Data Model

With Examples from “SQL for Smarties” from J.Celko
Agenda

• Relational – SQL
• Beyond Relational
  • Arrays
  • Graphs
• Time series
What is the meaning?

SELECT  s.RESTAURANT_NAME, t.TABLE_SEATING, to_char(t.DATE_TIME,'Dy, Mon FMDD') AS THEDATE, to_char(t.DATE_TIME,'HH:MI PM') AS THETIME,to_char(t.DISCOUNT,'99') || '%' AS AMOUNTVALUE,t.TABLE_ID, s.SUPPLIER_ID, t.DATE_TIME, to_number(to_char(t.DATE_TIME,'SSSSS')) AS SORTTIME
FROM TABLES_AVAILABLE t, SUPPLIER_INFO s,
  (SELECT          s.SUPPLIER_ID, t.TABLE_SEATING, t.DATE_TIME, max(t.DISCOUNT) AMOUNT, t.OFFER_TYPE
   FROM          TABLES_AVAILABLE t, SUPPLIER_INFO
   WHERE          t.SUPPLIER_ID = s.SUPPLIER_ID
   and (TO_CHAR(t.DATE_TIME, 'MM/DD/YYYY') != TO_CHAR(sysdate, 'MM/DD/YYYY') OR TO_NUMBER(TO_CHAR(sysdate, 'SSSSS')) < s.NOTIFICATION_TIME - s.TZ_OFFSET)
   and t.NUM_OFFERS > 0
   and t.DATE_TIME > SYSDATE
   and s.CITY = 'SF'
   and t.TABLE_SEATING = '2'
   and t.DATE_TIME between sysdate and (sysdate + 7)
   and to_number(to_char(t.DATE_TIME, 'SSSSS')) between 39600 and 82800
   and t.OFFER_TYPE = 'Discount'
   GROUP BY
   s.SUPPLIER_ID, t.TABLE_SEATING, t.DATE_TIME, t.OFFER_TYP ) u
WHERE
  t.SUPPLIER_ID   = s.SUPPLIER_ID
  and u.SUPPLIER_ID   = s.SUPPLIER_ID
  and t.SUPPLIER_ID   = u.SUPPLIER_ID
  and t.TABLE_SEATING = u.TABLE_SEATING
  and t.DATE_TIME     = u.DATE_TIME
  and t.DISCOUNT = u.AMOUNT
  and t.OFFER_TYPE    = u.OFFER_TYPE
  and (TO_CHAR(t.DATE_TIME, 'MM/DD/YYYY') != TO_CHAR(sysdate, 'MM/DD/YYYY') OR TO_NUMBER(TO_CHAR(sysdate, 'SSSSS')) < s.NOTIFICATION_TIME - s.TZ_OFFSET)
  and t.NUM_OFFERS > 0
  and t.DATE_TIME > SYSDATE and s.CITY = 'SF' and t.TABLE_SEATING = '2' and t.DATE_TIME between sysdate and (sysdate + 7)
  and to_number(to_char(t.DATE_TIME, 'SSSSS')) between 39600 and 82800 and t.OFFER_TYPE = 'Discount'
ORDER  BY AMOUNTVALUE DESC,  t.TABLE_SEATING ASC, upper(s.RESTAURANT_NAME) ASC,SORTTIME ASC, t.DATE_TIME ASC
Spot the Difference?

- SELECT ssnum
  FROM employee e1
  WHERE numfriends = ALL (SELECT COUNT(e2.ssnum)
    FROM employee e2, tech
    WHERE e2.dept = tech.dept
    AND e2.dept = e1.dept);

- INSERT INTO temp
  SELECT COUNT(ssnum) as numcolleagues, employee.dept
  FROM employee, tech
  WHERE employee.dept = tech.dept
  GROUP BY employee.dept;

SELECT ssnum
FROM employee, temp
WHERE numfriends = numcolleagues
  AND employee.dept = temp.dept;

Beware NULL and aggregates!
More NULL

- Employee (emp_name, birthday)
- Celebrity(celeb_name, birthday)

- Assume ('Katja Glamour', NULL) is a tuple in Celebrity

- Find the employees that were not born on the same day as a celebrity
More NULL

SELECT P1.emp_name
FROM Personal as P1
WHERE NOT EXISTS
  (SELECT * FROM Celebrities C1
   WHERE P1.birthday = C1.birthday);

SELECT P1.emp_name
FROM Personal as P1
WHERE P1.birthday NOT IN
  (SELECT * FROM Celebrities C1
   WHERE P1.birthday = C1.birthday);

Beware 3-valued logic
Time Intervals

- Guests (name, arrival_date, depart_date)
- Celebrations(name, start_date, finish_date)
- Which guests were present for which celebrations?
Time Intervals

- CREATE VIEW GuestCeleb(g_name, c_name) AS SELECT G.name, C.name FROM Guests G, Celebrations C WHERE NOT ((depart_date < start_date) OR (arrival_date > finish_date))

- SELECT * FROM GuestCeleb WHERE (arrival_date BETWEEN start_date AND finish_date) OR (depart_date BETWEEN start_date AND finish_date) OR (start_date BETWEEN arrival_date AND depart_date) OR (finish_date BETWEEN arrival_date AND depart_date)
T-join Problem

- Rooms(room_nr, room_size)
- Classes(class_nbr, class_size)
- Assign classes to available rooms: i.e., find pairs class_nbr, room_nbr (no class_nbr or room_nbr duplicate) so that class_size is lower than room_size.
T-join Problem

- CREATE VIEW AllClassRooms
  AS SELECT *
  FROM Classes, Rooms
  WHERE class_size < room_size;

- CREATE VIEW SmallestClassRooms
  AS SELECT *
  FROM AllClassRooms CR1
  WHERE room_size = (SELECT MIN(room_size)
  FROM AllClassRooms
  WHERE class_nbr = CR1.class_nbr);
T-join Problem

- SELECT *
  FROM SmallestClassRooms CR1
  WHERE class_size = SELECT MAX(class_size)
    FROM SmallestClassRooms
    WHERE room_nbr = CR1.room_nbr;
Agenda

- Relational – SQL
- Beyond Relational
  - Arrays
  - Graphs
  - Time series
Arrays

int [4] [4] d2;

• Representation in SQL / Other data models.
Arrays in SQL

• Ranking

• \((\text{NF})^2\): Non First Normal Form

• Sparseness: NULL
BigTable Data Model

- A Bigtable is a sparse, distributed, persistent multidimensional sorted map.

- The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.
SciDB Data Model

Basic 2D Ragged Array

Dimension 1

5
4
3
2
1

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Dimension 2

Cell

(a_1, a_2, )

Basic array: MyArray [3, 7]
Enhanced array: MyArray {16.3, 27.6}

From Cudre-Mauroux et al. - VLDB 2009
HDF-5

- HDF5 dataset: a multidimensional array of data elements, together with supporting metadata

```hdf5
HDF5 "dset.h5" {
GROUP "/" {
    DATASET "dset" {
        DATATYPE { H5T_STD_I32BE }
        DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
        DATA {
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0
        }
    }
}
}
```
Time Series in SQL

- Sequence generation (ad-hoc / Automatic)
- Time stamp attribute as key
- For example:
  - MyStock (sale_date, price)
  - Find the intervals where stock price is generally increasing (at no point in the interval is the price below its initial price).
SELECT S1.sale_date as start_date,
    S2.sale_date as finish_date
FROM MyStock as S1, MyStock as S2
WHERE S1.sale_date < S2.sale_date
    AND S1.price < S2.price
    AND NOT EXISTS (
        SELECT * FROM MyStock as S3
        WHERE S3.sale_date BETWEEN S1.sale_date AND S2.sale_date
        AND S3.price NOT BETWEEN S1.price AND S2.price);
Truviso Data Model

SQL with two extensions:

- Windowing
- Event patterns

CREATE STREAM trades (symbol varchar(5), price real, volume integer, tstamp timestamp CQTIME USER GENERATED SLACK ‘1 minute’) TYPE UNARCHIVED;

SELECT sum(price * volume) / sum(volume) AS vwap, sum(volume) AS volume, advance_agg(qtime) AS windowtime FROM trades < VISIBLE ‘1 minute’ ADVANCE ‘5 seconds’ > WHERE symbol = ‘MSFT’
Graphs in SQL

- Journeys (depart_town, arrival_town, distance)
- Find all composite journeys starting from Paris
Graphs in SQL

WITH RECURSIVE Journeys (arrival_town)
AS (SELECT DISTINCT depart_town
    FROM Journeys
    WHERE depart_town = 'Copenhagen'
    UNION ALL
    SELECT arrival_town
    FROM Journeys as R1,
        Journeys as R2
    WHERE R1.arrival_town = R2.depart_town)

SELECT DISTINCT arrival_town FROM Journeys;
Trees in SQL

• Adjacency List  R(N1, N2)

• Denormalization R(N1,ListOfPaths)

• Preorder traversal R(N, Lft, Rgt)
RDF

- Directed, labeled graph, where the edges represent named link between two resources (vertices).