Database Basics

What’s a Database?
- a repository of information
- a collection of records
Examples:
  - your phone book
  - Airline reservation system
  - Medical records and billing data
  - Employment records

Database management systems (DBMS)
- collections of tools used to manage databases
- Four basic functions performed by all DBMS are:
  - Create, modify, and delete data structures, e.g. tables
  - Add, modify, and delete data
  - Retrieve data selectively
  - Generate reports based on data

Schema:
- describes the objects that are represented in the database, and the relationships among them.

Database Models
- a number of different ways of modeling the database structure: these are known as “database models” (or data models).
- The model in most common use today is the relational model, which in layman's terms represents all information in the form of multiple related tables each consisting of rows and columns

Database Components
- Related tables/Relations
- Fields
- Records
Table
- A table is a collection of related records. For example, employee table, product table, customer, and orders tables.
- In a table, records are represented by rows and fields are represented as columns.

Fields
- an area (within a record) reserved for a specific piece of data.
- Examples: customer id, customer name, street address, city, phone, current balance.
- defined by:
  - Field name
  - Data type
  - Field size (Amount of space reserved for storing data)

Record/Tuple
- A record is the collection of values for all the fields pertaining to one entity: i.e. a person, product, company, transaction, etc.

Relationships
- One-to-One
- One-to-Many
- Many-to-Many

One-to-Many relationship: Customer-Order, each order has only one customer, but a customer can make many orders.
One-to-Many relationships consist of two tables, the "one" table, and the "many" table.

Many-to-Many relationship: Order-Product, an order can contain many products, and a product can be on many orders.
A Many-to-Many relationship consists of three tables: two "one" tables, both in a One-to-Many relationship with a third table. The third table is sometimes referred to as the join.
Key Fields

- **Primary key**: A Primary key is a field or a combination of two or more fields. The value in the primary key field for each record uniquely identifies that record.

  In our previous example, customer number is the Primary key for the Customer table. A customer number identifies one and only one customer in the Customer table.

- **Foreign key**: When a "one" table's primary key field is added to a related "many" table in order to create the common field which relates the two tables, it is called a foreign key in the "many" table.

  In our previous example, the primary key (customer number) from the Customer table ("one" table) is a foreign key in the Orders table ("many" table). For the "many" records of the Order table, the foreign key identifies with which unique record in the Customer table they are associated.

Example

<table>
<thead>
<tr>
<th>Customer</th>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bil</td>
<td>abc</td>
<td>12345</td>
</tr>
<tr>
<td>2</td>
<td>Ali</td>
<td>XYZ</td>
<td>56789</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Price</th>
<th>Quantity/Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Java: The Complete Reference</td>
<td>700</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>C++: The Complete Reference</td>
<td>300</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>CustomerID</th>
<th>Date</th>
<th>AmountPayable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>15/10/2006</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>20/10/2006</td>
<td>750</td>
</tr>
</tbody>
</table>

Data types

- **CHAR**: This will hold between 0 and 255 characters. You have to specify the maximum amount of characters that you will be putting in when you create your column.

- **VARCHAR**: This will hold as many characters as you like up to 2 gigabytes of space.

- **INT**: This will hold a value of 0 to 2 billion. It is primarily used for numeric and boolean issues. It is also referred to as a number or a whole field.

- **FLOAT**: This can hold a value of 0 or 1. It is primarily used for mathematical issues.

- **DATE**: This will hold a date and a time. It is also commonly referred to as a TIMESTAMP since it is primarily used to track when entries or updates to a row or table.

How Do I Create a Table?

- **Using an SQL statement.**

  ```sql
  CREATE TABLE contacts (
    contact_id INT IDENTITY (1, 1) NOT NULL ,
    first_name CHAR (30) NULL ,
    last_name CHAR (50) NULL ,
    email VARCHAR (75) NULL
  );
  ```

Designing a Table in a Database Tool? ....

- **Using Microsoft Access**

  **INSERT statement**

  ```sql
  INSERT INTO contacts
  (first_name, last_name, email)
  VALUES ('Bilal', 'Afzal', 'bilal@hotmail.com');
  ```
What is that **NULL** thing?
- NULL is how you tell the DBMS that you have nothing to enter in this column.
- Should I always list all of my table columns?
  - Yes, it improves readability for future code updates

**UPDATE Statement**

```
UPDATE contacts
SET
  email = 'bilal@hotmail.com'
WHERE
  contact_id = 1;
```

**SELECT Statement**

```
SELECT *
FROM contacts;
```

```
SELECT first_name, email
FROM contacts
WHERE
  first_name = 'Bilal';
```

```
SELECT * FROM contacts WHERE (first_name='Bilal' AND
  last_name='Afzal') OR last_name='Ahmed';
```

```
SELECT * FROM contacts ORDER BY last_name DESC;
```

```
SELECT * FROM contact WHERE last_name LIKE 'al';
```

**Example Queries**

**AND/OR operators**
- SELECT * FROM contacts WHERE (first_name='Bilal' AND
  last_name='Afzal') OR last_name='Ahmed';

**LIKE**
- SELECT * FROM contact WHERE last_name LIKE '%al';

**ORDER BY Clause:**
- SELECT * FROM contacts ORDER BY last_name DESC;

**Grouping**

Imagine a scenario where you are running a gift catalog business for corporations and you want to know how many items have been sold to each corporate client you have. To accomplish this task you will need to learn about COUNT and GROUP BY.

**COUNT**

- keeps a running total of the number of records retrieved.

```
SELECT COUNT(*) AS sales_total
FROM sales;
```

So, what you end up with here is the total number of all of the rows on the "sales" table which will be stored in the temporary column of "sales_total".
**GROUP BY**

```sql
SELECT company_name, COUNT(*) AS sales_by_company_total FROM sales GROUP BY company_name;
```

The output would look something like this:

<table>
<thead>
<tr>
<th>company_name</th>
<th>sales_by_company_total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dollar General Stores</td>
<td>5</td>
</tr>
<tr>
<td>CompUSA</td>
<td>512</td>
</tr>
<tr>
<td>Sears</td>
<td>222</td>
</tr>
<tr>
<td>Amazon.com</td>
<td>478</td>
</tr>
</tbody>
</table>

For sorted output:

```sql
SELECT company_name, COUNT(*) AS sales_by_company_total FROM sales GROUP BY company_name ORDER BY company_name ASC;
```

**Aggregate Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX()</td>
<td>Returns a selected column's highest value</td>
</tr>
<tr>
<td>MIN()</td>
<td>Returns a selected column's lowest value</td>
</tr>
<tr>
<td>AVG()</td>
<td>Returns the average value of a selected column</td>
</tr>
<tr>
<td>SUM()</td>
<td>Returns the sum of a selected column</td>
</tr>
<tr>
<td>COUNT()</td>
<td>Counts the number of rows in a selected column</td>
</tr>
</tbody>
</table>

**Examples using Aggregate Functions**

- SELECT MAX(sale_total) FROM daily_receipts WHERE sales_date = '01/01/2002';
- SELECT MIN(event_date) FROM event WHERE event_date > GETDATE();
- SELECT SUM(sales) FROM daily_receipts WHERE sales_date = '01/01/2002';
- SELECT AVG(sales) FROM daily_receipts WHERE sales_date = '01/01/2002';

**DELETE statement**

DELETE
FROM contacts
WHERE contact_id = 3;

- What if you don't provide the WHERE clause?

**What's So Great About DISTINCT?**

- SELECT DISTINCT category FROM discussion_group;
- SELECT COUNT(DISTINCT category) FROM discussion_group;
  - This will give you the total number of unique categories in your discussion group.

**References**