Normalization
The highlights of this topic include

- **What normalization is**
  - Normalization techniques
  - Raw database
  - Logical database
  - Data redundancy
  - The three normal forms

- **Benefits of normalization**

- **Drawback of normalization**
Normalizing a Database

- Normalization is a process of reducing redundancies of data in a database.
- Normalization is a technique that is used when designing and redesigning a database.
- The actual guidelines of normalization, called normal forms
The Raw Database

- A database that is not normalized may include data that is contained in one or more different tables for no apparent reason. This could be bad for security reasons, disk space usage, speed of queries, efficiency of database updates, and, maybe most importantly, data integrity.

- A database before normalization is one that has not been broken down logically into smaller, more manageable tables. *Figure 1* illustrates sample of database used before it was normalized.
### COMPANY_DATABASE

<table>
<thead>
<tr>
<th>emp_id</th>
<th>cust_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_name</td>
<td>cust_name</td>
</tr>
<tr>
<td>first_name</td>
<td>cust_address</td>
</tr>
<tr>
<td>middle_name</td>
<td>cust_city</td>
</tr>
<tr>
<td>address</td>
<td>cust_state</td>
</tr>
<tr>
<td>city</td>
<td>cust_zip</td>
</tr>
<tr>
<td>state</td>
<td>cust_phone</td>
</tr>
<tr>
<td>zip</td>
<td>cust_fax</td>
</tr>
<tr>
<td>phone</td>
<td>ord_num</td>
</tr>
<tr>
<td>pager</td>
<td>qty</td>
</tr>
<tr>
<td>position</td>
<td>ord_date</td>
</tr>
<tr>
<td>date_hire</td>
<td>prod_id</td>
</tr>
<tr>
<td>pay_rate</td>
<td>prod_desc</td>
</tr>
<tr>
<td>bonus</td>
<td>cost</td>
</tr>
<tr>
<td>date_last_raise</td>
<td></td>
</tr>
</tbody>
</table>
Any database should be designed with the end user in mind.

Logical database design, also referred to as the logical model, is the process of arranging data into logical, organized groups of objects that can easily be maintained.

The logical design of a database should reduce data repetition or go so far as to completely eliminate it. After all, why store the same data twice? Naming conventions used in a database should also be standard and logical.
Data Redundancy

- Data should not be redundant, which means that the duplication of data should be kept to a minimum for several reasons.

For example, it is unnecessary to store an employee's home address in more than one table.

- With duplicate data, unnecessary space is used.

Confusion is always a threat when, for instance, an address for an employee in one table does not match the address of the same employee in another table. Which table is correct? Do you have documentation to verify the employee's current address? As if data management were not difficult enough, redundancy of data could prove to be a disaster.
The Normal Forms

- The normal forms, an integral concept involved in the process of database normalization.
- Normal form is a way of measuring the levels, or depth, to which a database has been normalized. A database's level of normalization is determined by the normal form.
- The following are the three most common normal forms in the normalization process:
  - The first normal form
  - The second normal form
  - The third normal form
- Of the three normal forms, each subsequent normal form depends on normalization steps taken in the previous normal form.

For example, to normalize a database using the second normal form, the database must first be in the first normal form.
The First Normal Form

- The objective of the first normal form is to divide the base data into logical units called tables. When each table has been designed, a primary key is assigned to most or all tables. Examine Figure 2, which illustrates how the raw database shown in the previous figure has been redeveloped using the first normal form.
The first normal form

- To achieve the first normal form, data had to be broken into logical units of related information, each having a primary key and ensuring that there are no repeated groups in any of the tables.
- Instead of one large table, there are now smaller, more manageable tables: EMPLOYEE_TBL, CUSTOMER_TBL, and PRODUCTS_TBL.
- The primary keys are normally the first columns listed in a table, in this case: EMP_ID, CUST_ID, and PROD_ID.
The Second Normal Form

- The objective of the second normal form is to take data that is only partly dependent on the primary key and enter that data into another table. Figure 3 illustrates the second normal form.
- According to the figure, the second normal form is derived from the first normal form by further breaking two tables down into more specific units.
The second normal form

- EMPLOYEE_TBL split into two tables called EMPLOYEE_TBL and EMPLOYEE_PAY_TBL. Personal employee information is dependent on the primary key (EMP_ID), so that information remained in the EMPLOYEE_TBL (EMP_ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, ADDRESS, CITY, STATE, ZIP, PHONE, and PAGER).
On the other hand, the information that is only partly dependent on the EMP_ID (each individual employee) is used to populate EMPLOYEE_PAY_TBL (EMP_ID, POSITION, POSITION_DESC, DATE_HIRE, PAY_RATE, and DATE_LAST_RAISE).

Notice that both tables contain the column EMP_ID. This is the primary key of each table and is used to match corresponding data between the two tables.
The second normal form (cont2..

- CUSTOMER_TBL split into two tables called CUSTOMER_TBL and ORDERS_TBL. What took place is similar to what occurred in the EMPLOYEE_TBL.
- Columns that were partly dependent on the primary key were directed to another table. The order information for a customer is dependent on each CUST_ID, but does not directly depend on the general customer information in the original table.
The Third Normal Form

- The third normal form's objective is to remove data in a table that is not dependent on the primary key. *Figure 4* illustrates the third normal form.
The third normal form

- Another table was created to display the use of the third normal form. EMPLOYEE_PAY_TBL is split into two tables, one table containing the actual employee pay information and the other containing the position descriptions, which really do not need to reside in EMPLOYEE_PAY_TBL.

- The POSITION_DESC column is totally independent of the primary key, EMP_ID.
Normalization provides numerous benefits to a database. Some of the major benefits include the following:

- Greater overall database organization
- Reduction of redundant data
- Data consistency within the database
- A much more flexible database design
- A better handle on database security
Drawbacks of Normalization

- Although most successful databases are normalized to some degree, there is one substantial drawback of a normalized database: reduced database performance. The acceptance of reduced performance requires the knowledge that when a query or transaction request is sent to the database, there are factors involved, such as CPU usage, memory usage, and input/output (I/O).

- A normalized database requires much more CPU, memory, and I/O to process transactions and database queries than does a denormalized database.

- A normalized database must locate the requested tables and then join the data from the tables to either get the requested information or to process the desired data.