

Databases, SQL & ETL Understanding

K Kiran Kumar, INSOFE

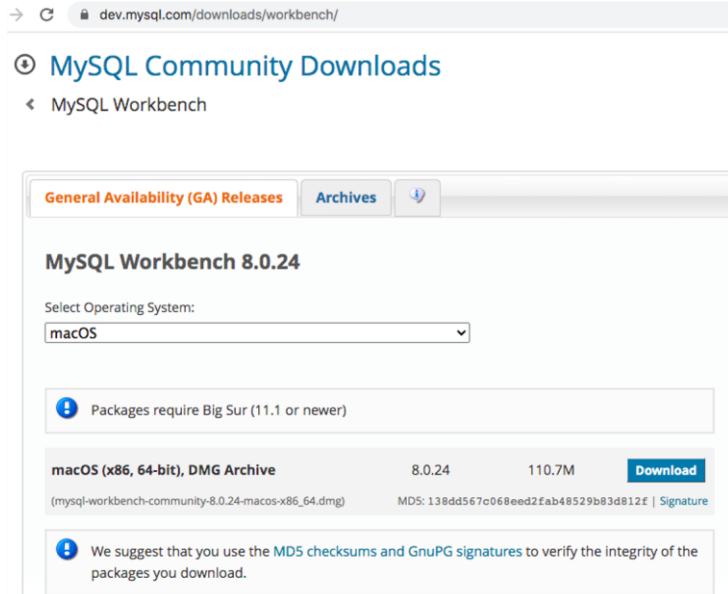
Learning Objective

At the end of this module, you will be introduced to the following concepts.

- Intro to SQL, Types and Operations
- SQL – CRUD Commands

Software Installation

Please follow the below steps to install MySQL workbench, which we will use for all of our lab activities.



The screenshot shows a web browser window with the address bar displaying `dev.mysql.com/downloads/workbench/`. The page title is "MySQL Community Downloads" and the breadcrumb is "MySQL Workbench". The main content area shows "MySQL Workbench 8.0.24" with a dropdown menu for "Select Operating System" set to "macOS". A note states "Packages require Big Sur (11.1 or newer)". Below this, a table lists the download options for macOS (x86, 64-bit), DMG Archive, version 8.0.24, with a size of 110.7M. A "Download" button is present. The table also includes the MD5 checksum and a link to the signature. A final note suggests using MD5 checksums and GnuPG signatures to verify the integrity of the packages.

Operating System	Version	Size	Action
macOS (x86, 64-bit), DMG Archive	8.0.24	110.7M	Download

(mysql-workbench-community-8.0.24-macos-x86_64.dmg) MD5: 138dd567c068eed2fab48529b83d812f | [Signature](#)

Please go to the site :

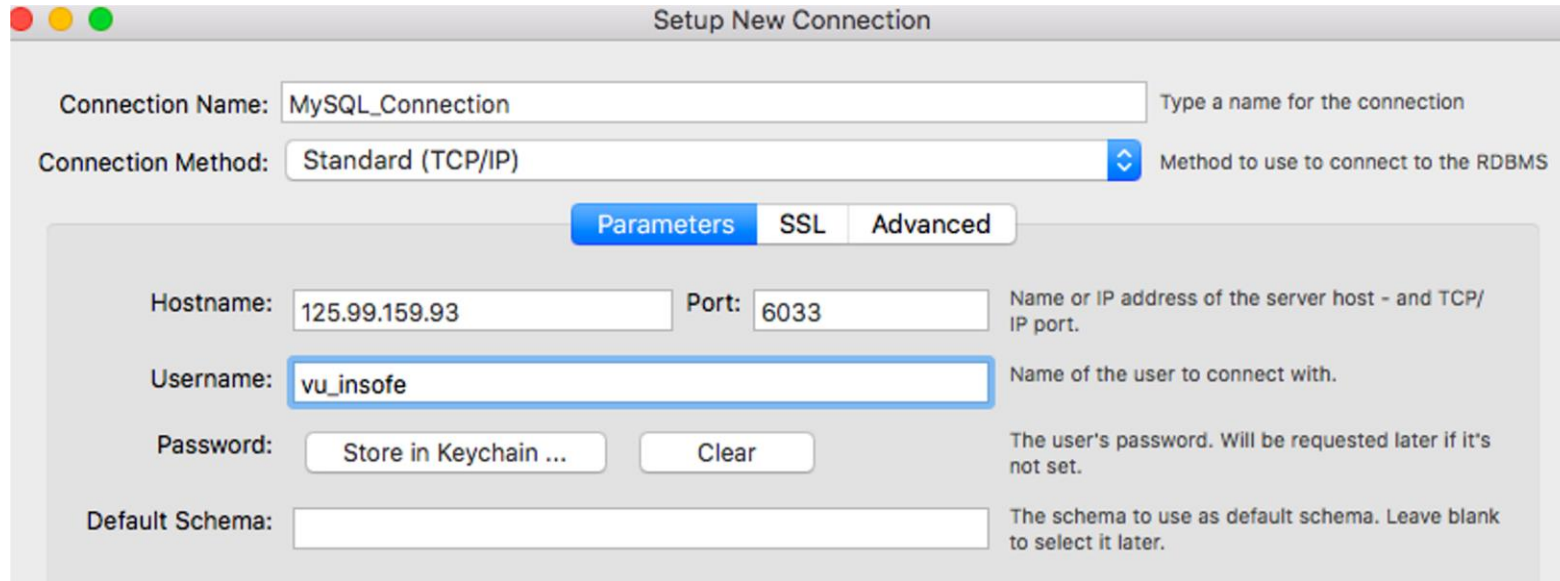
<https://dev.mysql.com/downloads/workbench/>

Download the version of the software as per your system configuration.

Connect to MySQL server.

Open MySQL workbench once you have installed it.

Use the option “MySQL Connections”, and enter the details as shown in the below image. Username and password will be same for all of you. Username - vu_insofe and Password Vu_1ns0f3_987



The screenshot shows the 'Setup New Connection' dialog box in MySQL Workbench. The dialog has a title bar with three colored buttons (red, yellow, green) on the left. The main content area is divided into sections. The first section contains 'Connection Name' (MySQL_Connection) and 'Connection Method' (Standard (TCP/IP)). Below this is a tabbed interface with 'Parameters', 'SSL', and 'Advanced' tabs. The 'Parameters' tab is active, showing fields for 'Hostname' (125.99.159.93), 'Port' (6033), 'Username' (vu_insofe), 'Password' (with 'Store in Keychain ...' and 'Clear' buttons), and 'Default Schema'. Each field has a descriptive text to its right.

Field	Value	Description
Connection Name	MySQL_Connection	Type a name for the connection
Connection Method	Standard (TCP/IP)	Method to use to connect to the RDBMS
Hostname	125.99.159.93	Name or IP address of the server host - and TCP/IP port.
Port	6033	
Username	vu_insofe	Name of the user to connect with.
Password	[Buttons: Store in Keychain ..., Clear]	The user's password. Will be requested later if it's not set.
Default Schema		The schema to use as default schema. Leave blank to select it later.

- SQL stands for Structured Query Language and a standard language for accessing and manipulating databases.
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.
- **A Powerful** language that performs the functions of data manipulation(**DML**), data definition(**DDL**) and data control or data authorization(**DAL/DCL**).
- **A Non procedural language** - the capability to act on a set of data and the lack of need to know how to retrieve it. An SQL can perform the functions of more than a procedure.
- The **De Facto** Standard query language for RDBMS
- **Very flexible.**

- SQL can be used to
 - ✓ Execute queries against a database.
 - ✓ Retrieve data from a database.
 - ✓ Insert, update/modify data into a database.
 - ✓ Create new databases.
 - ✓ Create new tables in a database.
 - ✓ Create views in a database.
 - ✓ Set access permissions on tables, databases and views.

- Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

- However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in similar manner.

- Also Most of the SQL database programs such as Oracle, MySQL, IBM DB2 and Microsoft SQL Server also have their own proprietary extensions in addition to the SQL standard!

- Data Definition Language (**DDL**):
 - ✓ To define the database objects such as databases, tables, views.
 - ✓ Commands include Create, Alter and Drop.

- Data Manipulation Language (**DML**):
 - ✓ To access and modify the data from the database objects.
 - ✓ Commands include Select, Insert, Update and Delete.

- Data Control Language (**DCL**):
 - ✓ To Secure the database objects.
 - ✓ Commands include Grant and Revoke.

SQL Operations:

- The following are the operations that can be performed by a SQL on the database tables.
 - ✓ Select
 - ✓ Project
 - ✓ Union
 - ✓ Intersection
 - ✓ Difference
 - ✓ Join
 - ✓ Divide

SQL – Operations Example:

```
SELECT * FROM employees LIMIT 10; ** Select the first 10 rows/records **
```

- Every table is broken up into smaller entities called fields. The fields in the `employees` table consist of `emp_no`, `birth_date`, `first_name`, `last_name`, `gender` and `hire_date`.
- A field is a column in a table that is designed to maintain specific information about every record in the table.
- A record, also called a row, is each individual entry that exists in a table. For example, there are 300024 records in the above `employees` table. A record is a horizontal entity in a table.
- A column is a vertical entity in a table that contains all information associated with a specific field in a table.

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24

Data Definition Language:

CREATE

This statement is used to create objects.

Syntax: For Creating a Table

```
CREATE TABLE <table name> (Column(s) Definitions)
PRIMARY KEY(Column(s)) / FOREIGN KEY(Column(s))
(referential constraints)
```

reference_option: RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

Referential constraints helps in enforcing the rules involved when there are dependencies between relations.

```
CREATE TABLE employees (
  emp_no int(11)
  NOT NULL,
  birth_date date
  NOT NULL,
  first_name
  varchar(14) NOT
  NULL,
  last_name
```

➤ Referential Integrity

- ✓ Foreign Key references database_name.table on 'relation condition for delete'
- ✓ Table1 references table2(target) - Table2's Primary key is the foreign key defined in Table1
- ✓ The Conditions that can be used are CASCADE, RESTRICT, SET NULL, NO ACTION, SET DEFAULT (referential constraint for the foreign key definition)
- ✓ Inserting (or updating) rows in the target is allowed only if there are no rows in the referencing table

Referential Integrity Example:

```
CREATE TABLE employees (  
  emp_no int(11) NOT NULL,  
  birth_date date NOT NULL,  
  first_name varchar(14) NOT NULL,  
  last_name varchar(16) NOT NULL,  
  gender enum('M','F') NOT NULL,  
  hire_date date NOT NULL,  
  PRIMARY KEY (emp_no)  
);
```

```
CREATE TABLE departments (  
  dept_no char(4) NOT NULL,  
  dept_name varchar(40) NOT NULL,  
  PRIMARY KEY (dept_no),  
  UNIQUE KEY dept_name (dept_name)  
);
```

```
CREATE TABLE dept_emp (  
  emp_no int(11) NOT NULL,  
  dept_no char(4) NOT NULL,  
  from_date date NOT NULL,  
  to_date date NOT NULL,  
  PRIMARY KEY (emp_no,dept_no),  
  KEY dept_no (dept_no),  
  CONSTRAINT dept_emp_ibfk_1  
  FOREIGN KEY (emp_no)  
  REFERENCES employees (emp_no)  
  ON DELETE CASCADE,  
  CONSTRAINT dept_emp_ibfk_2  
  FOREIGN KEY (dept_no)  
  REFERENCES departments (dept_no)  
  ON DELETE CASCADE  
);
```

The above referential integrity definition for the table dept_emp, ensures the deletion of the corresponding rows

1. An employee is deleted from the employees table. or
2. A department is deleted from the departments table.

Data Definition Language:

ALTER

This statement is used for altering all database objects.

Syntax: For altering a Table

```
ALTER TABLE <table name>  
ADD Column Data-type [ not null with default]
```

- Alter allows primary & Foreign key specifications to be changed.
- It does not support changes to width or data type of a column or dropping a column.

```
ALTER TABLE employees RENAME birth_date TO date_of_birth;
```

Data Definition Language:

DROP

This statement is used for dropping all database objects.

Syntax: For dropping a table

DROP TABLE <table name>

```
DROP TABLE employees;
```

SQL - Selection & Projection (DML):

- Select retrieves a specific number of rows from a table.
- Projection operation retrieves a specified subset of columns(but all rows) from the table.

`SELECT Column_1, Column_2 FROM <table_name>`

- The WHERE clause defines the Predicates for the SQL operation.
- The above WHERE clause can have multiple conditions using AND & OR .

Example:

```
SELECT emp_no, first_name, last_name
FROM employees
WHERE hire_date > '1986-01-01'
AND gender = 'F';
```

Other Clauses:

- Many other clauses can be used in conjunction with the **WHERE** clause to code the required predicate,
Some are :-
 - ✓ **Between / Not Between**
 - ✓ **In / Not In**
 - ✓ **Like / Not Like**
 - ✓ **IS NULL / IS NOT NULL**

SQL - Selection & Projection (DML) (contd...):

➤ **SELECT** using a range

✓ **BETWEEN** Clause

Example:

```
SELECT emp_no, first_name, last_name  
FROM employees  
WHERE emp_no BETWEEN 10000 AND 20000;
```

✓ **IN** Clause

Example:

```
SELECT emp_no, first_name, last_name  
FROM employees  
WHERE birth_date IN ('1952-04-19', '1963-06-01');
```

SQL - Selection & Projection (DML) (contd...):

➤ LIKE clause

Example:

- ✓ To find names beginning with john:

```
SELECT * FROM employees WHERE first_name LIKE 'john%';
```

- ✓ To find names ending with mary:

```
SELECT * FROM employees WHERE name LIKE '%mary';
```

- ✓ To find names containing a phil:

```
SELECT * FROM employees WHERE name LIKE '%phil%';
```

- ✓ To find names containing exactly five characters, use instances of the _pattern character:

```
SELECT * FROM employees WHERE name LIKE '_____';
```

- Note :- '_' for a single char ; '%' for a string of chars Escape '\' - escape char;
If precedes '_' or '%' overrides their meaning.

SQL - Selection & Projection (DML) (contd...):

➤ Working with NULL Values

- ✓ The **NULL** value can be surprising until you get used to it. Conceptually, NULL means “a missing unknown value” and it is treated somewhat differently from other values.
- ✓ To test for **NULL**, use the **IS NULL** and **IS NOT NULL** operators:
`SELECT 1 IS NULL, 1 IS NOT NULL;`
- ✓ One cannot use arithmetic comparison operators such as =, <, or <> to test for **NULL**. To demonstrate this for yourself, try the following query:
- ✓ Because the result of any arithmetic comparison with NULL is also NULL, you cannot obtain any meaningful results from such comparisons.

`SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;`

This returns **NULL**.

Order By and Group By clauses (DML) :

- Order by sorts retrieved data in the specified order; uses the WHERE clause.
- Group by operator causes the table represented by the FROM clause to be rearranged into groups, such that within one group all rows have the same value for the Group by column (not physically in the database). The Select clause is applied to the grouped data and not to the original table.
- Here 'HAVING' is used to eliminate groups, just like WHERE is used for rows.

Example:

```
SELECT ORDER_ID, SUM(ITEM_COUNTS)
FROM ORDER
GROUP BY ORDER_ID
HAVING AVG(ITEM_COUNTS) < 100
ORDER BY ORDER_ID;
```

Functions:

- Two Types of functions.
 - ✓ Column Functions:
 - Compute from a group of rows aggregate value for a specified column(s)
 - ✓ Scalar Functions:
 - Are applied to a column or expression and operate on a single value.

Functions (contd...):

➤ Column Functions Examples.

- ✓ AVG –
 - Returns Avg value of all the values of a given column.
- ✓ SUM –
 - Returns Sum/Total of all the values of a given column.
- ✓ COUNT –
 - Returns Total count of all the values of a given column.
- ✓ MIN –
 - Returns Minimum value from all the values of a given column.
- ✓ MAX –
 - Returns Maximun value from all the values of a given column.

Functions (contd...):

- Scalar Functions Examples.
 - ✓ LENGTH : Returns the length of the specified string (in bytes).
 - ✓ REPLACE :Replaces all occurrences of a specified string.
 - ✓ SUBSTR :Extracts a substring from a string (starting at any position).
 - ✓ SUBSTRING : Extracts a substring from a string (starting at any position).
 - ✓ TRIM : Removes leading and trailing spaces from a string.
 - ✓ DATEDIFF : Returns the difference in days between two date values.
 - ✓ DATE_FORMAT : Formats a date as specified by a format mask.
 - ✓ DATE_SUB : Returns a date after a certain time/date interval has been subtracted.
 - ✓ More...

Other DML Statements:

➤ INSERT

Example:

```
INSERT INTO Tablename(  
    column1, column2, column3, ...)  
VALUES  
    (value1, value2, value3, ...)
```

- If any column is omitted in an INSERT statement and that column is NOT NULL, then INSERT fails; if null it is set to null.
- If the column is defined as NOT NULL BY DEFAULT, it is set to that default value.
- Omitting the list of columns is equivalent to specifying all values.

```
INSERT INTO employees VALUES (  
300025, '1990-01-01', 'Philips', 'Mathew', 'M', '2018-01-01');
```


Other DML Statements (contd...):

➤ UPDATE

Example:

UPDATE Tablename

SET Columnname(s) = scalar expression

WHERE [condition]

- Single or Multiple rows can be updated.
- Can be updated using a sub-query. (Query can be written to update where the updated value can be a result of another query).

```
UPDATE employees  
SET birth_date = '1991-01-01'  
WHERE emp_no = 300025;
```

Other DML Statements (contd...):

➤ **DELETE**

Example:

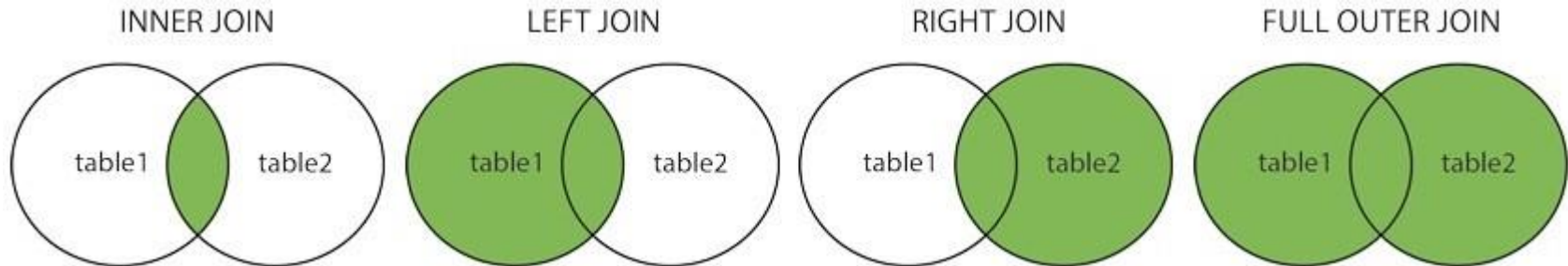
```
DELETE FROM Tablename  
WHERE [ condition ]
```

- Single or multiple row delete or deletion of all rows.

```
DELETE  
FROM employees  
WHERE emp_no = 300025;
```

SQL Joins

- A **JOIN** clause is used to combine rows from two or more tables, based on a related column between them.
- Different Types of SQL JOINS
 - ✓ **(INNER) JOIN**: Returns records that have matching values in both tables.
 - ✓ **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table.
 - ✓ **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table.
 - ✓ **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table.



```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

- The UNION operator is used to combine the result-set of two or more SELECT statements.
 - Each SELECT statement within UNION must have the same number of columns.
 - The columns must also have similar data types.
 - The columns in each SELECT statement must also be in the same order.
 - The column names in the result-set are usually equal to the column names in the first SELECT statement in the UNION.
-
- UNION and UNION ALL
 - ✓ The UNION operator selects only distinct values by default.
 - ✓ To allow duplicate values, use UNION ALL.

UNION

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2
;
```

Database Object Table

We can consider Table as a database object which can be used to organise the data in a structured manner with rows and columns.

We will learn how to create tables in MySQL using the CREATE TABLE command, but before to that we need to understand, about data types in MySQL.

When we discuss about data types, it is essentially means that, what kind of data you want to store, like if it will be of type Integer, Character or of Floating type of data.

Example of some.

Numeric Type

INT	Normal Integer type with a range from -2147483648 to 2147483647
SMALLINT	Small range than INT and do have a range from -32768 to 32767
DECIMAL (x,y)	Correspond to floating point numbers, x is the length of number and y is the number of decimal points allowed.

There are many other numeric types available, we have just mentioned some of the most used type.

Character/String Type

CHAR (number of characters)	Normal character type data, and we need to provide the size or length, maximum allowed is 255.
VARCHAR(size)	Normal character type but allowed alpha numeric type data, maximum size is 255

There are many other character types available, we have just mentioned some of the most used type.

Time/Date Type

DATE	Used to store date type data, and default format is yyyy-mm-dd.
DATETIME	Used to store date and time format data like yyyy-mm-dd hh:mm:ss
TIME	Only used to store time data like hh:mm:ss

There are many other date/time types available, we have just mentioned some of the most used type.

CREATE TABLE

Let us say we want to create a table with student's name, age and city, we can use the following syntax.

```
USE database <dbname>
```

```
CREATE TABLE Student  
(
```

```
    Name VARCHAR(50),  
    AGE  INT,  
    CITY VARCHAR(20)
```

```
);
```

To add a record to the table you can use the below statement.

```
INSERT INTO Student VALUES('Manas',50,'Pune');
```

CREATE VIEW

View can be considered as a virtual representation of table, where they do not contain any values.

View can be worked as a restriction mechanism which we can put on the base table, if we want certain features of a table to be not visible for a section of user.

```
create view student1_vw AS  
select name,age from Student;
```

INDEX

Index can be considered as a structure which we impose on a table to speed up the data retrieval process.

It also provides us a way to restrict any column/feature in a table to have only unique values and when someone will try to insert any duplicate values, it will flag an error.

We will learn more about the concepts of primary key , unique key etc. which automatically creates an index.

INDEX

```
7
8 Name VARCHAR(50),
9 AGE INT,
10 CITY VARCHAR(20),
11 Branch VARCHAR(20)
12 );
13
14
15 • INSERT INTO Student VALUES ('XXX',20,'Pune','DS');
16 • INSERT INTO Student VALUES ('YYY',21,'Pune','DS');
17 • INSERT INTO Student VALUES ('XYZ',22,'Pune','AI');
18 • INSERT INTO Student VALUES ('YXX',23,'Pune','AI');
19
20 • select * from Student;
```

100% 23:20

Result Grid Filter Rows: Search Export:

	Name	AGE	CITY	Branch
▶	XXX	20	Pune	DS
	YYY	21	Pune	DS
	XYZ	22	Pune	AI
	YXX	23	Pune	AI

INDEX

22 • EXPLAIN select Name from Student;

23

24

25

26

27

28

29

30

100% 34:22

Result Grid Filter Rows: Search Export:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Student	NULL	ALL	NULL	NULL	NULL	NULL	4	100.00	NULL

21 • CREATE INDEX ix_1 ON Student (Name);

22 • EXPLAIN select Name from Student;

23

24

25

26

27

28

29

30

100% 1:23

Result Grid Filter Rows: Search Export:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Student	NULL	index	NULL	ix_1	53	NULL	4	100.00	Using index

SEQUENCE

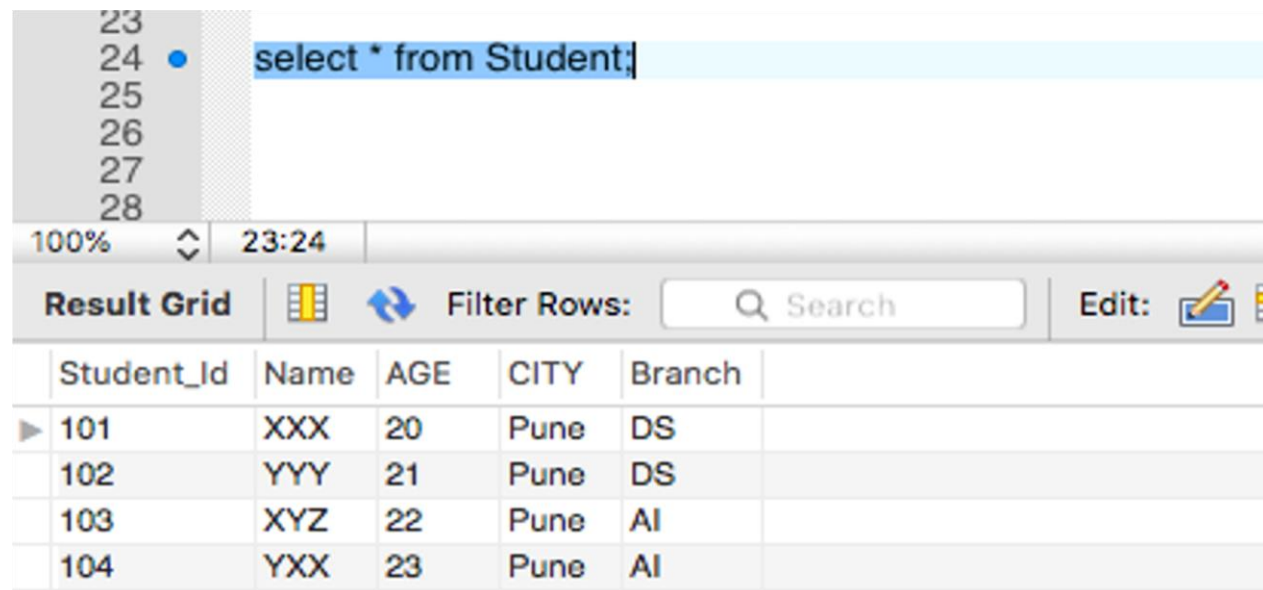
Sequence can be considered as a database object, which is used for generating unique integer values in sequence.

Sometimes, we need to have a unique column, in a table and we need to insert values in that column automatically whenever we insert any records in the table.

```
5 • CREATE TABLE Student
6   (
7     Student_Id INT NOT NULL AUTO_INCREMENT ,
8     Name VARCHAR(50),
9     AGE INT,
10    CITY VARCHAR(20),
11    Branch VARCHAR(20),
12    CONSTRAINT unique_ix UNIQUE KEY (Student_Id)
13  );
14
15
16 • ALTER TABLE Student AUTO_INCREMENT = 101;
```

```
17
18 • INSERT INTO Student(Name,AGE,CITY,Branch) VALUES ('XXX',20,'Pune','DS');
19 • INSERT INTO Student(Name,AGE,CITY,Branch) VALUES ('YYY',21,'Pune','DS');
20 • INSERT INTO Student(Name,AGE,CITY,Branch) VALUES ('XYZ',22,'Pune','AI');
21 • INSERT INTO Student(Name,AGE,CITY,Branch) VALUES ('YXX',23,'Pune','AI');
22
```

SEQUENCE



The screenshot shows a database query editor with a line of code selected: `select * from Student;`. Below the editor is a toolbar with a 'Result Grid' button, a 'Filter Rows' button, a search bar, and an 'Edit' button. The result grid displays a table with 6 columns: Student_Id, Name, AGE, CITY, Branch, and an empty column. The table contains 4 rows of data.

	Student_Id	Name	AGE	CITY	Branch	
▶	101	XXX	20	Pune	DS	
	102	YYY	21	Pune	DS	
	103	XYZ	22	Pune	AI	
	104	YXX	23	Pune	AI	

Store Procedures - Transactions

```
program MoneyTransfer;
var
exec sql begin declare section
    xAmount, Amount, xBalance: integer;
    Number, FromAccount, ToAccount: array [1..6] of char;
exec sql end declare section

begin
exec sql connect "UserId" identified by "Password";
{ Input data is read }
writeln('Write Amount, Withdrawals Account, Deposit Account');
read(Amount, FromAccount, ToAccount);
exec sql
    select Balance into :xBalance
    from SavingAccounts
    where Number = :FromAccount;
if xBalance < Amount
then
begin
writeln('Insufficient Balance'); rollback;
end
else
begin
exec sql
    update SavingAccounts
    set Balance = :xBalance - :Amount
    where Number = :FromAccount;
exec sql
    update CheckingAccounts
    set Balance = :xBalance + :Amount
    where Number = :ToAccount;
end;
if sqlcode = 0 then commit else rollback
end;
end {program}.
```


Reference Books

S.No	Name	Author	Publisher
1	SQL Cookbook.	Anthony Molinaro	O'Reilly Media
2	SQL QuickStart Guide	Walter Shields	ClydeBank Media LLC
3	Learning SQL	Alan Beaulieu	O'Reilly

References

- Slides adopted from - Database System Concepts - Abraham Silberschatz , Henry F. Korth , S. Sudarshan
- Relational DBMS Internals - Antonio Albano , Dario Colazzo, Giorgio Ghelli , Renzo Orsini
- MySQL Documentation : <https://dev.mysql.com/doc/>
- Entity Relationship Diagrams source : https://www.lucidchart.com/pages/er-diagrams/#section_2 , https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,1.html
- <https://www.geeksforgeeks.org/snowflake-schema-in-data-warehouse-model/>
- <https://www.vertabelo.com/blog/data-warehouse-modeling-the-star-schema/>
- www.innovativearchitects.com

Additional Slides

Integer Types (Exact Value) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT

Table 11.1 Required Storage and Range for Integer Types Supported by MySQL

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

◀ PREV HOME UP NEXT ▶

<https://dev.mysql.com/doc/refman/8.0/en/integer-types.html>

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
 - Business decision – What attributes should we record in the database?
 - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be divided into
 - The storage manager,
 - The query processor component,
 - The transaction management component.

Storage Manager

- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
 - Interaction with the OS file manager
 - Efficient storing, retrieving and updating of data
- The storage manager components include:
 - Authorization and integrity manager
 - Transaction manager
 - File manager
 - Buffer manager

Storage Manager (Cont.)

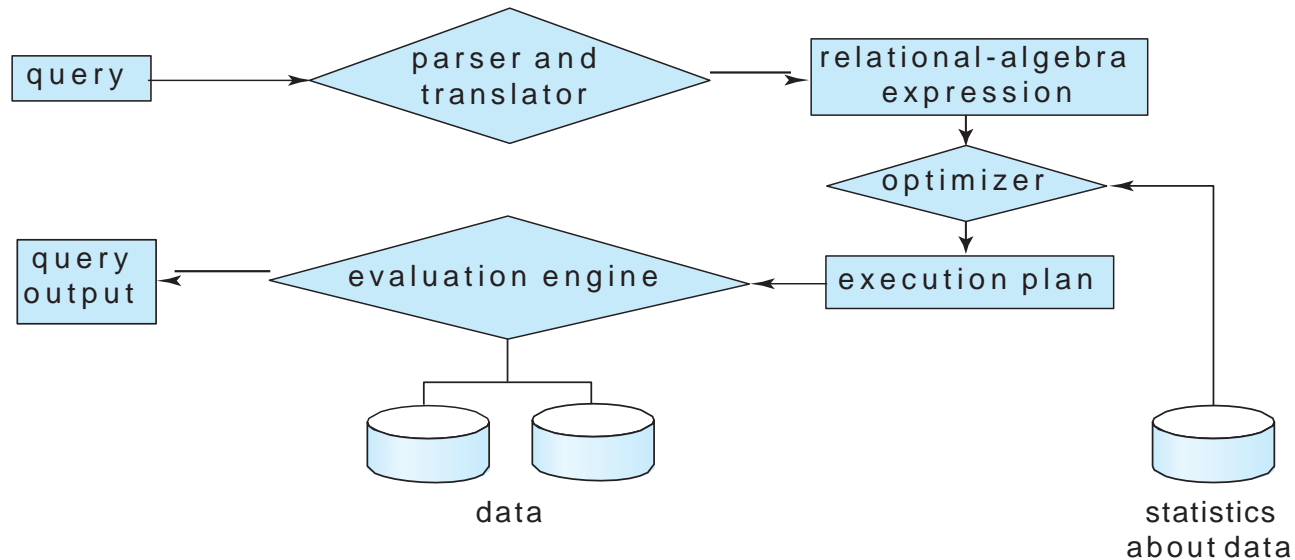
- The storage manager implements several data structures as part of the physical system implementation:
 - Data files -- store the database itself
 - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
 - Indices -- can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.

Query Processor

- The query processor components include:
 - DDL interpreter -- interprets DDL statements and records the definitions in the data dictionary.
 - DML compiler -- translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
 - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
 - Query evaluation engine -- executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Architecture

- Centralized databases
 - One to a few cores, shared memory
- Client-server,
 - One server machine executes work on behalf of multiple client machines.
- Parallel databases
 - Many core shared memory
 - Shared disk
 - Shared nothing
- Distributed databases
 - Geographical distribution
 - Schema/data heterogeneity