**Database Concepts**

## File Organization Terms and Concepts



Figure 6-1: The data hierarchy

An **entity** is basically the person, place, thing, or event on which you maintain information. Each characteristic or quality describing an entity is called an **attribute**. In the table below, each column describes a characteristic (**attribute**) of John Jones’ (who is the **entity**) address.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **First Name** | **Last Name** | **Street** | **City** | **State** | **Zip** | **Telephone** |
| John | Jones | 111 Main St | Center City | Ohio | 22334 | 555-123-6666 |

## Problems with the Traditional File Environment

***Data Redundancy and Inconsistency:*** Have you ever gotten two pieces of mail from the same organization? For instance, you get two promotional flyers from your friendly neighborhood grocery store every month. It may not necessarily be that you’re a popular person. It’s probably because your data was somehow entered twice into the business’s database. That’s **data redundancy**. Now, let’s say you change residences and, consequently, your address. You notify everyone of your new address including your local bank. Everything is going smoothly with your monthly statements. All of a sudden, at the end of the year, the bank sends a Christmas card to your new address and one to your old address. Why? Because your new address was changed in one database, but the bank maintains a separate database for its Christmas card list and your address was never changed in it. That’s **data inconsistency.** Just from these two simple examples you can see how data redundancy and inconsistency can waste resources and cause nightmares on a much larger scale.

***Program-Data Dependence:*** Some computer software programs, mainly those written for large, mainframe computers, require data to be constructed in a particular way. Because the data are specific to that program, it can’t be used in a different program. If an organization wants to use the same data in a different program, it has to reconstruct it. Now the organization is spending dollars and time to establish and maintain separate sets of data on the same entities because of **program-data dependence**.

***Lack of Flexibility:*** The Sales and Marketing manager needs information about his company’s new production schedule. However, he doesn’t need all of the data in the same order as the Production manager’s weekly report specifies. Too bad. The company’s database system lacks the flexibility to give the Sales manager the information he needs, how he needs it, and when he would like to receive it.

***Poor Security:*** Traditional file environments have little or no security controls that limit who receives data or how they use it. With all the data captured and stored in a typical business, that’s unacceptable.

***Lack of Data Share and Availability:*** What if the CEO of a business wants to compare sales of Widget A with production schedules. That might be difficult if production data on the widgets is maintained differently by the sales department. This problem happens far more frequently in older traditional file environments that lack the ability to share data and make it available across the organization.

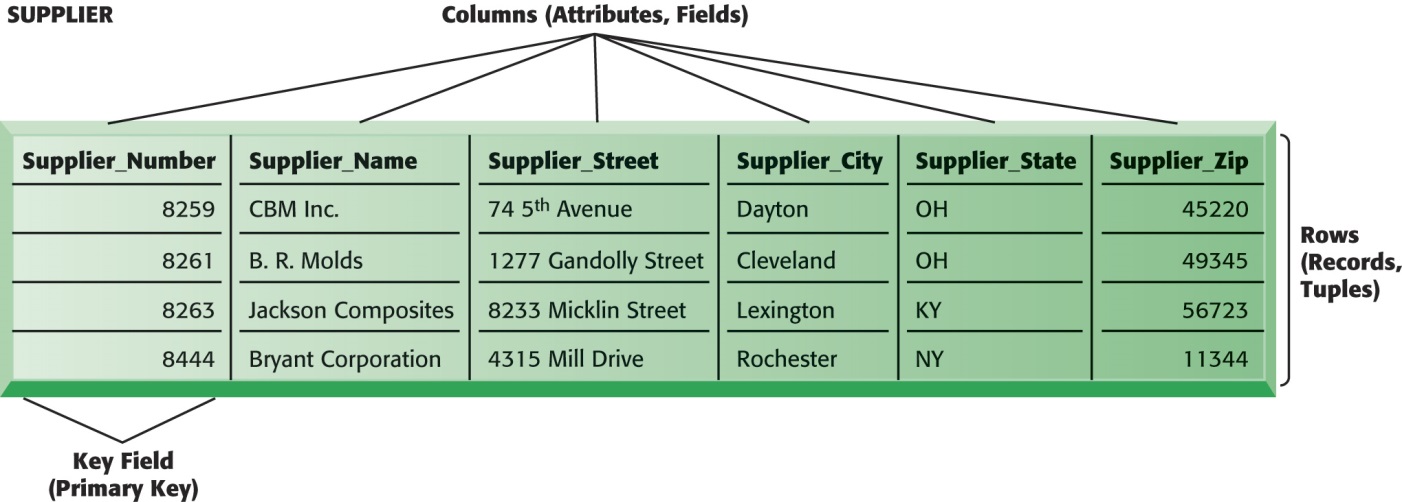
## Database Management Systems

A **Database Management System** (DBMS) is basically another software program like Word or Excel or e-mail. This type of software is more complicated; it permits an organization to centralize data, manage them efficiently, and provide access to the stored data by application programs.

### Relational DBMS

A **relational database** stores data in tables. The data are then extracted and combined into whatever form or format the user needs. The tables are sometimes called files, although that is actually a misnomer.

Data in each table are broken down into **fields.** A field, or column, contains a single attribute for an entity. A group of fields is stored in a **record** or **tuple** (the technical term for record). Figure 6-4 shows the composition of a relational database table.

****

A Relational Database Table.

Each record requires a **key field**, or unique identifier. The best example of this is your social security number: there is only one per person. That explains in part why so many companies and organizations ask for your social security number when you do business with them.

In a relational database, each table contains a **primary key**, a unique identifier for each record. To make sure the tables relate to each other, the primary key from one table is stored in a related table as a **foreign key**. For instance, in the customer table below the primary key is the unique customer ID. That primary key is then stored in the order table as the foreign key so that the two tables have a direct relationship.

|  |  |  |  |
| --- | --- | --- | --- |
| Customer Table | | Order Table | |
| *Field Name* | Description | Field Name | Description |
| Customer Name | Self-Explanatory | Order Number | Primary Key |
| Customer Address | Self-Explanatory | Order Item | Self-Explanatory |
| **Customer ID** | **Primary Key** | Number of Items Ordered | Self-Explanatory |
| **Order Number** | **Foreign Key** | **Customer ID** | **Foreign Key** |

**Wrong way:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Address** | **Telephone number** |
| John L. Jones | 111 Main St Center City Ohio 22334 | 555-123-6666 |

**Right way:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **First Name** | **Middle Initial** | **Last Name** | **Street** | **City** | **State** | **Zip** | **Telephone** |
| John | L. | Jones | 111 Main St | Center City | Ohio | 22334 | 555-123-6666 |

### Normalization and Entity-Relationship Diagrams

We mentioned before that you want to create the smallest data fields possible. You also want to avoid redundancy between tables and not allow a relationship to contain **repeating data groups**. You do not want to have two tables storing a customer’s name. That makes it more difficult to keep data properly organized and updated. What would happen if you changed the customer’s name in one table and forgot to change it in the second table? Minimizing redundancy and increasing the stability and flexibility of databases is called **normalization.**

Your goals for creating a good data model are:

* Including all entities and the relationships among them
* Organizing data to minimize redundancy
* Maximizing data accuracy
* Making data easily accessible

Whichever relationship type you use, you need to make sure the relationship remains consistent by enforcing **referential integrity**. That is, if you create a table that points to another table, you must add corresponding records to both tables.

Determine the relationships between each data entity by using an **entity-relationship diagram (below)**.

