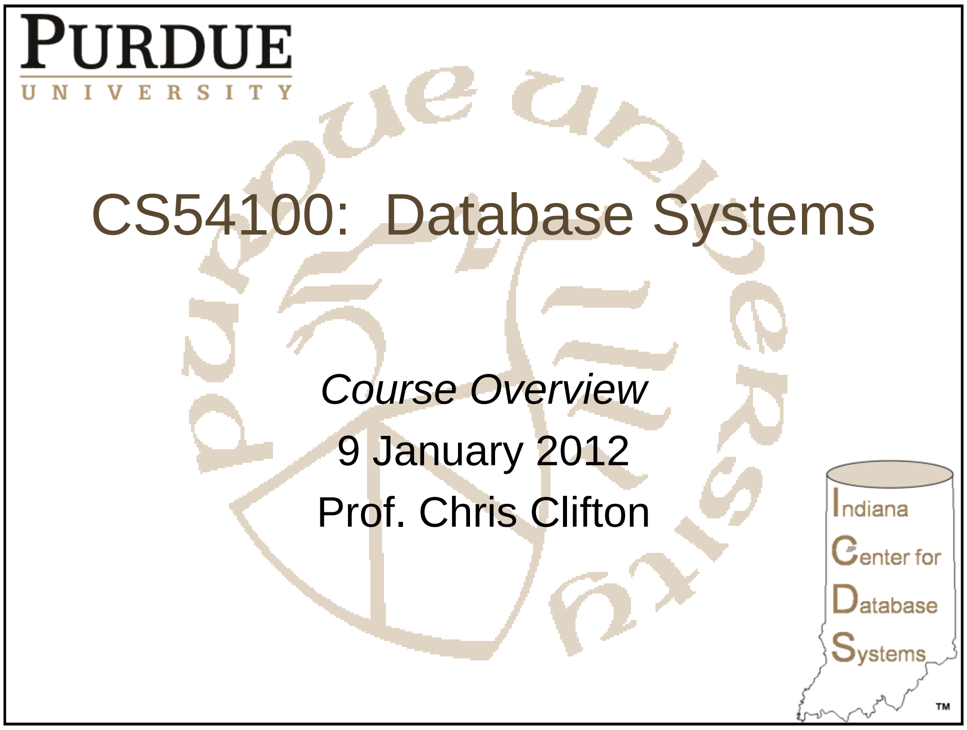


PURDUE
UNIVERSITY

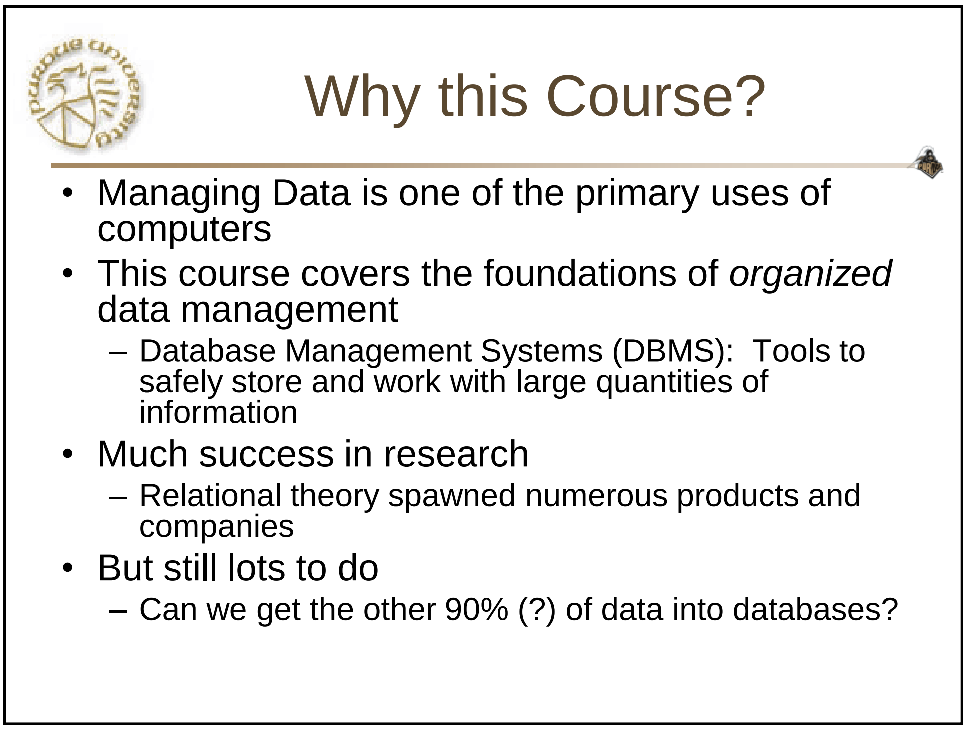
# CS54100: Database Systems

*Course Overview*

9 January 2012

Prof. Chris Clifton

Indiana
Center for
Database
Systems
™



## Why this Course?

- Managing Data is one of the primary uses of computers
- This course covers the foundations of *organized* data management
  - Database Management Systems (DBMS): Tools to safely store and work with large quantities of information
- Much success in research
  - Relational theory spawned numerous products and companies
- But still lots to do
  - Can we get the other 90% (?) of data into databases?

## Course Information

- Contact Information: Professor Clifton
  - Office: LWSN 2142F, x4-6005.
  - Office hours: TBD, for now generally 8-5
  - clifton@cs.purdue.edu
- Teaching Assistant: Duong Nguyen
  - Office: TBD
  - Office hours: TBD
- Course Web Page: http://www.cs.purdue.edu/homes/clifton/cs541/

## Course Methodology

- Lectures to present the concepts
  - Unless otherwise noted, all the material you are expected to know will be covered in class
  - Interaction (questions/discussion/thinking) encouraged
- Reading will fill in the details
  - Text: Database Management Systems, by Raghu Ramakrishnan and Johannes Gehrke. McGraw Hill, 2003, ISBN 0-07-246563-8
  - Other readings (e.g., research literature) will be made available where appropriate
- Homework and Projects get you to *understand* what you've read and heard
  - 5-6 written homeworks
  - 3-4 programming projects
  - Suggested exercises from the text (ungraded)

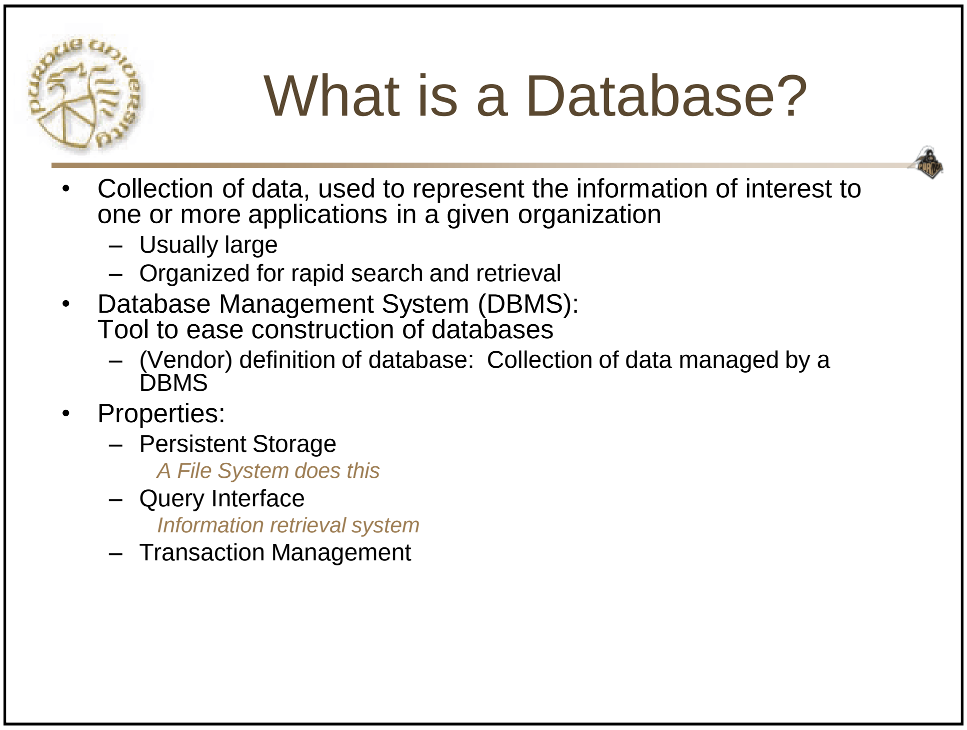
## Evaluation and Grading

- Points earned as follows:
  - Midterm (20%)
  - Final Exam during Finals week (30%)
  - Homeworks / projects (45%)
    - Larger projects may be given higher weight
  - Instructor's evaluation (5%)
    - In-class discussions/participation
    - Out of class discussions, email
    - Overall perception of quality of your work in ways that may not be reflected in your scores
- Late work penalized 10% per day
- Qualifying Exam: One hour supplement to regular exam
  - Passing the qualifier requires both suitable performance in the course and on the supplemental exam

*For more details see the course web page*

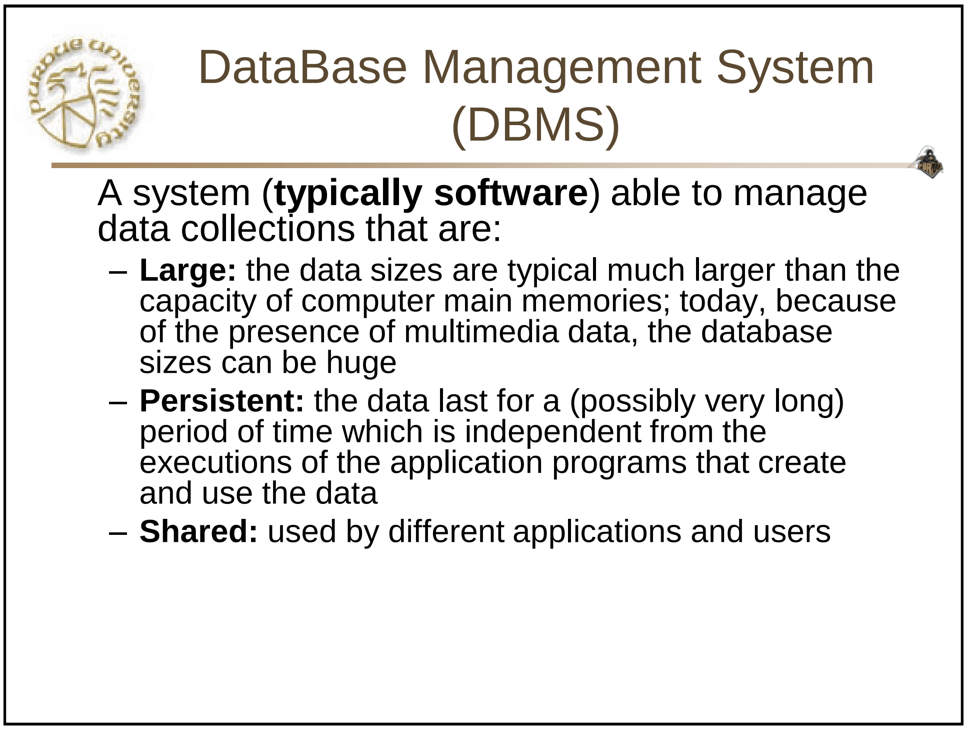
## Course Outline

1. Course Introduction
   - Intro / history lesson
   - Background check (Yes, a (take-home) **quiz**. It isn't graded – just so I know how fast to move through early material)
   - Relational Model
2. Data Modeling
   - Entity-Relationship Data Model
   - Constraints and Constraint Modeling
3. Relational Theory
   - Relational Algebra and Calculus
   - Keys and Dependencies
   - Normalization
4. Using a Relational Database
   - Views
   - Constraints
   - Triggers
5. Storage mechanisms
6. Putting the Data on Disk
7. Indexing
8. Hashing / Bitmap Indexes
9. Query Processing
10. Query Optimization
11. Handling Failure
12. Concurrency Control
13. Transaction Management
14. Research topics
15. Review



## What is a Database?

- Collection of data, used to represent the information of interest to one or more applications in a given organization
  - Usually large
  - Organized for rapid search and retrieval
- Database Management System (DBMS): Tool to ease construction of databases
  - (Vendor) definition of database: Collection of data managed by a DBMS
- Properties:
  - Persistent Storage
    *A File System does this*
  - Query Interface
    *Information retrieval system*
  - Transaction Management



## DataBase Management System (DBMS)

A system (**typically software**) able to manage data collections that are:

- **Large:** the data sizes are typical much larger than the capacity of computer main memories; today, because of the presence of multimedia data, the database sizes can be huge
- **Persistent:** the data last for a (possibly very long) period of time which is independent from the executions of the application programs that create and use the data
- **Shared:** used by different applications and users

## DBMS

- A DBMS must assure:
  - **Reliability:** the data must survive to hardware and software errors
  - **Confidentiality**: access to data must be controlled
- As the majority of computer systems, a DBMS must be **efficient** (by optimizing the resources of the underlying system) and **effective** (by allowing users to make productive use of data)

## Motivations for DB Technology
*Organization/Enterprise*

- It uses a set of resources, policies and regulation to execute the activities of interest for its own goals
  - Information and knowledge represent a key resource
- The *information system* is today always present in any organization we may think of
- The information system executes/manages information processes, that is, processes that involve information

## Motivations for DB Technology
*Information Systems*

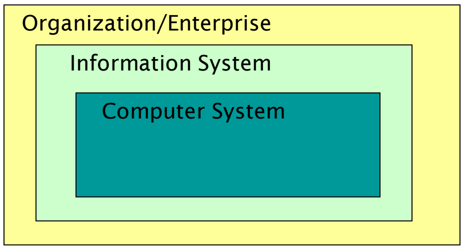
- It is a component (subsystem) of a given organization.
- It manages (that is, acquires, processes, stores, produces, and delivers) information of interest to the organization.
- Each organization has an information system, even though such system may not be explicitly present the organization structure
- In most cases, the information system supports other subsystems in the organization and therefore it may have to support users and applications from different sectors of the organization
- The information system is usually organized in subsystems (with a distributed or hierarchical organization); these subsystems may be tigthly or loosely coupled

## Motivations for DB Technology
*Automation of Information Systems*

- The concept of "information system" is independent from IT technology: there are organizations, the goal of which is to manage information (like in the case of demographic services), and that have been in place for centuries

## Motivations for DB Technology
*Computer system*

- Automated portion of the information system
- Component of the information system that manages information through the use of computer systems



## Motivations for DB Technology
*Database systems*

- Database systems represent the most important computer technology for implementing and supporting information systems
- Therefore, they represent a key technology for any modern organization/enteprise
- Because information are a key resources, database systems must be reliable, secure and efficient

## Motivations for DB Technology
*Information and Data*

- Computer systems represent information through data
- *Data* represents information. *Information* is the (subjective) interpretation of data
- ***Data* -** Physical phenomena chosen by convention to represent certain aspects of our conceptual and real world. The meaning we assign to data are called information. Data is used to transmit and store information and to derive new information by manipulating the data according to formal rules.

  from:

  P.Brinch Hansen. *Operating Systems Principles*. Prentice-Hall, 1973.

## Motivations for DB Technology
*Information and Data*

- The data are elementary facts that need to be interpreted in order to convey information
- Example

  Consider a data item represented by the integer number 3; such number does not provide any information

  By contrast, saying that 3 is the number of credits of CS541 provides some information

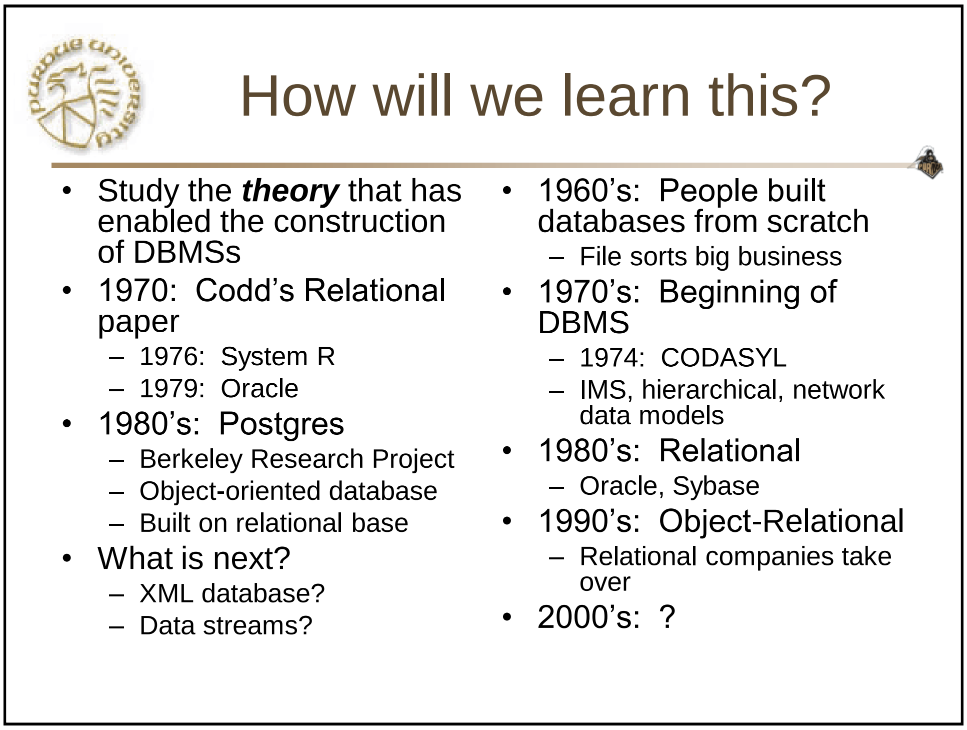
## Motivations for DB Technology
*Information and Data*

- One of the fundamental goals of a database management system (DBMS) is to provide <u>an interpretation context to a collection of data</u>, so that users can effectively access information encoded by this collection of data

## What are we studying?

- Methods to build databases
  - Data modeling
  - Query languages

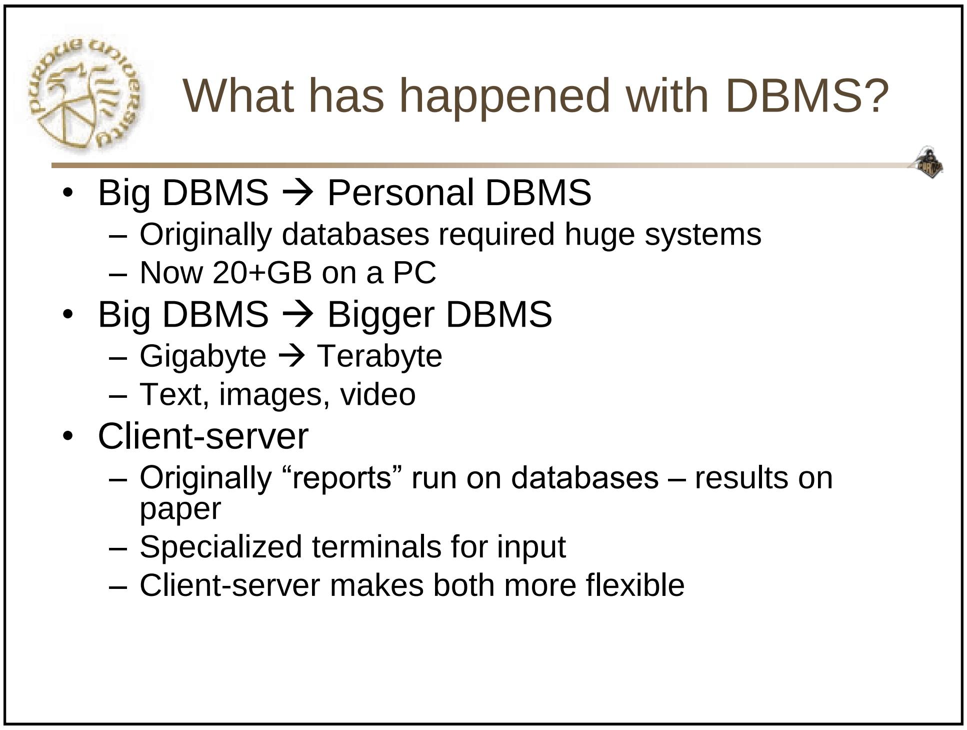
  *We'll try to cover this quickly – many may know it*
- Methods to build DBMSs
  - Storage (safe, persistent)
  - Query (how to make them fast)
  - Transactions (how to make a lot happen at once)



## How will we learn this?

- Study the ***theory*** that has enabled the construction of DBMSs
- 1970: Codd's Relational paper
  - 1976: System R
  - 1979: Oracle
- 1980's: Postgres
  - Berkeley Research Project
  - Object-oriented database
  - Built on relational base
- What is next?
  - XML database?
  - Data streams?

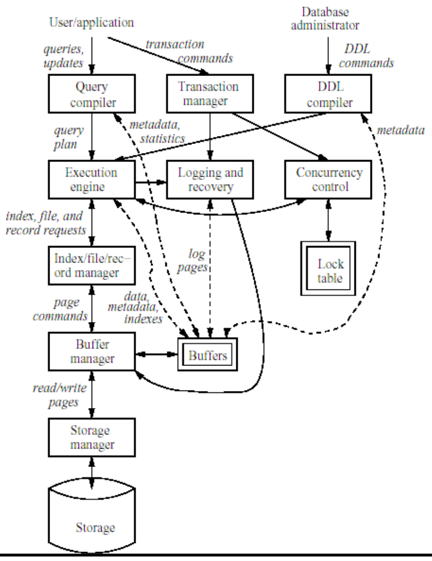
- 1960's: People built databases from scratch
  - File sorts big business
- 1970's: Beginning of DBMS
  - 1974: CODASYL
  - IMS, hierarchical, network data models
- 1980's: Relational
  - Oracle, Sybase
- 1990's: Object-Relational
  - Relational companies take over
- 2000's: ?



## What has happened with DBMS?

- Big DBMS → Personal DBMS
  - Originally databases required huge systems
  - Now 20+GB on a PC
- Big DBMS → Bigger DBMS
  - Gigabyte → Terabyte
  - Text, images, video
- Client-server
  - Originally "reports" run on databases – results on paper
  - Specialized terminals for input
  - Client-server makes both more flexible

## What goes in to a DBMS?



- Query Compilation
  - Turn a declarative query to procedural execution
  - What is the fastest way to get the result?
- Transaction Management
  - Try to run lots at once
  - Ensure queries don't interfere with each other
- Storage Management
  - Disks are slow – how do we get to the data fast?
  - Minimize trips to the disk

## Efficiency

- It is measured (like in all comuter systems) in terms of execution times and storage overhead (both main memory and secondary storage)
- The DBMS, because they need to implement a large variety of functions, are not necessarily more efficient than file systems
- The efficiency depends from the DBMS quality and the efficiency of the applications using the database

## Some Big Successes: Indexing

- B-tree: fast search on disk
  - Read a whole block at once
  - Minimize number of reads
- Multi-dimensional indexes
  - When a query asks for name AND date
- What next?
  - "Indexing" data streams?
  - Similarity – when the answer isn't exact?

## Big Successes: Query Processing

- Relational Algebra
  - Concept of equivalent queries
  - Different ways of execution guaranteed to give same result
- Cost-based optimization
  - Use knowledge about the data to decide best plan
- What next?
  - *What about non-relational databases?*

## Data Sharing

- Any organization (especially if large) is typically organized in different sectors or carries out different activities. Each sector or activity corresponds to a sub-system of the information systems
- Very often data pertaining the various sectors overlap
- A database is an **integrated resource**, shared among various sectors or applications

## Sharing

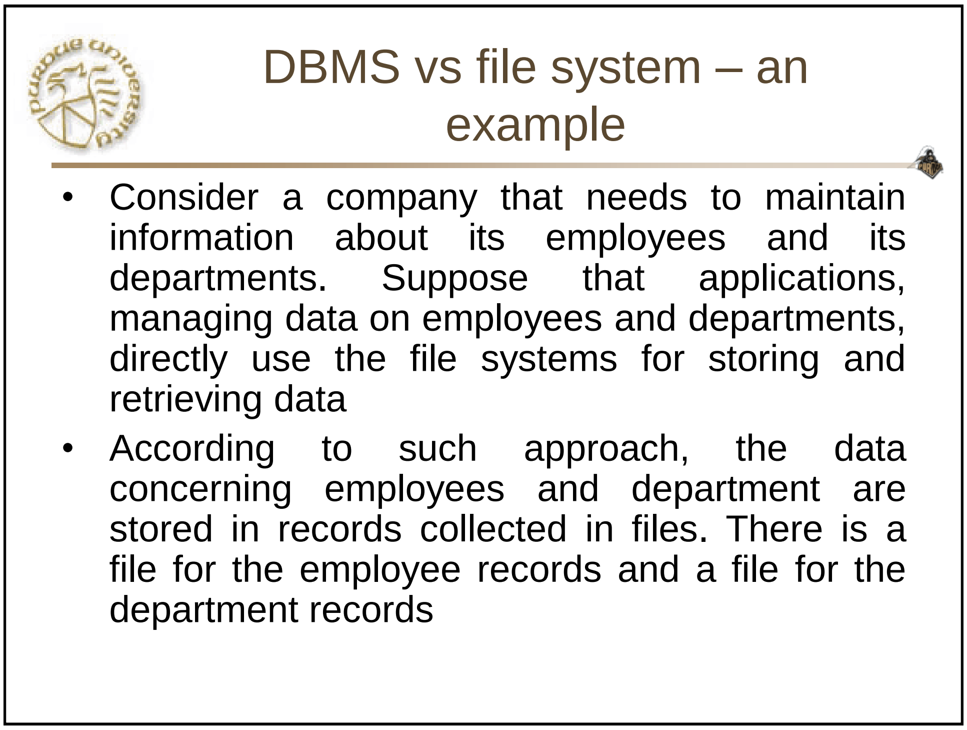
- Integration and sharing reduce data **redundancy** (thus preventing data duplication) and, as consequence, reduce data **inconsistency.**
- Because sharing is usually partial, DBMS provide mechanisms supporting data **confidentiality**, through the **access control mechanisms**
- Data sharing also requires synchronization of concurrent accesses by multiple users and application programs; such synchronization is achieved through the **concurrenty control** mechanism

## Big Successes: Transaction Management

- Serializability
  - Should appear as if queries/updates run sequentially
  - No question of interference!
- Two-phase locking
  - Achieves serializability
  - Allows significant concurrency
- Distributed Transactions
- What next?
  - Models other than serializability?

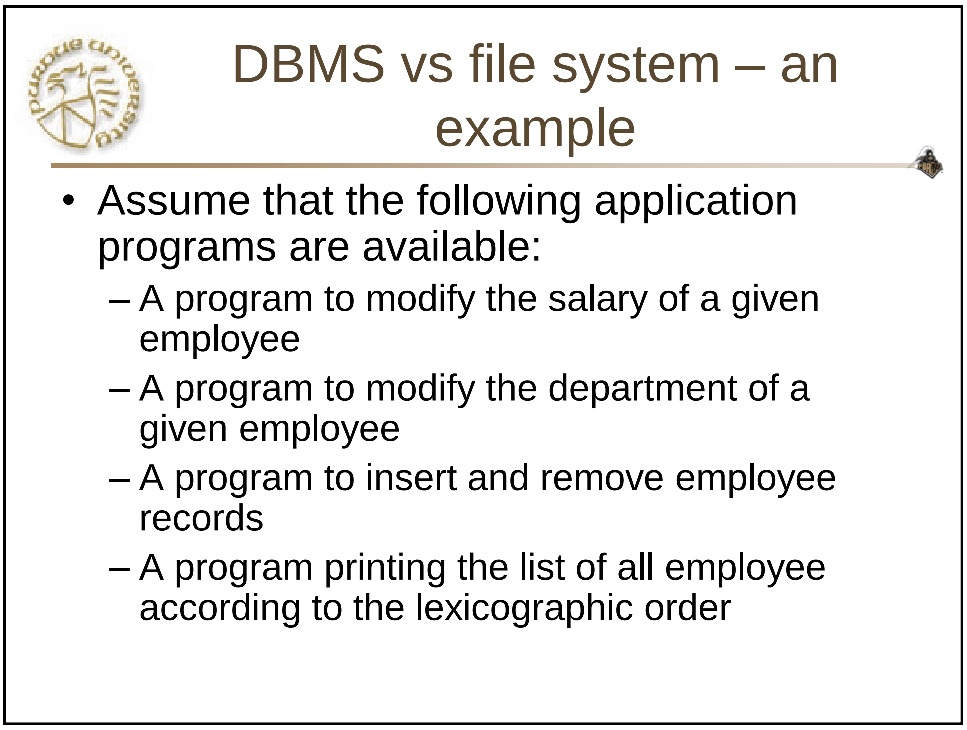
## DBMS vs file systems

- The management of large collections of persistent data is also supported by simplest systems (that is, the **file systems** provided as part of the operating systems). Such systems provide some rudimentary mechanisms for data sharing and data security
- The DBMS extend the main functionality of file systems by providing a large number of data management services and support for data integration
- Also DBMS provide *high-level declarative languages* for data definition and manipulation



## DBMS vs file system – an example

- Consider a company that needs to maintain information about its employees and its departments. Suppose that applications, managing data on employees and departments, directly use the file systems for storing and retrieving data
- According to such approach, the data concerning employees and department are stored in records collected in files. There is a file for the employee records and a file for the department records



## DBMS vs file system – an example

- Assume that the following application programs are available:
  - A program to modify the salary of a given employee
  - A program to modify the department of a given employee
  - A program to insert and remove employee records
  - A program printing the list of all employee according to the lexicographic order

## DBMS vs file system – an example

- The approach based on the use of file systems has the following drawbacks
  - Data redundancy and inconsistency
  - Difficulties in data access
  - Problems with concurrent accesses
  - Problems of fine-grained, content-based access control
  - Problems of data integrity

## Goals of a DBMS

| | | |
|---|---|---|
| Data Integration | $\longrightarrow$ | Enhances the accessibility of data, reduces redundancies and inconsistencies |
| Data Independency | $\longrightarrow$ | Simplifies the development of new applications, and the maintainance of existing applications |
| Centralized Data Control | $\longrightarrow$ | Assures data quality, confidentiality, and integrity |

## Goals of a DBMS

- The basic mechanism that makes possible to integrate data is a logical and centralized definition of the data, referred to as database **schema**
- A schema specifies, through a high level formalism referred to as **data model**, the contents of the database

## Goals of a DBMS

- Data independency is a second important goal. It guarantees that:
  - Changes to the physical data representation (for example, the use of a storage structure instead of another one) do not require changes to the existing applications – **physical data independency**
  - Changes to the logical representation of data do not require changes to the existing applications - **logical data independency**

## Goals of a DBMS

- The third goal is achieved by introducing a centralized control on the data through the DBMS that provide centralized mechanisms for data access and manipulation

## Services provided by a DBMS

| Service | Description |
|---|---|
| Data definition | To specify the data to be stored |
| Data manipulation | To access data, to insert new data, to modify and delete existing data |
| Semantic integrity | To prevent the storage of incorrect data |
| Storage structures | To represent in secondary storage the data model constructs, to store efficienty store and retrieve data |
| Query optimization | To determine the most efficient strategy for data access |
| Access control | To protect data from unauthorized accesses |
| Recovery | To prevent data inconsistency in case of errors and failures |
| Data dictionary | To determine the data stored in a database and access their definitions |

## Course Goal: Be Ready to Lead

- Understand the foundations on which today's work is built
  - Existing database systems
  - Research
- Know enough to:
  - Participate in building the *next* DBMS
  - Be prepared to develop the theory behind the one after that

## Data Models

- A data model is a "conceptual tool", or *formalism*, that includes three fundamental components:
  - One or more data structures.
  - A notation to specify the data through the data structures of the model.
  - A set of operations for managing data; these operations are defined in terms of the data structures of the model.

## Data Models

- A data model allows one to represent real-world entities of interest to a given set of applications
- It is thus useful to identity the basic concepts of such representation; relevant concepts include:
  - *Entity*: an "object" of the application domain
  - *Attribute*: a property of a given entity which meaningful, for the description of the application domain

    Each entity is thus characterized by one or more attributes; an attribute takes one or more values, referred to as *attribute values*, from a set of possible values; such set if referred to as *attribute domain*

## Data Models

- *Entity set*: it groups together a set of "objects" characterized by the same features; that is, it groups “similar” entities having the same attributes, even though these attributes do not necessarily have the same values
- A set of attributes the values of which uniquely identify an entity in a given entity set is a **key** for the entity set
- *Relationship*: a correspondence between the elements of two (or more) entity sets

## Data Models: Example

- Consider the example of employees and departments:
  - **Entities**: John Smith, the department #30
  - **Entity sets**: the set of all employees, the set of all departments
  - **Attributes**: employee name, salary, job, department number, department name
  - **Relationships**: the fact that John Smith works in the department #30

## Data Models

- Each data model must answer two fundamental questions:
  - (a) how to represent the entities and their attributes
  - (b) how to represent the relationships

(a) Almost all models use structures such as the record; each component in a record represents an attribute

(b) Data models widely in this respect; relationships can be represented as:

- specific structures, values, pointers (logical or physical)

## The Relational Data Model

- It is based on a single data structure – the *relation*
- A relation can be seen as a table with rows, called *tuples*, and columns containing values of specific types, such as integer numbers or strings

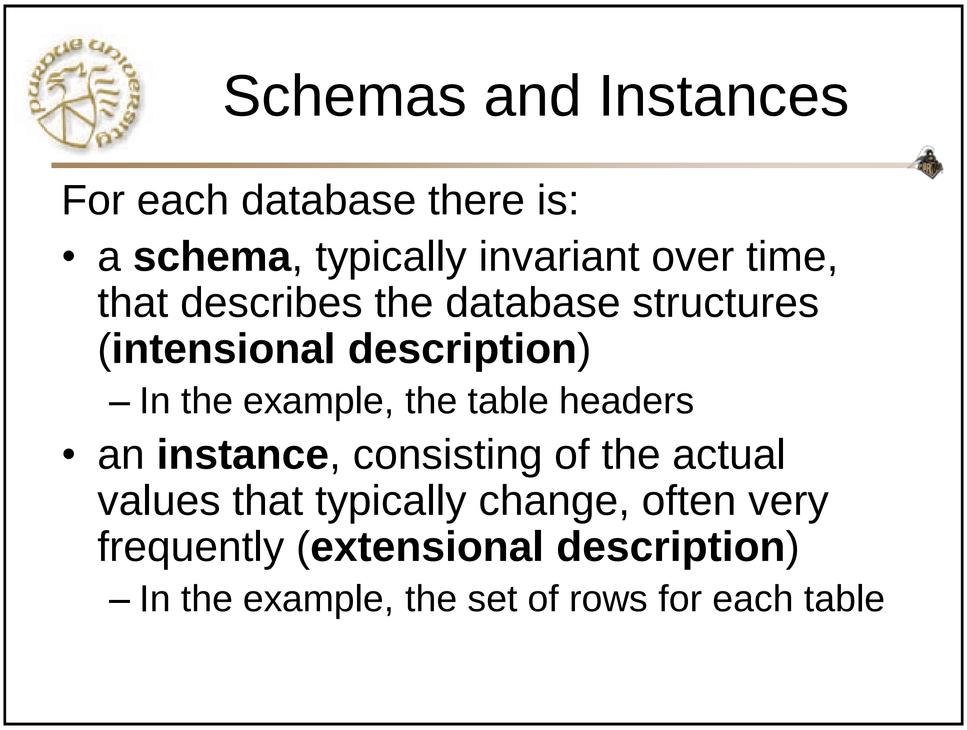
## Tables
## *representation of relationships*

Courses

| Course-Name | Instructor | Room-Name |
|---|---|---|
| Databases | Smith | DS1 |
| Operating Syst. | Black | N3 |
| Networks | White | N3 |
| Security | White | G |

Rooms

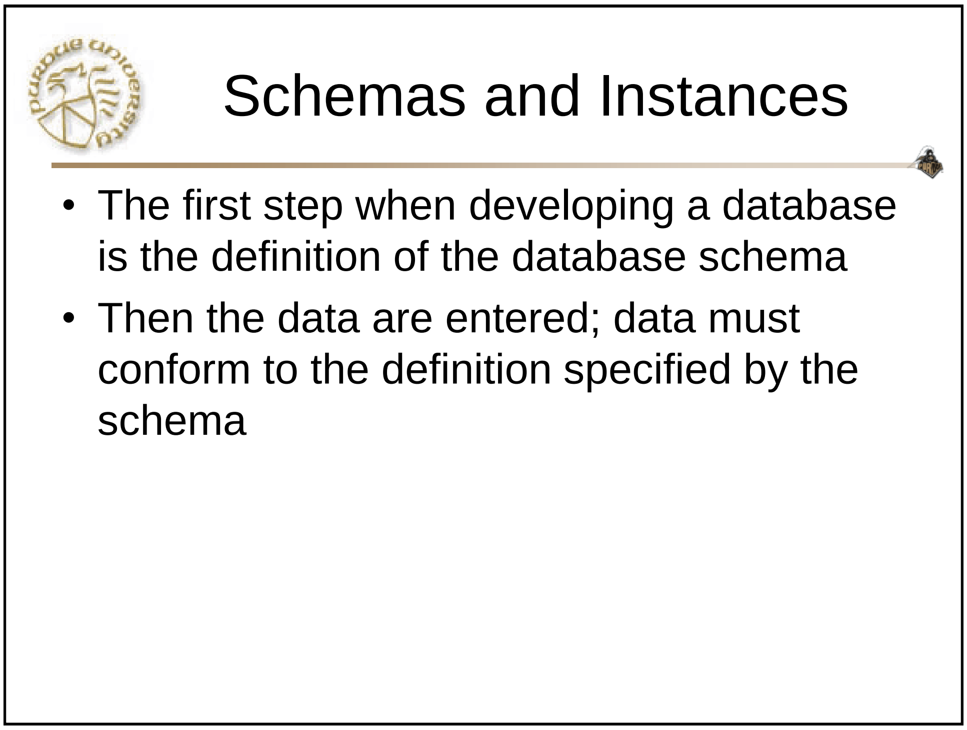
| Room-Name | Building | Floor |
|---|---|---|
| DS1 | Recitation | 1 |
| N3 | Recitation | 1 |
| G | Univ. Hall | 2 |



## Schemas and Instances

For each database there is:

- a **schema**, typically invariant over time, that describes the database structures (**intensional description**)
  - In the example, the table headers
- an **instance**, consisting of the actual values that typically change, often very frequently (**extensional description**)
  - In the example, the set of rows for each table



## Schemas and Instances

- The first step when developing a database is the definition of the database schema
- Then the data are entered; data must conform to the definition specified by the schema

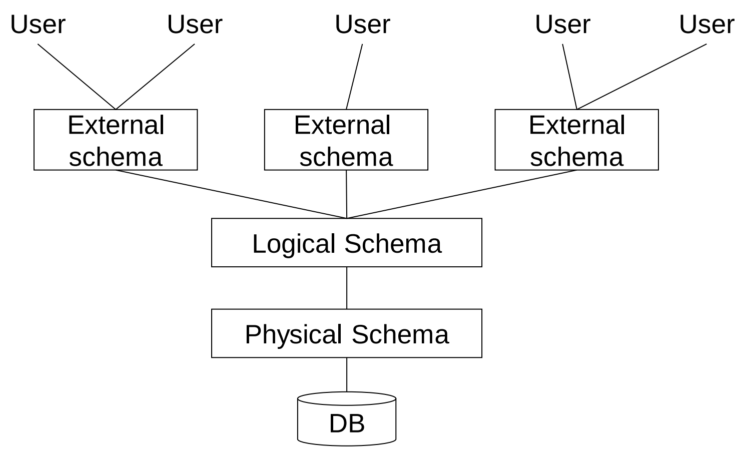
## Main categories of data models

- **Logical models**: used to describe, organize and access data in DBMS; application programs refers to such models. They are independent from the physical data structures
  - examples: relational data model, hierarchical data model, object-relational data model
- **Conceptual models**: support the representation of data independently from specific DBMS. Their goal is to provide representations, which are rich in semantics, of the real word entities, their properties and relationships. These models are mainly used for the conceptual design of databases
  - The **Entity-Relationship** is the most well known model in such category

## Logical Models: Evolution

- First generation (60):
  - Network data model – Codasyl
  - Hierarchical data model
- Relational data model (70)
- Post-relational data models:
  - Nested relational data models
  - Object-oriented data models
  - Deductive data models (e.g. Datalog and its extensions)
  - Object-relational data models
  - XML

## Architecture ANSI/SPARC Three representation levels



## DBMS: languages

- **Data Definition Language** (DDL). It allows one to define:
  - The logical schema of the DB
  - The semantic integrity constraints
  - The authorizations for data accesses
- **Data Manipulation Language** (DML)
  - It is used for data retrieval (query language) and for data updates
- **Storage Definition Language** (SDL)
  - It is used to define physical access structures

## Data Definition Language

- The schema of the DB is specified by using the DDL
- The DDL provides a syntax to specify the constructs of the data model
- The DDL supports the specification of the database name, of the names and definitions of all the logical units in the schema (relations and attributes in the case of the relational model) and of semantic integrity constraints

## Data Manipulation Language

- A database is accessed by users and applications through the DML
- The DML supports four basic types of operations:
  - *Insert*
  - *Query*
  - *Delete*
  - *Update*

## Storage Definition Language

- The correspondences between the logical data structures, specified in the logical schema, and the storage structures must be properly defined
- In all current DBMS such definition is automatically provided by the DBMS once that the schema is defined
- However expert users may require the allocation of additional data structures (such as B-Tree); other aspects of data storage need also to be specified by users, like DBA
- Such requests are issues through the commands of the storage definition language

## SQL, an interactive query language

Courses

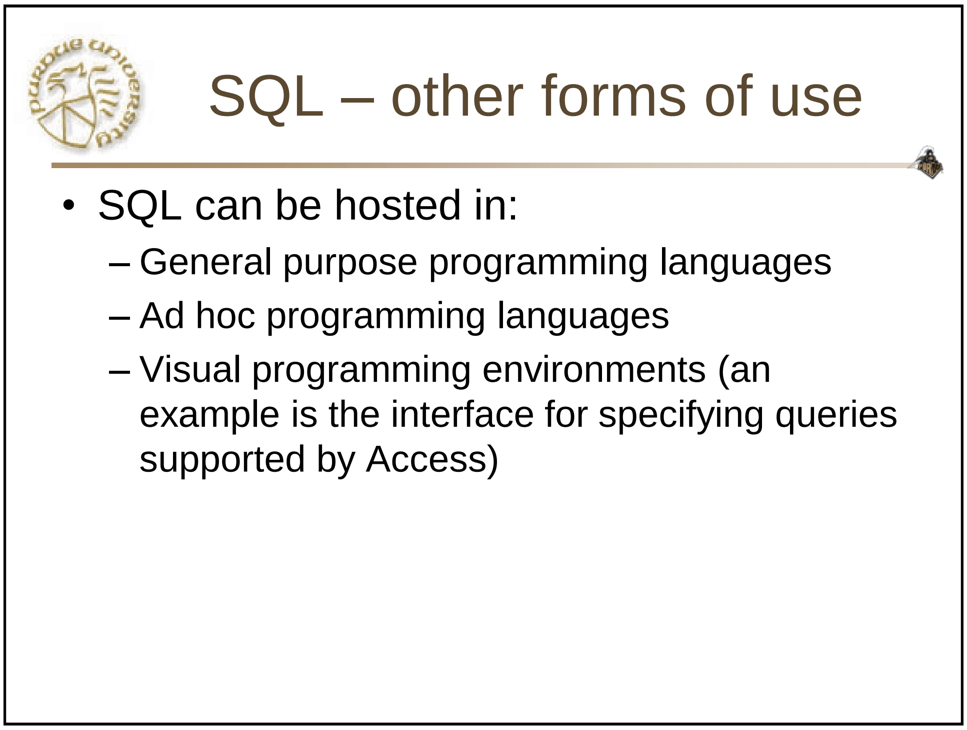
| Course-Name | Instructor | Room-Name |
|---|---|---|
| Database | Smith | DS1 |
| Operating Syst. | Black | N3 |
| Networks | White | N3 |
| Security | White | G |

Rooms

| Room-Name | Building | Floor |
|---|---|---|
| DS1 | Recitation | 1 |
| N3 | Recitation | 1 |
| G | Univ. Hall | 2 |

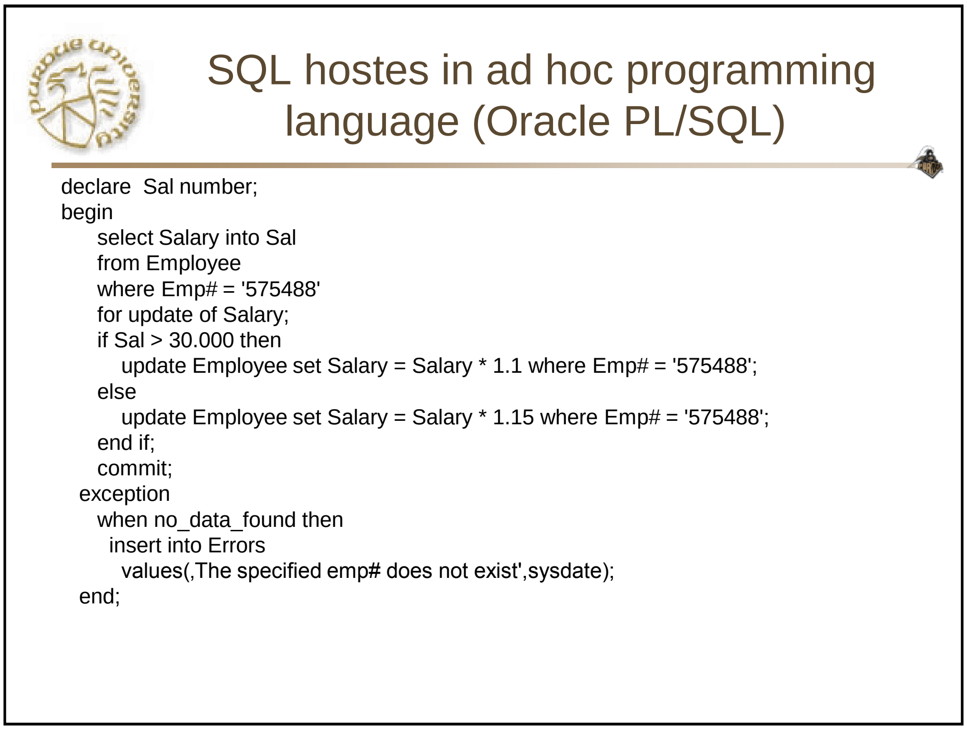
```
SELECT Course-Name, Room-Name, Floor
FROM Courses, Rooms WHERE
Courses.Room-Name =
   Rooms.Room-Name
AND  Instructor = ‘White’;
```

| Course-Name | Room-Name | Floor |
|---|---|---|
| Networks | N3 | 1 |
| Security | G | 2 |



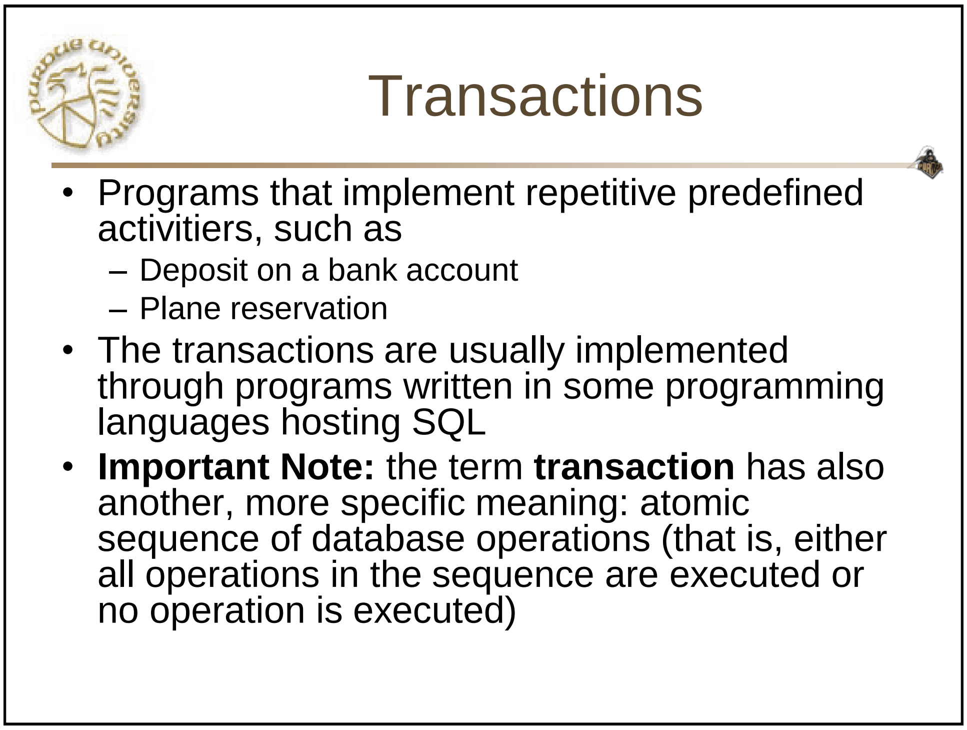
## SQL – other forms of use

- SQL can be hosted in:
  - General purpose programming languages
  - Ad hoc programming languages
  - Visual programming environments (an example is the interface for specifying queries supported by Access)



## SQL hostes in ad hoc programming language (Oracle PL/SQL)

```
declare  Sal number;
begin
      select Salary into Sal
      from Employee
      where Emp# = '575488'
      for update of Salary;
      if Sal > 30.000 then
          update Employee set Salary = Salary * 1.1 where Emp# = '575488';
      else
          update Employee set Salary = Salary * 1.15 where Emp# = '575488';
      end if;
      commit;
   exception
      when no_data_found then
        insert into Errors
          values(,The specified emp# does not exist',sysdate);
   end;
```



## Transactions

- Programs that implement repetitive predefined activitiers, such as
  - Deposit on a bank account
  - Plane reservation
- The transactions are usually implemented through programs written in some programming languages hosting SQL
- **Important Note:** the term **transaction** has also another, more specific meaning: atomic sequence of database operations (that is, either all operations in the sequence are executed or no operation is executed)