data
distributed
database
DBMS
DBMSs
shared
parallel
system
systems
architecture
processing
network
access
centralized
processors
performance
multiple
Distributed
DDBMS
memory
sites
site
single
applications
computers
nothing
technology
computer
Parallel
processor
may
fragments
require
physically
disk
stored
logically
number
sometimes
local
provide
decentralized
large
one
split
islands
bottleneck
interconnection
Figure
application
transparent
management
development
definition
nodes
logical
control
scalability
approach
improve
distinction
Shared
databases
typically
major
users
machines
within
like
Database
reliability
eliminates
replicated
make
also
many
discuss
often
directly
Date
com
disks
resources
based
defined
remote
controlled
architectures
scalable
order
referred
operational
communication
distribution
computing
collection
structure
provides
global
associated
used
least
capable
System
must
partitioned
consists
incompatible
way
administered
organizational
mode
user
dis
per
makes
Management
across
fact

Database System Lab (CSE 3103)

Session 04

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

Slide Copyright: Nazmus Sakib

OPERATOR

Operator	**Description**
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

©Nazmus Sakib, CSE, AUST

LIKE OPERATOR

- The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator:
- →The percent sign (%)
- →The underscore (_)
- The percent sign represents zero, one, or multiple characters. The underscore represents a single number or character. The symbols can be used in combinations.

LIKE OPERATOR

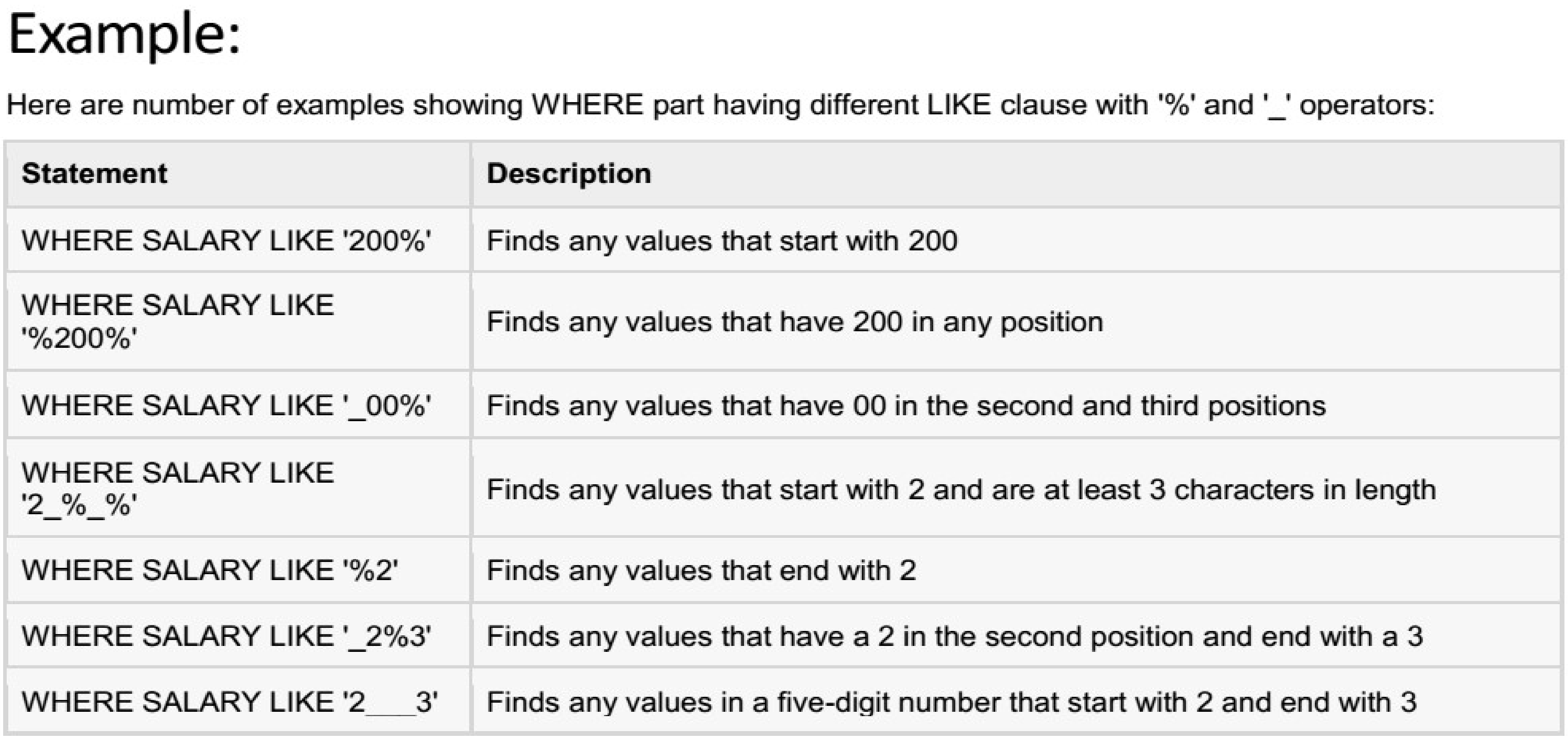
SYNTAX

```
SELECT column name(s)
FROM table name
WHERE column_ name LIKE pattern
```

You may also combine with the **NOT** keyword (Logical Negation). The NOT operator reverses the meaning of the logical operator with which it is used.

Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.

LIKE OPERATOR



Example:

Here are number of examples showing WHERE part having different LIKE clause with '%' and '_' operators:

Statement	Description
WHERE SALARY LIKE '200%'	Finds any values that start with 200
WHERE SALARY LIKE '%200%'	Finds any values that have 200 in any position
WHERE SALARY LIKE '_00%'	Finds any values that have 00 in the second and third positions
WHERE SALARY LIKE '2_%_%'	Finds any values that start with 2 and are at least 3 characters in length
WHERE SALARY LIKE '%2'	Finds any values that end with 2
WHERE SALARY LIKE '_2%3'	Finds any values that have a 2 in the second position and end with a 3
WHERE SALARY LIKE '2___3'	Finds any values in a five-digit number that start with 2 and end with 3

IN OPERATOR

- IN operator allows you to specify multiple values in WHERE clause

```
SELECT column name(s)
FROM table name
WHERE column_ name IN (value1, value2,….)
```

Example

```
SELECT *  FROM CUSTOMER WHERE AGE IN (25, 27);
```

BETWEEN OPERATOR

- BETWEEN operator selects a range of data between two values. Values can be numbers, text, or dates.

```
SELECT column name(s)
FROM table name
WHERE column_ name BETWEEN value1  AND value2
```

Example

```
SELECT *  FROM CUSTOMER WHERE AGE BETWEEN 23 AND 27;
```

OR | AND OPERATOR

- The AND operator displays a record if both the first condition and the second condition is TRUE.
- The OR operator displays a record if either the first condition or the second condition is true.
- Examples:

```
SELECT * FROM CUSTOMER WHERE AGE > =25 AND SALARY >=6500;

SELECT * FROM CUSTOMER WHERE AGE > =25 OR SALARY >=6500;

SELECT   *   FROM CUSTOMER WHERE NAME LIKE ‘KA%’ AND (AGE > =25 OR
SALARY >=6500);
```

TOP OPERATOR

- The TOP clause is used to specify the number of records to return.
- The TOP clasue can be very useful on large tables with thousands of records.

- SYNTAX

```
SELECT TOP number | percent column_name(s)
FROM table_name
```

- Examples :

```
SELECT TOP 3 * FROM CUSTOMER

SELECT TOP 60 percent FROM CUSTOMER
```

SQL Aggregate Functions

- SQL aggregate functions return a single value, calculated from values in a column.
- Useful aggregate functions:

✓AVG() - Returns the average value

✓COUNT() - Returns the number of rows

✓FIRST() - Returns the first value

✓LAST() - Returns the last value

✓MAX() - Returns the largest value

✓MIN() - Returns the smallest value

✓SUM() - Returns the sum

SQL Scalar functions

- SQL scalar functions return a single value, based on the input value.
- Useful scalar functions:

✓UPPER() - Converts a field to upper case

✓LOWER() - Converts a field to lower case

✓MID() - Extract characters from a text field

✓LEN() - Returns the length of a text field

✓ROUND() - Rounds a numeric field to the number of decimals specified

✓NOW() - Returns the current system date and time

✓FORMAT() - Formats how a field is to be displayed

GROUP BY FUNCTION

- The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.
- Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

Slide Copyright: Nazmus Sakib

GROUP BY FUNCTION

- Examples:

```
SELECT Age, MAX(Salary) FROM CUSTOMER
```

- Error message
- **error message**

```
Level 16, State 1, Line 1 Column 'CUSTOMER.Age' is invalid in the select list
because it is not contained in either an aggregate function or the GROUP BY clause
```

- Solution
- `SELECT Age, MAX(Salary) FROM CUSTOMER GROUP BY Age`

HAVING FUNCTION

- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.
- Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

HAVING FUNCTION

- Examples:

```
SELECT Age, MAX(Salary) FROM CUSTOMER GROUP BY Age
HAVING NAME LIKE 'Ka%'
```

- Error message
- error message

```
Level 16, State 1, Line 1 Column 'CUSTOMER.Age' is invalid in the select list
because it is not contained in either an aggregate function or the GROUP BY clause
```

- Solution

```
SELECT Age, MAX(Salary) FROM CUSTOMER GROUP BY Age
HAVING Age >=25
```

data
distributed
database
DBMS
shared
parallel
architecture
network
processing
system
systems
Distributed
DDBMS
DBMSs
access
centralized
processors
performance
multiple
memory
site
sites
single
applications
computer
nothing
disk
processor
Parallel
fragments
may
typically
major
require
physically
within
users
machines
computers
control
scalability
approach
improve
distinction
logical
application
management
transparent
Shared
databases
development
definition
nodes
technology
capable
interconnection
bottleneck
Figure
administered
organizational
mode
logically
stored
number
user
dis
Management
consists
partitioned
global
used
least
associated
collection
structure
incompatible
way
provides
must
System
provide
make
replicated
sites
local
decentralized
directly
com
Date
large
islands
resources
disks
remote
defined
controlled
based
often
discuss
many
also
reliability
eliminates
fact
across
sometimes
computing
one
split
order
scalable
architectures
referred
operational
communication
distribution
like
Database