Structured Query Language (SQL)

SQL is scripting syntax for running database queries and performing Create, Read, Update, and Delete (CRUD) operations in a relational database system. Depending on the database management system (DBMS), different versions of SQL will have slight syntactical variations. However, foundational CRUD syntax will typically be common among DBMSs (e.g. SELECT, INSERT, UPDATE, DELETE, etc.).

Here we will explore how SQL is used to build database architectures and navigate the data within it. We start by understanding how basic tables are constructed in a relational database, how SQL builds tables that comprise an architecture, and how SQL creates, reads, updates, and deletes data, among other operations.

**Primary Key:**  Primary keys in a database are records’ unique identifiers.  These values cannot be repeated across records within a given table (unique values), and are used to index records for efficient data retrieval.  When implemented properly in a relational database, primary keys are referenced in other tables as foreign keys.

* Often the first column in a table
* Each table in a relational database usually only assigns one column as its primary key field.
* Never null values
* Always unique values (never reused among records)
* Can never be modified once created

**Foreign Key:**  Foreign keys in a database are reference values with which records in a data table are related to records in another data table.  These reference values are primary keys in their source tables that encapsulate all the data in the corresponding record. Note that it is not good practice for a table reference its own values.

**Applying Keys to Database Tables**

The table below shows records for an office supply company. It tracks their products and indicates their respective product managers. While this data is valuable to track, its database implementation is inefficient given the number of repeated records.

A screenshot of a computer screen

AI-generated content may be incorrect.

Implementing primary keys and foreign keys is a valuable strategy for designing efficient databases. The goal is to reduce the number of repeated records in any given table, thus reducing the memory required to store the database. A key benefit in reducing repeated records in a database is more efficient queries.

The orange table above can be broken out into two tables to eliminate the repeated product manager records. By creating a managers table comprised of unique manager records, the products table becomes less cluttered by eliminating 4 columns of repeated values. This is where primary keys and foreign keys become useful!

A screenshot of a computer

AI-generated content may be incorrect.

The blue table above shows records for product managers' contact infromation. The 'manager\_id' column is defined as a primary key and cast as an integer type. The 'first\_name', 'last\_name', 'email\_address', and 'phone\_number' columns are cast as string types.

Note that the 'phone\_number' column contains numeric values, but the dashes in each value make the records ineligible for numeric types. Alternatively, the dashes could be removed in extract, transform, load (ETL) and the records can then be stored as numeric types, specifically integer types.

A screen shot of a product

AI-generated content may be incorrect.

The green table above implements primary keys and foreign keys to detail a company's products. The 'product\_id' column is defined as a primary key and cast as an integer type. The 'product\_number', 'product\_name', and 'unit' columns are cast as string types, and the 'product\_unit\_price' column is cast as a float type.

The 'product\_manager' column is defined as a foreign key to the managers table primary keys and cast as and integer type, which relates each product to a manager. This also simplifies manager reassignments, as only the foreign key value needs to be updated to the reassigned manager. Furthermore, the manager table can be used in other tables throughout the databaase, keeping a consistent link between managers, products, and other tracked instances.

The foreign keys in the 'product\_manager' column reference the primary keys in the managers table, relating the records in the products table to their product manager.

* The products table primary key, product\_id, 1 is Pencil, encapsulating all recorded pencil data within the key. Among the pencil's data is the foreign key, product\_manager, 3 which references the managers table primary key, manager\_id. So Pencil is associated Alex Tan.
* Similarly:
  + Alex Tan (3) is also the product manager for paper (2).
  + Ben Smith (1) is the product manager for eraser (3), tape (6), and stickynote (7).
  + Sam Gomez (2) is the product manager for paperclip (4)
  + Sarah Lewis (4) is the product manager for stapler (5) and folder (8)

Once the database is architected and loaded with data, SQL is used to interact with the data. Interactions include, but are not limited to:

* Retrieving existing data
* Inserting new data
* Modifying existing data
* Deleting existing data

Simply put, SQL performs queries, which are action requests.

***SQL Database Architecture Syntax***

Databases are structured storage containers that store data. They are comprised of tables that are formatted into rows and columns, facilitating data retrieval and manipulation.

Each row is a data record, comprised of data distributed among its column. Each database column, or field, is assigned a label to identify the data stored in that table cell.

Rows are not inherently ordered, so they must be assigned an identifying value, known primary keys. Using row indices or record primary keys with column names, records can be accessed by identifying their row-column intersection in the table.

Below are some useful SQL operations for navigating and building a database architecture. Note that SQL syntax can vary across DBMSs, so red notes have been added where syntax variation is possible.

|  |  |  |
| --- | --- | --- |
| BOOL | Boolean value | TRUE |
| INT | Integer number | 100 |
| DECIMAL(<M>, <N>) | Decimal, or floating point, that specifies the number of permissible digits, where <M> is the total number of digits in the value and <N> is the number of digits after the decimal | DECIMAL(5, 2): 100.25 |
| DOUBLE | Double-precision floating point number |  |
| DATE | Date formatted as YYYY-MM-DD | 1970-01-01 |
| TIME | Time formatted as HH:MM:SS | 12:30:56 |
| DATETIME | Combined date and time formatted as YYYY-MM-DD HH:MM:SS | 1970-01-01 12:30:56 |
| YEAR | Year formatted as YY or YYYY | 1970 |
| TIMESTAMP | Automatic date and time of record entry | 1970-01-01 12:30:56 |
| CHAR(<N>) | String of defined length up to 255 characters that is stored with padding | CHAR(15): 'HELLO WORLD\_ \_ \_ \_" |
| VARCHAR(<N>) | String of variable length up to 255 characters that is stored without padding | VARCHAR(15): 'HELLO WORLD" |
| TEXT | String of up to 65,535 characters | "HELLO WORLD" |
| BLOB | Binary type for variable data |  |
| ENUM(<VAL\_0>, <VAL\_1>) | Single string value from a defined list of multiple permitted values (only 1 value), up to thousands | ENUM('RED', 'YELLOW', 'BLUE') 'BLUE' |
| SET(<VAL\_0>, <VAL\_1>) | String of multiple strings from a defined list of multiple permitted values (one or more values), up to 64 options | ENUM('RED', 'YELLOW', 'BLUE') ('RED', 'BLUE') |

**Helpful Resources**

* [SQL in Easy Steps](https://a.co/d/aonx2jc) - Mike McGrath
* [W3Schools](https://www.w3schools.com/sql/default.asp)
* [Practice Exercises](https://www.w3schools.com/sql/exercise.asp?filename=exercise_select1)
* [Free Online Database Sandbox](https://sqlplayground.app/)