Database Concepts

Databases that you may use…..

Flipkart

Spotify

prime video

cleartrip

Expedia musafir.com goibibo.com

make my trip yatra OYO

travelguru The Hotel Experts redBus via.com

IRCTC

NETFLIX

amazon.com

Data

Data

- A necessity for almost any enterprise to carry out its business. Consists of raw facts, and when organized may be transformed into information

Database

- A collection of data organized to meet users’ needs

Database management system (DBMS)

- A group of programs that manipulate the database and provide an interface between the database and the user of the database or other application programs

DBMS ‘Discussion’

A collection of programs that enables you to store, modify, and extract information from a database. There are many different types of DBMSs, ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications:

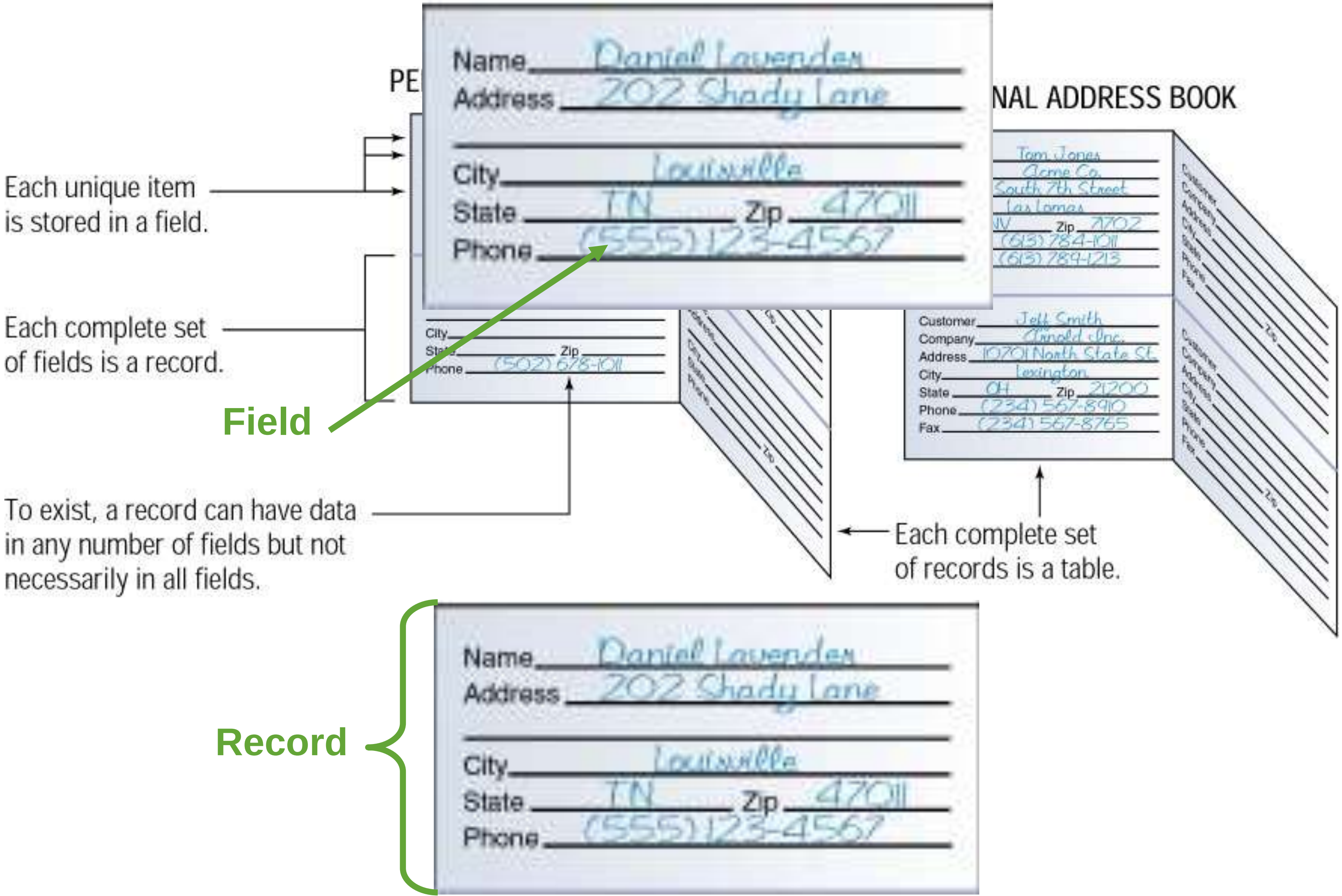
- computerized library systems
- automated teller machines
- flight reservation systems
- computerized parts inventory systems

Hierarchy of Data

Basics of Data Arrangement and Access

The Data Hierarchy

- Recall…8 bits => 1 byte => 1 character
- **Field** - a logical grouping of characters into a word, a small group of words, or a complete number
- **Record** - a logical grouping of related fields
- **File** - a logical grouping of related records
- **Database** - a logical grouping of related files



Each unique item is stored in a field.

Each complete set of fields is a record.

To exist, a record can have data in any number of fields but not necessarily in all fields.

Each complete set of records is a table.

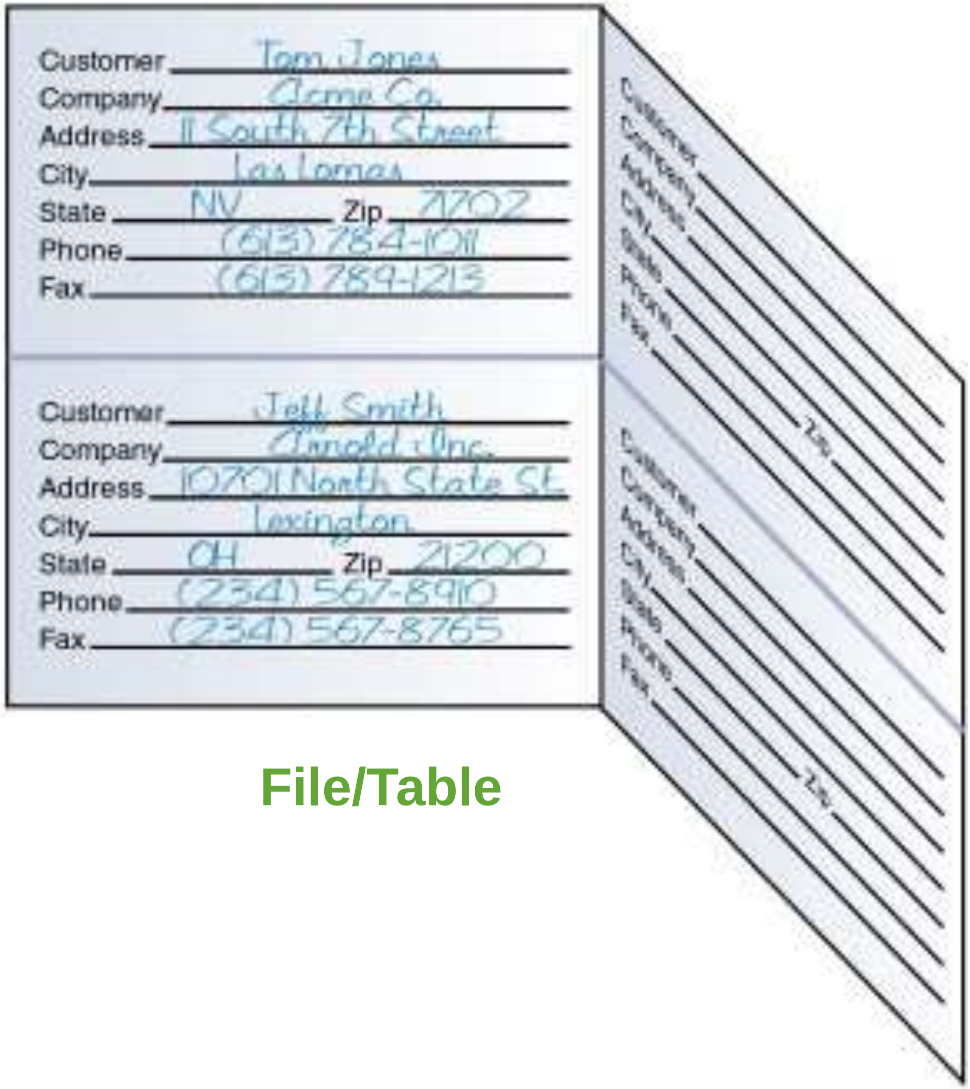
Field

Name Daniel Lavender
Address 202 Shady Lane
City Louisville
State TN Zip 47011
Phone (555) 123-4567

Record

Name Daniel Lavender
Address 202 Shady Lane
City Louisville
State TN Zip 47011
Phone (555) 123-4567

PROFESSIONAL ADDRESS BOOK



Customer Tom Jones
Company Acme Co.
Address 11 South 7th Street
City Las Lomas
State NV Zip 71702
Phone (613) 784-1011
Fax (613) 789-1213

Customer Jeff Smith
Company Arnold Inc.
Address 10701 North State St.
City Lexington
State OH Zip 21200
Phone (234) 567-8910
Fax (234) 567-8765

File/Table

The Hierarchy of Data

Hierarchy of data	Example	
Database	Personel file Department file Payroll file	(Project database)
Files	005-10-6321 Johns Francine 10-7-65 549-77-1001 Buckley Bill 2-17-79 098-40-1370 Fiske Steven 1-5-85	(Personnel file)
Records	098-40-1370 Fiske Steven 1-5-85 598	(Record containing SSN, last name, first name, date of hire)
Fields	Fiske	(Last name field)
Characters (bytes)	1000100	(Letter ‘F’ in ASCII)

Data Entities, Attributes, and Keys

Entity

- A generalized class of people, places, or things (objects) for which data are collected, stored, and maintained
- E.g., Customer, Employee

Attribute

- A characteristic of an entity; something the entity is identified by
- E.g., Customer name, Employee name

Keys

- A field or set of fields in a record that is used to identify the record
- E.g, A field or set of fields that uniquely identifies the record

Keys and Attributes

Employee #	Last name	First name	Hire date	Dept. #
005-10-6321	Johns	Francine	10-7-65	257
549-77-1001	Buckley	Bill	2-17-79	650
098-40-1370	Fiske	Steven	1-5-85	598

Key field

Attributes (fields)

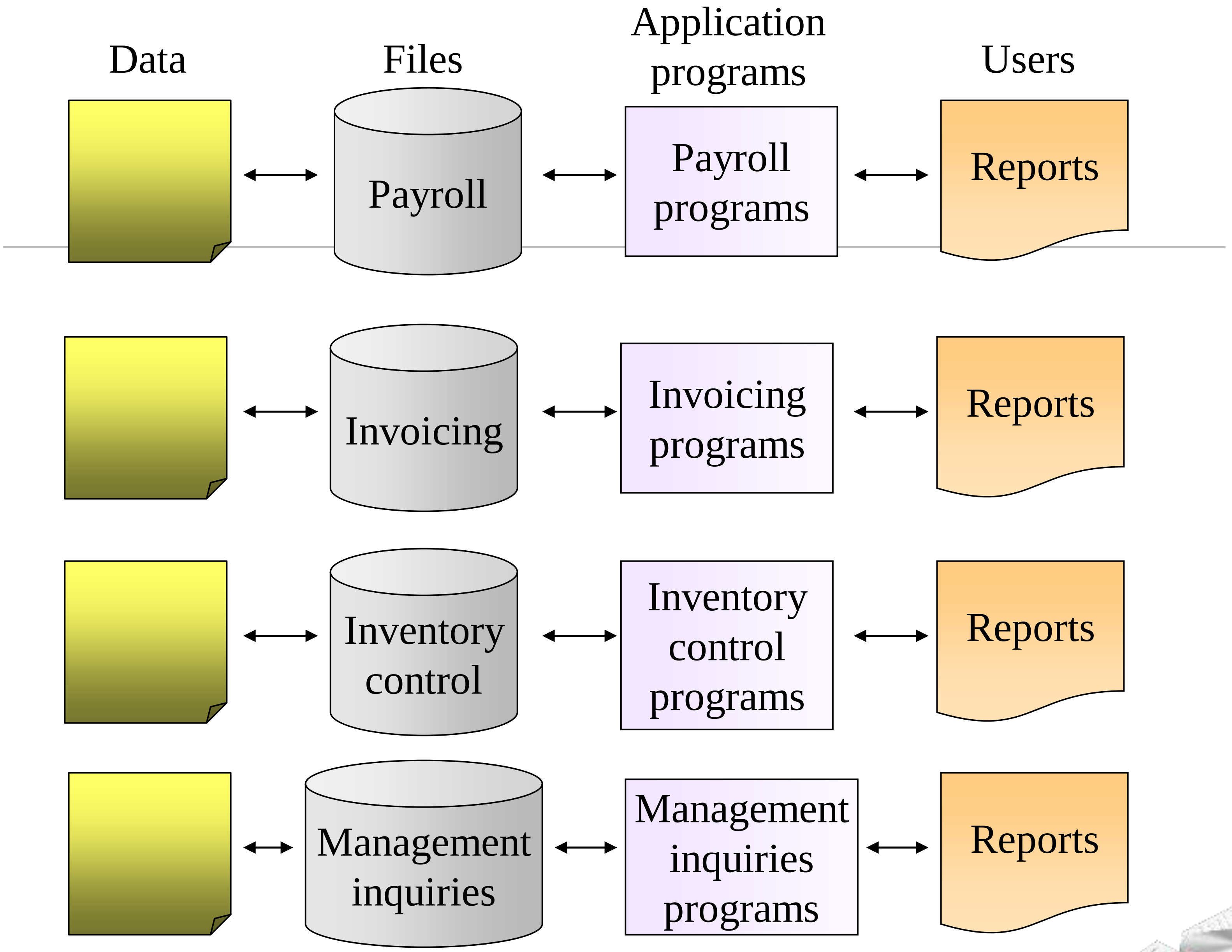
Entities (records)

The Traditional Approach

The traditional approach…

- Separate files are created and stored for each application program

Schematic



Data	Files	Application programs	Users
	Payroll	Payroll programs	Reports
	Invoicing	Invoicing programs	Reports
	Inventory control	Inventory control programs	Reports
	Management inquiries	Management inquiries programs	Reports

Drawbacks

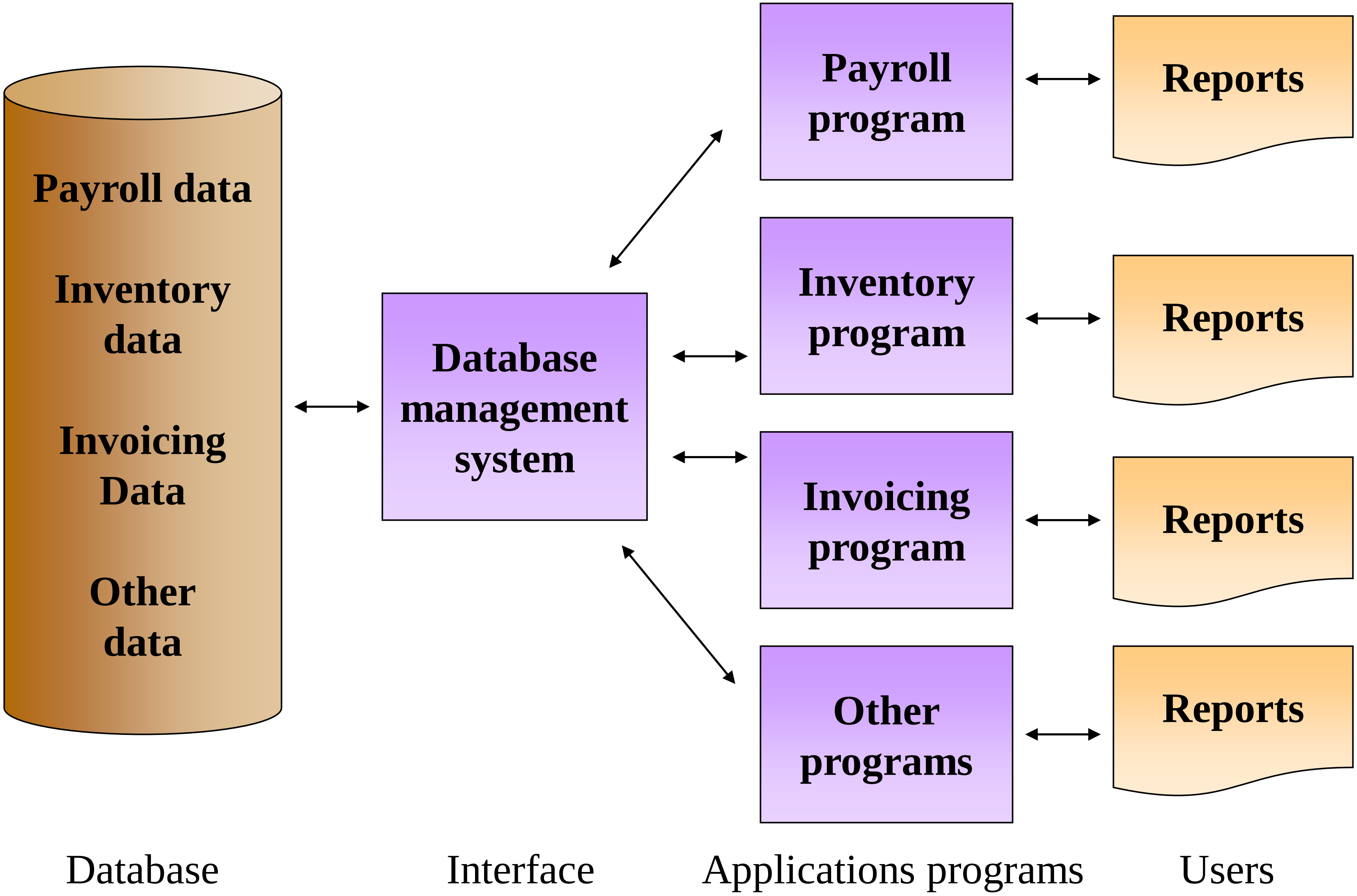
- Data redundancy
  - Duplication of data in separate files
- Lack of data integrity
  - The degree to which the data in any one file is accurate
- Program-data dependence
  - A situation in which program and data organized for one application are incompatible with programs and data organized differently for another application

Database Approach

The database approach…

- A pool of related data is shared by multiple application programs
- Rather than having separate data files, each application uses a collection of data that is either joined or related in the database

Schematic


Payroll data
Inventory data
Invoicing Data
Other data
Database management system
Payroll program
Inventory program
Invoicing program
Other programs
Reports
Reports
Reports
Reports
Database
Interface
Applications programs
Users

Database Management System (DBMS)

DBMS contains information about a particular enterprise

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both *convenient* and *efficient* to use

Database Applications:

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Databases touch all aspects of our lives

Purpose of Database Systems

In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data:

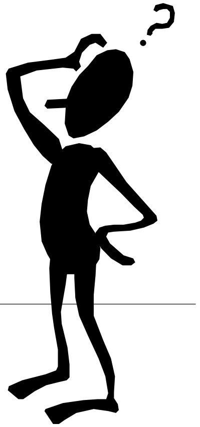
- **Data redundancy and inconsistency**
  - Multiple file formats, duplication of information in different files
- **Difficulty in accessing data**
  - Need to write a new program to carry out each new task
- **Data isolation — multiple files and formats**
- **Integrity problems**
  - Integrity constraints (e.g. account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems (cont.)

- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- **Concurrent access by multiple users**
  - Concurrent accessed needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance and updating it at the same time
- **Security problems**
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Why Use a DBMS?



- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Levels of Abstraction

Physical level: describes how a record (e.g., customer) is stored.

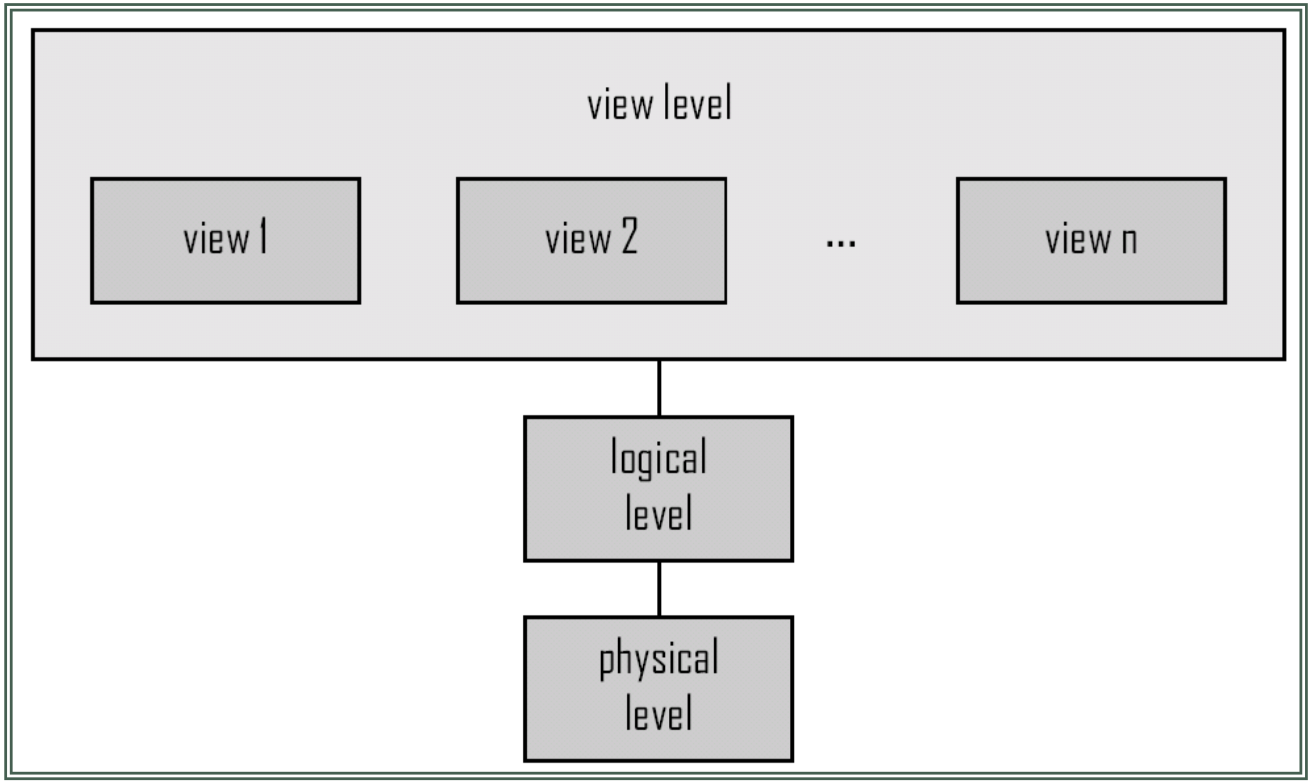
Logical level: describes data stored in database, and the relationships among the data.

```
type customer = record
            customer_id : string;
            customer_name : string;
            customer_street : string;
            customer_city : integer;
   end;
```

View level: application programs hide details of data types. Views can also hide information (such as an employee’s salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

Similar to types and variables in programming languages

Schema – the logical structure of the database

- Example: The database consists of information about a set of customers and accounts and the relationship between them)
- Analogous to type information of a variable in a program
- **Physical schema**: database design at the physical level
- **Logical schema**: database design at the logical level

Instance – the actual content of the database at a particular point in time

- Analogous to the value of a variable

Physical Data Independence – the ability to modify the physical schema without changing the logical schema

- Applications depend on the logical schema
- In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Database Design

The process of designing the general structure of the database:

Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.

- Business decision – What attributes should we record in the database?
- Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

Physical Design – Deciding on the physical layout of the database

Data Models

A collection of tools for describing

- Data
- Data relationships
- Data semantics
- Data constraints

Relational model

Entity-Relationship data model (mainly for database design)

Other models

- *Object-based data models (Object-oriented and Object-relational)*
- *Semi-structured data model (XML)*

Other older models:-

- *Network Model*
- *Hierarchical Model*

Database Languages

- Data Definition Language
- Data Manipulation Language
- Data Control Language

Data Definition Language (DDL)

Specification notation for defining the database schema

Example: **create table** *account* (
account-number **char**(10),
balance **integer**)

DDL compiler generates a set of tables stored in a *data dictionary*

Data dictionary contains metadata (i.e., data about data)

- Database schema
- Data *storage and definition* language
  - Specifies the storage structure and access methods used
- Integrity constraints
  - Domain constraints
  - Referential integrity (**references** constraint in SQL)
- Authorization

Data Manipulation Language (DML)

Language for accessing and manipulating the data organized by the appropriate data model

- DML also known as query language

Two classes of languages

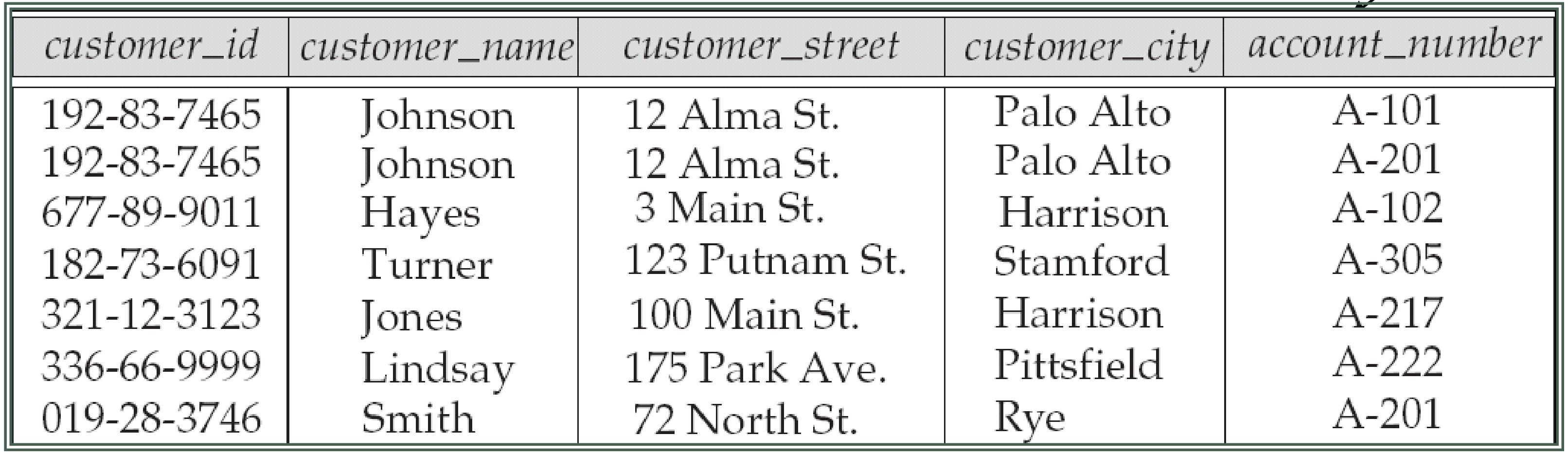
- **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data, expresses the logic of a computation without describing its control flow. It attempts to minimize or eliminate side effects by describing what the program should accomplish, rather than describing how to go about accomplishing it.
- **Procedural** – user specifies what data is required and how to get those data

SQL is the most widely used query language

Relational Model

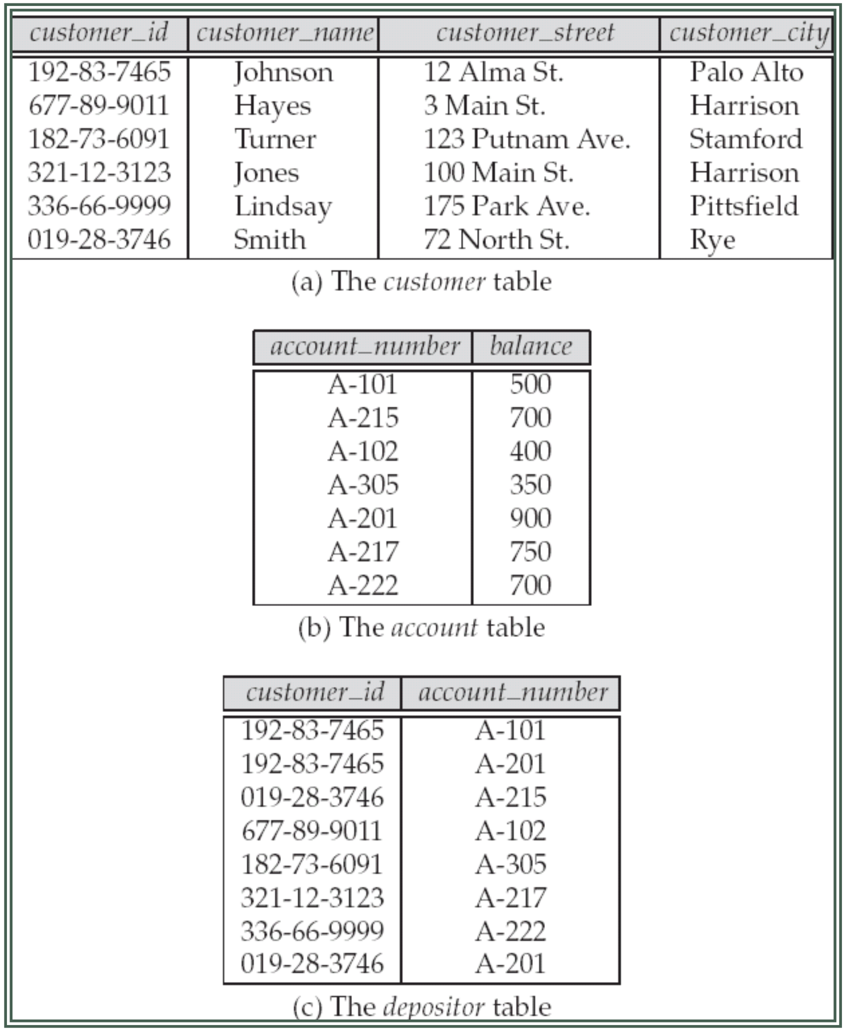
Example of tabular data in the relational model

Attributes



customer_id	*customer_name*	*customer_street*	*customer_city*	*account_number*
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201

A Sample Relational Database



customer_id	*customer_name*	*customer_street*	*customer_city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer_id	*account_number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

SQL

SQL: widely used non-procedural language

- Example: Find the name of the customer with customer-id 192-83-7465

```
select   customer.customer_name
from     customer
where    customer.customer_id = ‘192-83-7465’
```

Application programs generally access databases through one of

- Language extensions to allow embedded SQL
- Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Transaction Management

A **transaction** is a collection of operations that performs a single logical function in a database application

Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Architecture of Database Applications

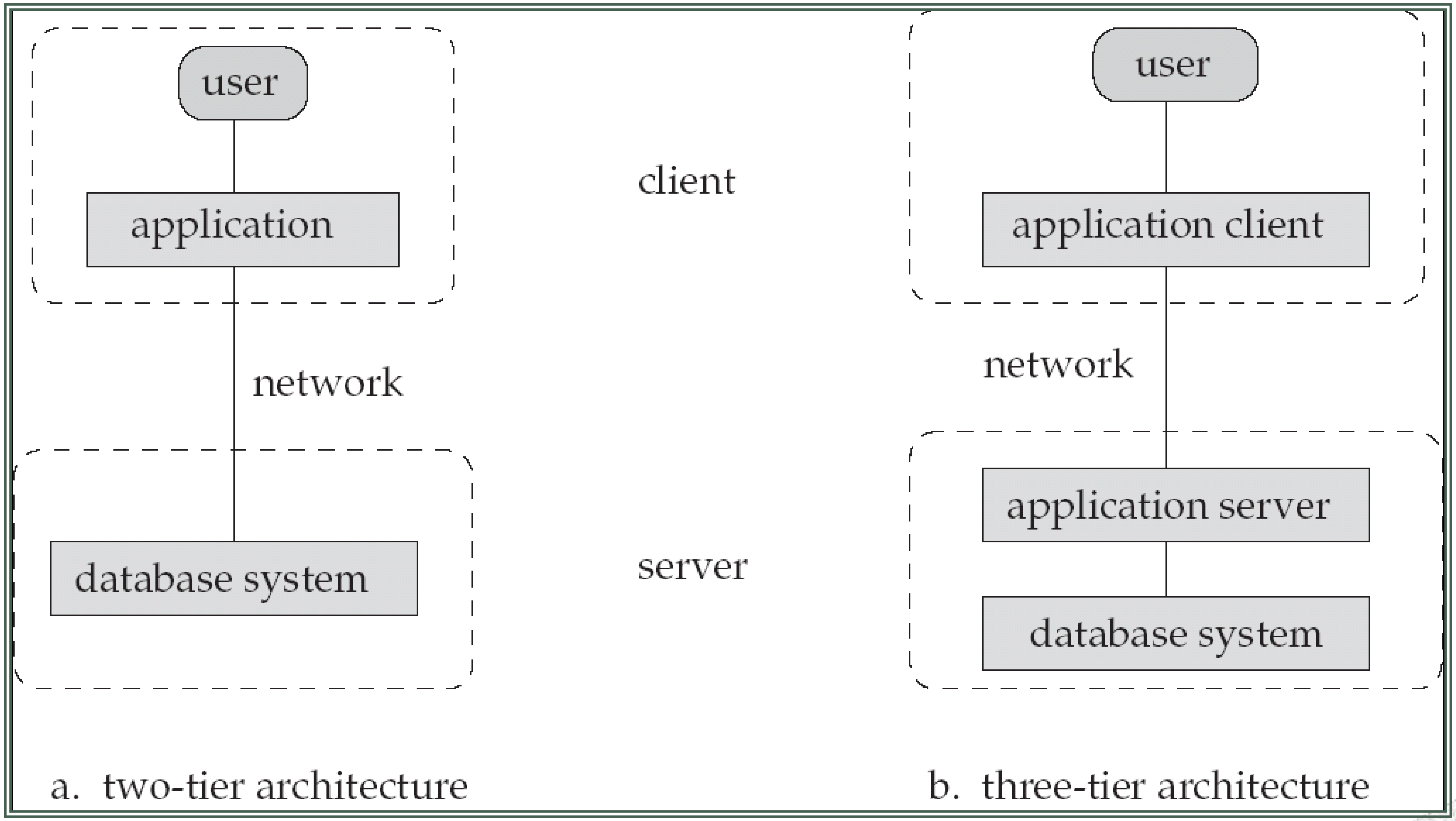
Database applications are usually partitioned into two or three parts

Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine

Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.

- The client end communicates with an application server, usually through a forms interface.
- The application server in turn communicates with a database system to access data.

Architecture



a. two-tier architecture

b. three-tier architecture

Database Users

Users are differentiated by the way they expect to interact with the system

Database Administrators

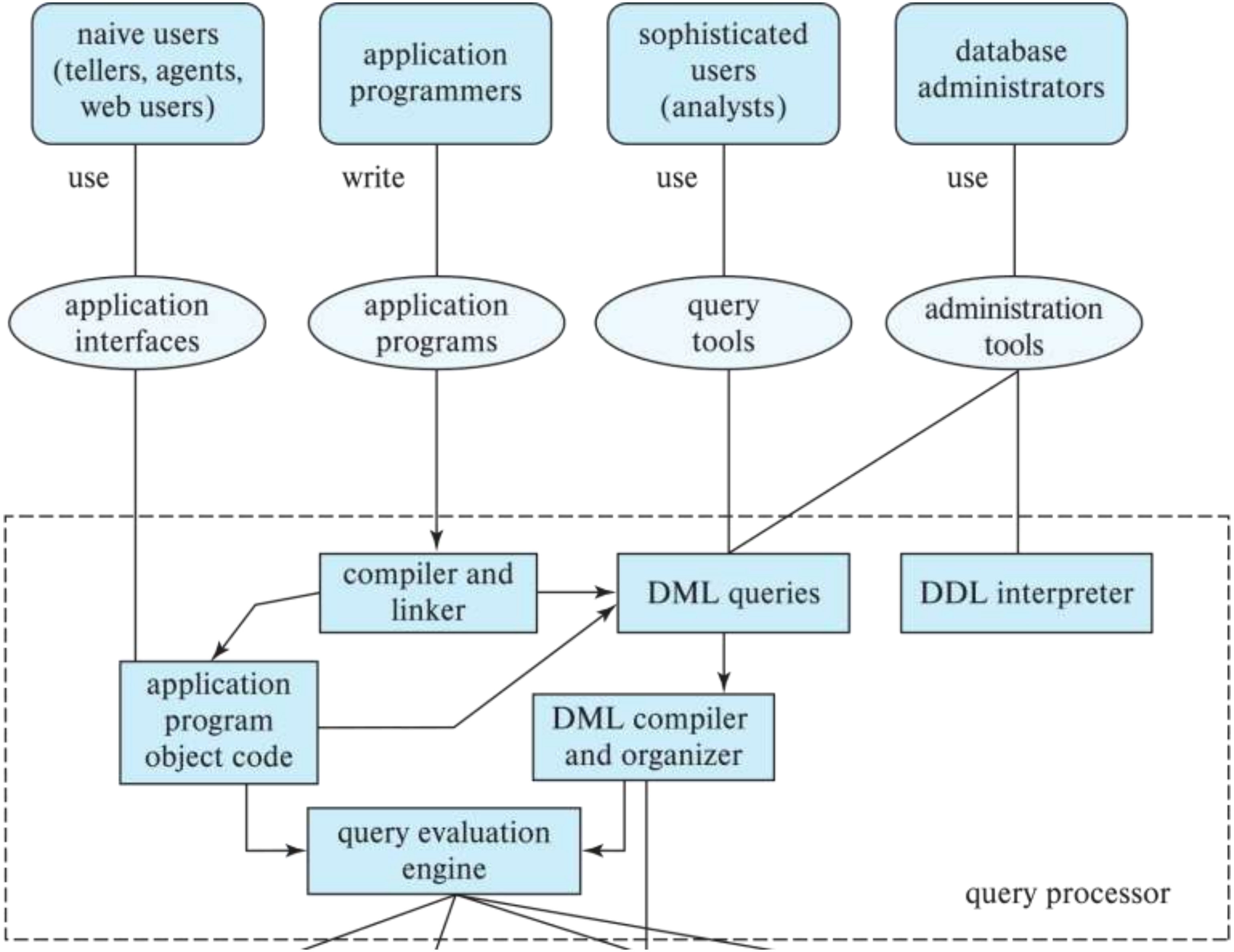
Application programmers – interact with system through DML calls

Sophisticated users – form requests in a database query language

Naïve users – invoke one of the permanent application programs that have been written previously

- Examples, people accessing database over the web, bank tellers, clerical staff

Database Users



Database Administrator

Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise’s information resources and needs.

Database administrator's duties include:

- Schema definition
- Storage structure and access method definition
- Schema and physical organization modification
- Granting user authority to access the database
- Specifying integrity constraints
- Acting as liaison with users
- Monitoring performance and responding to changes in requirements

Database System Structure

Database Engine

A database system is partitioned into modules that deal with each of the responsibilities of the overall system.

The functional components of a database system can be divided into

- **The storage manager,**
- **The query processor component,**
- **The transaction management component.**

Storage Management

Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible to the following ta sks:

- File manager
- Authorization and Integrity manager
- Transaction Manager
- Buffer Manager

Implements several data structures :

- Data files
- Data Dictionary
- Indices

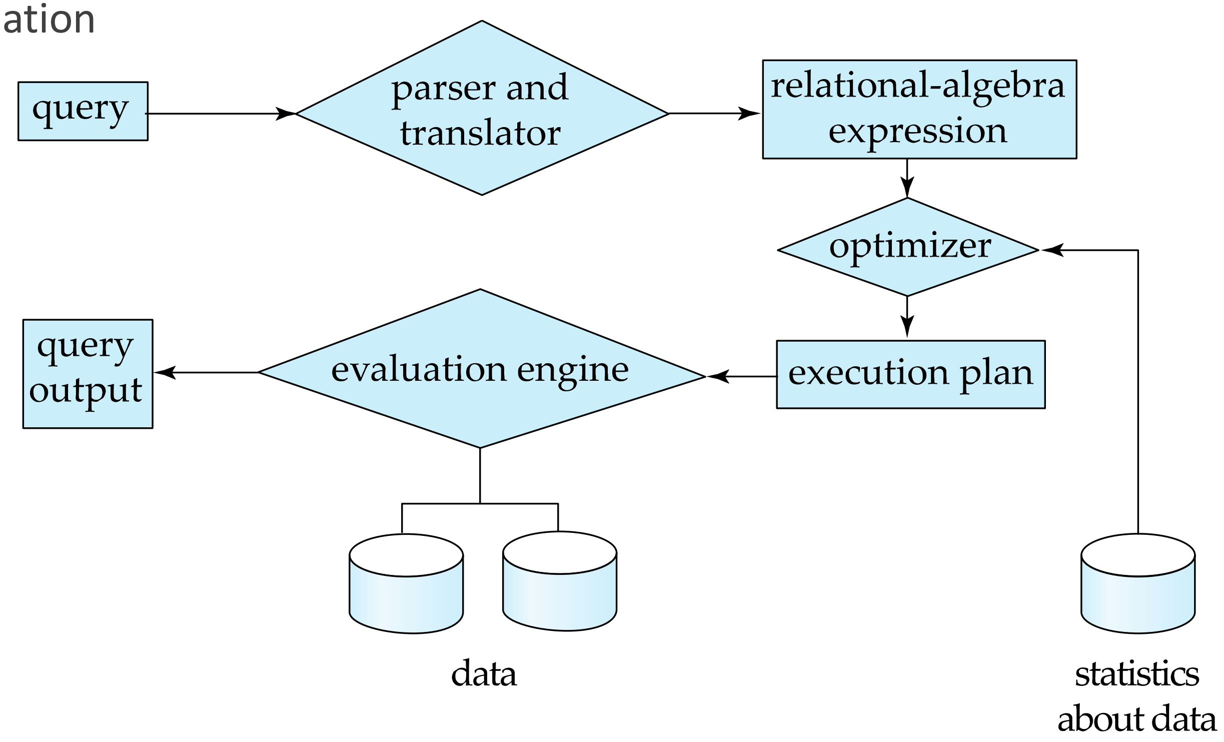
Query Processor

The query processor components include:

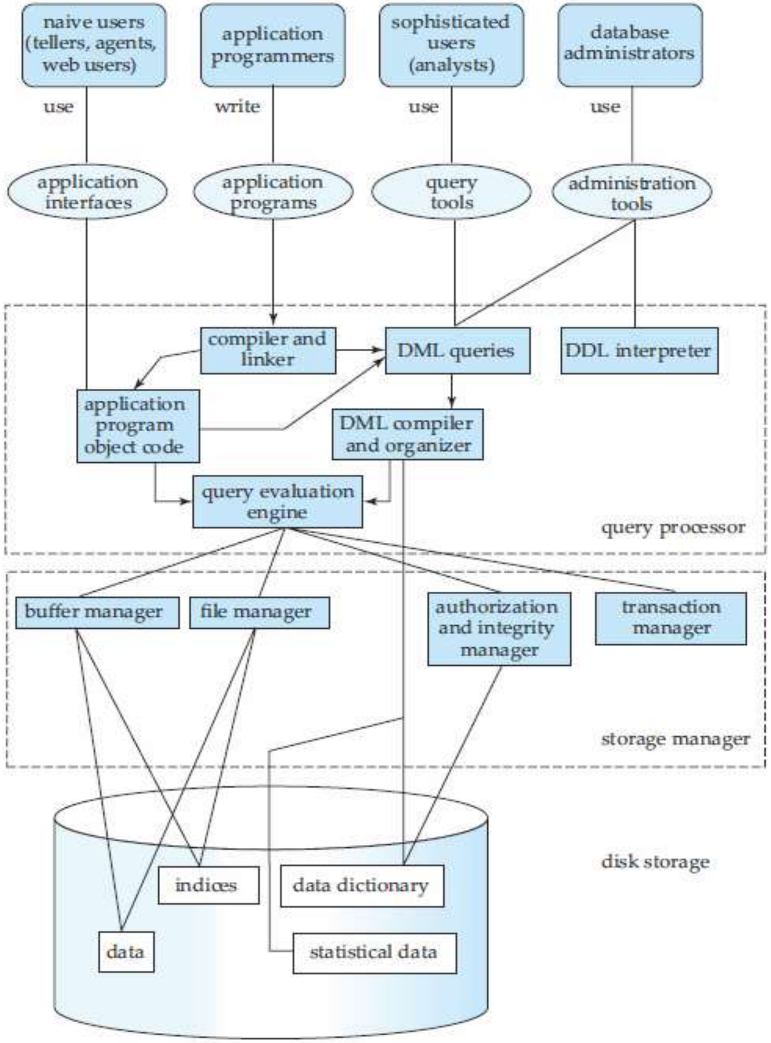
- DDL interpreter -- interprets DDL statements and records the definitions in the data dictionary.
- DML compiler -- translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
    - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
- Query evaluation engine -- executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Overall System Structure



Database Architecture

Centralized databases

- One to a few cores, shared memory

Client-server,

- One server machine executes work on behalf of multiple client machines.

Parallel databases

- Many core shared memory
- Shared disk
- Shared nothing

Distributed databases

- Geographical distribution
- Schema/data heterogeneity