

bite-sized.sql

# SQL: UPSERT WITH MERGE

# Suppose we have a table of customers

```
SELECT TOP 10 * FROM customers;
```

	CustomerKey	FirstName	LastName	BirthDate
1	11000	Jon	Yang	1971-10-06
2	11001	Eugene	Huang	1976-05-10
3	11002	Ruben	Torres	1971-02-09
4	11003	Christy	Zhu	1973-08-14
5	11004	Elizabeth	Johnson	1979-08-05
6	11005	Julio	Ruiz	1976-08-01
7	11006	Janet	Alvarez	1976-12-02
8	11007	Marco	Mehta	1969-11-06
9	11008	Rob	Verhoff	1975-07-04
10	11009	Shannon	Carlson	1969-09-29

# And some changes in another table

```
SELECT * FROM new_customers;
```


	FirstName	LastName	BirthDate	NewFirstName	NewBirthDate
1	Eugene	Huang	1976-05-10	Eugene	1976-06-10
2	Elizabeth	Johnson	1979-08-05	Beth	1979-08-05
3	Janet	Alvarez	1976-12-02	Janet	1976-12-02
4	Homer	Simpson	1956-05-12	Homer	1956-05-12
5	Marge	Simpson	1956-10-01	Marge	1956-10-01

Existing customers

New customers

# We could run an update for the existing customers

```
UPDATE c
SET c.FirstName = nc.NewFirstName,
    c.BirthDate = nc.NewBirthDate
FROM customers c
    INNER JOIN new_customers nc
        ON c.FirstName = nc.FirstName
        AND c.LastName = nc.LastName
        AND c.BirthDate = nc.BirthDate;
```



This INNER JOIN query updates any customers that exist in both tables

# And a separate insert for the new customers

```
INSERT INTO customers (FirstName, LastName, BirthDate)
SELECT FirstName, LastName, BirthDate
FROM new_customers
EXCEPT
SELECT FirstName, LastName, BirthDate
FROM customers;
```

This EXCEPT query gives us any customers in the new\_customers table that aren't in the customers table



	FirstName	LastName	BirthDate
1	Marge	Simpson	1956-10-01
2	Homer	Simpson	1956-05-12

# Or we can do both at once with **MERGE**

2

The **USING** keyword precedes the data we want to reference. This can be a table, a query or parameters from a calling process

1

The **MERGE** keyword precedes the target table

```
MERGE customers AS c
USING new_customers AS nc
  ON c.FirstName = nc.FirstName
  AND c.LastName = nc.LastName
  AND c.BirthDate = nc.BirthDate
WHEN MATCHED
  THEN UPDATE
    SET c.FirstName = nc.NewFirstName,
        c.BirthDate = nc.NewBirthDate
WHEN NOT MATCHED
  THEN INSERT (FirstName, LastName, BirthDate)
  VALUES (nc.FirstName, nc.LastName, nc.BirthDate);
```

3

We include join predicates as we would in any other query

4

The merge includes instructions for what to do when the join produces a match. In this case, **UPDATE** the rows with the new values

5

And instructions for when there is no match (in this example, **INSERT** the new rows)



Note: some databases  
use RETURNING instead  
of OUTPUT

bite-sized.sql

# We can also **OUTPUT** the changes

```
MERGE customers AS c
... --code collapsed for brevity
OUTPUT
```

In SQL Server, **\$action** is  
a text column which  
contains one of **INSERT**,  
**UPDATE** or **DELETE**

```
$action,
deleted.CustomerKey AS OldCustomerKey,
deleted.FirstName AS OrigFirstName,
deleted.LastName AS OrigLastName,
deleted.BirthDate AS OrigBirthDate,
inserted.CustomerKey AS NewCustomerKey,
inserted.FirstName AS NewFirstName,
inserted.LastName AS NewLastName,
inserted.BirthDate AS NewBirthDate;
```

The **deleted** result-set  
shows the state of the rows  
*before* the change

The **inserted** result-set  
shows the state of the rows  
*after* the change

	\$action	OldCustomerKey	OrigFirstName	OrigLastName	OrigBirthDate
1	INSERT	NULL	NULL	NULL	NULL
2	INSERT	NULL	NULL	NULL	NULL
3	UPDATE	11001	Eug		
4	UPDATE	11004	Eliz		
5	UPDATE	11006	Jan		

NewCustomerK...	NewFirstName	NewLastName	NewBirthDate
29484	Marge	Simpson	1956-10-01
29485	Homer	Simpson	1956-05-12
11001	Eugene	Huang	1976-06-10
11004	Beth	Johnson	1979-08-05
11006	Janet	Alvarez	1976-12-02

# **MERGE** clause:

1. **UPDATE, DELETE and/or INSERT in one statement**
2. **Specify action to take on matched and not matched rows**
3. **Optionally output a summary of the changes**