Structured Query Language (SQL) More Basic Statements

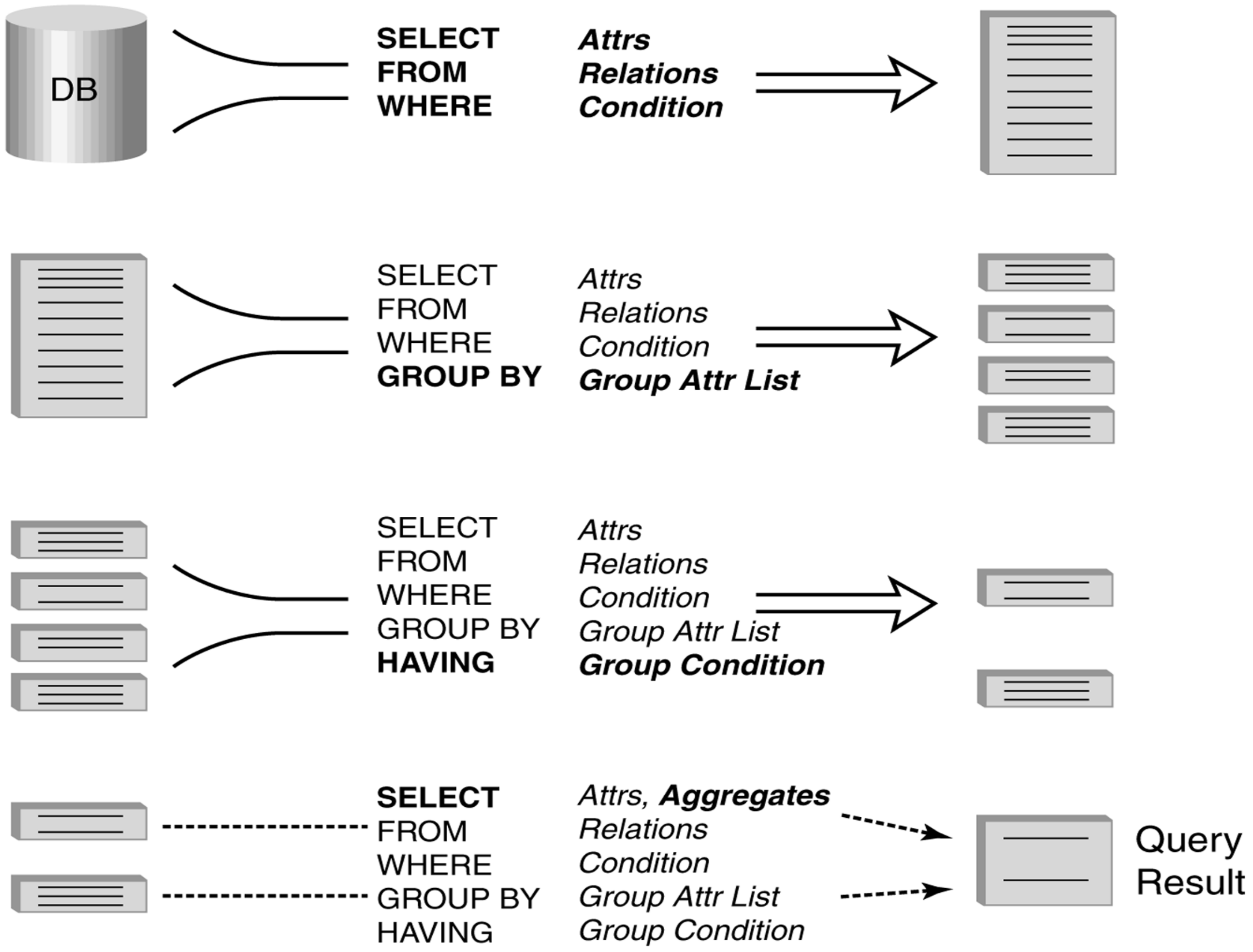
SQL

- Select statement

```
SELECT [DISTINCT] column_list
FROM table_list
[WHERE Predicate]
[GROUP BY column_list  [HAVING group_condition]]
[ORDER BY column_list [ASC|DESC] ] ;
```

SQL

- Select statement



SQL

- Examples:

```
select bname
from Branch;



select *
from Branch;

select *
from Account
where lower(bname) = ‘france’;
```

SQL

- Examples:

Print the name of all Customer with an account(s) at France branch

Print the name of all Customer with a loan(s) at France branch

SQL

- Examples:

Print the name of all Customer with an account(s) at France

```
select cname
from Account
where lower(bname) = ‘france’;
```

Print the name of all Customer with a loan(s) at France

```
select cname
from Loan
where lower(bname) = ‘france’;
```

SQL

- Examples:

  Print all Customer names with a loan, an account **or** both at France

SQL

- Examples:

  Print all Customer names with a loan, an account **or** both at France

```
select cname
from Account
where lower(bname) = ‘france’
Union
select cname
from Loan
where lower(bname) = ‘france’;
```

SQL

- Print the name of customers who have an account or a loan, or both in France branch except for those customers who have an account with bal of less than $800.

SQL

- Examples:

  Print all Customer names with a loan **and** an account at France

SQL

- Examples:

  Print all Customer names with a loan **and** an account at France

```
select cname
from Account
where lower(bname) = ‘france’
intersect
select cname
from Loan
where lower(bname) = ‘france’;
```

SQL

- Examples:

  Print all Customer names with an account **but not** a loan at France

SQL

- Examples:

  Print all Customer names with an account **but not** a loan at France

```
select cname
from Account
where lower(bname) = ‘france’
minus
select cname
from Loan
where lower(bname) = ‘france’;
```

Joins

- Joins are performed when the FROM clause has more than one table
  - Cartesian Product
    - When there are no conditions enforced
    - Example:

      Select * from Customer, Branch;

      This type of statements are hardly used.
  - Join
    - When there are is at least one condition enforced
    - There are different join types
      - Inner Join
      - Outer Join

SQL

- Inner Join (or Join)
  - Two or more columns are forced to be **equal**
  - The columns do not need to be named the same
  - Example

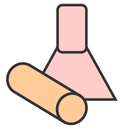
```
select Customer.cname, ccity
from Loan, Customer
where Loan.cname = Customer.cname
   AND lower(Loan.bname) = ‘france’;
```

SQL

- Examples:

  Print all Customer names who live in MPLS and have account(s) in branches that have more than 90,000 in assets and are located in the city Minnetonka.

SQL



- Examples:

  Print all Customer names who live in MPLS and have account(s) in branches that have more than 90,000 in assets and are located in the city Minnetonka.

```
select customer.cname
from Account, Customer, Branch
where lower(bcity) = 'minnetonka' AND
      assets > 90000 AND
      upper(ccity) = 'MPLS' AND
      account.cname = customer.cname AND
      branch.bname = account.bname;
```

SQL

- Outer Joins
  - Consider the following SQL statement against the Bank database:

```
SELECT account.cname, loan.cname
FROM Account, Loan
WHERE account.cname = loan.cname;
```

Question:

- As a business person what is it that you want that this statement does not give you?

SQL

- Left Outer Join

```
SELECT account.cname, loan.cname
FROM Account LEFT OUTER JOIN Loan
ON account.cname = loan.cname;
```

  - Returns customer who have an account and a loan and those customers who do have account but not a loan in the bank

- Right Outer Join

```
SELECT account.cname, loan.cname
FROM Account RIGHT OUTER JOIN Loan
ON account.cname = loan.cname;
```

  - Returns customer who have an account and a loan and those customers who do have a loan but not an account in the bank

- Full Outer Join

```
SELECT account.cname, loan.cname
FROM Account FULL OUTER JOIN Loan
ON account.cname = loan.cname;
```

  - Returns customer who have an account and a loan, customers who do have an account but not a loan in the bank, and those customers who do have a loan but not an account in the bank.

SQL

- Try it
  - Print the following:
    - If a customer has account in the bank, print the name of the customer and the branch name for their account.
    - If a customers does not have an account in the bank, then just print the name of the customer.

SQL

- Try it
  - Print the following:
    - If a customer has account in the bank, print the name of the customer and the branch name for their account.
    - If a customers does not have an account in the bank, then just print the name of the customer.

```
SELECT customer.cname, bname
FROM Customer LEFT OUTER JOIN account
ON Customer.cname = account.cname
order by Customer.cname;
```

SQL

- Try it
  - Repeat the previous question only for those customers who live in MPLS.

SQL

- Try it
  - Repeat the previous question only for those customers who live in MPLS.

```
SELECT customer.cname, bname
FROM Customer LEFT OUTER JOIN Account
ON customer.cname = account.cname
WHERE upper(ccity) = ‘MPLS’
order by Customer.cname;
```

Creating Views

- View is a relation derived from one or more **base** relations.
- Defined by a select statement.
- Selection, projection, join and union commonly used.
- Most DBMS’s don’t materialize views

Creating Views

- View are used for security and/or ease of use purposes
- A view can be defined on one or more tables
- CREATE VIEW *view_name* [**(***column-list***)**] AS

    select_statement;

Creating Views

- Examples:

```
Create view Names AS
    Select Fname, Lname, Minit
    From Employee;

Create view Cust_Account AS
    Select Loan.cname, Loan.L#, Account.A#
    From Loan, Account
    Where Loan.cname = Account.cname;
```