

# Introduction to Database Systems

## Introduction to Database Systems

- So, what is a database, anyway?
- An integrated, self-describing collection of data about related sets of things and the relationships among them

*If you burned down all our plants, and we just kept our people and our information files, we should soon be as strong as ever.*

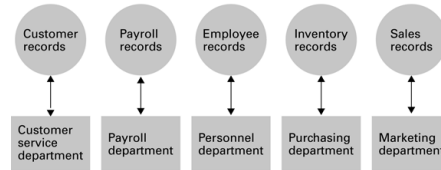
Thomas Watson, Jr. Former chairman of IBM

## Database Management Systems

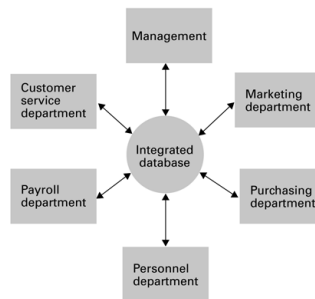
- Simple text files or office documents are one way to store data:
  - Fine for small amounts of data
  - But impractical for large amounts of data
- Businesses must maintain huge amounts of data
  - A *database management system (DBMS)* is the typical solution to the data needs of business
  - Designed to store, retrieve, & manipulate data
- Most programming languages can communicate with several DBMS
  - Tells DBMS what data to retrieve or manipulate

## File vs. Database organization

a. File-oriented information system

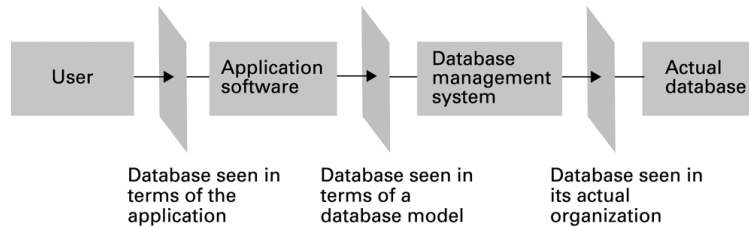


b. Database-oriented information system



## Layered Approach to Using a DBMS

- Applications that work with a DBMS use a layered approach
  - Application is topmost layer
  - Application sends instructions to next layer, the DBMS
  - DBMS works directly with data
- Programmer need not understand the physical structure of the data
  - Just need to know how to interact with the database



## Why not just use the file system?

Day8-21.txt

```
8,drive to work  
9,teach class  
10, ...
```

Day8-22.txt

```
8,drive to work  
9,eat donut  
10, ...
```

...

Could write programs to operate on this text file data.

## File Storage Problems

- Sharing data
- Same data may be duplicated many times
- Need to write custom programs to manipulate the data (e.g search, print)
- As file systems become more complex, managing files gets more difficult
- Making changes in existing file structures is important and difficult.
- Security, data integrity (redundancy, inconsistency, anomalies) features are difficult to implement and are lacking.

## File Storage Problems - Dependence

- *Structural Dependence*: A change in the file's structure requires the modification of all programs using that file.
- *Data Dependence*: A change in any file's data characteristics requires changes of all data access programs.

## Solution: DBMS

- Logically related data are stored in a single data repository.
- The database represents a change in the way end user data are stored, accessed, and managed efficiently.
- Easier to eliminate most of the file system's data inconsistency, data anomalies, and data structural dependency problems.
- Store data structures and relationships (schema)
- DBMS takes care of defining all the required access paths.

## Disadvantages of DBMS

- Cost of software and implementation
- Higher cost of processing routine batches
- Increase magnitude of potential disaster
- Lack of database technical capability

## Relational Database Model

- Introduced in the 60's and 70's and is the most common type of DBMS today
- Data elements stored in simple tables (related)
- General structure good for many problems
- Easy to understand, modify, maintain

Examples: MySQL, Access, Oracle, SQL Server

- We will focus on relational databases using Microsoft Access in our course

## The Relational Model

- Views entities as two-dimensional tables
  - Records are rows
  - Attributes (fields) are columns
- Tables can be linked
- Supports one-to-many, many-to-many, and one-to-one relationships

## Terminology

- *Database*: a collection of interrelated tables
- *Table*: a logical grouping of related data
  - A category of people, places, or things
  - For example, employees or departments
  - Organized into rows and columns
- *Field*: an individual piece of data pertaining to an item, an employee name for instance
- *Record*: the complete data about a single item such as all information about an employee
  - A record is a row of a table

## Database Table

- Each table has a *primary key*
  - Uniquely identifies that row of the table
  - Emp\_Id is the primary key in this example
  - Serves as an index to quickly retrieve the record
- Columns are also called *fields* or *attributes*
- Each column has a particular data type

Emp_Id	First_Name	Last_Name	Department
001234	Ignacio	Fleta	Accounting
002000	Christian	Martin	Computer Support
002122	Orville	Gibson	Human Resources
003400	Ben	Smith	Accounting
003780	Allison	Chong	Computer Support

Row (Record) →

↑ Column

↘ Field

## Choosing Column Names

- Define a column for each piece of data
- Allow plenty of space for text fields
- Avoid using spaces in column names
- For the members of an organization:

<u>Column Name</u>	<u>Type</u>	<u>Remarks</u>
Member_ID	int	Primary key
First_Name	varchar(40)	
Last_Name	varchar(40)	
Phone	varchar(30)	
Email	varchar(50)	
Date_Joined	smalldatetime	Date only, no time values
Meenings_Attended	smallint	
Officer	Yes/No	True/False values



## Issues with Redundant Data

- Database design minimizes redundant data
- In the following employee table:

<u>ID</u>	<u>First Name</u>	<u>Last Name</u>	<u>Department</u>
001234	Ignacio	Fleta	Accounting
002000	Christian	Martin	Computer Support
002122	Orville	Gibson	Human Resources
00300	Jose	Ramirez	Research & Devel
003400	Ben	Smith	Accounting
003780	Allison	Chong	Computer Support

- Same dept name appears multiple times
  - Requires additional storage space
  - Causes problems if misspelled
  - What if a department needs to be renamed?

## Eliminating Redundant Data

- Create a department table

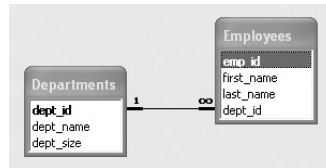
<u>Dept ID</u>	<u>Dept Name</u>	<u>Num Employees</u>
1	Human Resources	10
2	Accounting	5
3	Computer Support	30
4	Research & Development	15

- Reference department table in employee table

<u>ID</u>	<u>First Name</u>	<u>Last Name</u>	<u>Dept ID</u>
001234	Ignacio	Fleta	2
002000	Christian	Martin	3
002122	Orville	Gibson	1
003000	Jose	Ramirez	4
003400	Ben	Smith	2
003780	Allison	Chong	3

## One-to-Many Relationships

- The previous changes created a *one-to-many relationship*
  - Every employee has one and only one dept
  - Every department has many employees
  - DeptID in department table is a *primary key*
  - DeptID in employee table is a *foreign key*
- One-to-many relationship exists when primary key of one table is specified as a field of another table



## Normalization

- The previous example illustrated a technique used to make complex databases more efficient called Normalization
- Break one large table into several smaller tables
  - Eliminates all repeating groups in records
  - Eliminates redundant data
- Another example...

## Redundant Data

Student ID#	Student Name	Campus Address	Major	Phone	Course ID	Course Title	Instructor Name	Instructor Location	Instructor Phone	Term	Grade
A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 350	Intro. MIS	Van Deventer	T240C	555-2222	F'98	A
A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 372	Database	Hann	T240F	555-2224	F'98	B
A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 375	Elec. Comm.	Chatterjee	T240D	555-2228	F'98	B +
A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 448	Strategic MIS	Chatterjee	T240D	555-2228	F'98	A -
A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 474	Telecomm	Gilson	T240E	555-2226	F'98	C +
A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 350	Intro. MIS	Van Deventer	T240C	555-2222	F'98	A
A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 372	Database	Hann	T240F	555-2224	F'98	B -
A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 375	Elec. Comm.	Chatterjee	T240D	555-2228	F'98	A -
A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 448	Strategic MIS	Chatterjee	T240D	555-2228	F'98	C +
A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 350	Intro. MIS	Van Deventer	T240C	555-2222	F'98	A -
A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 372	Database	Hann	T240F	555-2224	F'98	A -
A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 375	Elec. Comm.	Chatterjee	T240D	555-2228	F'98	B +
A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 474	Telecomm	Gilson	T240E	555-2226	F'98	B
A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 350	Intro. MIS	Van Deventer	T240C	555-2222	F'98	A
A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 372	Database	Hann	T240F	555-2224	F'98	B +
A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 375	Elec. Comm.	Chatterjee	T240D	555-2228	F'98	B +
A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 474	Telecomm	Gilson	T240E	555-2226	F'98	A -
...	...	...	...	...	...	...	...	...	...	...	...

## Normalized Data

Student Table

Student ID#	Student Name	Campus Address	Major	Phone
A121	Joy Egbert	100 N. State Street	MIS	555-7771
A123	Larry Mueller	123 S. State Street	MIS	555-1235
A124	Mike Guon	125 S. Elm	MGT	555-2214
A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245
...	...	...	...	...

Enrolled Table

Student ID#	Course ID	Term	Grade
A121	MIS 350	F'98	A
A121	MIS 372	F'98	B
A121	MIS 375	F'98	B +
A121	MIS 448	F'98	A -
A121	MIS 474	F'98	C +
A123	MIS 350	F'98	A
A123	MIS 372	F'98	B -
A123	MIS 375	F'98	A -
A123	MIS 448	F'98	C +
A124	MIS 350	F'98	A -
A124	MIS 372	F'98	A -
A124	MIS 375	F'98	B +
A124	MIS 474	F'98	B
A126	MIS 350	F'98	A
A126	MIS 372	F'98	B +
A126	MIS 375	F'98	B +
A126	MIS 474	F'98	A -
...	...	...	...

Teaching Assignment

Course ID	Term	Instructor Name
MIS 350	F'98	Van Deventer
MIS 372	F'98	Hann
MIS 375	F'98	Chatterjee
MIS 448	F'98	Chatterjee
MIS 474	F'98	Gilson
...	...	...

Class Table

Course ID	Course Title
MIS 350	Intro. MIS
MIS 372	Database
MIS 375	Elec. Comm.
MIS 448	Strategic MIS
MIS 474	Telecomm
...	...

Instructor Table

Instructor Name	Instructor Location	Instructor Phone
Chatterjee	T240D	555-2228
Gilson	T240E	555-2226
Hann	T240F	555-2224
Valacich	T240D	555-2223
Van Deventer	T240C	555-2222

## Exercise

- Your company uses the following spreadsheet.  
How might it be normalized into database tables?

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Invoice No.	Date	Cust. No.	Cust. Name	Cust. Address	Cust. City	Cust. State	Item ID	Item Descr	Item Qty.	Item Price	Item Total	Order Total Price
2	125	9/13/2002	56	Foo, Inc.	23 Main St., Thorpleburg	Thorpleburg	TX	563	56" Blue Fre	4	\$ 3.50	\$ 14.00	\$ 82.00
3	125	9/13/2002	56	Foo, Inc.	23 Main St., Thorpleburg	Thorpleburg	TX	851	Spline End i	32	\$ 0.25	\$ 8.00	\$ 82.00
4	125	9/13/2002	56	Foo, Inc.	23 Main St., Thorpleburg	Thorpleburg	TX	652	3" Red Free	5	\$ 12.00	\$ 60.00	\$ 82.00
5	126	9/14/2002	2	Freens R Us	1600 Pennsylvania Avenu	Washington	DC	563	56" Blue Fre	500	\$ 3.50	\$1,750.00	\$ 10,750.00
6	126	9/14/2002	2	Freens R Us	1600 Pennsylvania Avenu	Washington	DC	652	3" Red Free	750	\$ 12.00	\$9,000.00	\$ 10,750.00

## Associations

- Relationships among the entities in the data structures
- Three types
  - One-to-one
  - One-to-many
  - Many-to-many
- Relationships set by placing primary key from one table as foreign key in another
  - Creates “acceptable” redundancy

## Association Examples

### One-to-one (1:1)

EMPLOYEE — SPOUSE

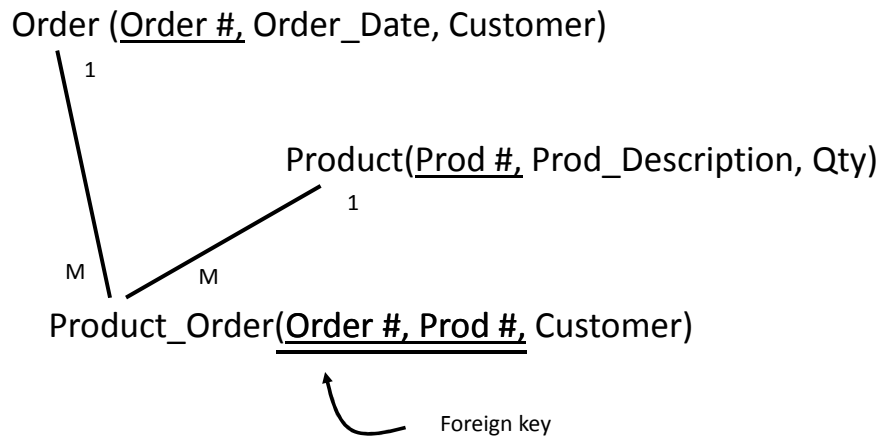
### One-to-many (1:M)

FACULTY — ADVISEE

### Many-to-many (N:M)

FACULTY > COURSE

## Associations



## Microsoft Access is Unique

- Provides DBMS functions
  - Not “industrial-strength”, designed for:
    - Individuals
    - Small workgroups
  - External application programs work with Access
- Provides built-in tools for reporting and for application development
  - Forms
  - Reports
  - Code modules using Visual Basic for Applications (VBA)
- Provides flexibility
  - Small, simple all-in-one environment
  - Data can be easily transferred to full-fledged DBMS

## Introduction to Access

- Sample databases
  - Northwind
    - Included with every version of Access since 2.0
- Demonstration of Access
  - Startup
  - Create tables
  - Link table relationships
  - Create queries/reports

## Access 2007 Example

<u>Student ID</u>	<u>Last Name</u>	<u>First Name</u>	<u>DOB</u>	<u>Address</u>
1	Mock	Kenrick	4-18-1968	123 Somewhere Ave
2	Cue	Barbie	3-21-1970	567 A Street
3	Obama	Barack	8-04-1961	123 Somewhere Ave

## Access 2007 Example

CS 101 Table		CS 201 Table	
<u>Student ID</u>	<u>Grade</u>	<u>Student ID</u>	<u>Grade</u>
1	A	1	B
2	B	2	A
3	B	3	C

## SQL

- Structured Query Language, abbreviated SQL
  - Usually pronounced “sequel” but also “ess-cue-ell”)
  - The common language of client/server database management systems.
  - Standardized – you can use a common set of SQL statements with all SQL-compliant systems.
  - Defined by E.F. Codd at IBM research in 1970.
  - Based on relational algebra and predicate logic

## SQL Data Retrieval

- Given an existing database, the SELECT statement is the basic statement for data retrieval.
  - Both simple and complex, and it may be combined with other functions for greater flexibility.

```
SELECT data_element1 [, {data_element2 | function(..)} ] Or *
FROM      table_1, [, table_2, ...]
[ WHERE condition_1 [, {not, or, and} condition_2] ]
[ GROUP BY data_1, ... ]
[ HAVING  aggregate function(...)... ]
[ORDER BY data1, ... ]
```



## SELECT statement

- Some sample aggregate functions:
  - COUNT(\*)      SUM(item)
  - AVG(item)      MAX(item)
  - MIN(item)
- Conditional Operators
  - =            Equal
  - <            Less than
  - >            Greater than
  - <>,!=        Not equal to
  - <=            Less than or equal to
  - >=            Greater than or equal to

## SELECT Examples

ORDERS

<u>CUST_ID</u>	<u>PROD_ID</u>	<u>COST</u>	<u>SALESPERSON</u>
100	P999	20	Jones
101	P999	30	Jones
101	X310	500	Parker
102	Z225	15	Smith

- Select every row, column from the table:
  - SELECT \* FROM Orders;
  - SELECT Orders.cust\_id, Orders.prod\_id, Orders.cost,  
Orders.salesperson  
FROM Orders;
- Returns a set of all rows that match the query

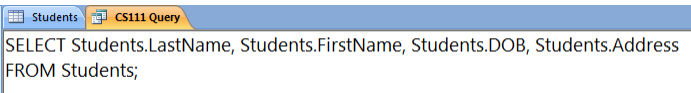
## SELECT

- If a table has spaces or certain punctuation in it, then Access needs to have the items enclosed in square brackets []. The previous query is identical to the following:
  - SELECT [orders].[cust\_id], orders.prod\_id, orders.cost, orders.[salesperson]  
FROM Orders;

## SELECT Query in Access

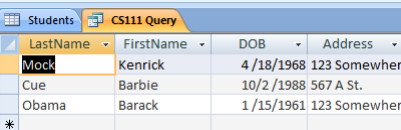
- Can flip back and forth between SQL View, Run, and Design Mode

SQL



```
SELECT Students.LastName, Students.FirstName, Students.DOB, Students.Address
FROM Students;
```

Run



LastName	FirstName	DOB	Address
Mock	Kenrick	4/18/1968	123 Somewher
Cue	Barbie	10/2/1988	567 A St.
Obama	Barack	1/15/1961	123 Somewher

Design

Field:	LastName	FirstName	DOB	Address
Table:	Students	Students	Students	Students
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:				
or:				

## More SELECT Statements

- Note that we can have duplicates as a result of the selection. If we want to remove duplicates, we can use the DISTINCT clause:

```
SELECT DISTINCT Orders.cust_id
FROM Orders;
```

- We can combine a selection and a projection by using the WHERE clause:

```
SELECT Orders.cust_id
FROM Orders
WHERE Salesperson = "Jones";
```

- This could be used if we wanted to get all the customers that Jones has sold to, in this case, CUST\_ID=101 and CUST\_ID=100. By default, Access is not case-sensitive, so "jones" would also result in the same table.

## More SELECT

- We can further refine the query by adding AND , OR, or NOT conditions. If we want orders from Jones or from Smith then the query becomes:

```
SELECT Orders.cust_id
FROM Orders
WHERE Salesperson = "Jones" or Salesperson = "Smith";
```

- Another refinement is to use the BETWEEN operator. If we want only those orders between 10 and 100 then we could define this as:

```
SELECT Orders.cust_id, Orders.cost
FROM Orders
WHERE Orders.cost >10 and Orders.cost <100;
```

- Or use the between operator:

```
SELECT Orders.cust_id, Orders.cost
FROM Orders
WHERE Orders.cost BETWEEN 10 and 100;
```

## More SELECT

- Finally, we might want to sort the data on some field. We can use the ORDER BY clause:

```
SELECT Orders.cust_id, Orders.cost
FROM Orders
WHERE Orders.cost >10 and Orders.cost <100
ORDER BY Orders.cost;
```

- This sorts the data in ascending order of cost. An example is shown in the table:

CUST_ID	COST
102	15
100	20
101	30

- If we wanted to sort them in descending order, use the DESC keyword:

```
SELECT Orders.cust_id, Orders.cost
FROM Orders
WHERE Orders.cost >10 and Orders.cost <100
ORDER BY Orders.cost DESC;
```

## Joining Data from Multiple Tables

- If our data is in multiple tables we can join them together in one query.
  - Use a JOIN operator (Access default w/Design view)
  - Add tables to the FROM, WHERE section (what we will use here)
- Say we have the following table in addition to Orders:

CUSTOMER

<u>CUST_ID</u>	<u>CUST_NAME</u>	<u>MEMBER_DATE</u>
100	Thomas Jefferson	9/27/99
101	Bill Clinton	9/26/99
102	George Bush	9/25/99

## Multiple Tables

```
SELECT Orders.cust_id, Customer.Cust_Name
FROM Orders, Customer
WHERE Orders.cost >10 and Orders.cost <100;
```

Result:

CUSTOMER

CUST_ID	CUST_NAME	MEMBER_DATE
100	Thomas Jefferson	9/27/99
101	Bill Clinton	9/26/99
102	George Bush	9/25/99

ORDERS

CUST_ID	PROD_ID	COST	SALESPERSON
100	P999	20	Jones
101	P999	30	Jones
101	X310	500	Parker
102	Z225	15	Smith

```
100 Thomas Jefferson
101 Thomas Jefferson
102 Thomas Jefferson
100 Bill Clinton
101 Bill Clinton
102 Bill Clinton
100 George Bush
101 George Bush
102 George Bush
```

PRODUCT of two tables!

- What do you expect from this query?

## Multiple Tables

- Need to link the tables by their common field, the customer ID:

```
SELECT Orders.cust_id, Customer.Cust_Name
FROM Orders, Customer
WHERE Orders.cust_id = Customer.Cust_Id and
Orders.cost >10 and Orders.cost <100;
```

CUSTOMER

CUST_ID	CUST_NAME	MEMBER_DATE
100	Thomas Jefferson	9/27/99
101	Bill Clinton	9/26/99
102	George Bush	9/25/99

ORDERS

CUST_ID	PROD_ID	COST	SALESPERSON
100	P999	20	Jones
101	P999	30	Jones
101	X310	500	Parker
102	Z225	15	Smith

Result:

```
100 Thomas Jefferson
101 Bill Clinton
102 George Bush
```

## INSERT command

- Allows you to insert single or multiple rows of data into a table
- `INSERT INTO table [(column-list)] [VALUES (value-list) | sql-query]`

## INSERT examples

Given mytable(field1 as currency, field2 as text, field3 as integer):

```
INSERT INTO mytable (field1, field2, field3)
VALUES (12.10, "bah",20);
```

Adds a new row to the table mytable

If you don't specify every field then fields left out get the default:

```
INSERT INTO mytable (field1, field2) VALUES(24.2, "zot");
```

Adds only for field1 and field2.

## INSERT Examples

ORDERS

<u>CUST_ID</u>	<u>PROD_ID</u>	<u>COST</u>	<u>SALESPERSON</u>
100	P999	20	Jones
101	P999	30	Jones
101	X310	500	Parker
102	Z225	15	Smith

```
INSERT INTO ORDERS (CUST_ID, PROD_ID, COST, SALESPESON)
VALUES (103, 'Y338', 55, 'Smith');
```

```
INSERT INTO ORDERS (PROD_ID, COST, SALESPESON)
VALUES ('Y638', 155, 'Smith');
```

Second might be useful if the CUST\_ID is an autonumber field

## DELETE

- Delete will remove a row from the table.
- DELETE FROM table\_name [WHERE search-condition]

Examples:

```
DELETE FROM mytable1;
```

Removes all rows!

```
DELETE FROM mytable1 WHERE field1 > 100;
```

Removes only rows with field1>100

## UPDATE

- Update lets you modify the contents of the data.

UPDATE table\_name

SET field\_name = expression [, field-name=expression ...]

[WHERE search-condition]

```
UPDATE mytable SET field1 = 0.0;
```

Changes all field1's to zero for every row!

```
UPDATE mytable SET field1 = 0.0, field2 = "woof";
```

Sets field1 to 0 and field2 to woof for all rows!

If this is a violation, access will prevent it from happening

```
UPDATE mytable SET field1 = 25.0 WHERE field2="foo";
```

Only updates the field where field2 is "foo"

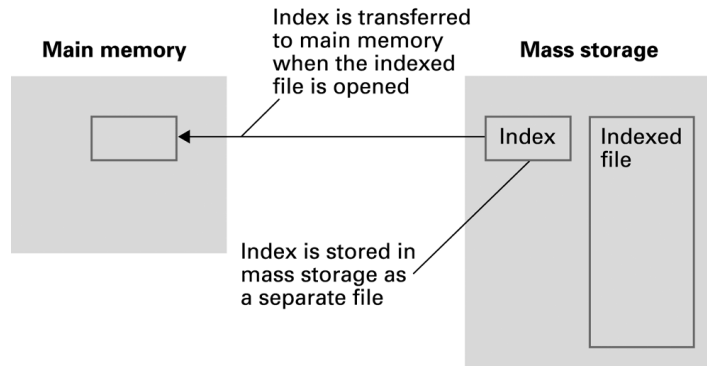
## SQL Queries

- There are a lot more queries, but that should give you an idea of what is possible and how it is done



## Indexed files

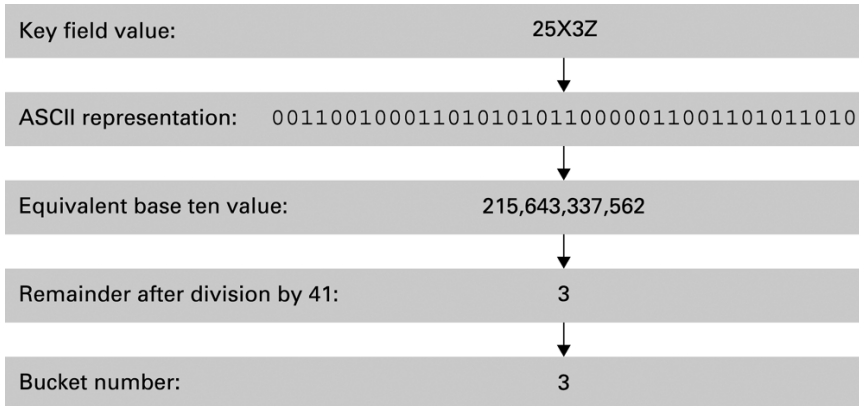
Mostly skipping implementation of database systems; a little on indices - key to quickly accessing a record



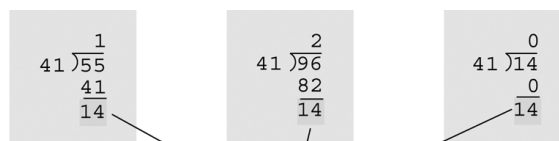
## Hashing

- Each record has a key field
- The storage space is divided into **buckets**
- A **hash function** computes a bucket number for each key value
- Each record is stored in the bucket corresponding to the hash of its key

## Hashing the key field value 25X3Z to one of 41 buckets

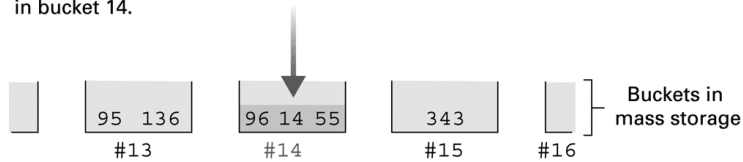


## The rudiments of a hashing system



Remainders

When divided by 41, the key field values of 14, 55, and 96 each produce a remainder of 14. Thus these records are stored in bucket 14.



## Collisions in Hashing

- **Collision:** The case of two keys hashing to the same bucket
  - Major problem when table is over 75% full
  - Solution: increase number of buckets and rehash all data

## Data Mining

- **Data Mining:** The area of computer science that deals with discovering patterns in collections of data
- **Data warehouse:** A static data collection to be mined
  - **Data cube:** Data presented from many perspectives to enable mining

## Social Impact of Database Technology

- Problems
  - Massive amounts of personal data are being collected
    - Often without knowledge or meaningful consent of affected people
  - Data merging produces new, more invasive information
  - Errors are widely disseminated and hard to correct
- Remedies
  - Existing legal remedies often difficult to apply
  - Negative publicity may be more effective