

Creating tables

Basic syntax

```
CREATE TABLE <table_name> (  
    <column1> <type1>,  
    <column2> <type2>,  
    ...  
    <columnN> <typeN>  
);
```

Example

```
CREATE TABLE Customer (  
    custid  varchar(10),  
    name    varchar(20),  
    city    varchar(30),  
    address varchar(30)  
);
```

Most common SQL data types

Strings

- ▶ **varchar** (n) – variable length, at most n characters

Numbers

- ▶ **smallint**
- ▶ **integer** or **int**
- ▶ **bigint**
- ▶ **numeric** (p, s) – arbitrary precision number
 p total digits, s digits in the fractional part

Date & Time

- ▶ **date** – e.g., '2023-10-03'
- ▶ **time** – time of the day: e.g., '21:09'
- ▶ **timestamp**

See the manual for the specific SQL version you use

Default values

Syntax

```
CREATE TABLE <table_name> (  
    <column1_name> <column1_type>,  
    <column2_name> <column2_type> DEFAULT <value>,  
    ...  
    <columnN_name> <columnN_type>  
) ;
```

Example

```
CREATE TABLE Account (  
    accnum varchar(12),  
    branch varchar(30),  
    custid varchar(10),  
    balance numeric(14,2) DEFAULT 0  
) ;
```

Populating tables

General syntax

```
INSERT INTO <table_name> VALUES (...), ..., (...);
```

Examples

```
INSERT INTO Account VALUES  
    ('243576', 'Edinburgh', 'cust1', -120);
```

```
INSERT INTO Customer VALUES  
    ('cust1', 'Renton', 'Edinburgh', '2 Wellington Pl'),  
    ('cust2', 'Watson', 'London', '221B Baker St'),  
    ('cust3', 'Holmes', 'London', '221B Baker St');
```

Populating tables with default values

Two possibilities:

1. Use the keyword **DEFAULT** in **INSERT**

Example

```
INSERT INTO Account VALUES  
('250018', 'London', 'cust3', DEFAULT)
```

2. List attributes explicitly (omitted ones will get the default)

Example

```
INSERT INTO Account (accnum, branch, custid) VALUES  
('250018', 'London', 'cust3')
```

Attributes without **DEFAULT** in **CREATE TABLE**
have default value **NULL** (meaning: the value is missing/unknown)

Changing the definition of a table

```
ALTER TABLE <name>  
    RENAME TO <new_name>;  
    RENAME <column> TO <new_column>;  
    ADD <column> <type>;  
    DROP <column>;  
    ALTER <column>  
        TYPE <type>;  
        SET DEFAULT <value>;  
        DROP DEFAULT;
```

Destroying tables

```
DROP TABLE <name>;
```

Many other changes are possible ...

Basic queries in SQL

Follow the basic pattern:

```
SELECT <list_of_attributes>  
FROM    <list_of_tables>  
WHERE   <condition>
```

Idea

1. Loop over all rows of the tables listed in **FROM**
2. Take those that satisfy the **WHERE** condition
3. Output the values of the attributes listed in **SELECT**

An extremely simple example

Customer			
ID	Name	City	Address
cust1	Renton	Edinburgh	2 Wellington Pl
cust2	Watson	London	221B Baker St
cust3	Holmes	London	221B Baker St

List all customers

```
SELECT *  
FROM    Customer
```

* means “all attributes”

What is the output to this query?

A very simple example

Customer			
ID	Name	City	Address
cust1	Renton	Edinburgh	2 Wellington Pl
cust2	Watson	London	221B Baker St
cust3	Holmes	London	221B Baker St

List name and address of all customers

```
SELECT Name, Address
FROM Customer
```

Output:

Name	Address
Renton	2 Wellington Pl
Watson	221B Baker St
Holmes	221B Baker St

A simple example

Customer			
ID	Name	City	Address
cust1	Renton	Edinburgh	2 Wellington Pl
cust2	Watson	London	221B Baker St
cust3	Holmes	London	221B Baker St

List name and address of customers living in Edinburgh

```
SELECT Name, Address
FROM Customer
WHERE City='Edinburgh'
```

Output:

Name	Address
Renton	2 Wellington Pl

More than one table in FROM

Table1	
A	B
1	2
3	4

Table2	
C	D
2	1

SELECT B, C
FROM Table1, Table2

1. Each row of Table1 is **concatenated** with each row of Table2

A	B	C	D
1	2	2	1
3	4	2	1

2. For each resulting row the values for attributes B and C are returned

B	C
2	2
4	2

Joining tables

Customer		
ID	Name	City
cust1	Renton	Edinburgh
cust2	Watson	London
cust3	Holmes	London

Account		
AccNum	CustID	Balance
123321	cust3	1330.00
243576	cust1	-120.00

List customers' names and their accounts' numbers

SELECT Name, AccNum
FROM Customer, Account
WHERE ID = CustID

Output:

Name	AccNum
Renton	243576
Holmes	123321

The basic WHERE clause

term :=

- ▶ attribute
- ▶ value

comparison :=

- ▶ term1 * term2, with $*$ $\in \{=, <>, <, >, <=, >=\}$
- ▶ term **IS NULL**
- ▶ term **IS NOT NULL**

condition :=

- ▶ condition1 **AND** condition2
- ▶ condition1 **OR** condition2
- ▶ **NOT** condition
- ▶ comparison

Database modification: Deletion

General form

```
DELETE FROM <table>  
WHERE <condition>
```

All rows in <table> satisfying <condition> are deleted

Example

Remove accounts with zero balance and unknown owner

```
DELETE FROM Account  
WHERE Balance=0 AND CustID IS NULL
```

Database modification: Replacement

General form

```
UPDATE <table>  
SET      <assignments>  
WHERE    <condition>
```

Replace the values of some attributes (using <assignments>) in each row of <table> that satisfies <condition>

Examples

Set a new balance on account 745622

```
UPDATE Account  
SET    balance=1503.82  
WHERE   accnum='745622'
```

Accounts in Burnaby with positive balance get a 0,2% bonus

```
UPDATE Account  
SET    balance = balance + 0.002*balance  
WHERE   branch='Burnaby' AND balance > 0
```

WHERE conditions in queries

- **filter** data within a table

```
SELECT Name, Address  
FROM    Customer  
WHERE   City='Vancouver'
```

- **join** data from different tables

```
SELECT Name, AccNum  
FROM    Customer, Account  
WHERE   ID = CustID
```

Filtering and join together

```
SELECT Name, Address, AccNum  
FROM    Customer, Account  
WHERE   ID = CustID AND City='Vancouver'
```


Explicit join syntax

```
table1 JOIN table2 ON <condition>
...
JOIN tableN ON <condition>
```

Logically separate join conditions from filters

```
SELECT Name, Balance
FROM Customer, Account
WHERE ID = CustID AND Balance < 0
```

```
SELECT Name, Balance
FROM Customer JOIN Account ON ID=CustID
WHERE Balance < 0
```

Qualification of attributes

Customer			Account		
CustID	Name	City	AccNum	CustID	Balance
cust1	Renton	Edinburgh	123321	cust3	1330.00
cust2	Watson	London	243576	cust1	-120.00
cust3	Holmes	London			

List the name of customers whose account is overdrawn

```
SELECT Customer.Name, Account.Balance
FROM Customer, Account
WHERE Account.CustID = Customer.CustID
AND Account.Balance < 0
```

Here, we specify the relations attributes are coming from

What is the output of this query?

Assigning new names to tables in FROM

```
SELECT Customer.Name, Account.Balance
FROM    Customer, Account
WHERE   Account.CustID = Customer.CustID
AND     Account.Balance < 0
```

```
SELECT C.Name, A.Balance
FROM    Customer C, Account AS A
WHERE   A.CustID = C.CustID
AND     A.Balance < 0
```

Note: “AS” is optional

```
SELECT C.Name, A.Balance
FROM    Customer C JOIN Account A ON C.CustID=A.CustID
WHERE   A.Balance < 0
```

Renaming attributes

```
SELECT C.Name CustName, A.Balance AS AccBal
FROM    Customer C, Account A
WHERE   A.CustID = C.CustID
AND     A.Balance < 0
```

This does not work:

```
SELECT C.Name CustName, A.Balance AS AccBal
FROM    Customer C, Account A
WHERE   A.CustID = C.CustID
AND     AccBal < 0
```

because the comparison happens before the output is produced,
but uses the new name AccBal of the output

Pattern matching

New comparison: term **LIKE** pattern

where **pattern** is a string consisting of
characters (case-sensitive!)

– (underscore) wildcard matching any one character

% (percent) wildcard matching any substring (including empty)

Example

Customers with a name that **begins with 'K'**
and has **at least 5 characters**

```
SELECT *  
FROM Customer  
WHERE name LIKE 'K_____ %' ;
```

Acknowledgements

[1] Database Systems: The Complete Book, 2nd Edition Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom Prentice Hall, 2009

[2] Database System Concepts, Seventh Edition Avi Silberschatz, Henry F. Korth, S. Sudarshan McGraw-Hill, March 2019 www.db-book.com