction 8 wants to read the data item

Versions of Data Items

	Value	read-timestamp	write-timestamp
V1	742	6 8	6
V2	1096	9	9

Transaction 8 gets the smaller nearest value
Read timestamp is changed

Problem 2

- Write request when an item read or written by smaller transaction

	Value	read-timestamp	write-timestamp
V1	742	8	6
V2	1096	9	9

Transaction 7 wants to write the item

The End

Thanks all

Timestamp Algorithm

➤ On read request

- Check the read TS of latest version
- If latest write TS < requested TS
 - Give the value of latest
- otherwise find the largest smaller write TS and return that value
- Update the read TS of data item

Timestamp Algorithm

➤ On write request

- Check the read and write TS of latest version
- If either one is greater than the requested TS, cancel the transaction and restart
- If none of latest is greater
 - Create a new version, place new value and current TS in write TS of data item

Database Management System

Lecture - 45