Lecture 10

SQL (III)

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 …

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, and the operation makes sense.
- The joined attributes don't have to be keys, although they usually are.
- 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

Join

- List all orders, showing order number, amount, name and credit limit of customer
  - Orders have order number and amount, but no customer names or credit limits
  - Customers has customer names and credit limit, but no order info

CustNbr	Company	CustRep	creditLimit
523	Enterprise	53	100000
522	AmaratungaEnterprise	89	75000
890	panhouse	53	1000000

RepNbr	Name	RepOffice	Quota	Sales
53	Bill Smith	1	100000	0
89	Jen Jones	2	50000	130000

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000	0.2
14	522	riveter	2	4000	0.3
4	522	Crane	1	500000	0.4
15	211	TestItem	NULL	200	NULL
8	523	Crane	1	500000	0.4

Join with 3 Tables

- 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;
```

Sub-query

Sub-query

- A sub-query is an additional method for handling multi-table manipulations. It is a SELECT statement that nests
  - Inside the WHERE or HAVING (and in some systems, SELECT or FROM) clause of another SELECT statement;
  - ! Inside an INSERT, UPDATE, or DELETE statement; or
  - Inside another sub-query.
- Sub-query always returns a single column of results when used in a criterion

Examples

- Find products sold (Amt) above average of all products sold.
  - Step 1: find the average
    - SELECT AVG(Amt) FROM orders;
  - Step 2: find orders amount that are above average
    - SELECT * FROM orders WHERE Amt > (SELECT AVG (Amt) FROM orders);
- List companies that have received discount rate of 0.4.
  - Step 1: find orders that has Disc = 0.4
    - SELECT DISTINCT cust FROM orders WHERE disc=0.4;
  - Step 2: find customers with disc=0.4
    - SELECT custnbr, company FROM customers WHERE custnbr IN (SELECT DISTINCT cust FROM orders WHERE disc=0.4);
  - We can in fact often use joins to do the same
    - SELECT custnbr, company FROM customers p, orders q WHERE p.custnbr=q.cust AND q.disc=0.4;

Customers

```
+---------+----------------------+---------+-------------+
| CustNbr | Company              | CustRep | creditLimit |
+---------+----------------------+---------+-------------+
|     522 | AmaratungaEnterprise |      89 |       75000 |
|     890 | Feni Fabricators     |      53 |     1000000 |
+---------+----------------------+---------+-------------+
```

salesreps

```
+--------+------------+-----------+--------+--------+
| RepNbr | Name       | RepOffice | Quota  | Sales  |
+--------+------------+-----------+--------+--------+
|     53 | Bill Smith |         1 | 100000 |      0 |
|     89 | Jen Jones  |         2 |  50000 | 130000 |
+--------+------------+-----------+--------+--------+
```

```
CREATE TABLE CUST1 (
       CustNbr INT PRIMARY KEY,
       Company VARCHAR(40),
       CustRep VARCHAR(40),
       CreditLimit INT
);

INSERT INTO CUST1 (
       SELECT CustNbr, Company, Name, CreditLimit
              FROM Customers, SalesReps
              WHERE Customers.CustRep=SalesReps.RepNbr
);
```


Views

Views (1/2)

- Views are virtual tables that present data in denormalized form to users
- They are not separate copies of the data; they reference the data in the underlying tables.
- Database stores definition of view; the data is updated when the underlying tables are updated
- Advantages:
  - Designed to meet specific needs of specific users
  - Much simpler queries for users on views constructed for them
  - Security: give access only for data in views
  - Independence: layers user or program away from change in underlying tables

Source: MIT 1.264 Lecture Notes

Views (2/2)

- To create a view:
  - CREATE VIEW *viewName [(columnName [, columnName]...)]* AS *select SQL statement*;
- Once a view is created, the view can be used almost the same as a table.
- To delete a view:
  - DROP VIEW *viewName*;

13

Examples

- Create an alias for customers table
  - CREATE VIEW c1 AS
    SELECT * FROM customers;
- Create a ID-Name view
  - CREATE VIEW c2 (ID, Name) AS
    SELECT CustNbr, Company FROM customers;
- A more elaborated view construction
  - CREATE VIEW customerOrders AS
    SELECT CustNbr, Company, Name, OrderNbr, Prod, Qty, Amt
    FROM Customers, SalesReps, Orders
    WHERE CustRep = RepNbr AND CustNbr = Cust;

14

Exercise

- Construct a SQL statement for finding the number of orders placed by each customer. Show customer's ID, customer's name, and number of orders placed.
  - SELECT cust, company, COUNT(*)
    FROM orders, customers
    WHERE Cust=CustNbr
    GROUP BY cust;
- Create a view noOrders with the previous SQL statement
  - CREATE VIEW noOrders (custID, Company, Orders) AS
    SELECT cust, company, COUNT(*) FROM orders, customers
    WHERE Cust=CustNbr GROUP BY cust;
- Find the customer with the most orders
  - SELECT * FROM noOrders WHERE orders=
    (SELECT MAX(Orders) FROM noOrders);

15

View Updates (1/2)

- It is possible to do updating through views
  - CREATE VIEW tinyCrane AS
    SELECT * FROM orders WHERE prod='tinyCrane';
  - UPDATE tinycrane SET cust=523 WHERE ordernbr=7;
- What happens here?
  - UPDATE tinyCrane SET prod='HugeCrane' WHERE ordernbr=7;
  - The record vanishes from the view.
  - To prevent this from happening
    CREATE VIEW tinyCrane1 AS
    SELECT * FROM orders WHERE prod='tinycrane'
    WITH CHECK OPTION;

16

View Update (2/2)

- Not all views can be updated. A view is read-only if:
  - DISTINCT is in the SELECT statement
  - Expressions (averages, totals, etc.)
  - References to views that are not updatable
  - GROUP BY or HAVING clauses
  - Sometimes: References to more than one table (defeats purpose)

17

Source: MIT 1.264 Lecture Notes

MS-SQL Server Management Studio

18

```
+----------+-----------------------+----------+-------------+
| CustNbr | Company               | CustRep | creditLimit |
+----------+-----------------------+----------+-------------+
|      522 | AmaratungaEnterprise |       89 |       75000 |
|      523 | Enterprise            |       53 |      100000 |
|      890 | Feni Fabricators      |       53 |     1000000 |
+----------+-----------------------+----------+-------------+
+-----------+---------+-------+--------+---------+--------+------------+
| OfficeNbr | City    | State | Region | Target  | Sales  | Phone      |
+-----------+---------+-------+--------+---------+--------+------------+
|         1 | Denver  | CO    | West   | 3000000 | 130000 | 9705863341 |
|         2 | New York | NY    | East   |  200000 | 300000 | 2129425574 |
|        55 | Dallas  | NULL  | West   |  200000 |      0 | NULL       |
|        57 | Dallas  | TX    | West   |       0 |      0 | 2147815342 |
+-----------+---------+-------+--------+---------+--------+------------+
| 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 |
|        6 |  522 | HugeCrane |    1 | 20000000 |  0.1 |
|        7 |  211 | TinyCrane |    1 |   100000 |  0.2 |
|        8 |  523 | Crane     |    1 |   500000 |  0.4 |
|        9 |  523 | Bulldozer |    2 |   100000 | NULL |
|       10 |  523 | TinyCrane |    5 |   500000 | NULL |
|       11 |  522 | TinyCrane |    7 |   210000 | NULL |
|       12 |  211 | Bulldozer |    2 |   100000 | NULL |
|       13 |  211 | TinyCrane |    2 |   100000 | NULL |
+----------+------+-----------+------+----------+------+
+--------+-----------+-----------+-------+--------+
| RepNbr | Name      | RepOffice | Quota | Sales  |
+--------+-----------+-----------+-------+--------+
|     53 | Bill Smith |         1 | 100000 |      0 |
|     89 | Jen Jones  |         2 |  50000 | 130000 |
+--------+-----------+-----------+-------+--------+
```

19

Demonstration (1/2)

New Query → Query → Design Query in Editors

1. `INSERT INTO orders VALUES (7, 522, Little Crane, 1, 2000000, 0.1);`
2. `SELECT RepNbr, Name, RepOffice, Quota, Sales, Sales-Quota FROM SalesReps;`
3. `SELECT RepNbr, Name, RepOffice, Quota, Sales, Sales-Quota FROM SalesReps WHERE Sales<Quota;`
4. `SELECT Avg(Amt) FROM Orders;`
5. `SELECT Avg(Orders.Amt) AS AmtOfAvg FROM Orders WHERE Cust=211;`
6. `DELETE FROM Customers WHERE CustNbr=522;`
7. `UPDATE Customers SET creditLimit = creditLimit*2 WHERE Company='AmaratungaEnterprise';`
8. `SELECT CustNbr AS 'Customer Number', Company, CustRep AS 'Customer Representative', creditLimit AS 'Credit Limit' FROM Customers WHERE Custnbr=522;`

20

Demonstration (2/2)

9. `SELECT * FROM Offices WHERE target>30000;`
10. `SELECT * FROM Offices WHERE State In ('TX', 'NY');`
11. `SELECT * FROM orders WHERE Qty BETWEEN 1 AND 5;`
12. `SELECT * FROM orders WHERE Prod LIKE '%crane%';`
13. `SELECT * FROM Orders ORDER BY Cust, Amt;`
14. `SELECT * FROM Orders ORDER BY Amt, Cust;`
15. `SELECT Region, Count(OfficeNbr) FROM Offices GROUP BY Region;`
16. `SELECT Cust, Sum(Amt) FROM Orders GROUP BY Cust;`
17. `SELECT OrderNbr, Company, Prod, Amt FROM Orders, Customers WHERE Customers.CustNbr = Orders.Cust;`

21

Exercises

Basic

1. List all customers with credit limit > 100000
2. Find offices in Dallas
3. List all products with 'crane' in it.
4. List sales representatives with sales < quota

Grouping

1. List number of offices in each city
2. List max, min, and average of amount of each customer's order
3. List the number of times each product got ordered.

22

ETL

Extract, Transform, Load

23

http://en.wikipedia.org/wiki/Extract,_transform,_load

ETL

- Extract, Transform, and Load (ETL) is a process in data warehousing that involves
  - Extracting data from outside sources
  - Transforming it to fit business needs (which can include quality levels)
  - Loading it into the end target

24

Extract

- Extract the data from the outside sources
  - May not be normalized
  - May not be of good quality
    - Need to clean it up
  - May be of different formats
- Extraction converts the data into a format for transformation processing.
- Parse the extracted data to check if the data meets an expected pattern or structure. If not, the data is rejected entirely.

25

http://en.wikipedia.org/wiki/Extract,_transform,_load

Transform

- Some data sources will require very little or even no manipulation of data.
- Transform involves
  - Selecting only certain columns
  - Translating coded values (e.g. 1 → male, 2 → female)
  - Encoding free-form values (e.g. male → 1, female → 2)
  - Deriving a new calculated value (e.g. sale_amount = qty * unit_price)
  - Joining together data from multiple sources (e.g., lookup, merge)
  - Aggregation (e.g. total sales for each store)

26

Transform (cont'd)

  - Filtering, Sorting
  - Generating surrogate key values
  - Transposing or pivoting (turning multiple columns into multiple rows or vice versa)
  - Splitting a column into multiple columns (e.g., putting a comma-separated list specified as a string in one column as individual values in different columns)
  - Data validation
- Most of the above transformations itself might result in an exception, (e.g. when a code-translation parses an unknown code in the extracted data.)
  - Reject entire dataset or partial dataset

27

Load

- Move transformed data into the target database.

28

Examples

- A table wants to break into the following three entities:
  - Product, ProductType, NutritionFacts
  - The relationship between Product and NutritionFacts is many-to-many → need an association table
- We need four tables:
  - Product, ProductType, NutritionFacts, and Product_NutritionFacts
  - How are we going to ETL from old table to the four new tables?
- The steps involve:
  - Transcoding, Transposition, Generating surrogate key values
  - Further, we probably need to demonstrate how to deduplicate records …

29