Created by Ahsan Arif

LIKE Condition

The LIKE condition allows you to use wildcards in the *where* clause of an SQL statement. This allows you to perform pattern matching. The LIKE condition can be used in any valid SQL statement - select, insert, update, or delete.

The patterns that you can choose from are:

% allows you to match any string of any length (including zero length)

_ allows you to match on a single character

Examples using % wildcard

The first example that we'll take a look at involves using % in the *where* clause of a select statement. We are going to try to find all of the suppliers whose name begins with 'Hew'.

```
SELECT * FROM suppliers
WHERE supplier_name like 'Hew%';
```

You can also using the wildcard multiple times within the same string. For example,

```
SELECT * FROM suppliers
WHERE supplier_name like '%bob%';
```

In this example, we are looking for all suppliers whose name contains the characters 'bob'.

You could also use the LIKE condition to find suppliers whose name does **not** start with 'T'. For example,

```
SELECT * FROM suppliers
WHERE supplier_name not like 'T%';
```

By placing the **not** keyword in front of the LIKE condition, you are able to retrieve all suppliers whose name does **not** start with 'T'.

Examples using _ wildcard

Next, let's explain how the _ wildcard works. Remember that the _ is looking for only one character.

For example,

```
SELECT * FROM suppliers
WHERE supplier_name like 'Sm_th';
```

This SQL statement would return all suppliers whose name is 5 characters long, where the first two characters is 'Sm' and the last two characters is 'th'. For example, it could return suppliers whose name is 'Smith', 'Smyth', 'Smath', 'Smeth', etc.

Here is another example,

```
SELECT * FROM suppliers
WHERE account_number like '12317_';
```

You might find that you are looking for an account number, but you only have 5 of the 6 digits. The example above, would retrieve potentially 10 records back (where the missing value could equal anything from 0 to 9). For example, it could return suppliers whose account numbers are:

123170
123171
123172
123173
123174
123175
123176
123177
123178
123179.

Examples using Escape Characters

Next, in Oracle, let's say you wanted to search for a % or a _ character in a LIKE condition. You can do this using an Escape character.

Please note that you can define an escape character as a single character (length of 1) ONLY.

For example,

```
SELECT * FROM suppliers
WHERE supplier_name LIKE '!%' escape '!';
```

This SQL statement identifies the ! character as an escape character. This statement will return all suppliers whose name is %.

Here is another more complicated example:

```
SELECT * FROM suppliers
WHERE supplier_name LIKE 'H%!%' escape '!';
```

This example returns all suppliers whose name starts with H and ends in %. For example, it would return a value such as 'Hello%'.

You can also use the Escape character with the _ character. For example,

```
SELECT * FROM suppliers
WHERE supplier_name LIKE 'H%!_' escape '!';
```

This example returns all suppliers whose name starts with H and ends in _. For example, it would return a value such as 'Hello_'.

Frequently Asked Questions

Question: How do you incorporate the Oracle upper function with the LIKE condition? I'm trying to query against a free text field for all records containing the word "test". The problem is that it can be entered in the following ways: TEST, Test, or test.

Answer: To answer this question, let's take a look at an example.

Let's say that we have a *suppliers* table with a field called *supplier_name* that contains the values TEST, Test, or test.

If we wanted to find all records containing the word "test", regardless of whether it was stored as TEST, Test, or test, we could run either of the following SQL statements:

```
select * from suppliers
where upper(supplier_name) like ('TEST%');
```

or

```
select * from suppliers
where upper(supplier_name) like upper('test%')
```

These SQL statements use a combination of the upper function and the LIKE condition to return all of the records where the *supplier_name* field contains the word "test", regardless of whether it was stored as TEST, Test, or test.

Practice Exercise #1:

Based on the *employees* table populated with the following data, find all records whose *employee_name* ends with the letter "h".

```
CREATE TABLE employees
( employee_number    number(10)        not null,
  employee_name      varchar2(50)      not null,
  salary             number(6),
  CONSTRAINT employees_pk PRIMARY KEY
  (employee_number)
);


INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1001, 'John Smith', 62000);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1002, 'Jane Anderson', 57500);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1003, 'Brad Everest', 71000);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1004, 'Jack Horvath', 42000);
```

Solution:

The following SQL statement would return the records whose *employee_name* ends with the letter "h".

```
SELECT *
FROM employees
WHERE employee_name LIKE '%h';
```

It would return the following result set:

EMPLOYEE_NUMBER	EMPLOYEE_NAME	SALARY

1001	John Smith	62000
1004	Jack Horvath	42000

Practice Exercise #2:

Based on the *employees* table populated with the following data, find all records whose *employee_name* contains the letter "s".

```
CREATE TABLE employees
( employee_number    number(10)        not null,
  employee_name      varchar2(50)      not null,
  salary             number(6),
  CONSTRAINT employees_pk PRIMARY KEY
  (employee_number)
);


INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1001, 'John Smith', 62000);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1002, 'Jane Anderson', 57500);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1003, 'Brad Everest', 71000);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1004, 'Jack Horvath', 42000);
```

Solution:

The following SQL statement would return the records whose *employee_name* contains the letter "s".

```
SELECT *
FROM employees
WHERE employee_name LIKE '%s%';
```

It would return the following result set:

EMPLOYEE_NUMBER	EMPLOYEE_NAME	SALARY
1002	Jane Anderson	57500

1003	Brad Everest	71000

Practice Exercise #3:

Based on the *suppliers* table populated with the following data, find all records whose *supplier_id* is 4 digits and starts with "500".

```
CREATE TABLE suppliers
( supplier_id          varchar2(10)        not null,
  supplier_name        varchar2(50)        not null,
  city                 varchar2(50),
  CONSTRAINT suppliers_pk PRIMARY KEY
  (supplier_id)
);
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES ('5008', 'Microsoft', 'New York');

INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES ('5009', 'IBM', 'Chicago');

INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES ('5010', 'Red Hat', 'Detroit');

INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES ('5011', 'NVIDIA', 'New York');
```

Solution:

The following SQL statement would return the records whose *supplier_id* is 4 digits and starts with "500".

```
select *
FROM suppliers
WHERE supplier_id LIKE '500_';
```

It would return the following result set:

SUPPLIER_ID	SUPPLIER_NAME	CITY
5008	Microsoft	New York
5009	IBM	Chicago