

Lecture 6

Introduction to Database Systems

Subject Lecturer: Kevin K.F. YUEN, PhD.

Acknowledgement: Slides were offered from Prof. Ken Yiu.
Some parts have been revised and indicated.

Outline



- ◆ Applications
- ◆ Problems in data management
- ◆ How to use DBMS?
- ◆ Why data independence is important?

Database and DBMS

- ◆ Database

- ◆ A collection of useful data for an organization
- ◆ E.g., database for a library

- ◆ Query

- ◆ An operation for retrieving data from a database
- ◆ E.g., find all users who have borrowed at least 10 books

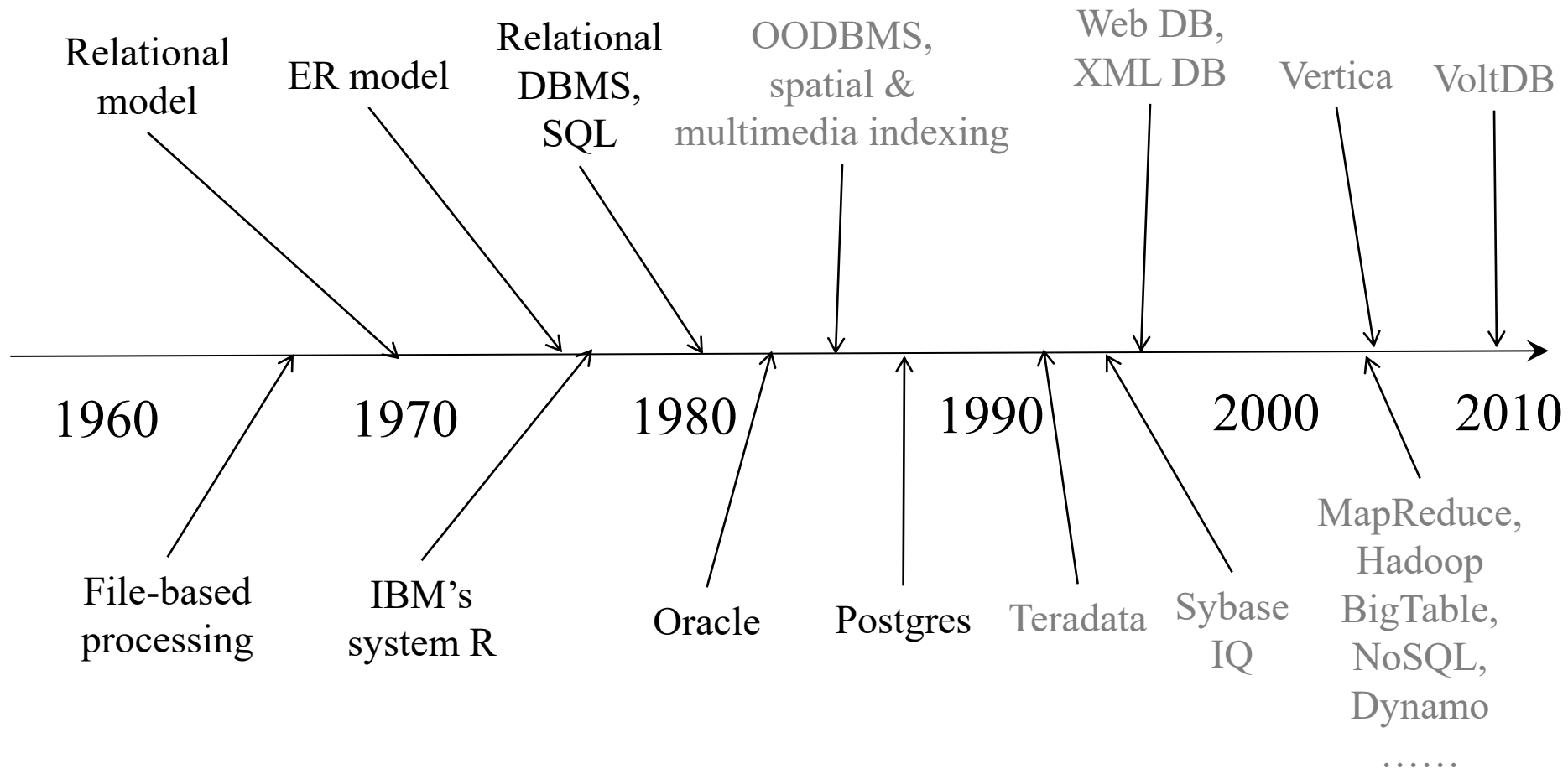
- ◆ Database management system (DBMS)

- ◆ Provide **convenient** way to store and access data
- ◆ **Efficient**

Applications of DBMS: Examples

- ◆ *Examples of applications*
 - ◆ banks, airline and hotel booking, university course management system
- ◆ Sales system
 - ◆ Customers, products, orders
- ◆ Library system
 - ◆ Students, books, loans
- ◆ Bank system
 - ◆ Customers, accounts, deposits, loans

History of Database Technology



Open source DBMS



Sound bite	“The world’s most advanced open source database”	“The world's most popular open source database”
Designed for	Database administrators	Application developers
Initial development priorities	Data integrity, security, standards	Ease-of-use, simple, fast
Start from	A research project/team in UC Berkeley in 1986	Two programmers in Sweden in 1996
Lines of codes (according to www.openhub.net)	1996 July: 103K 2023 Oct: 1,175K https://openhub.net/p/postgres	2000 July: 249K 2023 Oct: 4,010K https://openhub.net/p/mysql
Languages	C: 80%	C++: 73%, C: 16%
Industry users	https://en.wikipedia.org/wiki/PostgreSQL#Notable_users	https://www.mysql.com/customers/industry/

Commercial DBMS

- ◆ Oracle
- ◆ Microsoft SQL Server
- ◆ IBM DB2
- ◆ Examples of price lists
 - ◆ <https://www.oracle.com/assets/technology-price-list-070617.pdf>
 - ◆ <https://www.microsoft.com/en-gb/sql-server/sql-server-2019-pricing>

Outline

- ◆ Applications



- ◆ Problems in data management

- ◆ How to use DBMS?

- ◆ Why data independence is important?

Problems in data management

- ◆ Example scenario: online shopping
 - ◆ Manage data about customers, products, invoices

- ◆ Functions

- ◆ Find a customer, add a new customer
 - ◆ Prepare an invoice
 - ◆ Generate monthly report (for analysis)



- ◆ Before the invention of DBMS, what is the data management solution?
 - ◆ Answer: store data in files (in hard disk)
- ◆ What are the problems of this approach?

Before the invention of DBMS

- ◆ Solution: store data in files (in hard disk)
 - ◆ Supported by the file system / operating system
 - ◆ E.g., store customers in “customers.txt”
- ◆ Problems encountered
 - ◆ Hard to express queries
 - ◆ Hard to modify data format
 - ◆ Data isolation
 - ◆ Inconsistent data
 - ◆ Insecure
 - ◆ Inefficient
 - ◆ Concurrent access error
 - ◆ Error after crashes

```
//name,email,address  
James,james@yahoo.com,HungHom  
Mary,mary@gmail.com,Shatin  
Peter,peter@yahoo.com,MongKok  
.....
```

Problem 1: hard to express queries

- ◆ Examples of queries

- ◆ 1) find a customer called “Peter”
- ◆ 2) display customers in ascending order of address

```
//name,email,address  
James,james@yahoo.com,HungHom  
Mary,mary@gmail.com,Shatin  
Peter,peter@yahoo.com,MongKok  
.....
```

- ◆ Need to write a program for each query ☹

- ◆ DBMS solution

- ◆ Data manipulation language (DML)
 - express a query as a short statement

Lectures 8, 9

Problem 2: hard to modify data format

◆ Examples

- ◆ Add a phone number attribute for customer
- ◆ Extend the length of “email” to 50 characters
- ◆ Store attribute names in the first row (to support queries)

```
name,email,address,phone
James,james@yahoo.com,HungHom,111
Mary,mary@gmail.com,Shatin,222
Peter,peter@yahoo.com,MongKok,333
.....
```

- ◆ Need to write a program for each operation ☹

◆ DBMS solution

- ◆ Data definition language (DDL)
 - modify the table format (by using a short statement)

Lectures 8, 9

Problem 3: data isolation

- ◆ Boss: “Let’s recommend a new snack to snack lovers”
 - ◆ Search “invoices.txt” for customers who have bought snacks
 - ◆ E.g., Peter
 - ◆ Find the addresses of those customers in “customers.txt”
 - ◆ Oops! Two customers have the same name “Peter”!
 - ◆ Which “Peter” shall we refer to?
- ◆ How to uniquely identify each customer?
 - ◆ E.g., add an “identifier” (ID) attribute
- ◆ DBMS solution
 - ◆ Keys, joins



Lectures 7, 10

Problem 4: inconsistent data

◆ Examples

- ◆ Entered a duplicate ID for “Mary”
- ◆ Forgot to enter email for “Peter”

```
id,name,email,address  
1,James,james@yahoo.com,HungHom  
1,Mary,mary@gmail.com,Shatin  
3,Peter,MongKok  
.....
```

◆ DBMS solution

- ◆ Integrity constraints
 - enforce constraints to ensure valid data in the database

Problem 5: insecure



- ◆ Examples
 - ◆ Customer A: find the products he has bought
 - ◆ Customer B: find the popularity of each product
- ◆ The results can be found in “invoices.txt”
- ◆ How to avoid a customer from accessing other customers’ sales records?
- ◆ DBMS solution
 - ◆ Views, access rights



Problem 6: inefficient

- ◆ The number of customers, products, and invoices can be very large
 - ◆ Imagine there are 1,000,000 customers, 1 invoice per customer per day
 - ◆ Slow to modify data in “customers.txt”
 - ◆ Slow to query data in “invoices.txt”
- ◆ How to support efficient queries on a large database?
- ◆ DBMS solution
 - ◆ Index structures, query processing and optimization



Lectures 12, 13

Problem 7: concurrent access error

- ◆ Suppose that two salespersons edit “customers.txt” at the same time
 - ◆ 1) X opens the file (and reads data into memory)
 - ◆ 2) Y opens the file (and reads data into memory)
 - ◆ 3) X inserts a record for a new customer “Joe”
 - ◆ 4) X saves the file
 - ◆ 5) Y inserts a record for a new customer “Eric”
 - ◆ 6) Y saves the file → **overwrite** the version saved by X!
- ◆ The new record for “Joe” is missing!
- ◆ DBMS solution
 - ◆ Concurrency control

Problem 8: error after crashes

- ◆ The shop wishes to raise the price of each product by 10%
 - ◆ The data entry clerk updates the price of each product in “products.txt” one-by-one
 - ◆ Unfortunately, the system crashes in the middle
- ◆ We don't know which records have been updated!
- ◆ DBMS solution
 - ◆ Recovery system

Outline

- ◆ Applications

- ◆ Problems in data management



- ◆ How to use DBMS?

- ◆ Why data independence is important?

Scenario: online shopping



cust-id	name	email	address
1	James	james@yahoo.com	AB
2	Mary	mary@gmail.com	CD
3	Peter	peter@yahoo.com	EF
4	Peter	peter@gmail.com	null
...

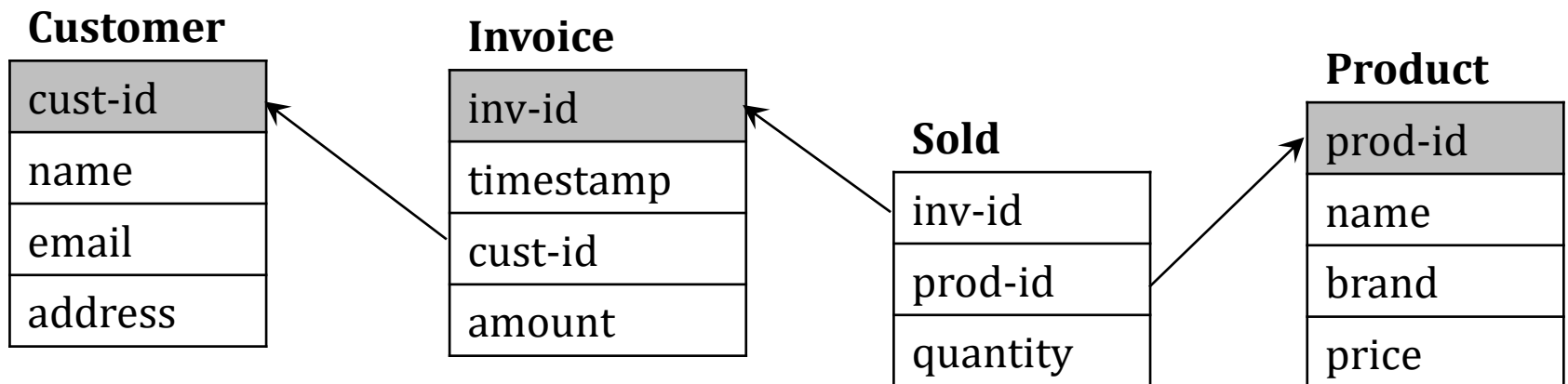
- ◆ (Relational) DBMS is used to store tabular data
- ◆ What kind of tables should we store in this scenario?
- ◆ We can create a “Customer” table to store the attributes of all customers
 - ◆ How to identify these attributes?
- ◆ Have we missed other tables?



Scenario: online shopping

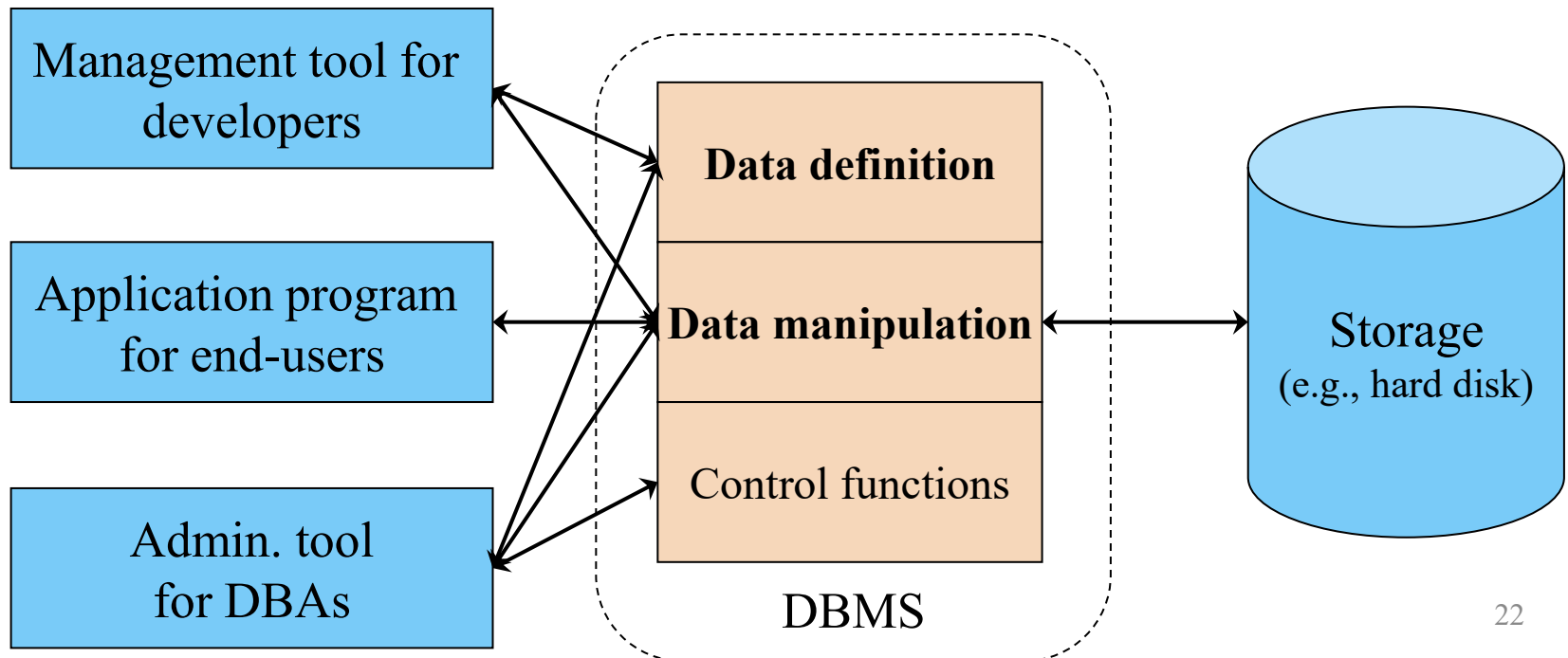


- ◆ Identify the requirements in a given scenario
- ◆ Draw a correct ER diagram accordingly, then convert it to relational schemas
 - ◆ *We'll study in Lecture 10*
- ◆ Apply database normalization techniques to reduce redundant information
 - ◆ *We'll study in Lecture 11*



How to use DBMS?

- ◇ *Data Definition Language (DDL)*:
 - ◆ define the database schema, i.e., the record type
- ◇ *Data Manipulation Language (DML)*:
 - ◆ access / update the database content
- ◇ SQL consists of both DDL and DML



Database Languages

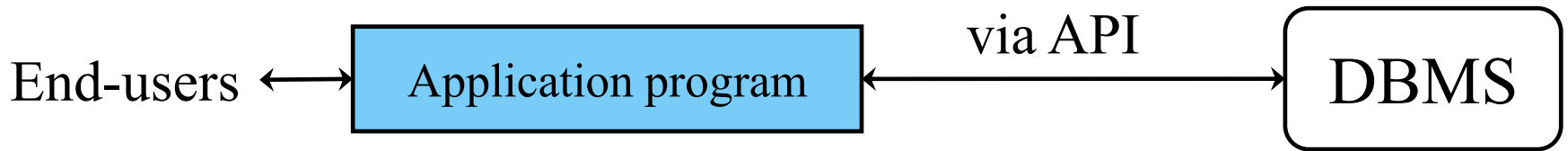
- ◆ Relational algebra
 - ◆ **Procedural** language: tell DBMS **how to do**
 - ◆ Used within DBMS (for query processing & optimization)
 - ◆ Form the foundation of SQL
 - ◆ *We'll study in Lecture 7*

$\sigma_{\text{price} > 8.0}(\textit{Product})$

- ◆ SQL (structured query language)
 - ◆ **Declarative** language: tell DBMS **what to do**
 - ◆ Used by human (e.g., developers, administrators)
 - ◆ *We'll study in Lectures 8, 9*

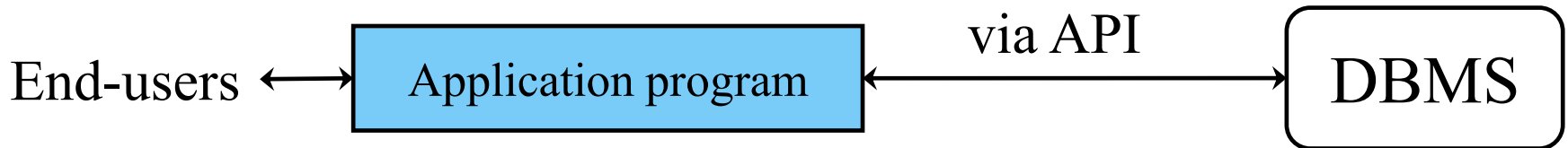
```
select *  
from Product  
where price > 8.0
```

How to use SQL in a program?



- ◆ An application program acts as the intermediary between users and the database
 - ◆ End-users *don't* use a query language like SQL
 - ◆ It **communicates** with a database server via **API** (application-program interface)
- ◆ An application program makes API calls to
 - ◆ **Connect** with the database server
 - ◆ **Send SQL commands** to the database server
 - ◆ **Fetch tuples** of result one-by-one into program variables

Examples of API



- ◆ JDBC (Java Database Connectivity) for Java applications
 - ◆ <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>
- ◆ ODBC (Open Database Connectivity) for applications written in C, C++, C#, and Visual Basic
 - ◆ <https://learn.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference?view=sql-server-ver16>
- ◆ Web interface, server-side scripting (for PHP)
 - ◆ <https://www.php.net/manual/en/mysqlinfo.api.choosing.php>
- ◆ Web interface, client-side scripting (for Javascript)
 - ◆ https://www.w3schools.com/nodejs/nodejs_mysql.asp

Application Architecture

- ◆ A large application: complex to develop and maintain
→ Use a multi-layered architecture

- ◆ Presentation layer

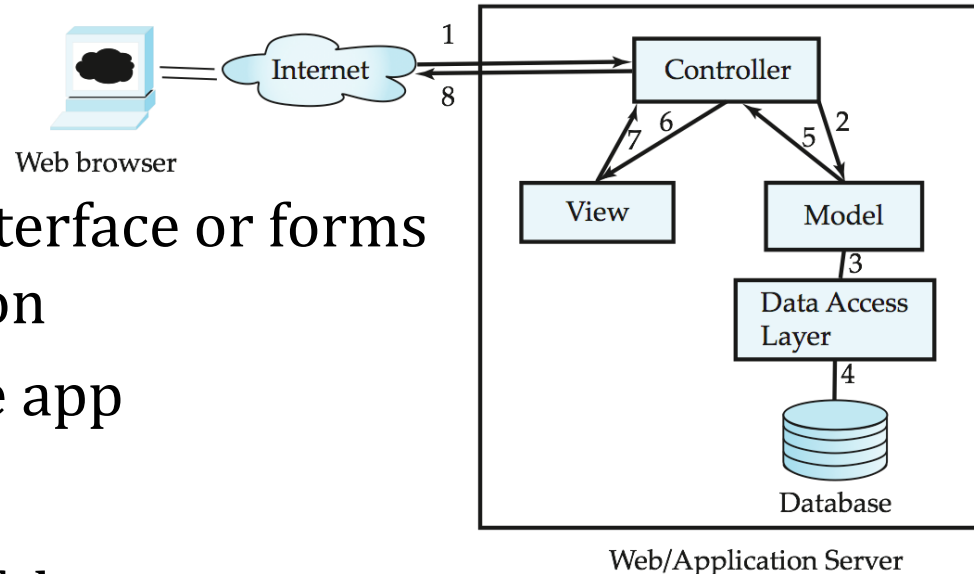
- ◆ Provides graphical-user interface or forms to deal with user interaction
- ◆ Runs as web app or mobile app

- ◆ Business-logic layer

- ◆ Provides high level view of data and actions on data

- ◆ Data access layer

- ◆ Interfaces between business logic layer and the underlying database



Outline

- ◆ Applications
- ◆ Problems in data management
- ◆ How to use DBMS?
- ◆ Why data independence is important?



Data independence

- ◆ Application developers like the following two rules
→ avoid rewriting an application program

- ◆ (1) Logical data independence

- ◆ The same application program can still be used even if the table format changes

- ◆ E.g.,

table T

id	A	B
1	c	e
2	d	f

vs.

table T1,

id	A
1	c
2	d

table T2

id	B
1	e
2	f

- ◆ (2) Physical data independence

- ◆ The same application program can still be used even if the organization of data in the storage changes

- ◆ E.g., store data in a sequential file vs. store data in a tree structure


Data independence

- ◆ DBMS supports data independence by using 3 levels
- ◆ View level
 - ◆ Part of a database that can viewed by a user
 - ◆ E.g., a customer can access his sales records only
 - ◆ Different views are provided to different types of users
- ◆ Logical level
 - ◆ Which attribute is stored in which table?
- ◆ Physical level
 - ◆ How data are stored in hard disk?

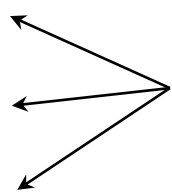
Introduction to relational model

- ◆ The relational model is used at the logical level
- ◆ Schema decides the format of a table
 - ◆ E.g., (id, name, email, address, phone)
- ◆ Example: table *customer*

attributes
(or columns)



id	name	email	address	phone
1	James	james@yahoo.com	HungHom	111
2	Mary	mary@gmail.com	Shatin	222
3	Peter	peter@yahoo.com	Mongkok	333



tuples
(or rows)

Summary

- ◆ Problems in data management
- ◆ Some concepts in DBMS
- ◆ Please read Chapter 1 in the book
“*Database System Concepts*”, 7th Edition
- ◆ In the next lecture, we will study:
 - ◆ Relational model
 - ◆ How to express queries in “relational algebra”?

$\sigma_{\text{price} > 8.0}(\textit{Product})$