



## Db2 Internals

By proceeding with this courseware you agree with [these terms and conditions](#). Interskill Learning Pty. Ltd. © 2019





## Objectives

---

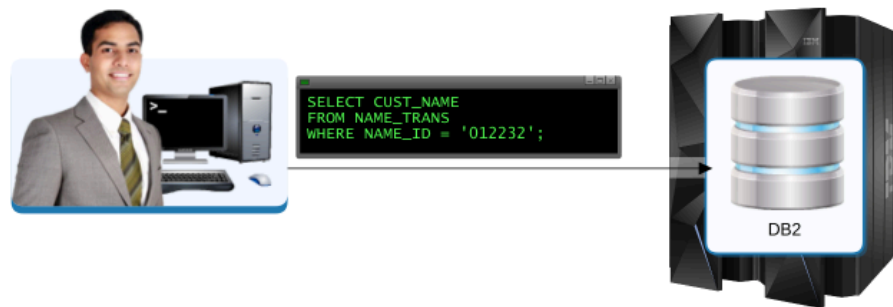
### Db2 Internals

In this module, you will discover how SQL accesses Db2 data and in more detail, look at the components that comprise Db2.

After completing this module, you will be able to:

- Identify How Db2 Data is Accessed
- Explain the Function of Db2 Data Structures
- Describe How Db2 System Components Are Utilized

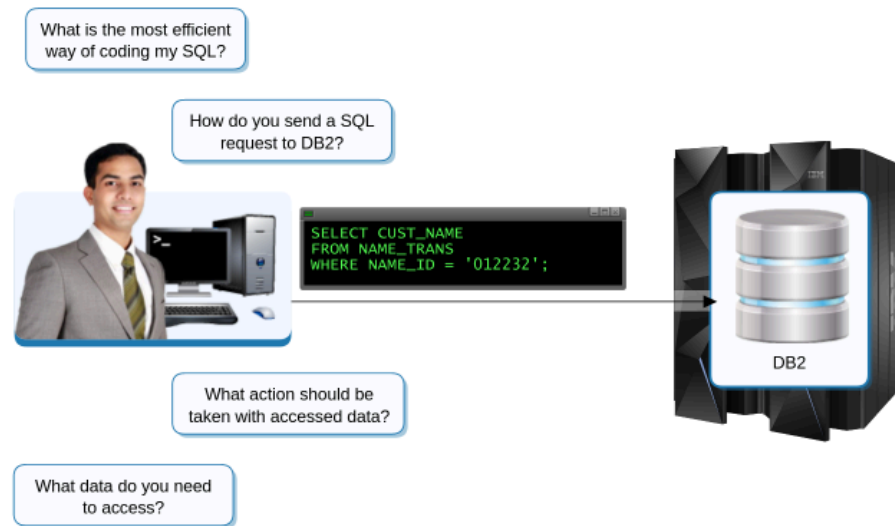




---

In the last module you discovered how Db2 could be configured to run on different systems including z/OS and were introduced to several interfaces that could be used to connect users to Db2 data.

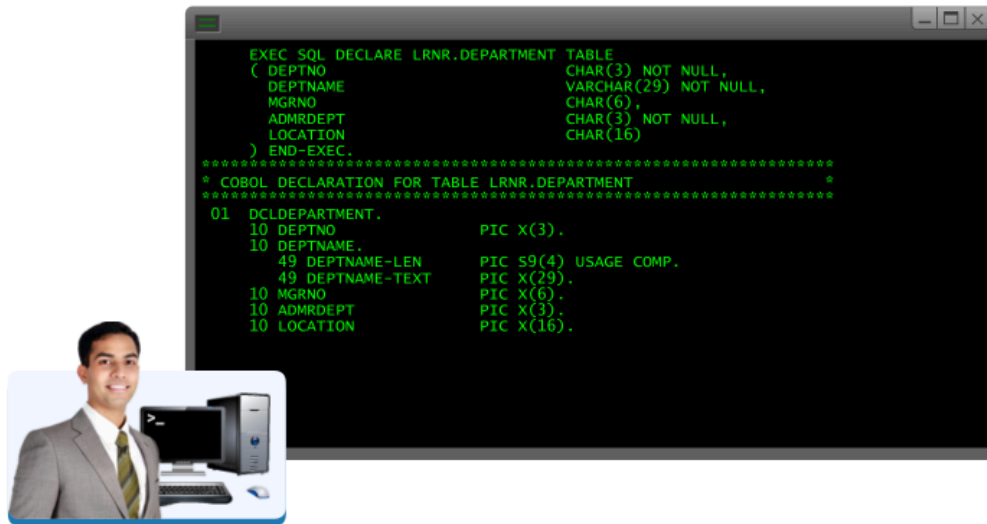
In this section you will look in more detail at the methods and products used by IT personnel to access Db2 data.



---

In the scenario shown here, an application programmer has coded some structured query statements to access specific Db2 data. SQL consists of over a 100 different statements that can be used to insert, query, update, delete and authorize access to Db2 data.


The intricacies of SQL are covered in the programming stream of Interskill courses so for now you will just look at how SQL is invoked.



```
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,
       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).
```

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 the END-EXEC indicates the end of the SQL stream. This coding standard is also required with C programs while Java uses #SQL to invoke SQL statements and REXX programs use EXEC SQL.



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

---

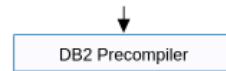
Rather than using static SQL statements as in the previous example, you can have SQL dynamically build your statements based on input provided to the application.

The example above shows that a PREPARE and EXECUTE statement 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.

```
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).
```



Source program with embedded SQL statements



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.

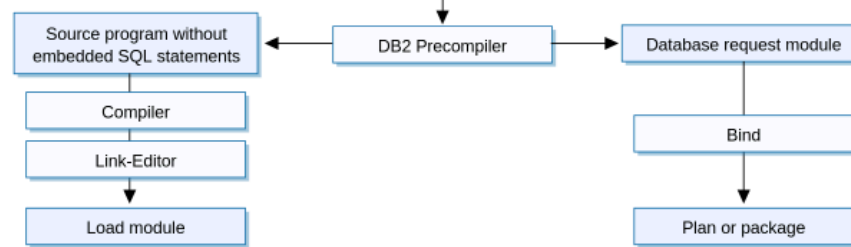
With static SQL statements, the Db2 precompiler or coprocessor checks the syntax of the SQL statements and if acceptable will convert them into host language statements that are used to access Db2 data.

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.

```
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).
```



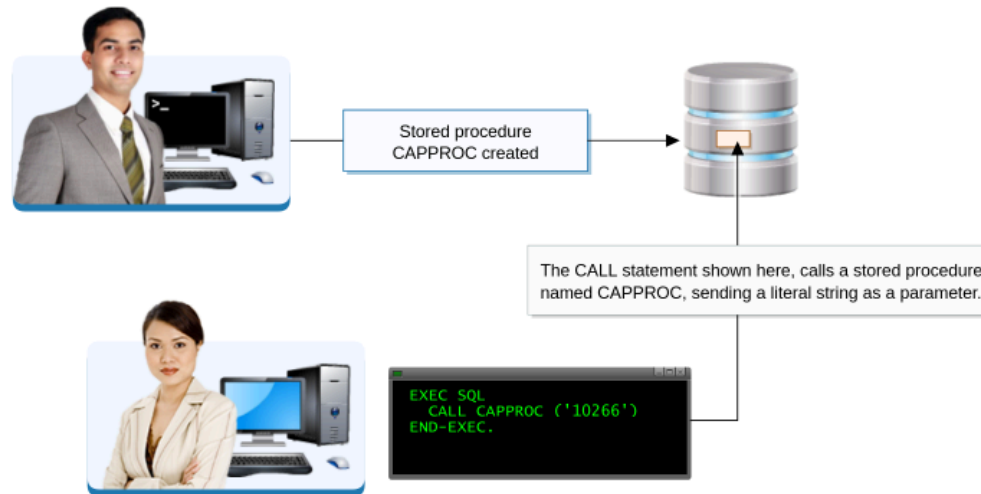
Source program with embedded SQL statements



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. Your Db2 application is now ready to run.





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 that:

- 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.

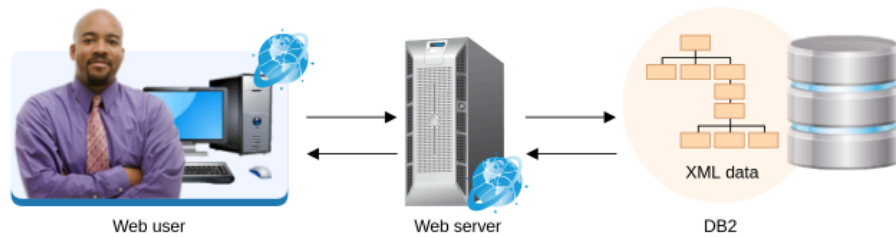
**Press Play** to view an example of this process.

```
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 the output data set name:       (Must be a 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.

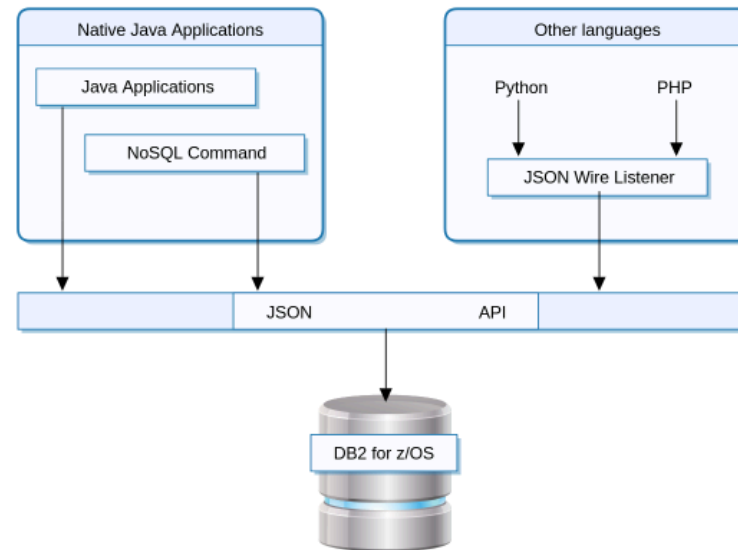
SQL statements can be sent to Db2 directly from a workstation. In the example shown here a TSO facility called SPUFI (SQL Processor Using File Input) allows you to execute SQL statements against an input file, without needing to embed them in an application program.



---

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.



Today's rapidly changing application environments has 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, and with Db2 version 11, these JSON documents can be stored in a Db2 database. The example shown here identifies various methods in which JSON documents can be queried and managed.

```
DSNB209I - DB2A DSNB1RTR - BUFFER MANAGER  
TABLESPACE/INDEXSPACE  
CLEANUP ROUTINE (DSNB1CFC)  
HAS BEEN SCHEDULED FOR  
DBNAME=GA7AGPBG  
SPACENAME=GA7A082A  
PSID=X'01400F8'
```



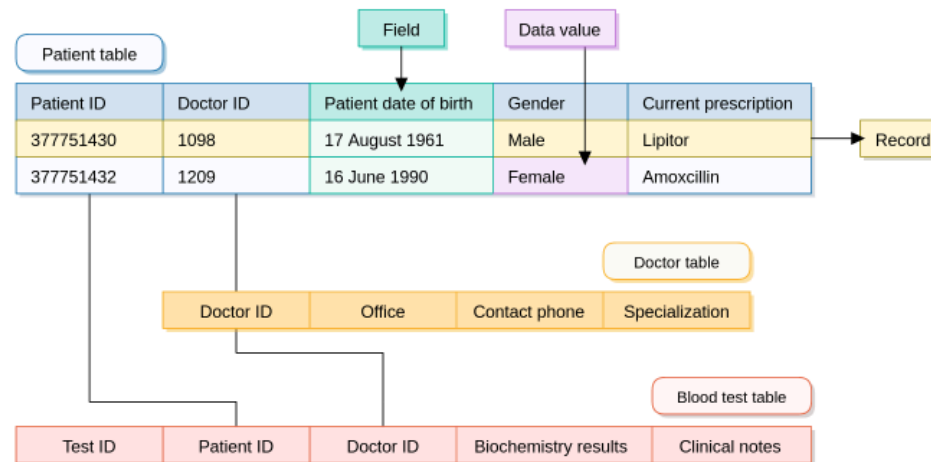
```
DSNU645I DB2X NO STORAGE GROUPS  
NAME FOUND
```

```
DSNL703I CDB TABLE SYSTA1.LUNAMES DOES NOT  
HAVE PROPER INDEX DEFINITIONS,  
INDEX SYSTA1.XLUNAMES IS MISSING OR  
INCORRECTLY DEFINED
```

```
DSNG007I - DSNB DB2 CATALOG LEVEL (A10)  
CODE LEVEL (A10) MODE (N)
```

---

In this section you will view the function of data components that comprise a Db2 database. This will provide you with an understanding on possible action you may need to take should messages indicate a problem or you need to create or re-configure a database.



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. This is discussed in detail in the course that follows this. 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, to result tables, sample tables, temporary tables, history tables and auxiliary tables.

Types of table spaces that can be created are:

- Universal table spaces
- Partitioned table spaces
- Segmented table spaces
- Large object table spaces
- Simple table spaces (these are backwardly supported but cannot be created using DB2 9.1 and above)
- XML table spaces



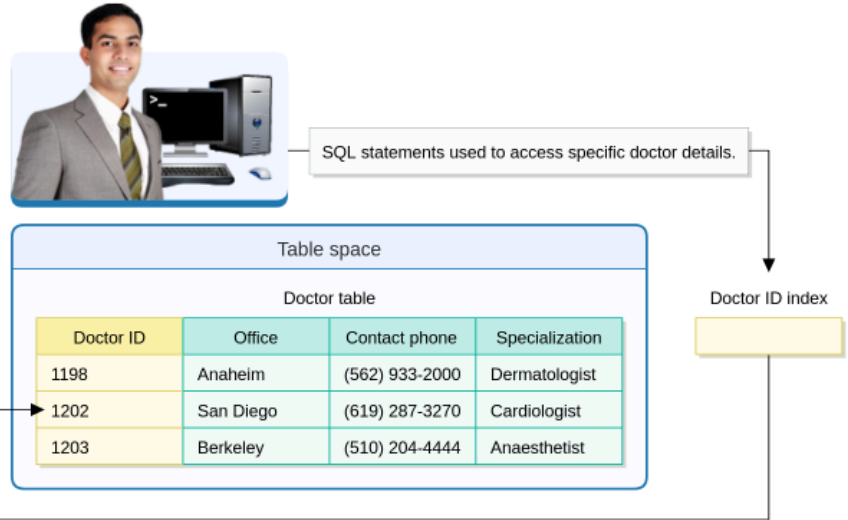
Table space

Doctor table

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

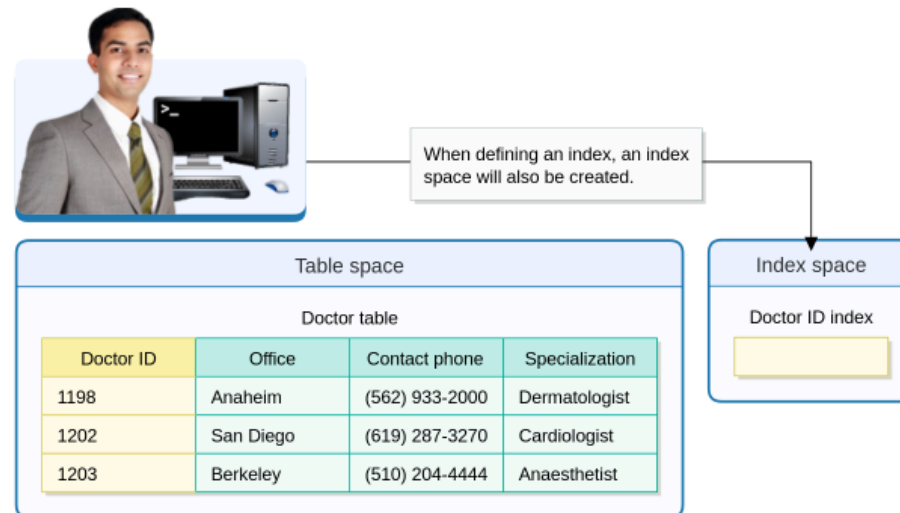
There are many types of table spaces that can be defined with the type you use being dependent on the type data being stored and the amount of space available.

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.



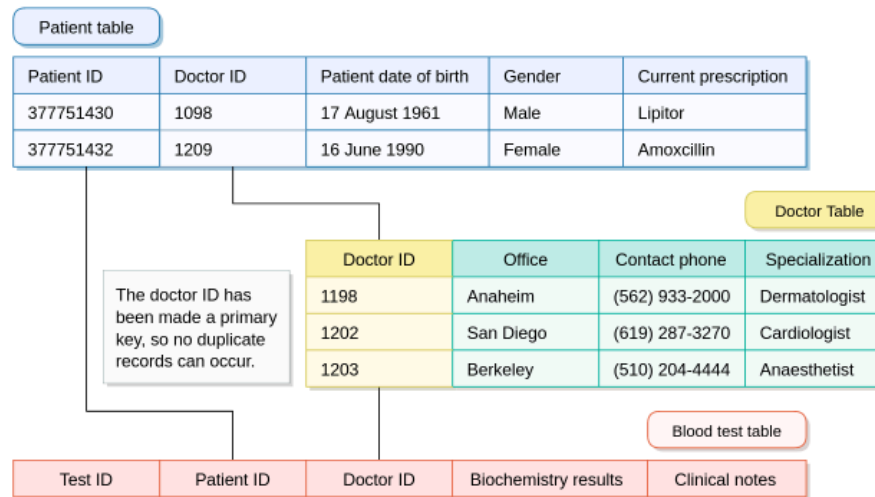
Just like an index for a book, Db2 indexes are used to quickly locate information without having to read all the content. In the example from the previous page, an index could be created for the Doctor ID column to locate a specific doctor's record. The main benefit derived from using an index is performance, because not all data in the table needs to be accessed during a request.





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





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.



DBA has access to all data within this table.

Doctor Table

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

View

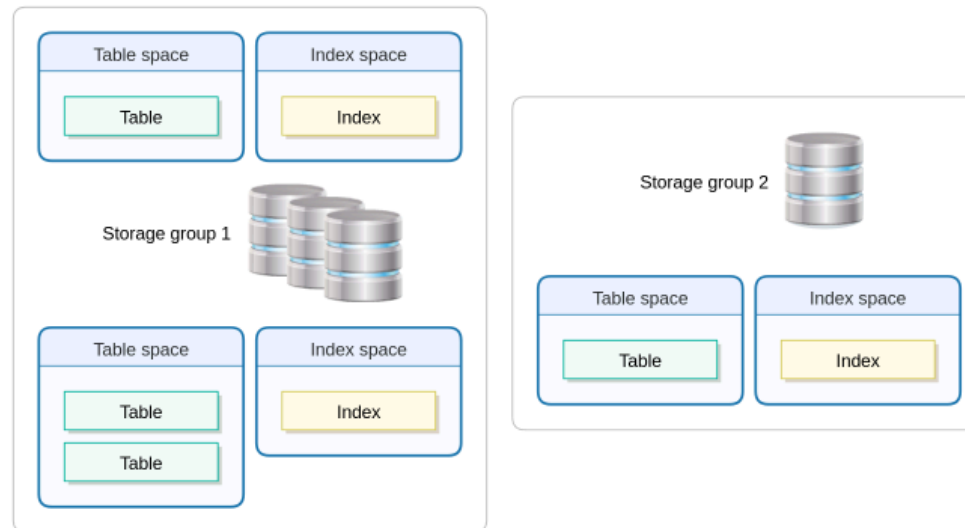
Doctor ID	Office
1198	Anaheim
1202	San Diego
1203	Berkeley



This user can only access a view of the table.

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.

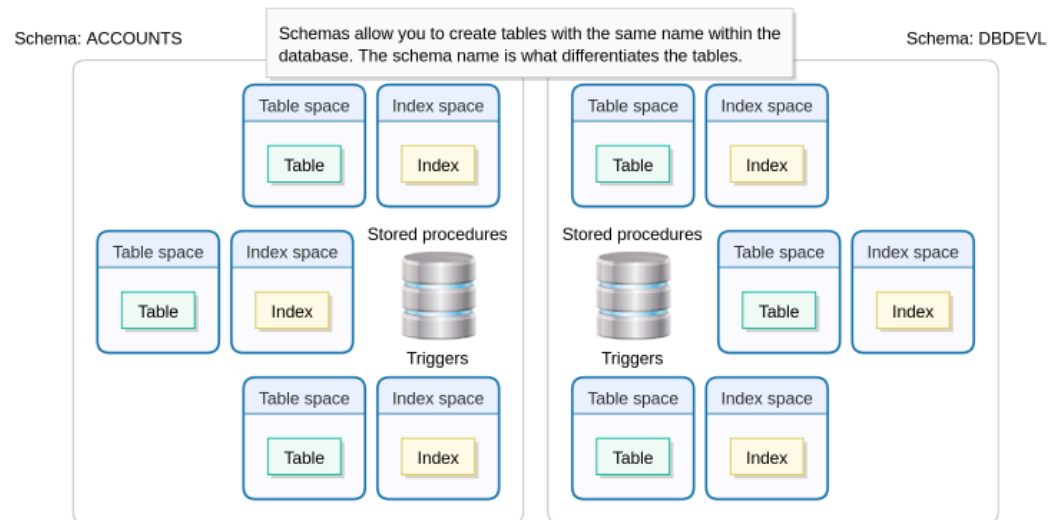
When a request for a new view is issued, only the definition supporting that view is stored. The data used to make up the view is already in existing tables, so creating separate data in views would result in much data duplication.



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.

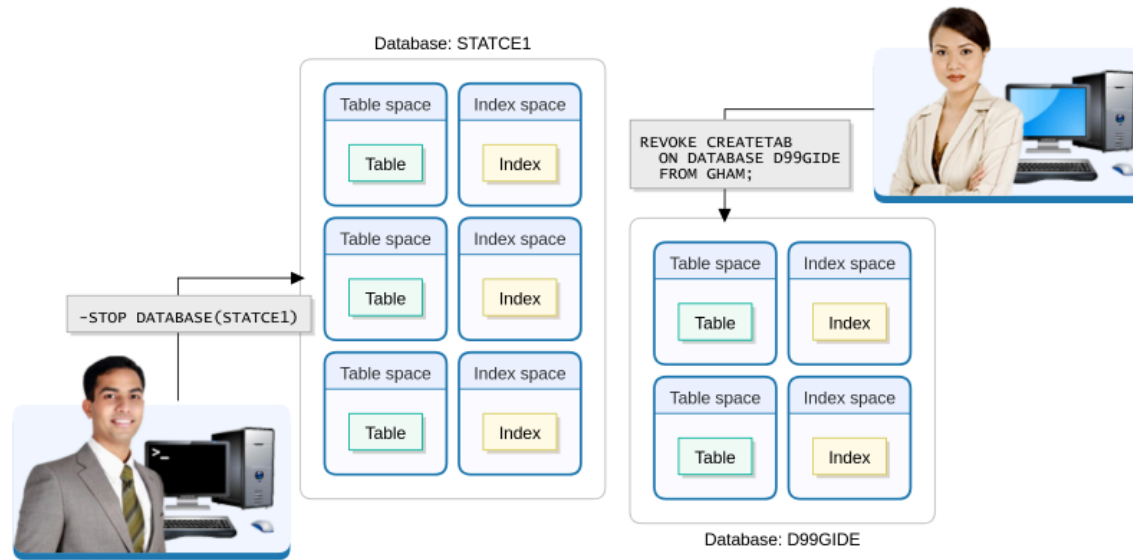
Note that these storage groups are not the same as the ones used by DFSMS.



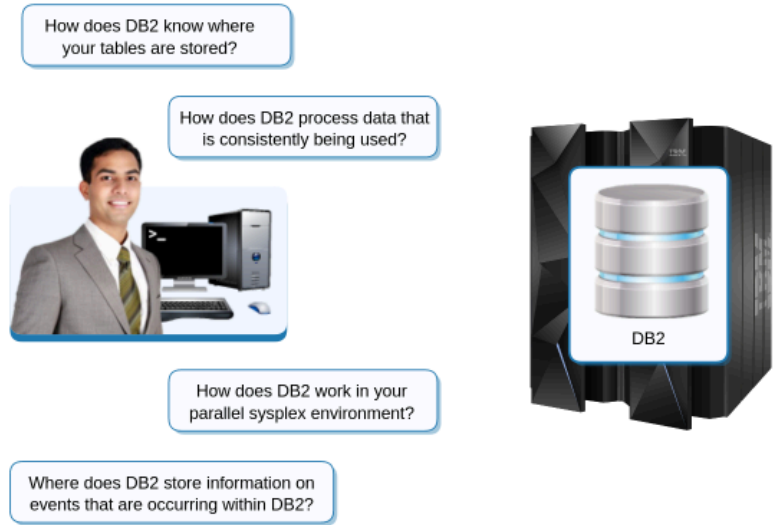


A schema is used to group a set of objects within the Db2 database together using a high level qualifier name. For example, if a schema called 'Accounts' is created and there are two tables in the database, Cust and Invoice, that are linked to the schema, then the fully qualified name of the tables is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

Schemas can also consist of indexes, table spaces, functions, stored procedures, and triggers. Whenever any of these elements is created it is automatically assigned a high level qualifier schema name.



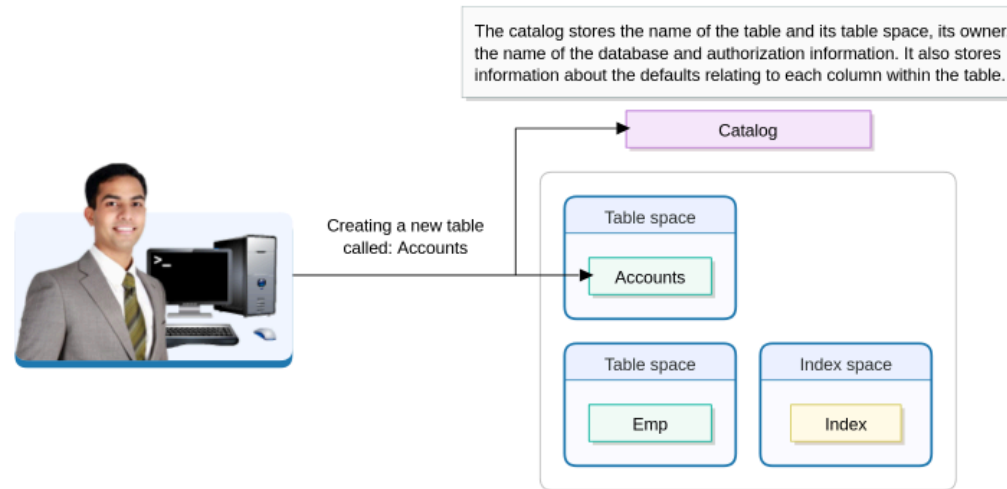
All of the data components you have looked at so far are linked together to form a database. Linking them like this enables you to control an entire application by entering commands to the database entity. For example you may need to prevent access to all data relating to that database because there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



---

The components discussed in the previous section were related to the storing of user data, with many aspects being able to be controlled using Db2 commands.

In this section you will look at the Db2 system components that are used to ensure database integrity and recoverability. Like the previous section, understanding these structures will assist you if you need to analyze and resolve Db2 error or warning messages.



---

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 when the objects mentioned above are created, modified or deleted.

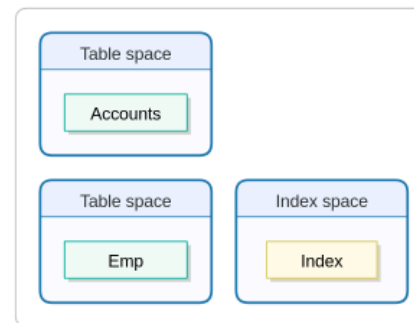
Because the catalog is a set of tables, it can be accessed using SQL.



The information stored in the directory also assists with recovery and restart processes.

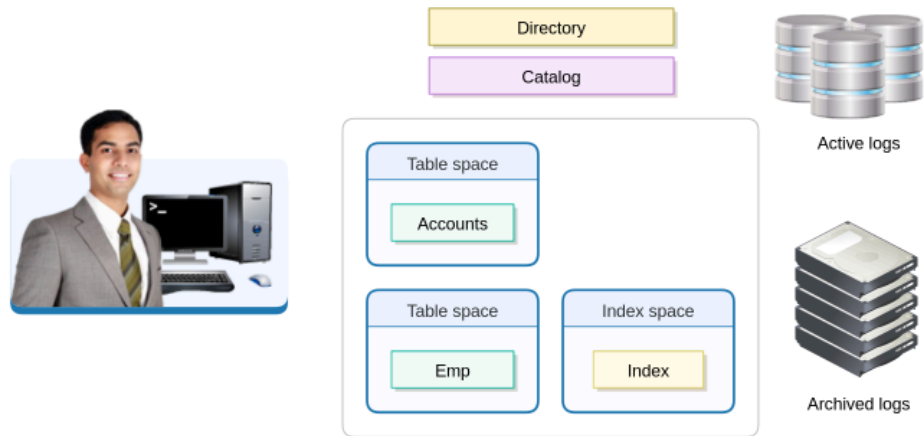
Directory

Catalog



---

The Db2 Directory consists of a number of directory tables stored in database DSND01. 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.

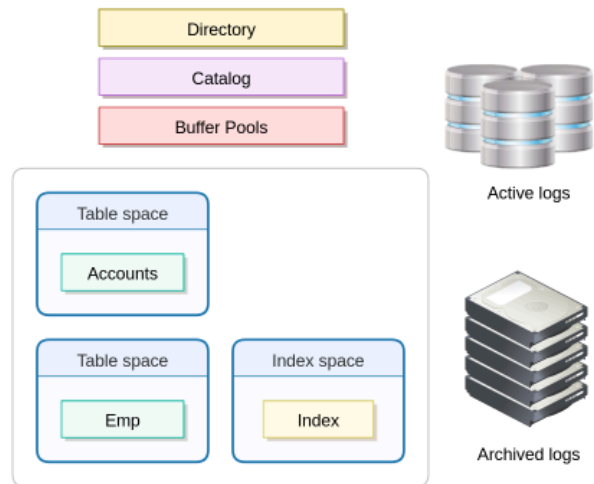


---

Like most products, Db2 stores information relating to data changes and major events in a log. During normal operation, a number of active logs are available for storing this data and when one of these active logs 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 backwards or forwards to a point where the database was known to be consistent.

As buffer pools can affect the performance of the DB2 database, they need to be monitored by the DBA.



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.

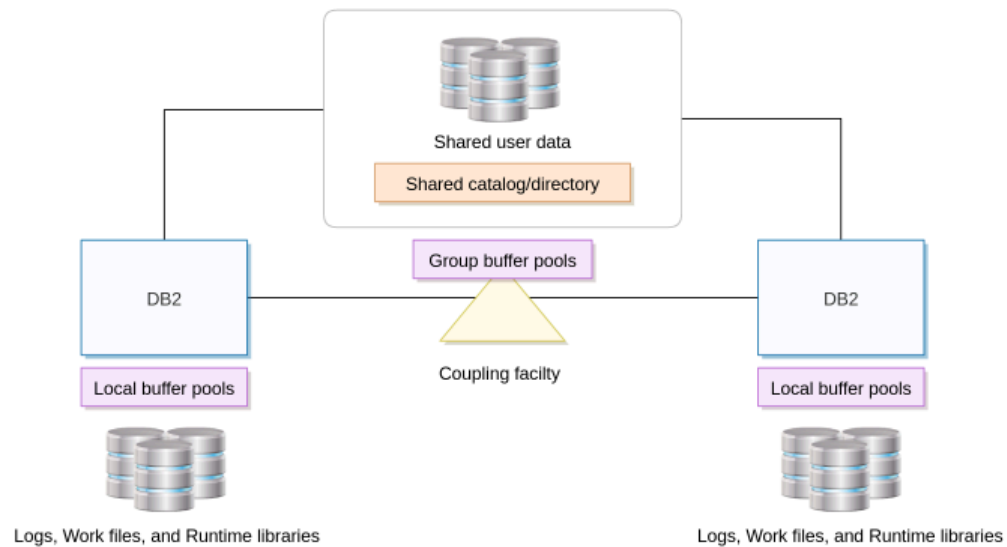
The data in the buffer pool can be read or modified and is copied back to the table or index following the completion of processing.



The resource limit facility consists of one or more tables that define the amount of processor resources that can be used by certain types of SQL statements.

Several related databases can be populated and used to provide support for Db2.

**Mouse-over** the databases for a description of their purpose.



If your organization connects z/OS systems using a Parallel Sysplex then you 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.



## Summary

---

### Db2 Internals

In this module, you discovered how SQL accesses Db2 data and looked at the data and system components that comprise Db2.

You should now be able to:

- Identify How Db2 Data is Accessed
- Explain the Function of Db2 Data Structures
- Describe How Db2 System Components Are Utilized





Question 5 of 12 - Accessing Db2 Data

That is incorrect

The source program and its SQL statements is put through the Db2 precompiler or compressor which creates a DBRM. The BIND command is then used to create a plan or package from this information. Your program meanwhile goes through the traditional compile and link-edit to produce a load module that contains a reference to the package or plan.

**Mouse-over** or **touch** the incorrect answers to view the correct options.

