# **sql developer**

A database is a storage of operational data that is integrated and stored so that multiple application systems can share it. A data set in which data necessary for operation is collected in a computer memory with minimal redundancy. A **Database Management System (DBMS)** is a set of software systems for managing and maintaining the database so that all application programs can share the database.

Database is used for (1) realtime accessibilities (immediately sends the generated data to a computer for processing), (2) real-time processing and response to queries (allows programmer to write and read whenever you want), (3) allows for continuous change (functions such as insert, delete, and update of new data are performed frequently), (4) concurrent sharing (multiple users can access and use the data they want at the same time) and (5) content Reference (data is not only referenced according to the address or location of data records, but according to the content requested by the user)

**- INTRO TO DBMS –**

Download: <https://www.oracle.com/downloads/>

* SQL Developer: <https://www.oracle.com/tools/downloads/sqldev-downloads.html>

|  |  |
| --- | --- |
| -- | Annotation (does not execute) |
| ; | Ending the command |
| Ctrl + enter | Execute |
| ‘*string’* | All data of type string or date needs to be in single quotation marks |
| “nickname” | Only nicknames use double quotation marks |
| % | Represents evverything |
| + | Prints alsot he fields that are 0 |

|  |  |
| --- | --- |
| SELECT \* FROM USER\_TABLES; | Table info that my account owns |
| SELECT \* FROM USER\_INDEXES; | Index info that my account owns |
| SELECT \* FROM USER\_CONSTRAINTS; | Constraint conditions my account has |
| SELECT \* FROM USER\_VIEWS; | View information my account has |

**-SELECT-**

|  |  |  |  |
| --- | --- | --- | --- |
| SELECT | The fields you want to print | SELECT \* FROM TAB; | Prints all the tables owned by the user |
| SELECT \* FROM *tableName*; | Prints the information in the table named: *tableName* |
| DESC | Describes the structure of the table | DESC *tableName*; |  |
| \* | Select all fields |  |  |
| DISTINCT | Remove repeating rows |  |  |
| FROM | Name of the table you want to extract information from |  |  |
| WHERE | Filtering condition for the precise information you want to print | SELECT \* FROM emp WHERE deptno=10; | Print all fields from the EMP table only if the department number is 10 |
| ORDER BY | Descending/ or Ascending (default) order |  |  |
| IN/NOT IN |  | SELECT \* FROM EMP WHERE EMPNO IN (7902,7788,7566); |  |
| LIKE/NOT LIKE | Searches whether it matches the pattern. | LIKE  SELECT ENAME, MGR, SAL FROM EMP WHERE ENAME LIKE 'M%' or ’%S’ or '%A% or '\_\_R%';  →Names that start with ‘m’  →Names that ends with ‘s’  →Names that includes an ‘a’  →Names that has ‘r’ as their third letter |  |
| IS NULL | Checking if the field is null |  |  |
| ALTER SESSION SET |  | ALTER SESSION SET NLS\_DATE\_FORMAT = 'MM-DD-YY'; | Changes the date to the format of your choice |
| NVL | Giving a substitute value to a field that can have a NULL values | SELECT ENAME, SAL, COMM, SAL\*12+NVL(COMM, 0)  FROM EMP; | If commission is null, change its value to 0 |

**- JOIN -**

1. **EQUI JOIN:** Joins the tables by connecting rows with the same column values ​​in both tables.

SELECT \* FROM EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;

1. **NON-EQUI JOIN:** Uses a comparison operator other than the ‘=’ operator for the join condition in the WHERE clause to check whether the join condition is **within a specific range**.

SELECT EMPNO, ENAME, JOB, MGR, SAL, 'grade'||GRADE GRADE

→Creates a new column called “GRADE”, checks in the salgrade table, which grade the employee’s salary fit in, and prints its grade

FROM EMP, SALGRADE

WHERE SAL BETWEEN LOSAL AND HISAL;

1. **OUTER JOIN:** A join in which rows that do not satisfy the join condition also appear.
2. **SELF JOIN:** Join within one table.

→When each employee has a manager, indicated by its manager’s employment number, we can use self-join to print the worker’s manager name.

SELECT W.EMPNO, W.ENAME, W.MGR, M.ENAME

FROM EMP W, EMP M

WHERE W.MGR = M.EMPNO;

**- SQL FUNCTIONS –**

1. **SINGLE FUNCTIONS**

|  |  |  |
| --- | --- | --- |
| Type | Operator | Description |
| Multiplicative Operator | \* | Multiplication |
| / | Division |
| Additive Operator | + | Addition |
| - | Subtraction |
| Relational Operator | < | Less than |
| <= | Less than (inclusive) |
| > | Greater than |
| >= | Greater than (inclusive) |
| = | Equality |
| !=  <>  ^= | Inequality |
| Logical Operators | AND | SELECT \* FROM EMP WHERE DEPTNO=10 AND job=’MANAGER’; |
| OR | SELECT \* FROM WHERE DEPTNO=10 OR JOB=’MANAGER’; |
| NOT | SELECT \* FROM EMP WHERE NOT DEPTNO = 10; |
| BETWEEN a AND b | SELECT ENAME, SAL FROM EMP WHERE SAL BETWEEN 100 and 1500; |
| Concatenation operator | || | SELECT ENAME || ‘ is a ‘ || JOB “Employees Details” FROM EMP;  →“Scott is a MANAGER” under the nickname “Employees Details. |
| Number Operator | ABS(x) | Absolute value |
| COS(x) | Cosine value |
| EXP(x) | ex (e=2.71828183) |
| FLOOR(x) | Abandons decimals |
| LOG(x) | Converts to log value |
| POWER(m, n) | mn |
| SIGN(x) | returns the sign of an argument. (-1 if x<0) |
| SIN(x) | Sin value |
| TAN(x) | Tan value |
| ROUND(x) | Rounds to the nearest decimal point |
| ROUND(x,y) | Rounds x to the yth decimal point |
| TRUNC(x, y) | Abandones all values below the yth decimal point |
| CEIL(x) | Rounds uup to nearest integer |
| MOD(x, n) | Remainder of x/n |
| String Operator | LOWER(str) | Converts to minuscule letter |
| UPPER(str) | Converts to capital letter |
| INITCAP(str) | Converts first character to capital and result to miniscules |
| CONCAT(str1, str2) | Links str1 and str2 (str1str2) |
| SUBSTR(str, starting point, number to extract) | Cut and extracts the characters. If the starting position is negative, count the digits from the end.  SELECT SUBSTR ('ORACLE',3,2)  FROM DUAL;  →AC |
| LENGTH(str) | Calculates string length |
| INSTR(str, character to find, starting position, number of occurrences and the number of searches) | How many times the character to find in str is found  SELECT INSTR('ABCABC', 'B')  FROM DUAL;  →2  SELECT INSTR ('ABCABC', 'B', -3)  FROM DUAL;  →2 |
| LPAD/RPAD(str, string length, string to fill it with) | Creates a string of specified length and fills the with the specified character (on the left)  SELECT LPAD ('ORACLE', 10, '#')  FROM DUAL;  →####ORACLE  SELECT RPAD ('ORACLE', 10, '#')  FROM DUAL;  →ORACLE#### |
| (1) TRIM(str)  (2) RTRIM(str)  (3) LTRIM(str) | (1) Drops all blanks  (2) Drops all blanks at the right side of the string  (3) Drops all blanks at the left side of the string |
| CHR(str) |  |
| ASCII(str) |  |
| REPLACE(str, ‘x’, ‘y’) | In the string str, replace all ‘x’ to ‘y’ |
|  |  |
| Date operators | SYSDATE | Current date  (SYSDATE+1 = tomorrow) |
| Converting types | TO\_CHAR | 0: fill character room even if there are not enough characters  SELECT TO\_CHAR (123456, '0,000,000.00')  DUAL;  →0,123,456.00 |
| 9: fill character room with only if there are enough characters  SELECT TO\_CHAR (123456, '9,999,999.99')  FROM DUAL; --123,456.00  →123,456.00 |
| ,: possible every 3 places  SELECT ENAME, TO\_CHAR (SAL, '$00,999')  FROM EMP  WHERE ENAME='SMITH';  →$00,800 |
| .: decimal point |
| $: Currency  SELECT ENAME, TO\_CHAR (SAL, '$99,999')  FROM EMP;  →$800 |
| L: Local currency units may be prefixed  SELECT TO\_CHAR (10000, 'L999,999')  FROM DUAL;  →￦10,000 |
| TO\_DATE | SELECT TO\_DATE ('20220412', 'YYYY-MMDD')  FROM DUAL;  →22/04/12 |
| EXTRACT | When you want to convert a year, month or date to a number:  SELECT EXTRACT (YEAR FROM SYSDATE)  FROM DUAL;  →2022 |

1. **GROUP FUNCTIONS**

* A group function is a function that brings a single result by applying a function to multiple rows or to all rows in a table.
* SUM, AVG, COUNT, MAX, MIN, STDDEV, VARIANCE
* We use the function GROUP BY, so that only one result is given per group.

SELECT DEPTNO, MAX(SAL), MIN(SAL), ROUND(AVG(SAL),2), SUM(SAL)

FROM EMP

GROUP BY DEPTNO;

* We use the HAVING clause to perform conditional comparison with group functions.

SELECT DNAME, MAX(SAL), MIN(SAL), ROUND(AVG(SAL),2)

FROM EMP E, DEPT D

WHERE E.DEPTNO = D.DEPTNO

AND SAL <5000

GROUP BY DNAME

HAVING AVG(SAL) >=1800

ORDER BY AVG(SAL);

* The TOP(n) can be used to limit the number of rows returned from ordered sets of data in SQL.

SELECT TOP(3)

ENAME, SAL,,EMPNO

FROM EMP

WHERE SAL >1000

ORDER BY SAL;

**- SUB-QUERY –**

In SQL a Subquery can be simply defined as a query within another query.

* Single row: = > >= < <= !=
* Multiple row: Multiple-row subqueries are nested queries that can return more than one row of results to the parent query (IN, ALL, ANY=SOME, EXISTS).

|  |  |  |
| --- | --- | --- |
| IN | ANY (SOME) | EXISTS |
| SELECT ENAME, HIREDATE, DEPTNO  FROM EMP  (DEPTNO, HIREDATE)  IN (SELECT DEPTNO,  MAX(HIREDATE) FROM  EMP); | SELECT \*  FROM EMP  WHERE SAL > ANY  (SELECT SAL FROM EMP  WHERE DEPTNO=30); | SELECT \*  FROM EMP MANAGER  WHERE EXISTS (SELECT \*  FROM EMP WHERE  MANAGER.EMPNO =  MGR); |

**- DDL/DML/DCL-**

1. **DDL** **– Data Definition Language**
   1. **CREATING TABLE**

|  |  |
| --- | --- |
| **Creating from scratch** | **Creating a similar table to a pre-existing one** |
|  |  |

* 1. **CHANGING TABLE LAYOUT**

|  |  |  |
| --- | --- | --- |
| ADDING | MODIFYING | DROPPING |
| ALTER TABLE EMP03  ADD (JOB VARCHAR2 (20),  SAL NUMBER (7,2) ); | ALTER TABLE EMP03  MODIFY (EMPNO VARCHAR2(4));  \*modification is only allowed when the the type of data is kept the same (VARCHAR(3) → VARCHAR(4)) | Eg.1:  ALTER TABLE EMP03  DROP COLUMN JOB;  Eg.2:  ALTER TABLE EMP03  DROP UNUSED COLUMNS; |

* 1. **DELETING TABLE**

If another table has reference data from this table, it cannot be deleted

DROP TABLE EMP01;

* 1. **RENAMING TABLE**

Changing table EMP02 to “TEST”

RENAME EMP02 TO TEST;

1. **DML** **– Data Manipulation Language**
   1. **INSERT**

|  |  |  |
| --- | --- | --- |
| **Adding values to all fields** | **Adding data to specific fields** | **Adding data retrieved from another table (using subquery)** |
| INSERT  INTO DEPT01  VALUES (50, 'ACCOUNTING', 'NEW YORK'); | INSERT  INTO DEPT01 (DEPTNO, LOC)  VALUES (60, 'SEOUL'); | INSERT  INTO DEPT01  SELECT \* FROM DEPT |

* 1. **UPDATE**

UPDATE DEPT01

SET LOC = (SELECT LOC FROM DEPT WHERE DEPTNO=60)

WHERE DEPTNO >=20;

* 1. **DELETE**

DELETE

FROM EMP01

WHERE ENAME='FORD';

1. **DCL** **– Data Control Language**
   1. **CREATE - CONSTRUCTING USER ADDRESS (scott2)**

CREATE USER Scott2 identified by 1234;

* 1. **GRANT – Giving access rights**

GRANT CREATE SESSION TO Scott2;

GRANT ALL ON EMP TO Scott2; -

GRANT ALL ON DEPT TO Scott2;

* 1. **REVOKE – Taking away access rights:**

REVOKE ALL ON EMP FROM Scott2

* 1. **DROP – Deleting a user**

DROP USER Scott2 CASCADE;

* 1. **COMMIT**
* It ends your current transaction and make permanent all changes performed in the transaction.

Until you commit a transaction, you can see any changes you have made during the transaction by querying the modified tables, and you can rollback (undo) any changes made during the transaction.

* This statement also erases all savepoints in the transaction and releases transaction locks.
  1. **ROLLBACK**
* It undoes work done in the current transaction.
  1. **SAVEPOINT**
* It marks different moments within the transaction that allows for a partial rollback till that position.

**- SEQUENCE -**

* A set of integers that are generated in order on a specific demand.
* Frequently used in the databases to give data a unique value.
  1. **CREATING SEQUENCE**

CREATE SEQUENCE FRIEND\_SEQ

\*: default (no need to specify)”

NOCACHE: indicate that values of the sequence are not pre-allocated. (otherwise database caches 20 sequence numbers by default)

START WITH 1 \*

INCREMENT BY 1 \*

MAXVALUE 9999

MNVALUE 1 \*

NOCACHE

NOCYCLE;

* 1. **ADDING NEXT VALUE**

FRIEND\_SEQ.NEXTVAL

* 1. **PRINTING CURRENT VALUE**

FRIEND\_SEQ.CURRVAL

* 1. **DELETING SEQEUNCE**

DROP SEQUENCE FRIEND\_SEQ

* 1. **EDIT SEQEUNCE**

ALTER SEQUENCE FRIEND\_SEQ [INCREMENT BY x] [START WITH x] [MAXVALUE x][MINVALUE x];

**-VIEW-**

* 1. **CONCEPT OF VIEW**
* A virtual table composed of rows and columns.
* Its purpose is to restrict access to an existing table
* A table derived from existing table that has takes up **NO** **physical storage and data.**
* When data in the base table is changed, it is also reflected in the view.
  1. **ADVANTAGES OF VIEW**
* Provides security for data by restricting access to the base table
* Easy way to manipulate data without affecting the base table.
  1. **TYPE OF VIEWS**

1. **Simple view:**

* A view that is composed out of one table
* Can use DML statements such as INSERT, DELETE, and UPDATE’
* Functions and group data can be used

CREATE OR REPLACE VIEW EMPv0

AS SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP;

1. **Composite view:**

* A view made up of one or more base tables.
* Limited use of DML statements
* Function or group data can be used

CREATE OR REPLACE VIEW EMPv0

AS SELECT EMPNO, ENAME, JOB, DNAME FROM EMP E, DEPT D

WHERE E.DEPTNO = D.DEPTNO;