Lecture 09

SQL (II)

- C: Insert a new order
- R: Show all orders
- U: Update Disc. rate
- D: Delete the new order

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

Review

- SQL
  - C: Create
    - create database, create table,
    - **INSERT INTO** table … **VALUES** …
  - R: Retrieve
    - **SELECT** … **FROM** table …
  - U: Update
    - **UPDATE** table **SET** …
  - D: Delete
    - **DELETE FROM** table …

Advanced SELECT

- SELECT is the real heart of SQL.[1]
- SELECT [ALL | DISTINCT] `SELECTion_list`
  FROM `{tableName | viewName} [, …]`
  [WHERE `criteria`]
  [GROUP BY `columnName [, …]`]
  [HAVING `criteria`]
  [ORDER BY `{columnName | SELECT_list_number}`]
  [DESC]
- SQL is a free-form language. However, the order of these clauses has to be in the order shown above to be syntax correct.

[1] The Practical SQL Handbook, Judith et al. Addison-Wesley, ISBN 0201447878

1. SELECTion_list

- SELECT [ALL | DISTINCT] SELECTion_list
  FROM {tableName | viewName} [, …]
  1. SELECTion_list contains column names to be picked. The order of column names can be arranged in any order as you pleased.
  2. SELECTion_list can also be used to specify new table display labels
  3. SELECTion_list can do computations: (),*/,+-, functions
  4. SELECTion_list can also be specified by using tableName.columnName;

2. FROM clause

- SELECT [ALL | DISTINCT] SELECTion_list
  FROM {tableName | viewName} [, …]
  - tableName can be given an alias, which is important when we're dealing with joins (to be discussed)

SELECT **O**.Cust FROM Orders **O**

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

1. SELECT Qty, Amt, Prod FROM Orders
2. SELECT Qty **AS** Quantity, Amt **AS** Amount, Prod **AS** Product FROM Orders
3. SELECT **AVG**(Amt) FROM Orders;
   SELECT OrderNbr, Qty***Amt FROM Orders
4. SELECT **Orders**.Qty, **Orders**.Prod, **Orders**.Amt FROM Orders

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

3. WHERE clause

- SELECT [ALL | DISTINCT] SELECTion_list
  FROM {tableName | viewName} [, …]
  [WHERE criteria]
  - Criteria can have
    - Comparison operators (=, <, >, <=, >=, <>)
    - Combinations or logical negations (AND, OR, NOT)
    - Lists (IN, NOT IN)
    - Ranges (BETWEEN and NOT BETWEEN)
    - Unknown values (IS NULL and IS NOT NULL)
    - Character matches (LIKE and NOT LIKE)
      - %: matching any length of characters
      - _: matching one character

	OfficeNbr	City	State	Region	Target	Sales	Phone
1	1	Denver	CO	West	3000000	130000	9705863341
2	2	New York	NY	East	200000	300000	2129425574
3	57	Dallas	TX	West	0	0	2147815342

```
SELECT * FROM offices;
SELECT * FROM offices WHERE target > 300000;
SELECT * FROM offices WHERE target <= 200000;
SELECT * FROM offices WHERE region='west' AND target>1000000;
SELECT * FROM offices WHERE state IN ('TX', 'NY');
```

9

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

```
SELECT * FROM orders;
SELECT * FROM orders WHERE qty BETWEEN 1 AND 5;
SELECT * FROM orders WHERE qty IS NULL;
SELECT * FROM orders WHERE prod LIKE '%crane%';
```

10

4. ORDER BY clause

- SELECT [ALL | DISTINCT] `SELECTion_list`
  FROM `{tableName | viewName} [, …]`
  [WHERE `criteria`]
  [GROUP BY `columnName [, …]`]
  [HAVING `critera`]
  [ORDER BY `{columnName | SELECT_list_number}`]
  [DESC]
  - Column name or SELECT_list_number
  - DESC suggested the sorted list should be in descending order
  - The list can have multiple column names

11

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

```
SELECT * FROM orders;
SELECT * FROM orders ORDER BY amt;
SELECT * FROM orders ORDER BY qty desc;
SELECT * FROM orders ORDER BY prod;

SELECT * FROM orders ORDER BY cust, amt;
SELECT * FROM orders ORDER BY cust, amt DESC;
SELECT * FROM orders ORDER BY 2, 5 DESC;
SELECT DISTINCT cust FROM orders;
```

12

5. GROUP BY clause

- SELECT [ALL | DISTINCT] `SELECTion_list`
  FROM `{tableName | viewName} [, …]`
  [GROUP BY `columnName [, …]`]
  [HAVING `criteria`]
  - Aggregate using the entire table
    - SUM(exp), AVG(exp), COUNT(exp), COUNT(*), MAX(exp), MIN(exp)
    - SUM(DISTINCT exp), AVG(DISTINCT exp), COUNT(DISTINCT exp)
  - Aggregate with GROUP BY
  - WHERE criteria that is a pre-filtering criteria to SELECT records before aggregation; while HAVING criteria is post-filtering criteria to SELECT aggregated records after aggregation.

13

Examples
Aggregation of the entire table

- Summation of sales amount
  - SELECT SUM(amt) AS total FROM orders;
- Count the number of orders
  - SELECT COUNT(cust) FROM orders;
- NULLs are not counted!
  - SELECT COUNT(qty), COUNT(*) FROM orders;
- Various aggregated functions applied
  - SELECT MAX(amt), MIN(amt), SUM(amt), AVG(amt) FROM orders WHERE prod LIKE '%CRANE%';
- Find distinct customers
  - SELECT COUNT(DISTINCT cust), COUNT(cust) FROM orders;

14

Example
Grouping

- Find the total amount of product purchased by each customer
  - SELECT cust, SUM(amt), COUNT(cust), SUM(qty) FROM orders GROUP BY cust;
- Find the number of distinct products purchased by each customer
  - SELECT cust, COUNT(DISTINCT prod) FROM orders GROUP BY cust;
- Find the distinct products purchased by each customer
  - SELECT cust, prod FROM orders GROUP BY cust, prod;
- Find distinct customers of each product purchased
  - SELECT prod, cust FROM orders GROUP BY prod, cust;

15

Example
Having

- Find total amount of purchase of each customer
  - SELECT cust, SUM(amt) AS total FROM orders GROUP BY cust;
- Find out who is our VIP
  - SELECT cust, SUM(amt) AS total FROM orders GROUP BY cust HAVING total > 2000000; (Invalid in T-SQL)
  - SELECT cust, SUM(amt) AS total FROM orders GROUP BY cust HAVING SUM(amt) > 2000000;
- Compare the following SQL statement
  - SELECT cust, SUM(amt) AS total FROM orders WHERE amt > 2000000 GROUP BY cust;

16

Join

Join

- List all orders, showing order number, amount, name and credit limit of customer

SELECT OrderNbr, Amt, Company, creditLimit
FROM orders, customers
WHERE *Cust*=*CustNbr*

CustNbr	**Company**	CustRep	**creditLimit**
211	Connor Co	89	50000
522	AmaratungaEnterprise	89	40000
890	Feni Fabricators	53	1000000

OrderNbr	***Cust***	Prod	Qty	**Amt**	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

Join

- Join allows us to SELECT data FROM multiple tables in a single SELECT statement
- Any columns in tables can be joined, as long as data types match
- The joined attributes are often keys.
- Good joins:
  - Join column is usually key column: either primary key or foreign key
  - Join columns must have compatible data types
  - **Null will never join!!**

Source: MIT 1.264 Lecture Notes

CustNbr	**Company**	CustRep	**creditLimit**
211	Connor Co	89	50000
522	AmaratungaEnterprise	89	40000
890	Feni Fabricators	53	1000000

OrderNbr	***Cust***	Prod	Qty	**Amt**	Disc
1	211	Bulldozer	7	31000	0.2
2	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
5	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4
7	211	TinyCrane	1	100000	0.2
6	522	HugeCrane	1	20000000	0.1

SELECT OrderNbr, Amt, Company, creditLimit
FROM orders, customers
WHERE *Cust*=*CustNbr*

OrderNbr	Amt	Company	creditLimit
1	31000	Connor Co	50000
2	4000	AmaratungaEnterprise	40000
4	500000	AmaratungaEnterprise	40000
5	200	Connor Co	50000
7	100000	Connor Co	50000
6	20000000	AmaratungaEnterprise	40000

Major kinds of join

- **INNER JOIN**
- **OUTER JOIN**
  - **LEFT JOIN**
  - **RIGHT JOIN**
  - **FULL JOIN**
- *CROSS JOIN (will not be discussed)*
- *NATURAL JOIN (will not be discussed)*


```
+------------+--------------+------+
| name       | phone        | pid  |
+------------+--------------+------+
| Mr. Brown  | 01225 708225 |    1 |
| Ms. Smith  | 01225 899360 |    2 |
| Mr. Pullen | 01380 724040 |    3 |
+------------+--------------+------+
```

```
+------+------+----------------------+
| pid  | spid | selling              |
+------+------+----------------------+
|    1 |    1 | Old House Farm       |
|    3 |    2 | The Willows          |
|    3 |    3 | Tall Trees           |
|    3 |    4 | The Melksham Florist |
|    4 |    5 | Dun Roamin           |
+------+------+----------------------+
```

Note both tables have a field pid.

```
SELECT name, phone, selling
FROM people, properties
WHERE people.pid = properties.pid;
```

```
+------------+--------------+----------------------+
| name       | phone        | selling              |
+------------+--------------+----------------------+
| Mr. Brown  | 01225 708225 | Old House Farm       |
| Mr. Pullen | 01380 724040 | The Willows          |
| Mr. Pullen | 01380 724040 | Tall Trees           |
| Mr. Pullen | 01380 724040 | The Melksham Florist |
+------------+--------------+----------------------+
```


Inner join

- All previous examples are “inner join”s. The following SQL statements are equivalent:

```
SELECT OrderNbr, Amt, Company, creditLimit
FROM orders, customers
WHERE Cust=CustNbr
```

```
SELECT OrderNbr, Amt, Company, creditLimit
FROM orders INNER JOIN customers
ON Cust=CustNbr
```

```
SELECT OrderNbr, Amt, Company, creditLimit
FROM orders JOIN customers
ON Cust=CustNbr
```

- “Inner join” joins data FROM different tables where the data in the joined column are “COMMON” or “INTERSECTION” between different tables.
- This is the common usage when the stored data are proper!

Outer Join - LEFT JOIN

- There are times data on the joined columns do not match properly.

```
SELECT name, phone, selling
FROM people LEFT JOIN properties
ON people.pid = properties.pid;
```

```
+------------+--------------+------+
| name       | phone        | pid  |
+------------+--------------+------+
| Mr. Brown  | 01225 708225 |    1 |
| Ms. Smith  | 01225 899360 |    2 |
| Mr. Pullen | 01380 724040 |    3 |
+------------+--------------+------+
```

```
+------+------+----------------------+
| pid  | spid | selling              |
+------+------+----------------------+
|    1 |    1 | Old House Farm       |
|    3 |    2 | The Willows          |
|    3 |    3 | Tall Trees           |
|    3 |    4 | The Melksham Florist |
|    4 |    5 | Dun Roamin           |
+------+------+----------------------+
```

```
+------------+--------------+----------------------+
| name       | phone        | selling              |
+------------+--------------+----------------------+
| Mr. Brown  | 01225 708225 | Old House Farm       |
| Ms. Smith  | 01225 899360 | NULL                 |
| Mr. Pullen | 01380 724040 | The Willows          |
| Mr. Pullen | 01380 724040 | Tall Trees           |
| Mr. Pullen | 01380 724040 | The Melksham Florist |
+------------+--------------+----------------------+
```

Outer Join – RIGHT JOIN

- There are times data on the joined columns do not match properly.

```
SELECT name, phone, selling
FROM people RIGHT JOIN properties
ON people.pid = properties.pid;
```

name	phone	pid
Mr. Brown	**01225 708225**	**1**
Ms. Smith	01225 899360	2
Mr. Pullen	**01380 724040**	**3**

pid	spid	selling
1	**1**	**Old House Farm**
3	2	The Willows
3	3	Tall Trees
3	4	The Melksham Florist
4	5	Dun Roamin

name	phone	selling
Mr. Brown	01225 708225	Old House Farm
Mr. Pullen	01380 724040	The Willows
Mr. Pullen	01380 724040	Tall Trees
Mr. Pullen	01380 724040	The Melksham Florist
NULL	**NULL**	**Dun Roamin**

Outer Join – FULL JOIN

```
SELECT name, phone, selling
FROM people FULL JOIN properties
ON people.pid = properties.pid;
```

name	phone	pid
Mr. Brown	**01225 708225**	**1**
Ms. Smith	01225 899360	2
Mr. Pullen	**01380 724040**	**3**

pid	spid	selling
1	**1**	**Old House Farm**
3	**2**	**The Willows**
3	**3**	**Tall Trees**
3	**4**	**The Melksham Florist**
4	5	Dun Roamin

name	phone	selling
Mr. Brown	**01225 708225**	**Old House Farm**
Ms. Smith	**01225 899360**	**NULL**
Mr. Pullen	**01380 724040**	**The Willows**
Mr. Pullen	**01380 724040**	**Tall Trees**
Mr. Pullen	**01380 724040**	**The Melksham Florist**
NULL	**NULL**	**Dun Roamin**

Three-Way Join

- List orders over $25,000, including the name of the salesperson who took the order and the name of the customer who placed it.

```
SELECT OrderNbr, Amt, Company, Name
FROM Orders, Customers, SalesReps
WHERE Cust=CustNbr AND CustRep=RepNbr AND Amt>=25000;
```

  - So far it is okay because we use different names on the joined field in different tables. What if we need to join two tables with identical field names in different tables?
  - Use tableName.fieldName = tableName2.fieldName2!
  - Use table alias helps reducing typing
    - SELECT cust, creditLimit FROM table1 a, table2 b where a.cust=b.cust;

Self-join

EmpNbr	Name	Title	Mgr
105	Mary Smith	Analyst	104
109	Jill Jones	Sr Analyst	107
107	Pat Brown	Manager	111
104	Sally Silver	Manager	111
111	Eileen Howe	President	105

To list the analysts and the names of their managers

Trial #1

```
SELECT name, name FROM exp1, exp1 WHERE Mgr=EmpNbr;
```

Trial #2

```
SELECT emp.name AS emp, mgr.name AS mgr
FROM exp1 emp, exp1 mgr
WHERE emp.Mgr=mgr.EmpNbr;
```