

CTEC1731  
Enterprise  
Computing  
III  
Fall 2009



Introduction to Database Systems

# DBMS (1)

- A *database management system* (DBMS) is a collection of interrelated data and a set of programs to access that data.
- The collection of data is called a *database*.

# Goals of a database system

- The primary goal of a DBMS is to provide an environment that is both *convenient* and *efficient* to use in retrieving information from and storing data into the database.

# Data Management (1)

- The management of data involves the following:
  - structures for storing data
  - ways to manipulate the data
  - protection for stored data

# DBMS (2)

- A database management system is a collection of interrelated files and a set of programs that allow several users to access and modify these files.

# Data Management (2)

- A major purpose of a database system is to provide users with an *abstract* view of the data.
- The system hides certain details of how the data is stored and maintained.

# Data Abstraction (1)

- **Physical level.** This is the lowest level of abstraction, where you describe *how* the data are actually stored.
- At this level, complex, low-level data structures are described in detail.

# Data Abstraction (2)

- **Conceptual level.** This is the next higher level of abstraction, where you describe *what* data is actually stored in the database, and the relationships that exist among the data.

# Data Abstraction (2)

- **Conceptual level** (*continued*)
- This level is used by *database administrators (DBAs)*, who must decide what information is to be kept in the database.

# Data Abstraction (3)

- **View level.** This is the highest level of abstraction where you describe only part of the entire database.
- Most end-users only need a part of the database, and there may be many views for the same database.

# RDBMS

- The *relational* database model is one of the most widely-accepted data models.
- The data and the relationships among data are represented by a collection of tables, each of which has a number of columns with unique names.

# Relational Database (1)

- To illustrate this, consider a database consisting of customers and the accounts that they have. A sample relational database follows.

# Relational Database (2)

name	street	city	number
Flair	Maple	Oakville	900
Austin	North	Mississauga	556
Austin	North	Mississauga	647
Hogan	Bloor	Toronto	801
Hogan	Bloor	Toronto	647

number	balance
900	55.33
556	100000.00
647	105366.37
801	10533.22

## Relational Database (3)

- It shows, for example, that customer Hogan lives on Bloor in Toronto, that he has two accounts, one numbered 647 with a balance of \$105,366.37, and the other 801 with a balance of \$10,533.22. Note that customers Austin and Hogan share account number 647 (they may share a business venture).

# Relational Database (4)

- A *relationship* is an association among *entities*.
- An *entity* is represented by a set of *attributes*.

# Relational Database (5)

- In the above example, we are dealing with two entities, *customer* and *account*.
- Possible attributes of the *customer* entity are *name*, *street*, and *city*.
- Possible attributes of the *account* entity are *number* and *balance*.

# Relational Database (6)

- For example, we may define a relationship which associates customer “Austin” with account 556. This specifies that Austin is a customer with bank account number 556.
- The relationship is implemented as a *join*, using one or more attributes that are common to the joined entities.

# Relational Database (7)

name	street	city	number
Flair	Maple	Oakville	900
Austin	North	Mississauga	556
Austin	North	Mississauga	647
Hogan	Bloor	Toronto	801
Hogan	Bloor	Toronto	647

number	balance
900	55.33
556	100000.00
647	105366.37
801	10533.22

# Relational Database (8)

- A *relationship set* is a set of relationships with the same type.
- To illustrate this, consider the two entity sets *customer* and *account*. We define the relationship set *CustAcct* to denote the association between customers and the bank accounts that they have.

# SQL (1)

- SQL, pronounced “sequel”, stands for *Structured Query Language*.
- It is used for *data definition*, *data manipulation*, and *data extraction*.
- RDBMS software has a layer (i.e., an *interface*) which interprets SQL instructions (queries).

# SQL (2)

- In the example database, we can define the *customer* entity and attributes using SQL:

```
create table customer
(
  name varchar(30) not null,
  street varchar(40) not null,
  city varchar(20) not null,
  number integer not null
)
```

# SQL (3)

- Similarly, the *account* entity and attributes:

```
create table account
```

```
(
```

```
  number integer not null primary key,
```

```
  balance decimal(12,2) not null
```

```
)
```

# SQL (4)

- To populate a row in the customer table:

```
insert into customer (  
  name, street, city, number  
) values (  
  "Austin", "North", "Mississauga", 556  
)
```

# SQL (5)

- To display all customers, and their bank account information:

```
select name, street, city,  
       account.number, balance  
from customer, account  
where  
       customer.number = account.number
```

# SQL (7)

- The *CustAcct* relationship set can be implemented as a *view*:

```
create view CustAcct as
  select name, street, city,
         number as account.number, balance
  from customer, account
 where
   customer.number = account.number
 group by name, street, city
 order by number
```

# SQL (8)

- Selecting the view runs the underlying **select** statement, as if the user typed it in.

```
select * from CustAcct
```

# “ACID”

- Atomicity
- Consistency
- Isolation
- Durability

## “ACID” (2)

- If you need a relational database, you want these. If you want flat files with SQL syntax, you don't. You can read the details here.

<http://en.wikipedia.org/wiki/ACID>