# DATABASE MANAGEMENT SYSTEM

1

# Basic Definitions

* **Database:**
	+ A logical coherent collection of data representing the mini-world such that change in the mini-world brings about change in database collected for a particular purpose and for a group of intended users.
* **Data:**
	+ Meaningful facts, text, graphics, images, sound, video segments that can be recorded and have an implicit meaning.
* **Metadata:**
	+ Data that describes data
* **File Processing System**
	+ A collection of application programs that perform services for the end-users such as production of reports
	+ Each program defines and manages its own data
* **Database Management System (DBMS):**
	+ A software package/ system to facilitate the creation and

maintenance of a computerized database.

* **Database System:**
	+ The DBMS software together with the data itself. Sometimes, the applications are also included. Database + DBMS 2

# fig01_01Simplified database system environment

3

# Evolution of DB Systems

* Flat files - 1960s - 1980s
* Hierarchical – 1970s - 1990s
* Network – 1970s - 1990s
* Relational – 1980s - present
* Object-oriented – 1990s - present
* Object-relational – 1990s - present
* Data warehousing – 1980s - present
* Web-enabled – 1990s - present

4

## Purpose of Database Systems

Database management systems were developed to handle the difficulties of typical file-processing systems supported by conventional operating systems

5

# Disadvantages of File Processing

* **Program-Data Dependence**
	+ **File structure is defined in the program code.**
	+ **All programs maintain metadata for each file they use**
* **Duplication of Data (Data Redundancy)**
	+ **Different systems/programs have separate copies of the same data**
* **Same data is held by different programs.**
* **Wasted space and potentially different values and/or different formats for the same item.**
* **Limited Data Sharing**
	+ **No centralized control of data**
	+ **Programs are written in different languages, and so cannot easily access**

**each other’s files.**

* **Lengthy Development Times**
	+ **Programmers must design their own file formats**
* **Excessive Program Maintenance**
	+ **80% of of information systems budget**
* **Vulnerable to Inconsistency**
	+ **Change in one table need changes in corresponding tables as well otherwise data will be inconsistent** 6

# Advantages of Database Approach

* Data independence and efficient access.
* Data integrity and security.
* Uniform data administration.
* Concurrent access, recovery from crashes.
* Replication control
* Reduced application development time.
* Improved Data Sharing
	+ Different users get different views of the data
* Enforcement of Standards
	+ All data access is done in the same way
* Improved Data Quality
	+ Constraints, data validation rules
* Better Data Accessibility/ Responsiveness
	+ Use of standard data query language (SQL)
* Security, Backup/Recovery, Concurrency
	+ Disaster recovery is easier

7

# Costs and Risks of the Database Approach

* Up-front costs:
	+ Installation Management Cost and Complexity
	+ Conversion Costs
* Ongoing Costs
	+ Requires New, Specialized Personnel
	+ Need for Explicit Backup and Recovery
* Organizational Conflict
	+ Old habits die hard

8

# Database Applications

* Database Applications:
	+ Banking: all transactions
	+ Airlines: reservations, schedules
	+ Universities: registration, grades
	+ Sales: customers, products, purchases
	+ Manufacturing: production, inventory, orders, supply chain
	+ Human resources: employee records, salaries, tax deductions
* Databases touch all aspects of our lives

9

# Levels of Abstraction

* Many *views*, single *conceptual (logical) schema* and *physical schema*.

Conceptual Schema

View 3

View 2

View 1

* + Views describe how users see the data.

Physical Schema

* + Conceptual schema defines

logical structure

* + Physical schema describes the files and indexes used.

\* *Schemas are defined using DDL; data is modified/queried using DML*.

10

# Example: University Database

* Conceptual schema:
	+ *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
	+ *Courses(cid: string, cname:string, credits:integer)*
	+ *Enrolled(sid:string, cid:string, grade:string)*
* Physical schema:
	+ Relations stored as unordered files.
	+ Index on first column of Students.
* External Schema (View):
	+ *Course\_info(cid:string, enrollment:integer)*

11

# Instances and Schemas

* Similar to types and variables in programming languages
* Schema – the logical structure of the database (e.g., set of customers and accounts and the relationship between them)
* Instance – the actual content of the database at a particular point in time

12

# Data Independence

* Ability to modify a schema definition in one level without affecting a schema definition in the other levels.
* The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
* Two levels of data independence
	+ Physical data independence:- Protection from changes in

*logical* structure of data.

* + Logical data independence:- Protection from changes in

physical structure of data.

13

# Instances and Schemas

* Similar to types and variables in programming languages
* **Schema** – the logical structure of the database
	+ e.g., 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.

14

# Database Languages

Data Definition Language (DDL)

* Specification notation for defining the database schema
* DDL compiler generates a set of tables stored in a data dictionary
* Data dictionary contains ***metadata*** (data about data)
* Data storage and definition language – special type of DDL in which the storage structure and access methods used by the database system are specified

Data Manipulation Language (DML)

* Language for accessing and manipulating the data organized by the

appropriate data model

* Two classes of languages
	+ Procedural – user specifies what data is required and how to get those data
	+ Nonprocedural – user specifies what data is required without

specifying how to get those data

15

# Database Users

* Users are differentiated by the way they expect to interact with the system
* Application programmers – interact with system

through DML calls

* Sophisticated users – form requests in a database query language
* Specialized users – write specialized database applications that do not fit into the traditional data processing framework
* Naïve users – invoke one of the permanent application programs that have been written previously
	+ E.g. people accessing database over the web, bank tellers, clerical staff

16

# 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

17

# Data Models

* A collection of tools for describing:
	+ Data
	+ Data relationships
	+ Data semantics
	+ Data constraints
* Object-based logical models
	+ Entity-relationship model
	+ Object-oriented model
	+ Semantic model
	+ Functional model
* Record-based logical models
	+ Relational model (e.g., SQL/DS, DB2)
	+ Network model
	+ Hierarchical model (e.g., IMS)

18

# Entity-Relationship Model

* The basics of Entity-Relationship modelling

u Entities (objects)

– E.g. customers, accounts, bank branch

u Attributes

u Relationships between entities

* E.g. Account A-101 is held by customer Johnson
* Relationship set *depositor* associates customers with accounts
* Widely used for database design
	+ Database design in E-R model usually converted to design in the relational model which is used for storage and processing

19

# ER Model Basics

**lot**

**Employees**

**name**

**ssn**

* *Entity:* Real-world object distinguishable from other objects. An entity is described using a set of *attributes*. Each attribute has a *domain*.
* *Entity Set*: A collection of similar entities. E.g., all employees.
	+ All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
	+ Each entity set has a *key*.

*Weak Entities:* A *weak entity* can be identified uniquely only

by considering the primary key of another (*owner*) entity.

20

# ER Model Basics

**ssn**

**name**

**lot**

**Employees**

**Departments**

**name**

**Works\_In**

**budget**

**did**

**dname**

**since**

**Employees**

**lot**

**ssn**

**super-**

**visor**

**subor-**

**dinate Reports\_To**

* *Relationship*: Association among two or more entities. E.g.,

Attishoo works in Pharmacy department.

* *Relationship Set*: Collection of similar relationships.
	+ An n-ary relationship set R relates n entity sets E1 ... En; each

relationship in R involves entities e1 E1, ..., en En

* + Same entity set could participate in different relationship

sets, or in different “roles” in same set.

21

# E-R Diagrams

n Rectangles represent entity sets.

n Diamonds represent relationship sets.

n Lines link attributes to entity sets and entity sets to relationship sets.

n Ellipses represent attributes

l Double ellipses represent multivalued attributes.

l Dashed ellipses denote derived attributes.

n Underline indicates primary key attributes (will study later)

22

# Mapping Cardinality Constraints

* Express the number of entities to which another entity can be associated via a relationship set.
* Most useful in describing binary relationship sets.
* For a binary relationship set the mapping cardinality must be one of the following types:
	+ One to one
	+ One to many
	+ Many to one
	+ Many to many

23

# Mapping Cardinalities

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

24

# Participation Constraints

* Does every department have a manager?

– If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).

* Every Department entity must appear in an instance of the relationship Works\_In (have an employee) and every Employee must be in a Department
* Both Employees and Departments participate totally in Works\_In

**name**

**since**

**dname**

**ssn lot did budget**

**Manages**

**Works\_In**

**since**

**Employees**

**Departments**

25

# Keys

* + A **super key** of an entity set is a set of one or more attributes

whose values uniquely determine each entity.

* + A **candidate key** of an entity set is a minimal super key
		- *Customer\_id* is candidate key of *customer*
		- *account\_number* is candidate key of *account*
	+ Although several candidate keys may exist, one of the candidate

keys is selected to be the **primary key**.

* + Alternate key **is t**he candidate key which are not selected as primary key.
	+ Foreign key are the attributes of an entity that points to the primary key of another entity. They act as a cross-reference between entities.
	+ Composite Key consists of two or more attributes that uniquely identify an entity.

Non-key attributes are the attributes or fields of a table, other than candidate key attributes/fields in a table.

* + Non-prime Attributes are attributes other than Primary Key

attribute(s)..

26

# Relational Model

Example of tabular data in the relational model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **ssn** | **street** | **city** | **account-number** |
| Johnson | 192-83-7465 | Alma | Palo Alto | A-101 |
| Smith | 019-28-3746 | North | Rye | A-215 |
| Johnson | 192-83-7465 | Alma | Palo Alto | A-201 |
| Jones | 321-12-3123 | Main | Harrison | A-217 |
| Smith | 019-28-3746 | North | Rye | A-201 |

|  |  |
| --- | --- |
| **account-number** | **balance** |
| A-101 | 500 |
| A-201 | 900 |
| A-215 | 700 |
| A-217 | 750 |

27

# Relational Model (Basic)

The **relational model** used the basic **concept** of a relation or table.

Tuple:- A tuple is a row in a table.

Attribute:- An attribute is the named column of a relation.

Domain:- A domain is the set of allowable values for one or more attributes.

Degree:- The number of columns in a table is called the degree of

relation.

Cardinality:- The number of rows in a relation,is called the cardinality of the relation.

28

# Integrity Constraints

Integrity constraints guard against accidental damage to the database, by ensuring that authorized changes

to the database do not result in a loss of data consistency.

* Domain Constraints:- It specifies that the value of each attribute x must be an atomic value from the domain of x.
* Key Constraints:- Primary Key must have unique value in the relational table.
* Referential Integrity:-It states that if a foreign key in table A refers to the primary key of table B then, every value of the foreign key in table A must be null or be available in table B.
* Entity Integrity:- It states that no attribute of a primary key can have a null value.

29

# A Sample Relational Database

******

30

## SQL Introduction

Standard language for querying and manipulating data

**S**tructured **Q**uery **L**anguage Many standards out there:

* ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), ….
* Vendors support various subsets: watch for fun discussions in class !

31

## SQL

* + Data Definition Language (DDL)
		- Create/alter/delete tables and their attributes
		- Following lectures...
	+ Data Manipulation Language (DML)
		- Query one or more tables – discussed next !
		- Insert/delete/modify tuples in tables

32

Table name

## Tables in SQL

Attribute names

Product

|  |  |  |  |
| --- | --- | --- | --- |
| PName | Price | Category | Manufacturer |
| Gizmo | $19.99 | Gadgets | GizmoWorks |
| Powergizmo | $29.99 | Gadgets | GizmoWorks |
| SingleTouch | $149.99 | Photography | Canon |
| MultiTouch | $203.99 | Household | Hitachi |

Tuples or rows

33

## Tables Explained

* The *schema* of a table is the table name and its attributes:

Product(PName, Price, Category, Manfacturer)

* A *key* is an attribute whose values are unique; we underline a key

Product(PName, Price, Category, Manfacturer)

34

## Data Types in SQL

* + Atomic types:
		- Characters: CHAR(20), VARCHAR(50)
		- Numbers: INT, BIGINT, SMALLINT, FLOAT
		- Others: MONEY, DATETIME, …
	+ Every attribute must have an atomic type
		- Hence tables are flat
		- Why ?

35

## Tables Explained

* + A tuple = a record
		- Restriction: all attributes are of atomic type
	+ A table = a set of tuples
		- Like a list…
		- …but it is unorderd:

no **first()**, no **next()**, no **last()**.

36

## SQL Query

Basic form: (plus many many more bells and whistles)

SELECT <attributes>

FROM <one or more relations>

WHERE <conditions>

37

## Simple SQL Query

Product

|  |  |  |  |
| --- | --- | --- | --- |
| PName | Price | Category | Manufacturer |
| Gizmo | $19.99 | Gadgets | GizmoWorks |
| Powergizmo | $29.99 | Gadgets | GizmoWorks |
| SingleTouch | $149.99 | Photography | Canon |
| MultiTouch | $203.99 | Household | Hitachi |

SELECT \*

FROM Product

WHERE category=‘Gadgets’

“selection”

|  |  |  |  |
| --- | --- | --- | --- |
| PName | Price | Category | Manufacturer |
| Gizmo | $19.99 | Gadgets | GizmoWorks |
| Powergizmo | $29.99 | Gadgets | GizmoWorks |

38

## Simple SQL Query

Product

|  |  |  |  |
| --- | --- | --- | --- |
| PName | Price | Category | Manufacturer |
| Gizmo | $19.99 | Gadgets | GizmoWorks |
| Powergizmo | $29.99 | Gadgets | GizmoWorks |
| SingleTouch | $149.99 | Photography | Canon |
| MultiTouch | $203.99 | Household | Hitachi |

SELECT PName, Price, Manufacturer FROM Product

WHERE Price > 100

“selection” and

“projection”

|  |  |  |
| --- | --- | --- |
| PName | Price | Manufacturer |
| SingleTouch | $149.99 | Canon |
| MultiTouch | $203.99 | Hitachi |

39

## Notation

Input Schema

Product(PName, Price, Category, Manfacturer)

SELECT PName, Price, Manufacturer FROM Product

WHERE Price > 100

Answer(PName, Price, Manfacturer)

40

Output Schema

## Keys and Foreign Keys

Key

Product

Company

|  |  |  |
| --- | --- | --- |
| CName | StockPrice | Country |
| GizmoWorks | 25 | USA |
| Canon | 65 | Japan |
| Hitachi | 15 | Japan |

Foreign key

41

|  |  |  |  |
| --- | --- | --- | --- |
| PName | Price | Category | Manufacturer |
| Gizmo | $19.99 | Gadgets | GizmoWorks |
| Powergizmo | $29.99 | Gadgets | GizmoWorks |
| SingleTouch | $149.99 | Photography | Canon |
| MultiTouch | $203.99 | Household | Hitachi |

## Joins

Product (pname, price, category, manufacturer)

Company (cname, stockPrice, country)

Find all products under $200 manufactured in Japan; return their names and prices.

SELECT PName, Price FROM Product, Company

Join

between Product and Company

WHERE Manufacturer=CName AND Country=‘Japan’

AND Price <= 200

42

## Joins

Product Company

Cname GizmoWorks Canon

Hitachi

StockPrice 25

65

15

Country USA

Japan

Japan

|  |  |  |  |
| --- | --- | --- | --- |
| PName | Price | Category | Manufacturer |
| Gizmo | $19.99 | Gadgets | GizmoWorks |
| Powergizmo | $29.99 | Gadgets | GizmoWorks |
| SingleTouch | $149.99 | Photography | Canon |
| MultiTouch | $203.99 | Household | Hitachi |

|  |  |
| --- | --- |
| PName | Price |
| SingleTouch | $149.99 |

43

SELECT PName, Price FROM Product, Company

WHERE Manufacturer=CName AND Country=‘Japan’

AND Price <= 200

## More Joins

Product (pname, price, category, manufacturer)

Company (cname, stockPrice, country)

Find all Chinese companies that manufacture products both in the ‘electronic’ and ‘toy’ categories

SELECT cname

FROM

WHERE

44

## NULLS in SQL

* Whenever we don’t have a value, we can put a NULL
* Can mean many things:
	+ Value does not exists
	+ Value exists but is unknown
	+ Value not applicable
	+ Etc.
* The schema specifies for each attribute if can be null (*nullable* attribute) or not
* How does SQL cope with tables that have NULLs ?

45

## Outer Joins

* Left outer join:
	+ Include the left tuple even if there’s no match
* Right outer join:
	+ Include the right tuple even if there’s no match
* Full outer join:
	+ Include the both left and right tuples even if

there’s no match

46

## Modifying the Database

Three kinds of modifications

* Insertions
* Deletions
* Updates

Sometimes they are all called “updates”

47

## Insertions

General form:

INSERT INTO R(A1,…., An) VALUES (v1,…., vn)

Example: Insert a new purchase to the database:

INSERT INTO Purchase(buyer, seller, product, store)

VALUES (‘Joe’, ‘Fred’, ‘wakeup-clock-espresso-machine’, ‘The Sharper Image’)

Missing attribute  NULL.

May drop attribute names if give them in order.

48

## Insertions

INSERT INTO PRODUCT(name)

SELECT DISTINCT Purchase.product

FROM Purchase

WHERE Purchase.date > “10/26/01”

The query replaces the VALUES keyword. Here we insert *many* tuples into PRODUCT

49

## Insertion: an Example

Product(name, listPrice, category)

Purchase(prodName, buyerName, price)

prodName is foreign key in Product.name

Suppose database got corrupted and we need to fix it:

Purchase

Product

|  |  |  |
| --- | --- | --- |
| prodName | buyerName | price |
| camera | John | 200 |
| gizmo | Smith | 80 |
| camera | Smith | 225 |

|  |  |  |
| --- | --- | --- |
| name | listPrice | category |
| gizmo | 100 | gadgets |

Task: insert in Product all prodNames from Purchase

50

## Insertion: an Example

INSERT INTO Product(name)

|  |  |  |  |
| --- | --- | --- | --- |
| SELECT | DISTINCT | prodName |  |
| FROM | Purchase |  |  |  |
| WHERE | prodName | NOT IN (SELECT | name FROM | Product) |

|  |  |  |
| --- | --- | --- |
| name | listPrice | category |
| gizmo | 100 | Gadgets |
| camera | - | - |

51

## Insertion: an Example

INSERT INTO Product(name, listPrice)

SELECT DISTINCT prodName, price

FROM Purchase

WHERE prodName NOT IN (SELECT name FROM Product)

Depends on the implementation

|  |  |  |
| --- | --- | --- |
| name | listPrice | category |
| gizmo | 100 | Gadgets |
| camera | 200 | - |
| camera ?? | 225 ?? | - |

52

## Deletions

Example:

DELETE

FROM

PURCHASE

WHERE

seller = ‘Joe’ AND

product = ‘Brooklyn Bridge’

Factoid about SQL: there is no way to delete only a single

occurrence of a tuple that appears twice in a relation.

53

## Updates

Example:

UPDATE PRODUCT

SET price = price/2 WHERE Product.name IN

(SELECT product FROM Purchase

WHERE Date =‘Oct, 25, 1999’);

54