3.2: Database Management

Database Design

Planning out how a database will be structured is easily overlooked in the beginning stages of a development project. The key to this process is to start with a template that will allow for both future expansion of your project while also maintaining that only essential information is kept in data storage. The main reason for this is that many types of databases will become very resource intensive as the amount of data grows to a large size. There are several components of database design that will be explained in this section that should be kept in mind.

Storage Options

Databases are often stored on a hard disc which can also be from the same disc containing the applications file system or it may be on a separate drive altogether. For the purpose of this lesson plan, ignore any systems that use a type of RAM (Random Access Memory) storage for holding temporary databases. When choosing the storage options for a database, you want to look at the types of data that you will be working with. Text, numbers and chemical identifiers work well in almost every type of database. However, storing things like images for chemical structures and long text documents can quickly cause database bloating so it may be necessary to design a system that also relies on file storage. A good practice to get into on choosing your storage options is to only design the database to store things like text and numbers. When working with images and documents, assign these items to the file storage and only keep the object link stored in the database.

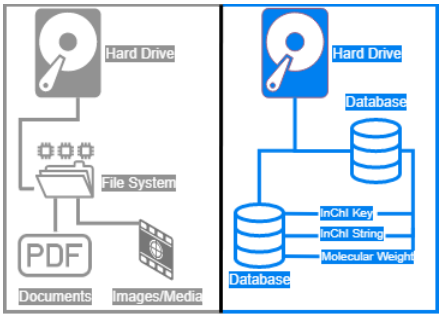

Figure 3.1.4-1 Shown above are two systems for using storage within an application. On the left is a file system which illustrates keeping documents and media stored as a file within a hard drive. This is much more feasible then storing the binary of a file inside of a database which overloads and causes database bloating. The right side shows a database which is also stored on a hard drive, however ,it is an application within itself that allows the structuring and organization of smaller pieces of information such as simple text strings. This allows for quick and easy access to data retrieval and storage without having to open and parse large documents.

Prepare for Expansion

Working with chemical data can present many problems for managing a database. The first and most obvious challenge is to recognize that the amounts of data will continue to expand when new contributions are made. Databases that are poorly designed can be challenging to expand if the available storage limits have been reached. On the other hand, a properly designed database can easily be expanded by just adding more memory or resources. It is the design of the original database that will be important in making sure that the expansions integrate properly.

If the database will be required to house a lot of similar data entries with fewer categories or indexes needed, then a single large capacity database may work fine. Another alternative would be to add new tables with the same structure every time the index number reaches a certain range. This would allow for allocating multiple database engines for faster data retrieval. The actual design of such a system is way beyond the scope of this course, however its important to recognize the process as large online chemical databases may be using similar technology.

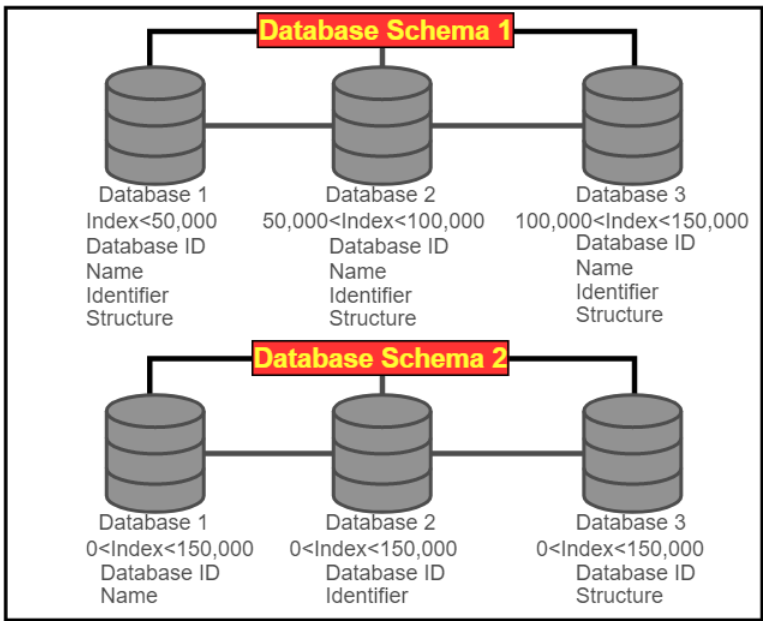

Figure 2 Shown above are two different database schemas for working with large amounts of data. Although both schemas represent only 4 fields for storing data, think of this as a small sample for data that could have hundreds of fields for each chemical entry. Database schema shows that the entries could be broken up simply by having a separate database for each additional 50,000 entries and keeping each chemical with all field entries kept together. Database schema 2 keeps all of the different fields together in the same database, but breaks up separate fields into different databases linked by the chemical database ID for proper retrieval. Both schemas work well for handling large amounts of data, however database schema 1 may be easier to set up for administrators without lots of experience in database design.

Designing Relationships

Challenges to building a database often arise when looking to set up relationships to show how information can be linked. With chemical data, there are endless possibilities on what can be linked and related within a data set. Among the most common relationships would be a chemical needing to be linked with its identifiers and properties such as a melting point or molecular weight. The relationship can be applied directly within a single database table in which the name and other values are only separated by different fields under the same index number. This would be the simplest and most direct way to design a relationship within a database. A problem might arise when the number of fields within this database table grows to a point that it causes data retrieval to slow down.

Solving the problem of building a database with lots of fields, the designer may link different types of tables or possibly using separate databases completely. The benefit to this process is that you can have a field dedicated to linking an index value between separate tables while allowing the resources that power the database to have smaller individual jobs.

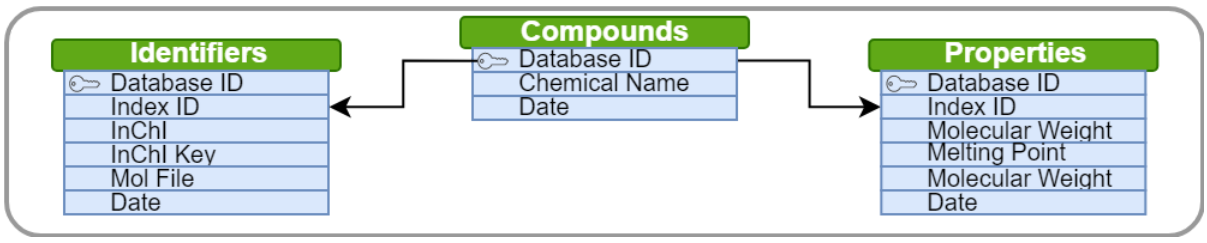

Figure 3.1.4-1 Shown above is an example database relationship. Using the above configuration multiple values for both properties and identifiers could be linked to a table of different chemical names. This might be useful in a situation where only a single InChI matches a particular compound, however that compound may have several different chemical names. This can also be useful in recording the same properties of a compound at different temperatures, pressures, etc.

Creating a Database

There are many options and configurations that can be used when creating a database. To review a few basic types of databases, look at section 1.3 of this text. To look up installing and working with a specific database then it's best to start with the official documentation provided by the creators of the software.

For the purpose of this lesson we will cover the basic concepts of creating the database layout and what to include. Start with a template that you either draw out on paper or electronically that includes some basic fields. The first item to include will be the unique index for all entries. This is most often a number that will automatically increment as data is loaded. It is important that each number be unique as this will be your primary key for the database. When using a spreadsheet or text file for your database, this primary key will usually be your first column and it is what the database will associate all fields with identification of the data. Next, you should define all fields that will be needed to house all of the collected data. These can be text fields, URL placeholders and number fields that will be used to store the related data to your chemical entries.

The last things that should be created are the relationships and indexes which will be used for searching and retrieving the stored data. A relationship should be created to link similar data values and compounds. An example of a popular relationship that is used could be set up through a field that defines types of chemicals. Under this relationship, the database could link all chemicals that fall under a ketone, alcohol, aldehyde, etc. This allows for a database to be queried against all chemicals that share certain functional groups.

Adding Data

Once the layout has been setup for a database, it is important to set up ways to put data into the tables. It is a little beyond the scope of this course, but setting up a web interface is a very common way to add entries into a database. There are many programming languages that can perform this operation such as PHP (Hypertext Preprocessor) and Python. Another common way to add items into a database is through an API (Application Programming Interface). There are many tutorials and videos online for performing each step to connect with a database.

When adding data to a database, it is important to remember that the unique database ID should be created at the same time as the values that are put into their relevant fields. You will want to make sure that the ID does not overwrite any previous entries and that it is set to auto-increment.

Editing Data

Sometimes a database will need to have edits made to previously entered values. A good example would be for a field that stores the location of an image, such as one made to represent a 3D view of a compound. If the user would like to change the image to one of higher quality, then the new uploaded image would have to overwrite the old location to reflect the new image. The commands used to do this should also include deleting the actual image from storage so that space is not used to keep unused files.

Data Removal

It may be bad practice to remove any chemical data, however mistakes do happen and sometimes it may be useful to remove data. This could be for several reasons, but an example could be that the original data may have contained errors and should be removed until a new collection is made. Keep in mind that when removing database entries, a script should also be included that will delete any associated files to that entry. A common practice for deleting database entries is that all deleted data is sent to a temporary storage folder for a short amount of time in case the data needs to be restored. Its always a good idea to keep regular backups of a database should something happen that leaves users unable to access data.