**STRUCTURED DATA**

Accessing structured data with SQL is quite different from the full text search of documents on the Web. Structured data in the relational model means data that can be represented in tables -- rows and columns. Each row in a table represents a different object, and the columns represent various "attributes" of the object. The columns have names and integrity constraints that specify valid values.

Since the column values are named and are represented in a consistent format, you can select rows very precisely, based on their contents. This is especially helpful in dealing with numeric data. You can also join together data from different tables, based on matching column values. You can do useful types of analysis, listing objects that are in one table and missing (or present, or have specific attributes) from a related table. You can extract from a large table precisely those rows of interest, regrouping them and generating simple statistics on them.

This document contains examples of:

* [creating a table](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#create table)
* [creating a view](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#create view)
* [inserting rows](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#insert)
* [updating rows](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#update)
* [deleting rows](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#delete)
* [commit -- making changes permanent](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#commit)
* [rollback -- undoing temporary changes](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#rollback)
* [enforcing integrity constraints](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#constraint)
* [using an Embedded C program](http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm#program)

**INTERACTIVE SQL EXAMPLES**

**create a table to store information about weather observation stations:**   
-- No duplicate ID fields allowed

CREATE TABLE STATION   
(ID INTEGER PRIMARY KEY,   
CITY CHAR(20),   
STATE CHAR(2),   
LAT\_N REAL,   
LONG\_W REAL);

**populate the table STATION with a few rows:**

INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);   
INSERT INTO STATION VALUES (44, 'Denver', 'CO', 40, 105);   
INSERT INTO STATION VALUES (66, 'Caribou', 'ME', 47, 68);

**query to look at table STATION in undefined order:**

SELECT \* FROM STATION;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **CITY** | **STATE** | **LAT\_N** | **LONG\_W** |
| 13 | Phoenix | AZ | 33 | 112 |
| 44 | Denver | CO | 40 | 105 |
| 66 | Caribou | ME | 47 | 68 |

**query to select Northern stations (Northern latitude > 39.7):**   
-- selecting only certain rows is called a "restriction".

SELECT \* FROM STATION   
WHERE LAT\_N > 39.7;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **CITY** | **STATE** | **LAT\_N** | **LONG\_W** |
| 44 | Denver | CO | 40 | 105 |
| 66 | Caribou | ME | 47 | 68 |

**query to select only ID, CITY, and STATE columns:**   
-- selecting only certain columns is called a "projection".

SELECT ID, CITY, STATE FROM STATION;

|  |  |  |
| --- | --- | --- |
| **ID** | **CITY** | **STATE** |
| 13 | Phoenix | AZ |
| 44 | Denver | CO |
| 66 | Caribou | ME |

**query to both "restrict" and "project":**

SELECT ID, CITY, STATE FROM STATION   
WHERE LAT\_N > 39.7;

|  |  |  |
| --- | --- | --- |
| **ID** | **CITY** | **STATE** |
| 44 | Denver | CO |
| 66 | Caribou | ME |

**create another table to store normalized temperature and precipitation data:**   
-- ID field must match some STATION table ID   
(so name and location will be known).   
-- allowable ranges will be enforced for other values.   
-- no duplicate ID and MONTH combinations.   
-- temperature is in degrees Fahrenheit.   
-- rainfall is in inches.

CREATE TABLE STATS   
(ID INTEGER REFERENCES STATION(ID),   
MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),   
TEMP\_F REAL CHECK (TEMP\_F BETWEEN -80 AND 150),   
RAIN\_I REAL CHECK (RAIN\_I BETWEEN 0 AND 100),   
PRIMARY KEY (ID, MONTH));

**populate the table STATS with some statistics for January and July:**

INSERT INTO STATS VALUES (13, 1, 57.4, 0.31);   
INSERT INTO STATS VALUES (13, 7, 91.7, 5.15);   
INSERT INTO STATS VALUES (44, 1, 27.3, 0.18);   
INSERT INTO STATS VALUES (44, 7, 74.8, 2.11);   
INSERT INTO STATS VALUES (66, 1, 6.7, 2.10);   
INSERT INTO STATS VALUES (66, 7, 65.8, 4.52);

**query to look at table STATS in undefined order:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .31 |
| 13 | 7 | 91.7 | 5.15 |
| 44 | 1 | 27.3 | .18 |
| 44 | 7 | 74.8 | 2.11 |
| 66 | 1 | 6.7 | 2.1 |
| 66 | 7 | 65.8 | 4.52 |
|  |  |  |  |

**query to look at table STATS, picking up location information by joining with table STATION on the ID column:**   
-- matching two tables on a common column is called a "join".   
-- the column names often match, but this is not required.   
-- only the column values are required to match.

SELECT \* FROM STATION, STATS   
WHERE STATION.ID = STATS.ID;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **CITY** | **ST** | **LAT\_N** | **LONG\_W** | **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | Phoenix | AZ | 33 | 112 | 13 | 1 | 57.4 | .31 |
| 13 | Phoenix | AZ | 33 | 112 | 13 | 7 | 91.7 | 5.15 |
| 44 | Denver | CO | 40 | 105 | 44 | 1 | 27.3 | .18 |
| 44 | Denver | CO | 40 | 105 | 44 | 7 | 74.8 | 2.11 |
| 66 | Caribou | ME | 47 | 68 | 66 | 1 | 6.7 | 2.1 |
| 66 | Caribou | ME | 47 | 68 | 66 | 7 | 65.8 | 4.52 |

**query to look at the table STATS, ordered by month and greatest rainfall, with columns rearranged:**

SELECT MONTH, ID, RAIN\_I, TEMP\_F   
FROM STATS   
ORDER BY MONTH, RAIN\_I DESC;

|  |  |  |  |
| --- | --- | --- | --- |
| **MONTH** | **ID** | **RAIN\_I** | **TEMP\_F** |
| 1 | 66 | 2.1 | 6.7 |
| 1 | 13 | .31 | 57.4 |
| 1 | 44 | .18 | 27.3 |
| 7 | 13 | 5.15 | 91.7 |
| 7 | 66 | 4.52 | 65.8 |
| 7 | 44 | 2.11 | 74.8 |
|  |  |  |  |

**query to look at temperatures for July from table STATS, lowest temperatures first, picking up city name and latitude by joining with table STATION on the ID column:**

SELECT LAT\_N, CITY, TEMP\_F   
FROM STATS, STATION   
WHERE MONTH = 7   
AND STATS.ID = STATION.ID   
ORDER BY TEMP\_F;

|  |  |  |
| --- | --- | --- |
| **LAT\_N** | **CITY** | **TEMP\_F** |
| 47 | Caribou | 65.8 |
| 40 | Denver | 74.8 |
| 33 | Phoenix | 91.7 |
|  |  |  |

**query to show MAX and MIN temperatures as well as average rainfall for each station:**

SELECT MAX(TEMP\_F), MIN(TEMP\_F), AVG(RAIN\_I), ID   
FROM STATS   
GROUP BY ID;

|  |  |  |  |
| --- | --- | --- | --- |
| **MAX(TEMP\_F)** | **MIN(TEMP\_F)** | **AVG(RAIN\_I)** | **ID** |
| 91.7 | 57.4 | 2.73 | 13 |
| 74.8 | 27.3 | 1.145 | 44 |
| 65.8 | 6.7 | 3.31 | 66 |
|  |  |  |  |

**query (with subquery) to show stations with year-round average temperature above 50 degrees:**   
-- rows are selected from the STATION table based on related values in the STATS table.

SELECT \* FROM STATION   
WHERE 50 < (SELECT AVG(TEMP\_F) FROM STATS   
WHERE STATION.ID = STATS.ID);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **CITY** | **ST** | **LAT\_N** | **LONG\_W** |
| 13 | Phoenix | AZ | 33 | 112 |
| 44 | Denver | CO | 40 | 105 |
|  |  |  |  |  |

**create a view (derived table or persistent query) to convert Fahrenheit to Celsius and inches to centimeters:**

CREATE VIEW METRIC\_STATS (ID, MONTH, TEMP\_C, RAIN\_C) AS   
SELECT ID,   
MONTH,   
(TEMP\_F - 32) \* 5 /9,   
RAIN\_I \* 0.3937   
FROM STATS;

**query to look at table STATS in a metric light (through the new view):**

SELECT \* FROM METRIC\_STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_C** | **RAIN\_C** |
| 13 | 1 | 14.1111111 | .122047 |
| 13 | 7 | 33.1666667 | 2.027555 |
| 44 | 1 | -2.6111111 | .070866 |
| 44 | 7 | 23.7777778 | .830707 |
| 66 | 1 | -14.055556 | .82677 |
| 66 | 7 | 18.7777778 | 1.779524 |
|  |  |  |  |

**another metric query restricted to January below-freezing (0 Celsius) data, sorted on rainfall:**

SELECT \* FROM METRIC\_STATS   
WHERE TEMP\_C < 0 AND MONTH = 1   
ORDER BY RAIN\_C;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_C** | **RAIN\_C** |
| 44 | 1 | -2.6111111 | .070866 |
| 66 | 1 | -14.055556 | .82677 |
|  |  |  |  |

**Interactive SQL Update Examples**

**update all rows of table STATS to compensate for faulty rain gauges known to read 0.01 inches low:**

UPDATE STATS SET RAIN\_I = RAIN\_I + 0.01;

**and take a look:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .32 |
| 13 | 7 | 91.7 | 5.16 |
| 44 | 1 | 27.3 | .19 |
| 44 | 7 | 74.8 | 2.12 |
| 66 | 1 | 6.7 | 2.11 |
| 66 | 7 | 65.8 | 4.53 |
|  |  |  |  |

**update one row, Denver's July temperature reading, to correct a data entry error:**

UPDATE STATS SET TEMP\_F = 74.9   
WHERE ID = 44   
AND MONTH = 7;

**and take a look:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .32 |
| 13 | 7 | 91.7 | 5.16 |
| 44 | 1 | 27.3 | .19 |
| 44 | 7 | 74.9 | 2.12 |
| 66 | 1 | 6.7 | 2.11 |
| 66 | 7 | 65.8 | 4.53 |
|  |  |  |  |

**make the above changes permanent:**   
-- they were only temporary until now.

COMMIT WORK;

**update two rows, Denver's rainfall readings:**

UPDATE STATS SET RAIN\_I = 4.50   
WHERE ID = 44;

**and take a look:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .32 |
| 13 | 7 | 91.7 | 5.16 |
| 44 | 1 | 27.3 | 4.5 |
| 44 | 7 | 74.9 | 4.5 |
| 66 | 1 | 6.7 | 2.11 |
| 66 | 7 | 65.8 | 4.53 |
|  |  |  |  |

**Oops! We meant to update just the July reading! Undo that update:**   
-- undoes only updates since the last COMMIT WORK.

ROLLBACK WORK;

**and take a look:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .32 |
| 13 | 7 | 91.7 | 5.16 |
| 44 | 1 | 27.3 | .19 |
| 44 | 7 | 74.9 | 2.12 |
| 66 | 1 | 6.7 | 2.11 |
| 66 | 7 | 65.8 | 4.53 |
|  |  |  |  |

**now update Denver's July rainfall reading and make it permanent:**

UPDATE STATS SET RAIN\_I = 4.50   
WHERE ID = 44   
AND MONTH = 7;

COMMIT WORK;

**and take a look:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .32 |
| 13 | 7 | 91.7 | 5.16 |
| 44 | 1 | 27.3 | .19 |
| 44 | 7 | 74.9 | 4.5 |
| 66 | 1 | 6.7 | 2.11 |
| 66 | 7 | 65.8 | 4.53 |
|  |  |  |  |

**delete July data and East Coast data from both tables:**   
-- note that we use longitude values from the related STATION table to determine which STAT stations were east of 90 degrees.

DELETE FROM STATS   
WHERE MONTH = 7   
OR ID IN (SELECT ID FROM STATION   
WHERE LONG\_W < 90);

DELETE FROM STATION WHERE LONG\_W < 90;

COMMIT WORK;

**and take a look:**

SELECT \* FROM STATION;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **CITY** | **ST** | **LAT\_N** | **LONG\_W** |
| 13 | Phoenix | AZ | 33 | 112 |
| 44 | Denver | CO | 40 | 105 |
|  |  |  |  |  |

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 13 | 1 | 57.4 | .32 |
| 44 | 1 | 27.3 | .19 |
|  |  |  |  |

**View METRIC\_STATS, a Fahrenheit-to-Centigrade and inches-to-centimeters conversion of table STATS, reflects the updates made to the underlying table.**

SELECT \* FROM METRIC\_STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_C** | **RAIN\_C** |
| 13 | 1 | 14.1111111 | .125984 |
| 44 | 1 | -2.6111111 | .074803 |
|  |  |  |  |

**SQL Constraints**

**SQL enforces data integrity constraints.**

**Attempt to insert a row for an unknown observation station.**   
-- The ID value of 33 does not match a station ID value in the STATION table.   
-- This is a violation of referential integrity.

INSERT INTO STATS VALUES (33,8,27.4,.19);

|  |
| --- |
| **error message** |
| violation of constraint STATS\_FOREIGN1 caused operation to fail |

**Attempt to update a row with a temperature below the range -80 TO 150.**

UPDATE STATS SET TEMP\_F = -100 WHERE ID = 44 AND MONTH = 1;

|  |
| --- |
| **error message** |
| violation of constraint STATS\_CHECK2 caused operation to fail |

**Attempt to insert a row with negative rainfall measurement, outside the range 0 to 100.**

INSERT INTO STATS VALUES (44,8,27.4,-.03);

|  |
| --- |
| **error message** |
| violation of constraint STATS\_CHECK3 caused operation to fail |

**Attempt to insert a row with month 13, outside the range of 1 to 12.**

INSERT INTO STATS VALUES (44,13,27.4,.19);

|  |
| --- |
| **error message** |
| violation of constraint STATS\_CHECK1 caused operation to fail |

**Attempt to insert a row with a temperature above the range -80 TO 150.**

INSERT INTO STATS VALUES (44,8,160,.19);

|  |
| --- |
| **error message** |
| violation of constraint STATS\_CHECK2 caused operation to fail |

**Attempt to insert a row with no constraint violations.**

INSERT INTO STATS VALUES (44,8,27.4,.10);

|  |
| --- |
| **status message** |
| 1 row inserted |

**and take a look:**

SELECT \* FROM STATS;

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **MONTH** | **TEMP\_F** | **RAIN\_I** |
| 44 | 8 | 27.4 | .10 |
| 13 | 1 | 57.4 | .32 |
| 44 | 1 | 27.3 | .19 |

**Attempt to insert a second row of August statistics for station 44.**   
-- This is a violation of the primary key constraint.   
-- Only one row for each station and month combination is allowed.

INSERT INTO STATS VALUES (44,8,160,.19);

|  |
| --- |
| **error message** |
| violation of constraint STATS\_PRIMARY\_ID\_MONTH caused operation to fail |

**Embedded SQL C Program Example**

Embedded C program to do the following:

Starting with a station name (Denver, in this example), look up the station ID.   
Print the station ID.   
List all rows for that station ID.

* shows single-row select and use of cursor
* note that all C-language variables used in SQL statements are declared in the DECLARE SECTION.
* note that all SQL statements begin with the syntax EXEC SQL and terminate with a semicolon.

|  |
| --- |
| #include<stdio.h>  #include<string.h>  EXEC SQL BEGIN DECLARE SECTION;  long station\_id;  long mon;  float temp;  float rain;  char city\_name[21];  long SQLCODE;  EXEC SQL END DECLARE SECTION;  main()  {  /\* the CONNECT statement, if needed, goes here \*/  strcpy(city\_name,"Denver");  EXEC SQL SELECT ID INTO :station\_id  FROM STATION  WHERE CITY = :city\_name;  if (SQLCODE == 100)  {  printf("There is no station for city %s\n",city\_name);  exit(0);  }  printf("For the city %s, Station ID is %ld\n",city\_name,station\_id);   printf("And here is the weather data:\n");  EXEC SQL DECLARE XYZ CURSOR FOR  SELECT MONTH, TEMP\_F, RAIN\_I  FROM STATS  WHERE ID = :station\_id  ORDER BY MONTH;  EXEC SQL OPEN XYZ;  while (SQLCODE != 100) {  EXEC SQL FETCH XYZ INTO :mon, :temp, :rain;  if (SQLCODE == 100)  printf("end of list\n");  else  printf("month = %ld, temperature = %f, rainfall = %f\n",mon,temp,rain);  }  EXEC SQL CLOSE XYZ;  exit(0);  } |

**Execution log:**

|  |
| --- |
| For the city Denver, Station ID is 44   And here is the weather data:  month = 1, temperature = 27.299999, rainfall = 0.180000   month = 7, temperature = 74.800003, rainfall = 2.110000   end of list |

[return to SQL Table of Contents](http://www.itl.nist.gov/div897/ctg/dm/contents_sql.htm)