CSCD 327: Relational Database Systems

Simple queries

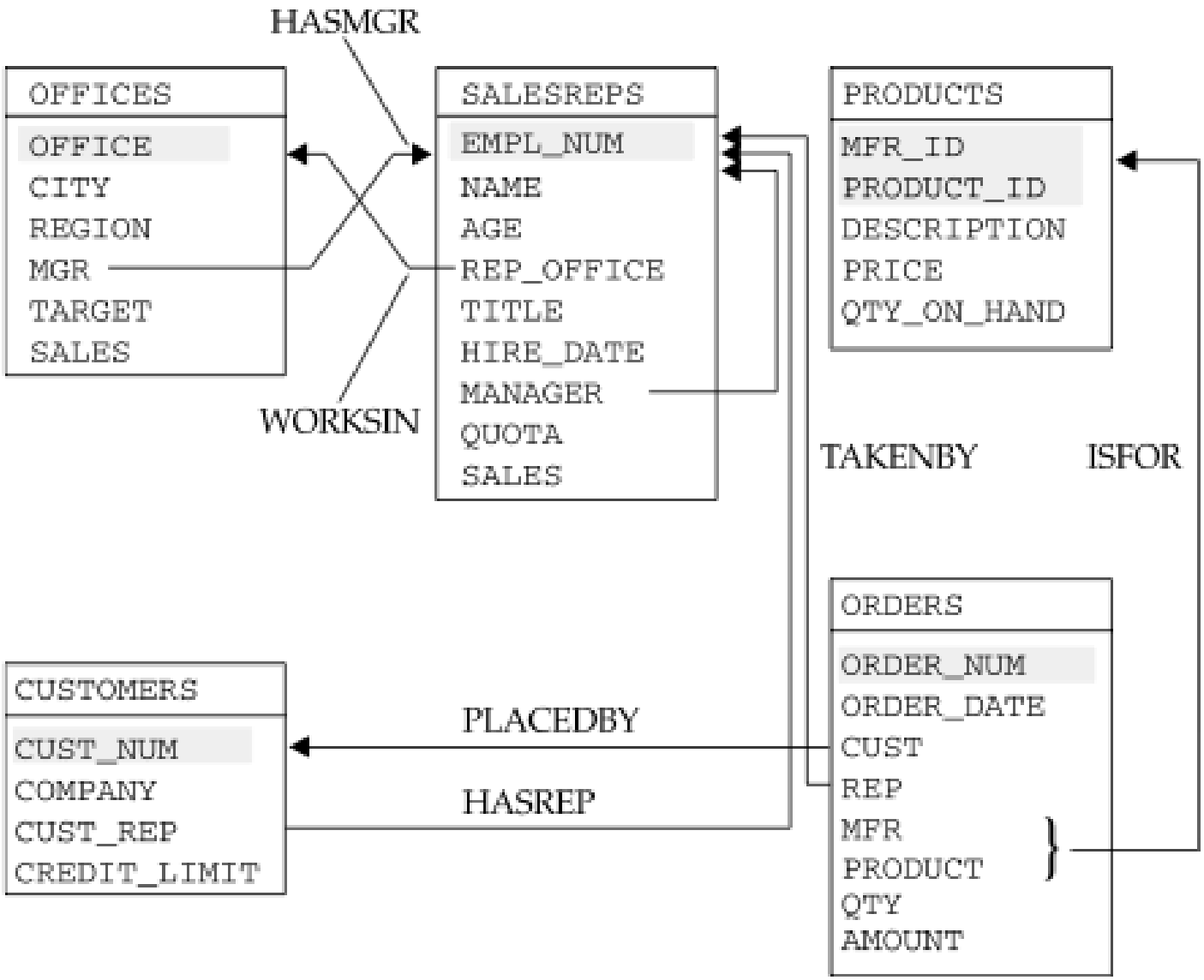
Instructor: Dr. Dan Li

SQL (DDL + DML)

- SQL is the Structured Query Language
- It is used to interact with the DBMS
- SQL can
  - Create Schemas in the DBMS (DDL - Create)
  - Alter Schemas (DDL - Alter)
  - Add Data (DML - Insert)
  - Remove Data (DML - Delete)
  - Change Data (DML - Update)
  - **Access Data (DML - Select)**
    - The main way of accessing data is using the DML command **SELECT**.
    - The abilities of the SELECT command forms the majority of this material on SQL

Sample Database

- Appendix A
- **PRODUCTS** Contains one row for each type of product that is available for sale.
- **OFFICES** Contains one row for each of the company's five sales offices where the salespeople work.
- **SALESREPS** Contains one row for each of the company's ten salespeople.
- **CUSTOMERS** Contains one row for each of the company's customers.
- **ORDERS** Contains one row for each order placed by a customer. For simplicity, each order is assumed to be for a single product.


HASMGR
OFFICES
OFFICE
CITY
REGION
MGR
TARGET
SALES
WORKSIN
SALESREPS
EMPL_NUM
NAME
AGE
REP_OFFICE
TITLE
HIRE_DATE
MANAGER
QUOTA
SALES
PRODUCTS
MFR_ID
PRODUCT_ID
DESCRIPTION
PRICE
QTY_ON_HAND
TAKENBY
ISFOR
ORDERS
ORDER_NUM
ORDER_DATE
CUST
REP
MFR
PRODUCT
QTY
AMOUNT
CUSTOMERS
CUST_NUM
COMPANY
CUST_REP
CREDIT_LIMIT
PLACEDBY
HASREP

Simple SELECT

- The SELECT statement retrieves data from a database and returns it to you in the form of query results.
- ***SELECT column list FROM table list***

e.g., *list the sales offices with their targets and actual sales.*

```
SELECT CITY, TARGET, SALES FROM OFFICES;
```

- Query results - the result of every relational operation is a relation (Not really. Why? Duplicates are allowed)

CITY	TARGET	SALES
Denver	300000.00	186042.00
New York	575000.00	692637.00
Chicago	800000.00	735042.00
Atlanta	350000.00	367911.00
Los Angeles	725000.00	835915.00

Formatting

- SPACES do not matter
- NEWLINES do not matter
- Good practice to put ; at the end of the query.
- CASE (except between single quotes) does not matter. (But, table names in MySQL are case sensitive. Why?)
- These are all valid:

```
SELECT CITY, TARGET, SALES FROM OFFICES;
```

OR

```
SElecT City, Target,       Sales
   From OfficES
      ;
```

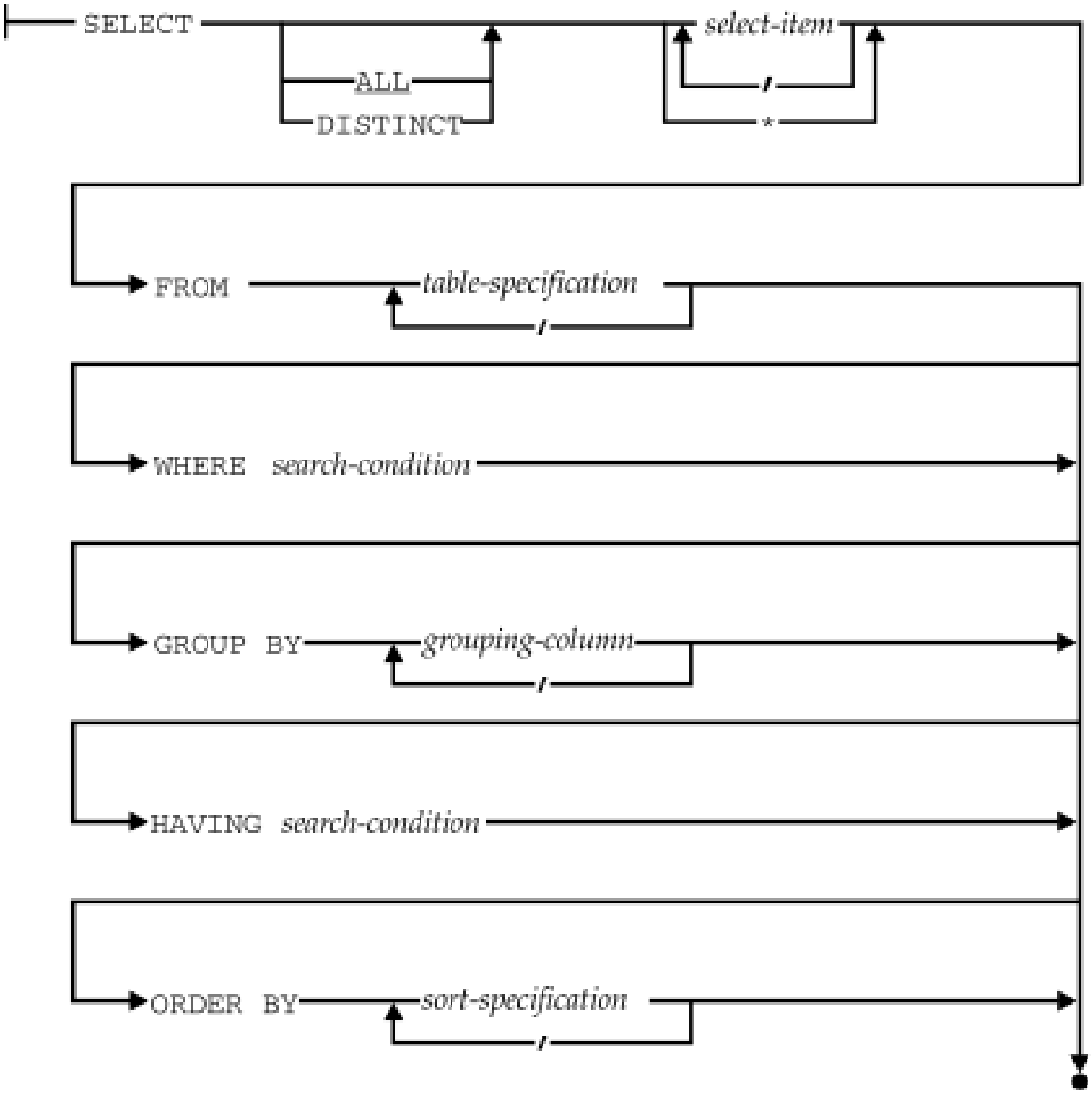
Comments

CITY	TARGET	SALES
Denver	300000.00	186042.00
New York	575000.00	692637.00
Chicago	800000.00	735042.00
Atlanta	350000.00	367911.00
Los Angeles	725000.00	835915.00

- To give you the ability to make notes in queries you are allowed to have comments.
- Comments are not executed
- A comment starts with '--' and ends with a newline (Can we start with '---'? At least MySQL doesn't recognize this as a comment.)
- They are also permitted within a query.

```
SELECT CITY, TARGET, SALES
FROM Offices -- The offices table
;
```

Complex SELECT Statement



SELECT Statement Explained

- The ***SELECT*** clause lists the data items to be retrieved by the SELECT statement.
- The ***FROM*** clause lists the tables and views that contain the data to be retrieved by the query.
  - From single table (6.1)
  - From multiple tables (6.2)
  - From views (chapter 8)
- The ***WHERE*** clause tells SQL to include only certain rows of data in the query results. A *search condition* is used to specify the desired rows.
  - Basic use (6.1)
  - Subqueries (6.3)
- The ***GROUP BY*** clause specifies a summary query (6.4).
- The ***HAVING*** clause tells SQL to include only certain groups produced by the GROUP BY clause in the query results (6.4)
- The ***ORDER BY*** clause sorts the query results based on the data in one or more columns (6.1).

SELECT Clause

- Select ALL: select all rows (default)

  select mgr from offices;

  select **ALL** mgr from offices;

mgr
104
105
106
108
108

- Select DISTINCT: remove duplicates

  select **DISTINCT** mgr from offices;

mgr
104
105
106
108

- Select *: select all columns

  select * from offices;

name	year(hire_date)
Sam Clark	2006
Mary Jones	2007
Bob Smith	2005
Larry Fitch	2007
Bill Adams	2006
Sue Smith	2004
Dan Roberts	2004
Tom Snyder	2008
Paul Cruz	2005
Nancy Angelli	2006

- Select *expression/function/constant*

  select city, sales**, sales*1.2** from offices;

city	sales	sales*1.2
Denver	186042.00	223250.400
New York	692637.00	831164.400
Chicago	735042.00	882050.400
Atlanta	367911.00	441493.200
Los Angeles	835915.00	1003098.000

  select name, **year**(hire_date) from salesreps;

  select 1 from offices;

1
1
1
1
1
1

- Use AS to rename a column

  Select empl_num AS 'Employee ID'

  From salesreps;

WHERE Clause

- The WHERE clause is used to specify the search conditions.
- If the search condition is **TRUE**, the row is included in the query results.
- If the search condition is **FALSE**, the row is excluded from the query results.
- If the search condition has a **NULL** (unknown) value, the row is excluded from from the query results.

select name, manager from salesreps;

name	manager
Sam Clark	*NULL*
Mary Jones	106
Bob Smith	106
Larry Fitch	106
Bill Adams	104
Sue Smith	108
Dan Roberts	104
Tom Snyder	101
Paul Cruz	104
Nancy Angelli	108

select name, manager
from salesrep
where manager = 106;

name	manager
Mary Jones	106
Bob Smith	106
Larry Fitch	106

Search Condition

- Comparison test (=, <>, <, <=, >, >=): Compares the values of one expression with the value of another expression.

select name, manager
from salesreps;

name	manager
Sam Clark	*NULL*
Mary Jones	106
Bob Smith	106
Larry Fitch	106
Bill Adams	104
Sue Smith	108
Dan Roberts	104
Tom Snyder	101
Paul Cruz	104
Nancy Angelli	108

select name, manager
from salesreps
where manager <> 106;

name	manager
Bill Adams	104
Sue Smith	108
Dan Roberts	104
Tom Snyder	101
Paul Cruz	104
Nancy Angelli	108

If either (left or right) of the two expressions Produces a NULL value, the comparison yields a NULL result.

Search Condition

- Range test (BETWEEN): Tests whether the value of an expression falls within a specified range of values.

select name, hire_date
from salesreps;

name	hire_date
Sam Clark	2006-06-14
Mary Jones	2007-10-12
Bob Smith	2005-05-19
Larry Fitch	2007-10-12
Bill Adams	2006-02-12
Sue Smith	2004-12-10
Dan Roberts	2004-10-20
Tom Snyder	2008-01-13
Paul Cruz	2005-03-01
Nancy Angelli	2006-11-14

select name, hire_date
from salesreps
Where hire_date between '2006-06-14' and '2007-12-31';

Use single quote for date

name	hire_date
Sam Clark	2006-06-14
Mary Jones	2007-10-12
Larry Fitch	2007-10-12
Nancy Angelli	2006-11-14

NOT BETWEEN: to check for values that fall outside the range.

Search Condition

- Set membership test (IN): Checks whether the value of an expression matches one of a set of values;

```
select name, manager
from salesreps
where manager in (104, 106);
```

name	manager
Mary Jones	106
Bob Smith	106
Larry Fitch	106
Bill Adams	104
Dan Roberts	104
Paul Cruz	104

Same as:

manager = 104 or manager = 106

Search Condition

- Pattern matching test (LIKE): Checks whether the value of a column containing **string** data matches a specified pattern.
  - The LIKE test must be applied to a column with a string data type.
- The percentage sign (%) wildcard character matches any sequences of zero or more characters.
- The underscore (_) wildcard character matches any single character.
- How to match the wildcard characters themselves as literal characters? – ESCAPE clause (Not all SQL products support ESCAPE.)

```
select order_num, product
from orders
where product like 'A$%BC%' ESCAPE '$';
```

The first % is treated as a literal percent sign;
The second functions as a wildcard.

LIKE Examples

- Name LIKE 'Jim Smith' e.g. Jim Smith
- Name LIKE '_im Smith' e.g. Tim Smith
- Name LIKE '___ Smith' e.g. Bob Smith
- Name LIKE '% Smith' e.g. Frank Smith
- Name LIKE '% S%' e.g. Brian Smart
- Name LIKE 'Bob %' e.g. Bob Martin
- Name LIKE '%' i.e. match anyone

- LIKE is more expensive than =
- If you are not using wildcards, always use = rather than LIKE.

Search Condition

- Null value test (IS NULL): Checks whether a column has a NULL value.

```
SELECT name
FROM salesreps
WHERE manager IS NULL
```

name
Sam Clark

Can we use this?

```
SELECT name
FROM salesreps
WHERE manager = NULL;
```

This is a legal statement, but it won't return the above table.

"**MySQL returned an empty result set**"

Compound Search Conditions (AND, OR, NOT)

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

TABLE 6-1 The AND Truth Table

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

TABLE 6-2 The OR Truth Table

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

TABLE 6-3 The NOT Truth Table

ORDER BY Clause

- Followed by a list of sort specifications separated by commas.
- By default, sort data in ascending sequence, but still can use ASC to specify an ascending sort.
- Use DESC to specify a descending sort.

```
SELECT city, region, (sales - target)
FROM offices
ORDER BY region, 3 DESC
```

city	region	(sales - target)
New York	Eastern	117637.00
Atlanta	Eastern	17911.00
Chicago	Eastern	-64958.00
Los Angeles	Western	110915.00
Denver	Western	-113958.00

```
SELECT city, region, (sales - target)
FROM offices
ORDER BY region DESC, 3 DESC
```

city	region	(sales - target)
Los Angeles	Western	110915.00
Denver	Western	-113958.00
New York	Eastern	117637.00
Atlanta	Eastern	17911.00
Chicago	Eastern	-64958.00

Single-Table Query Summary

- Format

  Select *columns/expression*

  From *tablename*

  Where *search_condition*

  Order by *columns*

- Query results are generated in this order:
  - The FROM clause is applied first
  - The WHERE clause is applied next
  - Then, the SELECT clause is applied
  - Finally, the ORDER BY clause is applied

Set Operation - UNION

- UNION combines query results from two or more queries.

```
SELECT MFR_ID, PRODUCT_ID
FROM PRODUCTS
WHERE PRICE > 2000.00
```

MFR_ID	PRODUCT_ID
ACI	4100Y
REI	2A44L
ACI	4100Z
REI	2A44R

```
SELECT DISTINCT MFR, PRODUCT
FROM ORDERS
WHERE AMOUNT > 30000.00
```

MFR	PRODUCT
REI	2A44L
REI	2A44R
IMM	775C

```
SELECT MFR_ID, PRODUCT_ID
FROM PRODUCTS
WHERE PRICE > 2000.00
UNION
SELECT DISTINCT MFR, PRODUCT
FROM ORDERS
WHERE AMOUNT > 30000.00
```

MFR_ID	PRODUCT_ID
ACI	4100Y
ACI	4100Z
IMM	775C
REI	2A44L
REI	2A44R

UNION

- The **data type** of each column selected from the first table must be the same as the data type of the corresponding column selected from the second table.
- The **column names** of the two queries do not have to be identical.
- The UNION operation automatically **eliminates duplicates**, unlike the SELECT clause.
- Neither of the two tables can be sorted with the ORDER By clause.
- Combined query results can be sorted by adding ORDER BY after the second SELECT statement.

Other Set Operations

- INTERSECT
- EXCEPT or MINUS
- Not all SQL implementations support these two operations.