

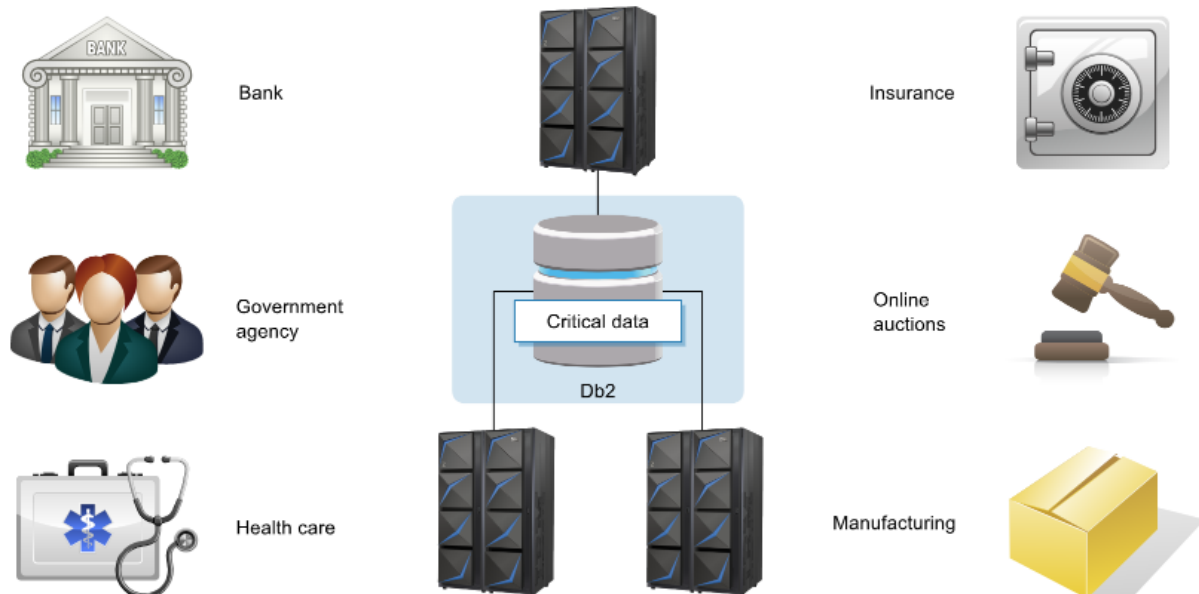
# Table of Contents

|   |          |
|---|----------|
| <b>Table of Contents</b>                | <b>1</b> |
| <b>Overview of DB2</b>                  | <b>3</b> |
| Reality of Digital Data                 | 3        |
| Introducing DB2                         | 3        |
| Advantages of DB2 for z/OS              | 4        |
| DB2 - A Relational Database             | 4        |
| Accessing DB2 Data                      | 5        |
| DB2 - Users                             | 5        |
| DB2 - User Groups                       | 6        |
| DB2 - Publications                      | 6        |
| The DB2 Environment                     | 6        |
| DB2 - Components                        | 6        |
| DB2 - Capabilities                      | 7        |
| DB2 - Tools                             | 7        |
| Middleware                              | 8        |
| <b>Accessing DB2 Data</b>               | <b>9</b> |
| Structured Query Language (SQL)         | 9        |
| Embedding SQL in an Application Program | 9        |
| Dynamic Preparation of SQL              | 10       |
| Preparing the Source Program            | 10       |
| Stored SQL Procedures                   | 11       |
| Interactively Invoking SQL              | 12       |
| XML Data                                | 12       |
| JSON                                    | 12       |
| DB2 Data Components                     | 14       |
| Tables                                  | 14       |
| Table Spaces                            | 14       |
| Indexes                                 | 15       |
| Index Spaces                            | 15       |
| Keys                                    | 15       |
| Views                                   | 16       |
| Storage Groups                          | 16       |
| Schemas                                 | 16       |
| Databases                               | 17       |
| DB2 System Components                   | 18       |
| DB2 - Catalog                           | 18       |
| DB2 - Directory                         | 18       |
| DB2 - Logs                              | 19       |
| DB2 - Buffer Pools                      | 19       |
| Databases                               | 20       |
| DB2 and the Parallel Sysplex            | 20       |

# Overview of DB2

## Reality of Digital Data

It is estimated that the amount of digital data increases at a rate of 60% each year. With 80% of the world's data processed by mainframes, organizations need to ensure that their data is stored securely and that it can be accessed reliably and quickly.



Organizations such as banks, health care, insurance and government departments need to be able to share their data between sites and applications and have it available 24 hours a day.

## Introducing DB2

Many of the organisations mentioned above have turned to DB2 to manage their data. Today's DB2 can be run on a variety of platforms, but the majority of information in these notes will focus on DB2 for z/OS, which combines a robust relational database (DB2) with the dependability of the mainframe (z/OS).

Within this structure, data can be made extremely secure, while providing the organisation with continuous availability of data and a scalable solution for future data growth.

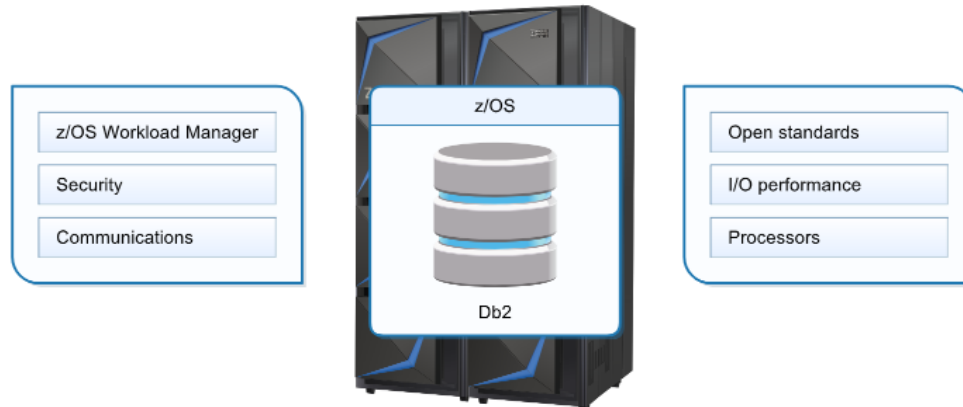
Some examples of data issues faced by organisations:

- **Banks** - May be dealing with new financial reforms requiring it to store additional records and produce new reports.
- **Health Care** - May have recently acquired a previous competitor, whose data systems are in a different format and are more geographically spread.
- **Government Agency** - May be looking to reduce energy costs and improve operational efficiency, the availability of data and access times to it.

- **Online Auctions** - May be consolidating the IT systems that are currently running different operating systems on various hardware platforms.

### **Advantages of DB2 for z/OS**

DB2 by itself provides you with a data structure that:

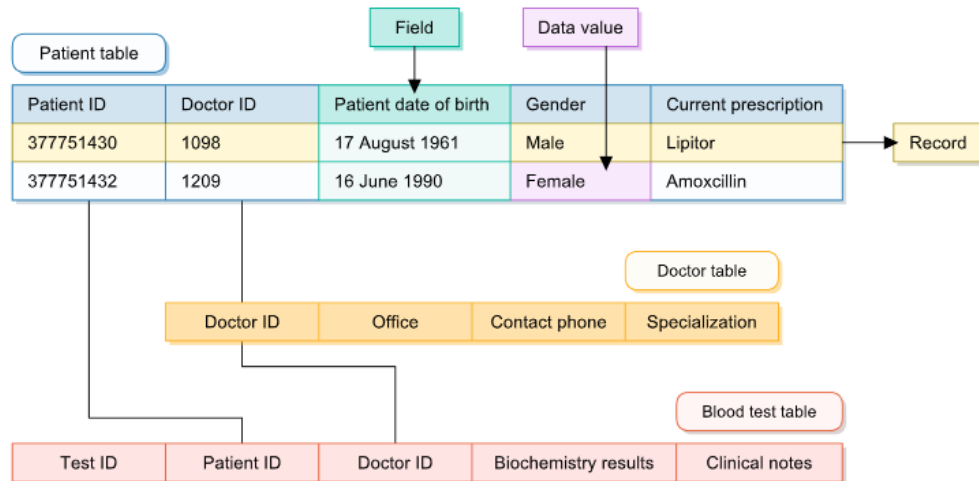


- Is easy to understand. Data from DB2 databases is presented to users in table-like format. This is one of the more familiar ways of displaying data, used in, for example, telephone books and transport timetables.
- Allows design flexibility. A table of data is relatively easy to design. All you need to know are the key components of your database, such as Name, Address and Telephone number.
- Provides ease of access. The user is shielded from the internal workings of Db2. Therefore, the user does not need to know the physical relationships of the database they are accessing.

### **DB2 - A Relational Database**

Db2 itself is a relational database, which is basically a collection of related data that is stored in multiple tables that can be reassembled to form meaningful user data.

The example displayed here shows a simple relational database structure for a healthcare organization. Three tables are displayed that contain doctor, patient, and blood test results. In reality there would be anywhere between ten and a thousand tables for this type of organization. The columns of each table form field names, while a row of data values forms a specific record.



## Accessing DB2 Data

The data stored in Db2 is accessed using SQL, which is a computer language specifically designed to communicate with relational databases. SQL is both simple and powerful, since a single SQL statement can generate an entire report, which would take numerous lines of conventional programming code.

As an example, if you wanted to list all doctors and sort them by the office they work in, you would need to access the Doctor table using the following SQL statement:

```
SELECT * FROM DOCTOR  
ORDER BY OFFICE;
```

SQL statements can be invoked from a number of sources if the appropriate interface software to Db2 has been installed. Common z/OS environment interfaces include:

- **TSO** - Allows you to interact with DB2 using the TSO attachment facility, the call attachment facility (CAF), and the Resource Recovery Services (RRS) attachment facility.
- **Batch** - Batch jobs accessing DB2 data use the same attachment facilities as those mentioned for TSO.
- **CICS** - The CICS Transaction Server provides access to DB2 through the CICS attachment facility.
- **IMS** - This is a hierarchical database and information management system that can access DB2 data using the IMS attachment facility.
- **WebSphere** - A suite of WebSphere products have been designed to be able to access DB2 data using the Resource Recovery Services (RRS) attachment facility.

You can also start Db2 sessions from other environments on clients such as Microsoft Windows or UNIX by using interfaces like ODBC, JDBC, and SQLJ.

## DB2 - Users

If a problem occurs with Db2, it may be your responsibility to identify the person it is escalated to, so it is important to understand the functions performed by IT personnel who interact with DB2.

- **Application Developer** - This person is responsible for all phases of development and maintenance associated with the development of DB2 related applications.
- **Database Administrator** - This person is responsible for the logical and physical design of databases, and implementation of those databases to support the business applications. They may also be involved in monitoring DB2 space usage and database performance.
- **DB2 Systems Programmer** - This person is usually responsible for the installation, upgrade and migration of the DB2 subsystem and related software. They may also be responsible for performance monitoring and tuning as well as reporting and resolving software issues.
- **Computer Operator** - This person monitors the overall computing system, including DB2. They may be responsible for starting and stopping the database or subsystem and responding to system messages produced by this product.

## **DB2 - User Groups**

If you are working with DB2, then there are a considerable range of options available to keep up to date with DB2 developments and interact with people.

User groups, and their websites and resources, can benefit you by expanding your resource pool and provide you with exposure to technical personnel with common areas of interest.

**The International DB2 Users Group** is one such organization, whose website contains information on conferences, blogs, and webcasts.

Many individual states have formed their own DB2 user groups, making it easier to attend meetings and meet people face to face.

## **DB2 - Publications**

There are a number of sources that provide DB2 users with publications such as technical user guides, useful tips, case studies, and general articles on various DB2 products and utilities.

**IBM's online Knowledge Center** is always a great place to start when looking for in-depth information, while IBM's DB2 product page contains links to articles and blogs on DB2 products.

## **The DB2 Environment**

### **DB2 - Components**

As mentioned previously, DB2, or the DB2 data server as it is more commonly known, and its associated products can be implemented and run on a number of different operating systems including: z/OS, IBM i, Linux, UNIX, and Microsoft Windows. This is useful if an organization consists of a mixture of these systems.

While each version shown here is tweaked to support the individual capabilities of that system, SQL is virtually the same in each environment, allowing data sharing to occur between data servers and application code with minimal modifications.



## **DB2 - Capabilities**

Db2 data servers support a large number of different clients and languages, and with additional support can integrate data from other sources.

| Data sources                        | Languages               | Clients      |
|-------------------------------------|-------------------------|--------------|
| IMS                                 | APL2                    | AIX          |
| Informix                            | Assembler               | Eclipse      |
| Oracle                              | C                       | HP-UX        |
| Microsoft® SQL Server, Excel        | C++                     | Linux        |
| Sybase                              | C#                      | Solaris      |
| JDBC                                | COBOL                   | Windows      |
| Databases that support the JDBC API | Fortran                 | Web browsers |
| OLE DB                              | Java                    |              |
| Teradata                            | .NET                    |              |
| EMC Documentum                      | Perl                    |              |
|                                     | PHP                     |              |
|                                     | PL/I                    |              |
|                                     | Python                  |              |
|                                     | REXX                    |              |
|                                     | Ruby on Rails           |              |
|                                     | SQL procedural language |              |
|                                     | Visual Basic .NET       |              |

## **DB2 - Tools**

There is a long list of tools that may be implemented by your organization to support aspects of DB2 for z/OS performance, administration, backup and recovery, automation, and replication and application management. DB2 for z/OS includes a comprehensive toolset, but many additional tools are optionally available from IBM as well as from third-party software vendors.

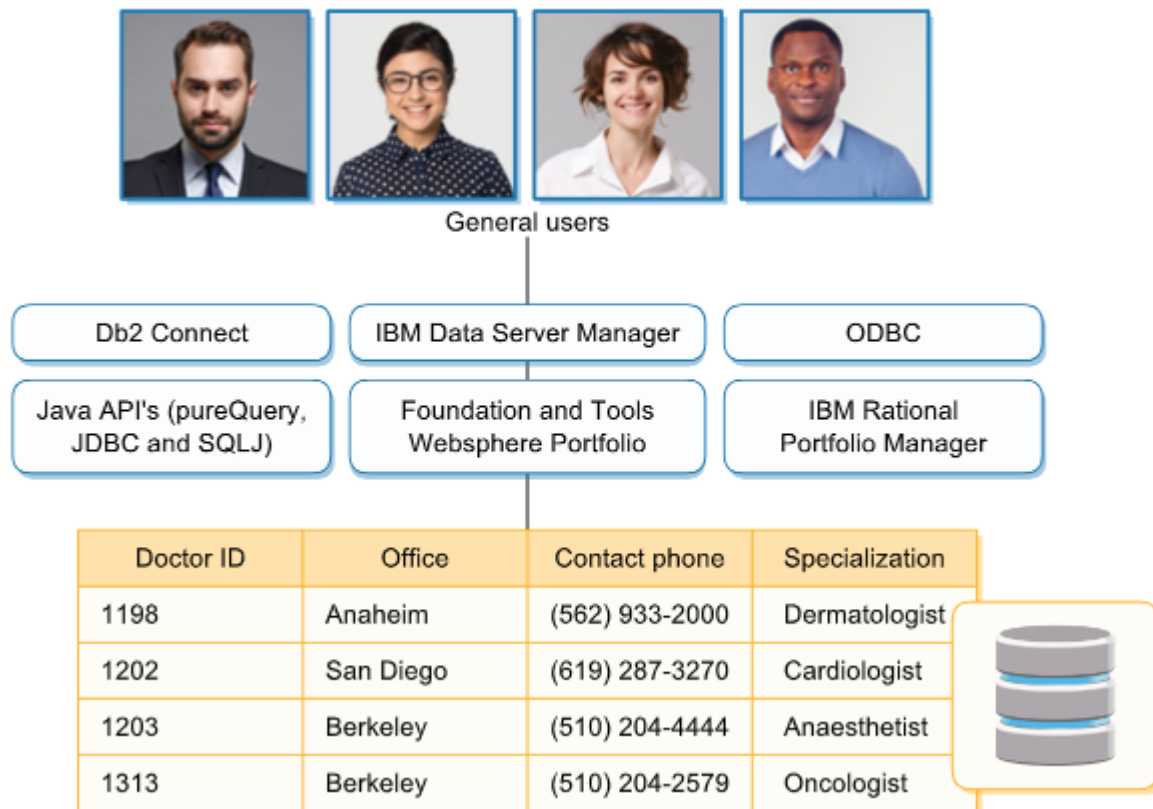
The tools mentioned below are provided by IBM.

- **DB2 Utilities Solution Pack for z/OS V4.2**
  - This solution provides you with the framework to better execute and manage DB2 utilities, ensuring improved performance optimisation and resource utilisation. The DB2 Automation Tool for z/OS can be set up to work with the Data Server Manager.
- **DB2 Performance Solution Pack for z/OS V1.5**
  - This solution enables you to quickly identify, diagnose, solve and prevent performance problems in your DB2 for the z/OS environment.
- **DB2 Administration Solution Pack for z/OS V3.1.0**
  - This solution manages the complexity, growth, change and cloning of DB2 for the z/OS objects and scheme throughout the application lifecycle.

Many of the DB2 for z/OS tools display data using a graphical user interface or ISPF and allow you to perform a number of DB2 tasks interactively.

## **Middleware**

Today's on-demand environment requires powerful and efficient middleware products and APIs to define and manage application development and facilitate communication between clients and the Db2 content.

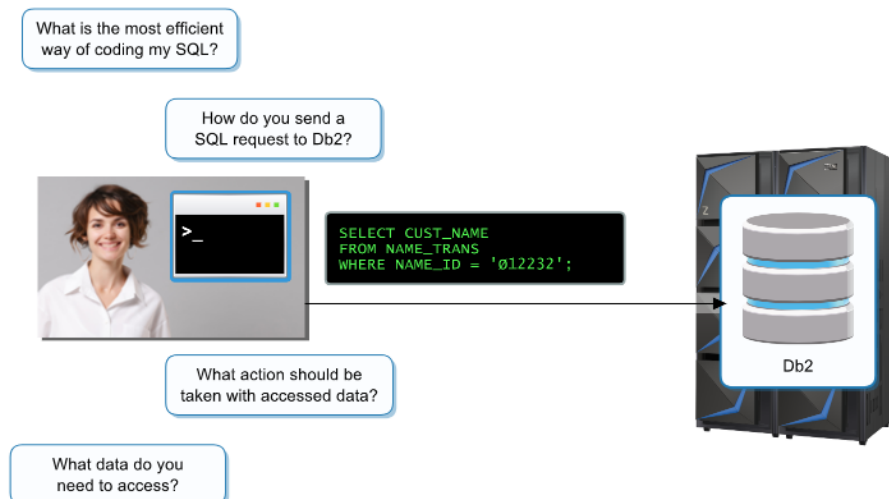




# Accessing DB2 Data

## Structured Query Language (SQL)

In the scenario shown here, an application programmer has coded some structured query statements to access specific DB2 data. SQL consists of over 100 different statements that can be used to insert, query, update, delete, and authorize access to DB2 data.



## Embedding SQL in an Application Program

SQL statements can be inserted into source application programs. This type of invocation is referred to as an embedded statement.

In the COBOL example shown here, the keywords EXEC SQL are required to invoke SQL statements and END-EXEC indicates the end of the SQL stream.

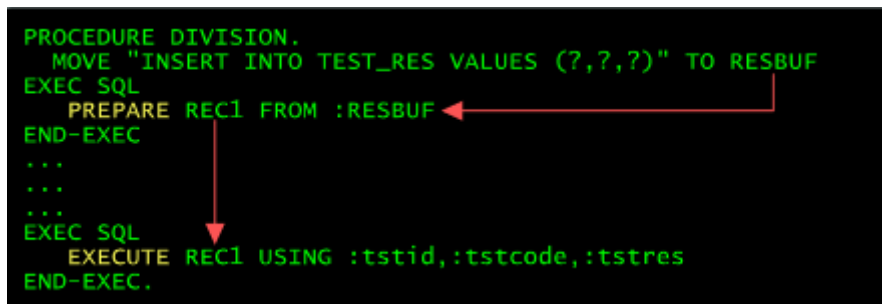
- For C and C++ programs, each SQL statement must begin with EXEC SQL and end with a semicolon (;).
- For REXX, each SQL statement must begin with EXEC SQL (in either upper-, lower-, or mixed-case) and be followed by either an SQL statement (enclosed in single or double quotation marks) or a REXX variable that contains an SQL statement.
- For Java, embedded SQL is executed within SQLJ executable clauses which begin with #sql and end with a semicolon.

```
EXEC SQL DECLARE LRNR.DEPARTMENT TABLE
( DEPTNO          CHAR(3) NOT NULL,
  DEPTNAME        VARCHAR(29) NOT NULL,
  MGRNO           CHAR(6),
  ADMRDEPT        CHAR(3) NOT NULL,
  LOCATION        CHAR(16)
) END-EXEC.
*****
* COBOL DECLARATION FOR TABLE LRNR.DEPARTMENT *
*****
01 DCLDEPARTMENT.
10 DEPTNO          PIC X(3).
10 DEPTNAME        PIC X(29).
49 DEPTNAME-LEN    PIC S9(4) USAGE COMP.
49 DEPTNAME-TEXT   PIC X(29).
10 MGRNO           PIC X(6).
10 ADMRDEPT        PIC X(3).
10 LOCATION        PIC X(16).
```

## Dynamic Preparation of SQL

Rather than using static SQL statements as in the previous example, you can dynamically build SQL statements based on input provided to the application.

```
PROCEDURE DIVISION.  
  MOVE "INSERT INTO TEST_RES VALUES (?, ?, ?)" TO RESBUF  
  EXEC SQL  
    PREPARE REC1 FROM :RESBUF  
  END-EXEC  
  ...  
  ...  
  ...  
  EXEC SQL  
    EXECUTE REC1 USING :tstid, :tstcode, :tstres  
  END-EXEC.
```



The example here shows that PREPARE and EXECUTE statements have been used to create an executable SQL statement from a string and then execute it.

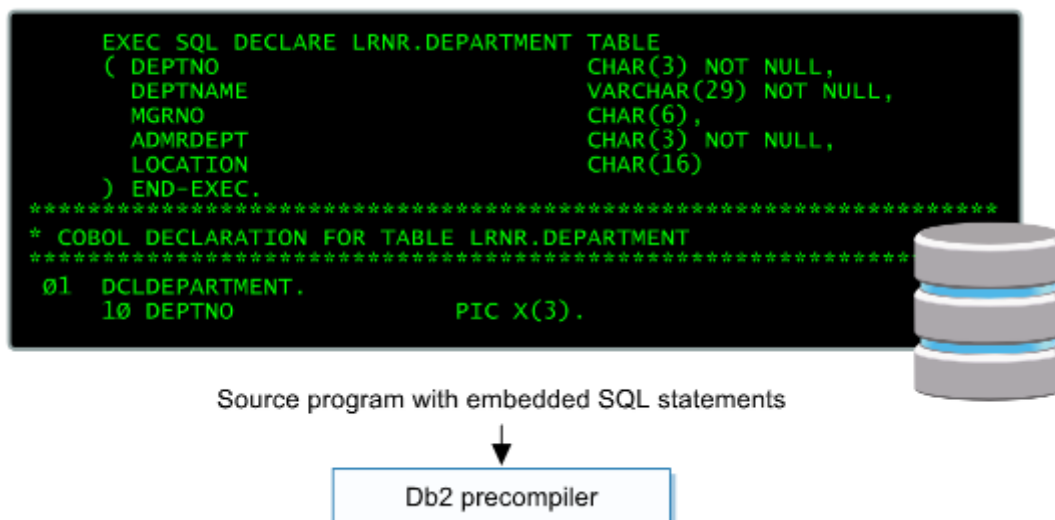
- All non-Java languages use this code or the EXECUTE IMMEDIATE statement.
- While Java uses Statement, PreparedStatement, and CallableStatement classes to perform the same function.

## Preparing the Source Program

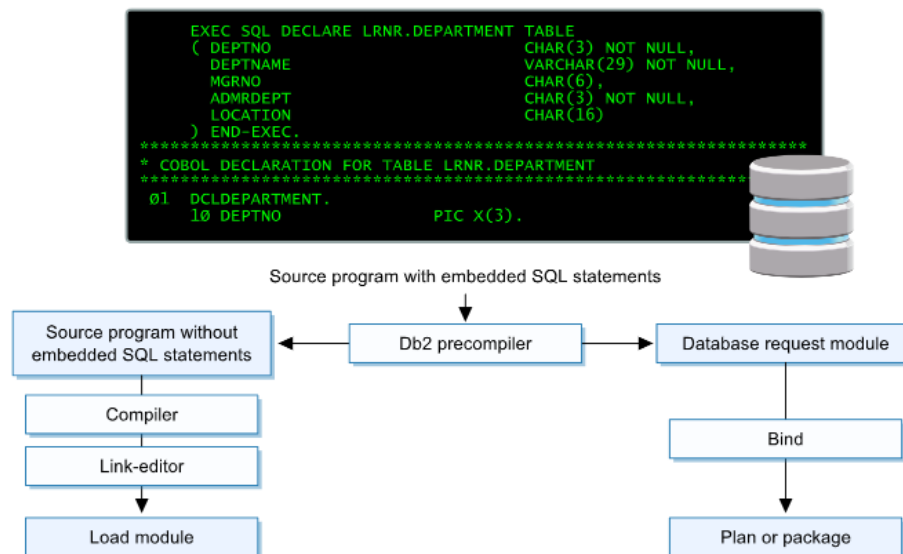
In traditional programming languages, whether there are static or dynamic SQL statements in your source program, they both need to be processed before the program is compiled.

With **Static SQL** statements, the DB2 precompiler or coprocessor checks the syntax of the SQL statements and if acceptable will convert them into host language statements that are used to access DB2 data.

**Dynamic SQL** statements are handled differently in that they are not precompiled but are instead passed to DB2 as a character string by the program. These statements are then processed at run time.



The DB2 precompiler stage produces a Database Request Module (DBRM) that contains the SQL statements and host variable information from the source program. This information is used when the BIND command is invoked, and will create a plan or package that contains control structures and processing options.

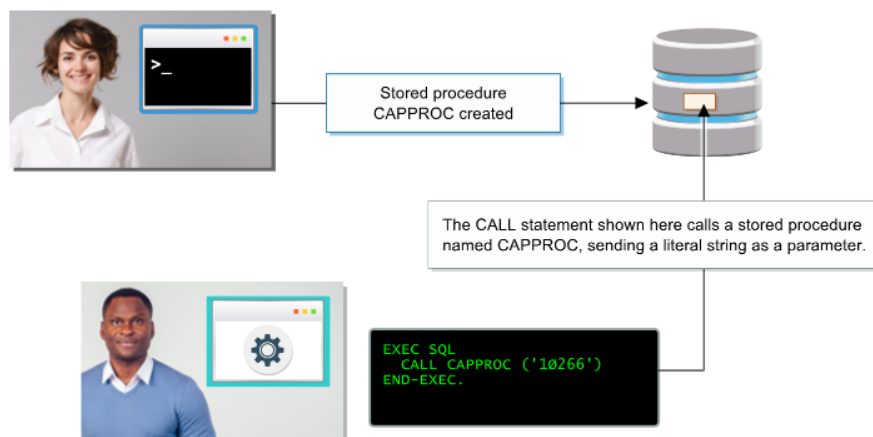


Finally an executable load module needs to be created from your source program using a link-edit procedure. The DB2 application is now ready to run.

## Stored SQL Procedures

Where an application program containing SQL statements performs a common function, it can be placed as a stored procedure on the database server. The stored procedure is then invoked from an application program using the CALL statement. The benefits of having a stored procedure are the following:

- There is less network traffic as the procedure is stored in close proximity to the data.
- Less duplication of code as it is only written once, which means that any changes will also only need to be made to one set of code.
- Security can be enhanced by assigning database privileges to the stored procedure rather than to users.



## Interactively Invoking SQL

SQL statements can be sent to DB2 directly from a workstation. In the example shown here, a TSO facility called **SPUFI (SQL Processor Using File Input)** allows you to execute SQL statements from an input file, without needing to embed them in an application program.

```
====>                                SPUFI                                SSID: DB11

Enter the input data set name:         (Can be sequential)
1 DATA SET NAME ... ==> 'BAY1.SPUFI.INPUT'
2 VOLUME SERIAL ... ==>
3 DATA SET PASSWORD ==>              (Enter if not cataloged)
                                       (Enter if password required)

Enter the output data set name:        (Must be sequential data set)
4 DATA SET NAME ... ==> 'BAY1.SPUFI.OUTPUT'

Specify processing options:
5 CHANGE DEFAULTS ... ==> YES
6 EDIT INPUT ..... ==> YES
7 Execute ..... ==> YES
8 AUTOCOMMIT ..... ==> YES
9 BROWSE OUTPUT ... ==> YES          (Y/N - Browse output data set?)

For remote SQL processing:
10 CONNECT LOCATION ==>

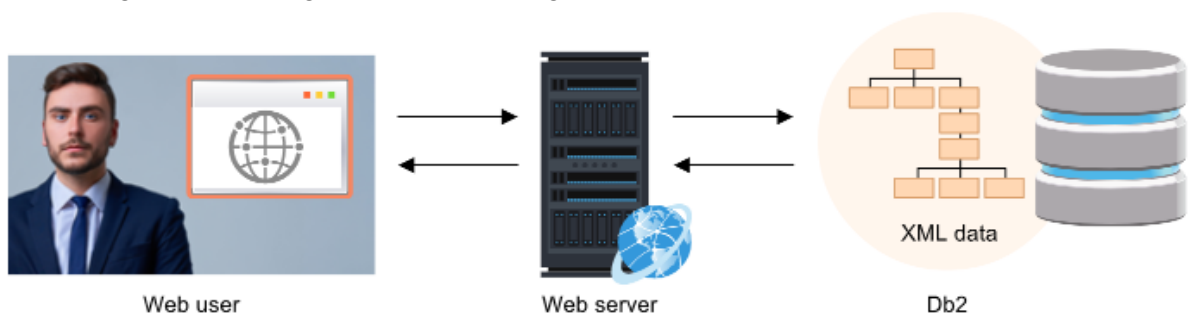
PRESS: ENTER to process  END to exit  HELP for more information
```

The input data set or PDS member contains either existing SQL statements or is empty, allowing you to enter your own statements.

The result from processing the SQL statements is placed in the output data set, which is automatically opened in browse mode following the processing.

## XML Data

XML data can also be stored in a DB2 table and is accessed, with a few exceptions, using the same SQL statements. This XML data can be used by organizations for document processing and providing information through its Web services.



DB2 uses a feature called **pureXML** to manage XML data stored in DB2 tables.

## JSON

Today's rapidly changing application environments have seen the emergence of JavaScript Object Notation (JSON) as a key technology to achieving an organization's mobile, social, big data analytics, and cloud data needs.

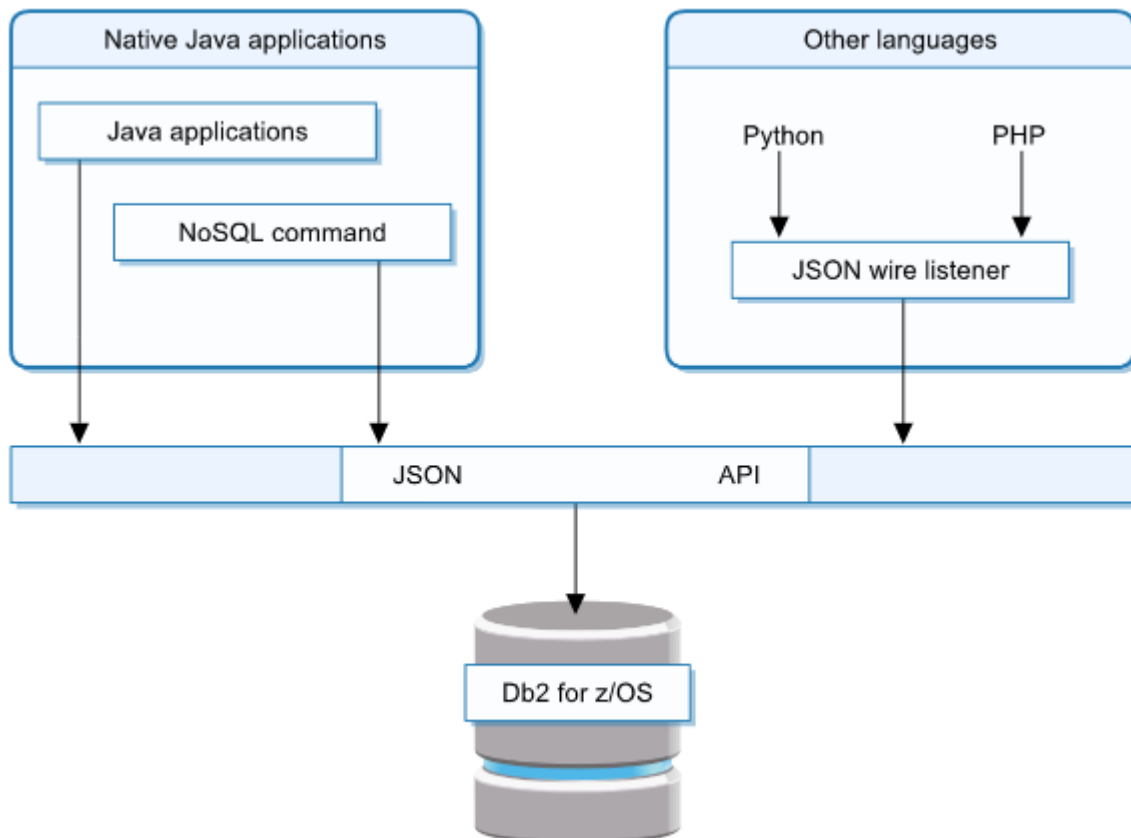
New web applications often use JSON for storing and exchanging information. With DB2 V10 for z/OS, document storage support for JSON was added, relying on DB2 NoSQL JSON APIs.

DB2 V11 for z/OS added two additional SQL interfaces, User Defined Functions:

- One to convert JSON to BSON
- The other is to convert BSON to JSON.

These allow SQL to be used to perform basic manipulation of JSON data. Db2 V12 for z/OS improves support for web and mobile applications with two ways of working with JSON: the Java driver for the JSON API or via SQL extensions.

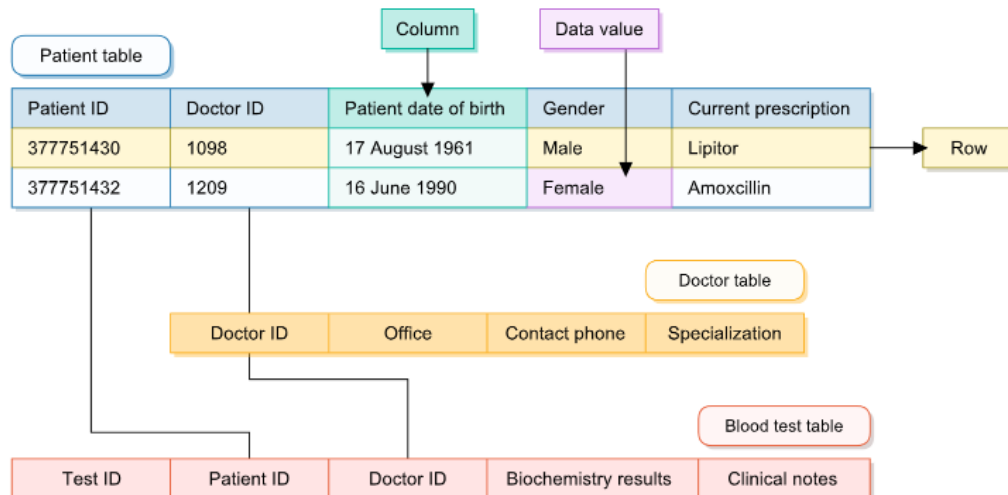
The example shown here identifies various methods in which JSON documents can be queried and managed.



## DB2 Data Components

## Tables

One of the most basic structures within DB2 is the table, which contains data your organization needs to store. A primary key is used when you need to create a referential integrity relationship with a foreign key in another table. If creating a primary key, it must have a corresponding index created, the process of which you will see shortly.



There are several types of tables that can be created from the commonly used base table, result tables, sample tables, temporary tables, history tables, auxiliary tables, and others.

## Table Spaces

All tables within DB2 are stored in table spaces, which is a storage structure that consists of **VSAM data** sets. Table spaces can be created by issuing a **CREATE TABLESPACE** command or by DB2 automatically when a table is created.

Types of table space that DB2 can work with include:

- Partition-by-growth universal table spaces (PBG UTS)
- Partition-by-range universal table spaces (PBR UTS)
- Segmented non-UTS table spaces (deprecated)
- Partitioned non-UTS table spaces (deprecated)
- Simple table spaces (deprecated)
- Large object table spaces
- XML table spaces

**Note:** As of DB2 12 for z/OS, non-UTS table spaces for base tables are deprecated and are likely to be unsupported in the future.



There are many types of table spaces that can be defined with the type you use being dependent on the type of data being stored and the amount of space available.

The diagram illustrates a database structure. At the top level is a 'Table space' container. Inside this container is a 'Doctor table'. The 'Doctor table' is represented as a table with four columns: 'Doctor ID', 'Office', 'Contact phone', and 'Specialization'. It contains three rows of data:

| Doctor ID | Office    | Contact phone  | Specialization |
|-----------|-----------|----------------|----------------|
| 1198      | Anaheim   | (562) 933-2000 | Dermatologist  |
| 1202      | San Diego | (619) 287-3270 | Cardiologist   |
| 1203      | Berkeley  | (510) 204-4444 | Anaesthetist   |

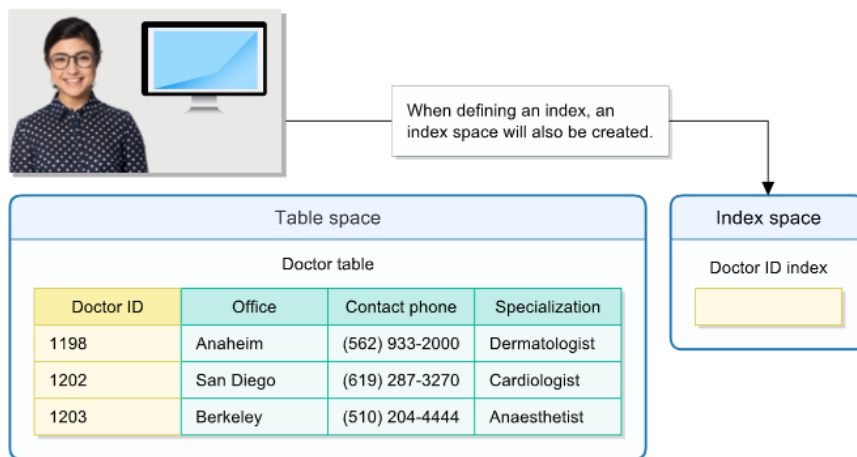
## Indexes

Just like an index for a book, DB2 indexes are used to quickly locate information without having to read all the content. In the example from the previous section, an index could be created for the Doctor ID column to locate a specific doctor's record.

The main benefit derived from using an index is performance, because not all data in the table needs to be accessed during a request.

## Index Spaces

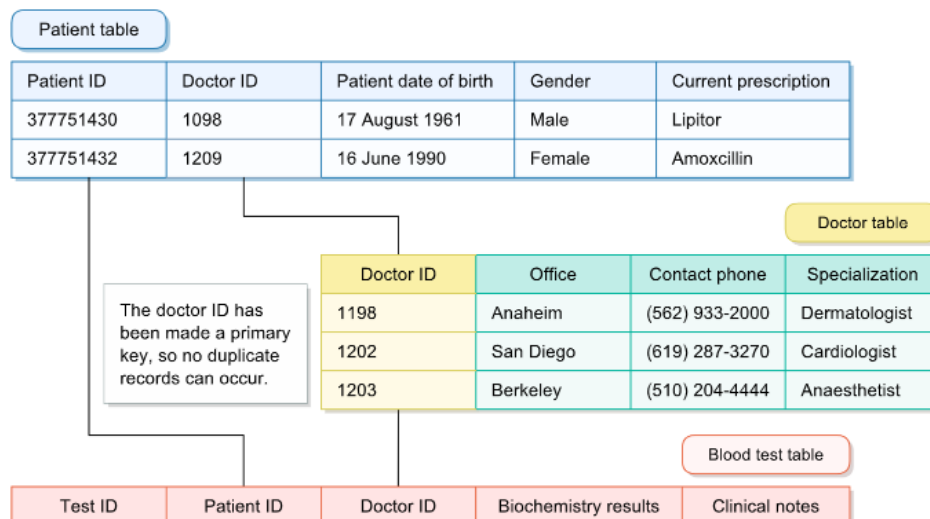
Just as tables have their own table space defined, so do indexes. A separate index space is created for every index.



## Keys

Tables can, and often do, contain columns that are defined as keys. These keys are used to ensure that each record in the table is unique, for example, that there are no duplicate doctor IDs, and that data from one table cannot be removed if it has related data in another table.

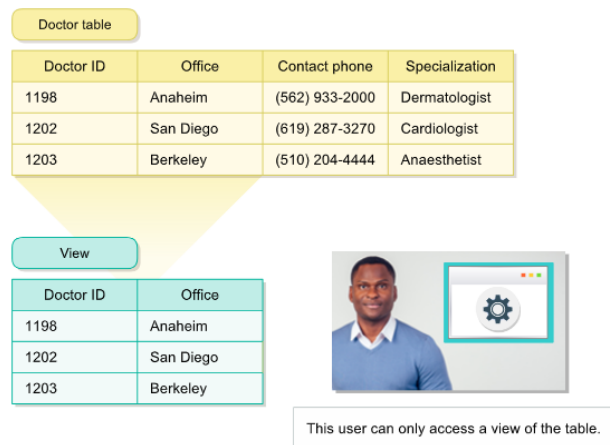
You will often see a primary key defined for a table, which is a unique key that is not allowed to contain null values. An index can be created for this type of key and is called a primary index. DB2 also supports a number of other keys including unique keys, parent keys, foreign keys, and composite keys.



## Views

There may be occasions where specific data from several tables is often requested, or users are not authorized to view certain data in a table. In these scenarios, views can be created which are subsets of existing data.

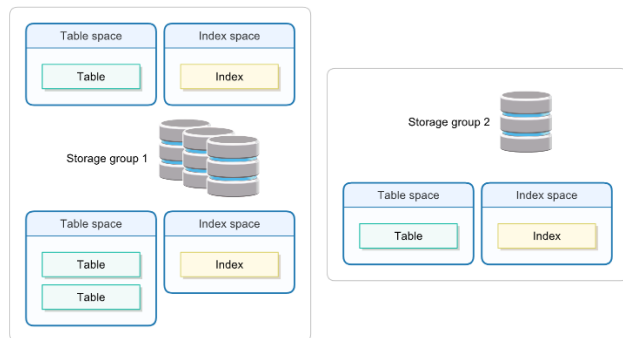
When a request for a new view is issued, only the definition supporting that view is stored. The data used to make up the view is already in existing tables, so creating separate data in views would result in much data duplication..



## Storage Groups

A storage group is a defined area on disk used to store tables and indexes. When these components are created you can specify the storage group whose space is to be utilized.

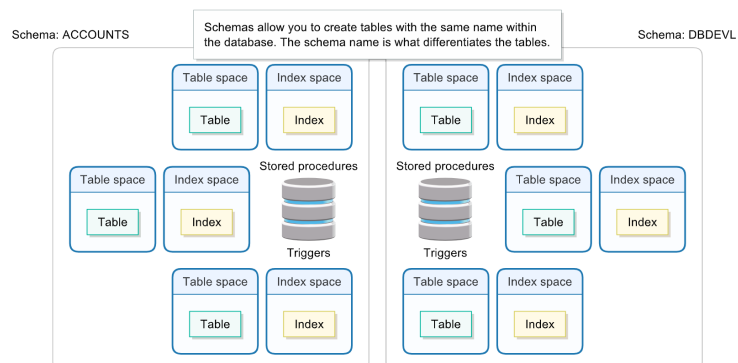
**Note:** These storage groups are not the same as the ones used by DFSMS.



## Schemas

A schema is used to group a set of objects within the Db2 database together using a high-level qualifier name. For example, if a schema called Accounts is created and there are two tables in the database, Cust and Invoice, that are linked to the schema, then the fully qualified names of the tables are Accounts.Cust and Accounts.Invoice.

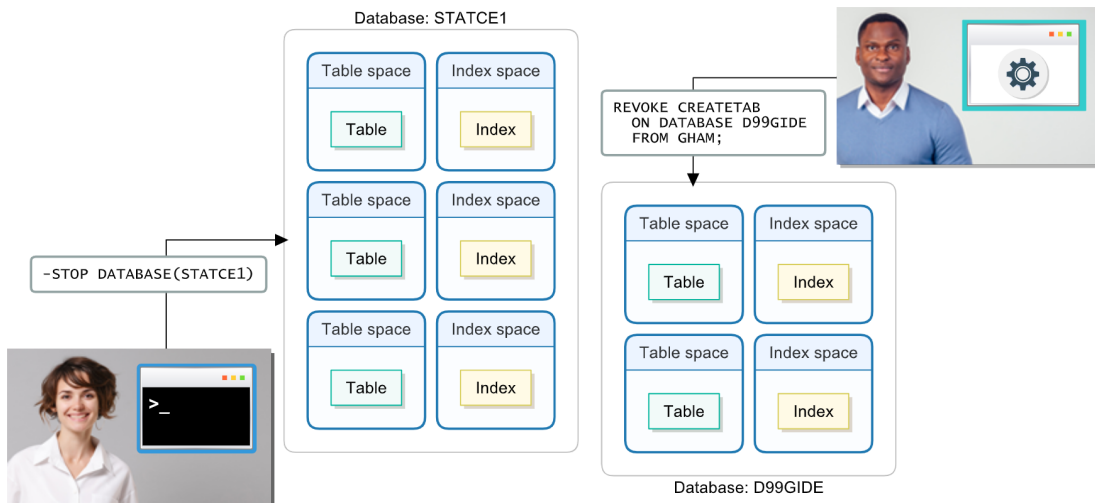
Schemas can also consist of indexes, table spaces (note that the schema for a table space is actually the database it belongs to), functions, stored procedures, and triggers. Whenever any of these elements is created, it is automatically assigned a high-level qualifier schema name.





## Databases

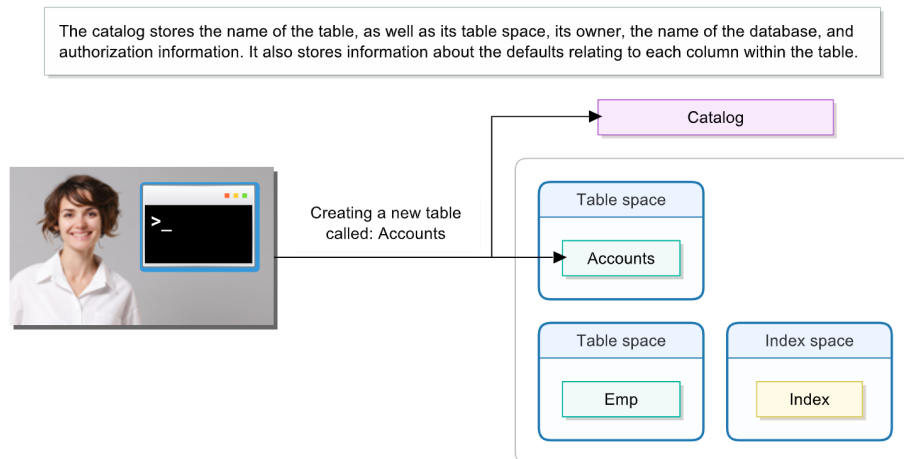
All of the data components you have looked at so far are linked together to form a database. Linking them like this enables you to control an entire application by entering commands to the database entity. For example, you may need to prevent access to all data relating to that database because there is a problem, or you may want to grant access to a user that needs to use the information in that database.



## DB2 System Components

### **DB2 - Catalog**

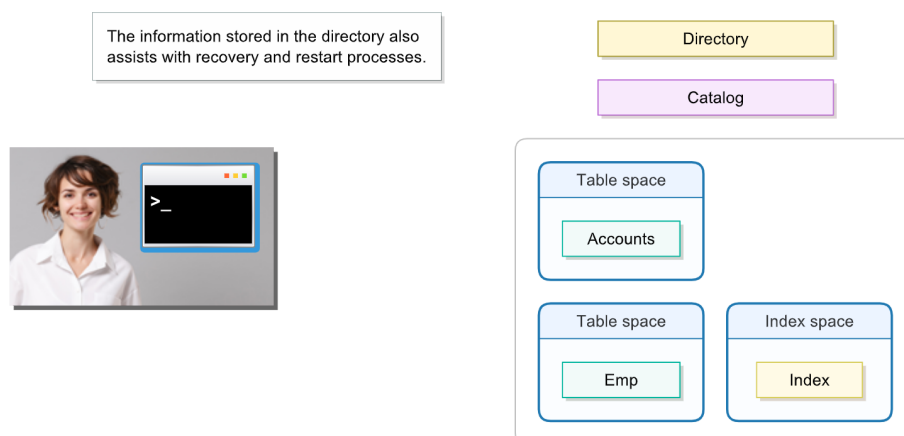
DB2 keeps track of objects such as tables, table spaces, indexes, index spaces, views, and storage groups within the DB2 database using a catalog. The catalog itself is a set of tables that contains details of when the objects just mentioned are created, modified, or deleted.



Because the catalog is a set of tables, it can be accessed using SQL.

### **DB2 - Directory**

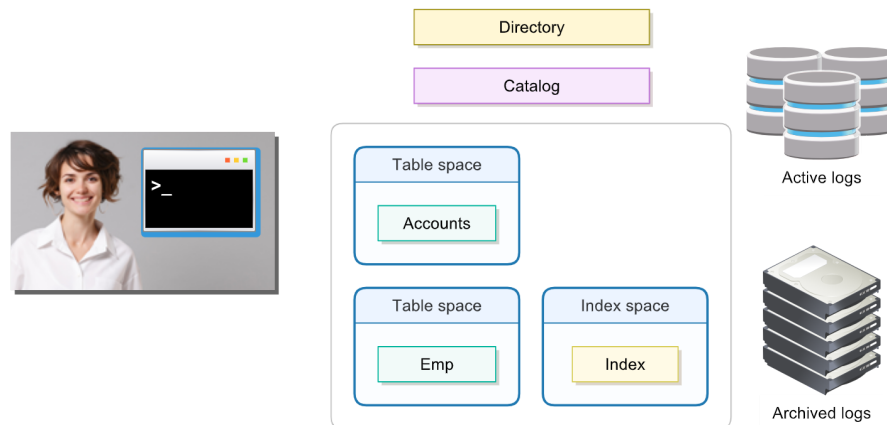
The DB2 directory consists of a number of directory tables stored in database DSNDB01. These tables contain system related information that is designed to be used internally by DB2, but a system administrator with appropriate authority can access data from them using SQL.



For example, details of active or stopped utility jobs can be obtained from the directory.

## **DB2 - Logs**

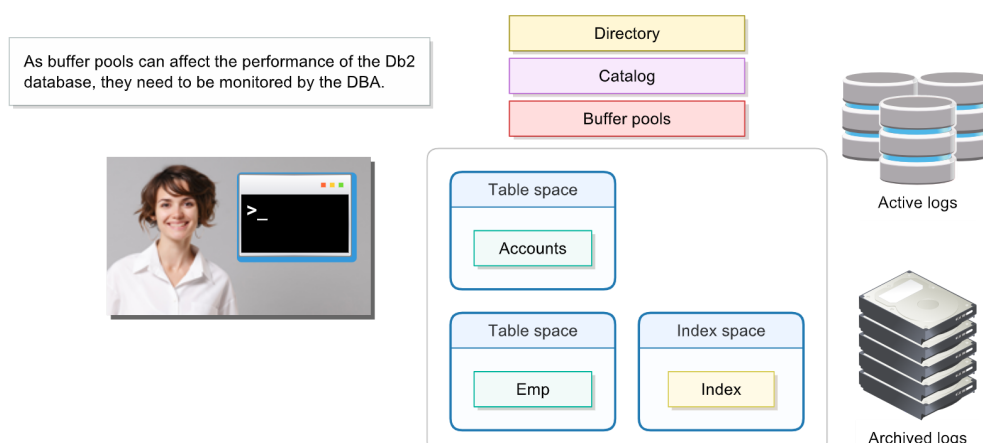
Like most products, DB2 stores information relating to data changes and major events in a log. During normal operation, an active log copy will have a number of data sets available for storing this data, and when one of these active log data sets becomes full, it is automatically backed up to tape or disk and is referred to as an archived log.



The logs can be browsed to help analyze problems and are also used in recovery scenarios to roll changes backward or forwards to a point where the database was known to be consistent.

## **DB2 - Buffer Pools**

When an application program needs to use Db2 table or index data, that data is accessed and stored in a temporary area called a buffer pool. Access to data in this buffer pool is much quicker than if the application program had to request it again from the table or index on DASD.



The data in the buffer pool can be read or modified and is copied back to the table or index following the completion of processing.

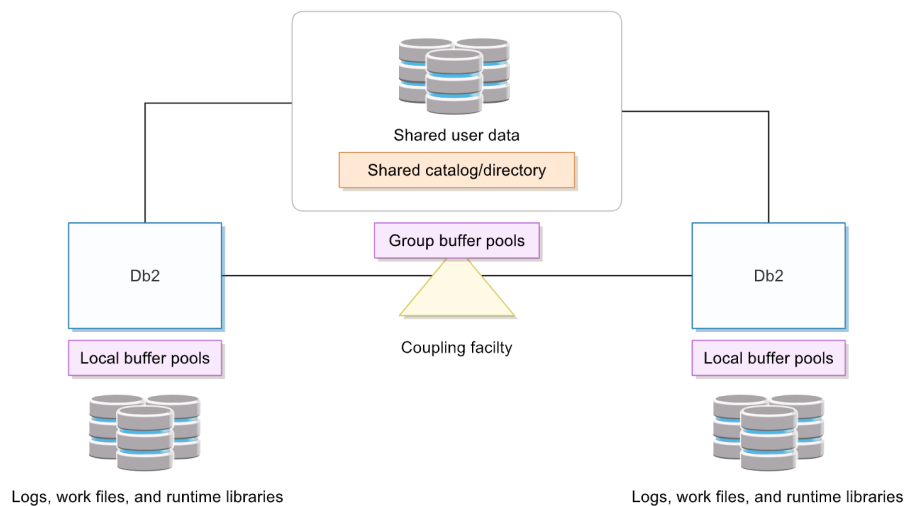
## **Databases**

Several related databases can be populated and used to provide support for DB2. For example:

- Data Definition Control Support Database - This database contains information that is used to prevent certain data definition language statements from being submitted.
- Resource Limit Facility - This consists of one or more tables that define the amount of processor resources that can be used by certain types of SQL statements.
- Work File Database - If SQL statements require additional space for processing, this database can be used.
- Communications Database - This database is used to communicate with remote database management systems.

## **DB2 and the Parallel Sysplex**

If an organization connects z/OS systems using a Parallel Sysplex then it will probably want to configure DB2 so that it can share data across the network.



The graphic displayed here shows how such a configuration can look and the data that is shared.