# Outline

1. What is a Query? Query Language?
2. Example Database Tables
3. SQL Overview: 3 Components
4. SELECT statement with 1 table
5. Multi-table SELECT statements
6. Why spatial extensions are needed?
7. 1-table spatial queries
8. Multi-table spatial queries
9. Trends



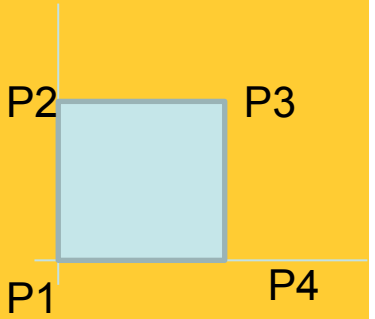
# Learning Objectives

- After this segment, students will able to
  - Explain why spatial extensions were added to SQL
  - Illustrate Semantic Gap between old SQL & Spatial Query
    - Also called Impedance Mismatch



# Why Extend SQL for Spatial Data?

- Original SQL had simple atomic data types
  - Examples: integer, dates, string, currency



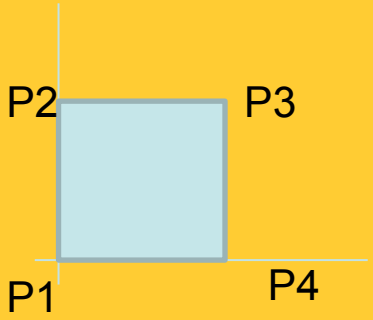

- Not convenient for spatial applications
  - Spatial Data: points, edges, rectangles, …
  - Example Queries:
    - Q1: List all rectangle with point( x = 0, y = 0) as a corner point.
    - Q2: List all rectangle with point( x = 0, y = 0) as an inside point.
    - …



# How old SQL modeled Spatial Data?

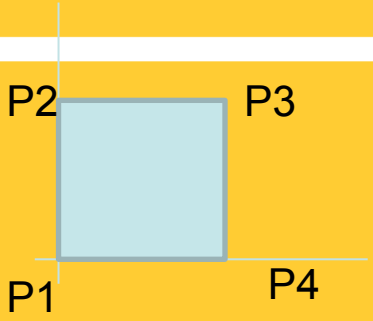
- Recall spatial data had
  - Points, Edges, Rectangles

- Old Table Design (3rd Normal Form)
  - Point ( Pid, x, y)
  - Edge (Eid, Length)
  - Rectangle (Rid, Rname)
  - Starts_or_Ends ( Eid, Pid )
  - Boundary ( Rid, Eid )






# Old Tabular Representation of Unit Square!




Point

| Pid | x | y |
|---|---|---|
| P1 | 0 | 0 |
| P2 | 0 | 1 |
| P3 | 1 | 1 |
| P4 | 1 | 0 |

Start_or_Ends

| Eid | Pid |
|---|---|
| E1 | P1 |
| E1 | P2 |
| E2 | P2 |
| E2 | P3 |
| E3 | P3 |
| E3 | P4 |
| E4 | P4 |
| E4 | P1 |

Edge

| Eid | Length |
|---|---|
| E1 | 1 |
| E2 | 1 |
| E3 | 1 |
| E4 | 1 |

Boundary

| Rid | Eid |
|---|---|
| R1 | E1 |
| R1 | E2 |
| R1 | E3 |
| R1 | E4 |

Rectangle

| Rid | Rname |
|---|---|
| R1 | UnitSq |

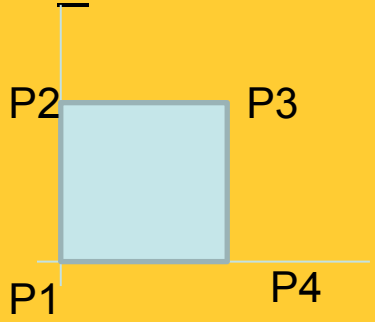


# List all rectangles with origin as a corner point.

You may be able to compose this SQL query

- However, joining 5 tables is challenging for average programmer
- Also, joining 5 tables to retrieve a simple object is costly!

```
SELECT R.Rid, R.Rname
FROM Rectangle R, Edge E, Point P, Boundary B, Starts_Or_Ends S
WHERE (R.Rid=B.Rid) AND (B.Eid=E.Eid)
        AND (E.Eid=S.Eid) AND (S.Pid=P.Pid)
        AND (P.x = 0)     AND      (P.y = 0)
```



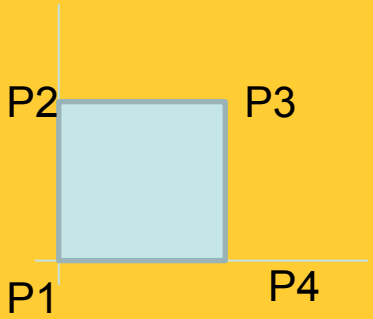



# List rectangles with origin as an inside point.

**Steps:** A. Gather properties of each rectangle in a row!

B. Compare x- and y-coordinates of query point and each rectangle

```
SELECT R.Rid, R.Rname, MIN(P.x), MIN(P.y), MAX(P.x), MAX(P.y)
FROM Rectangle R, Edge E, Point P, Boundary B, Starts_Or_Ends S
WHERE (R.Rid=B.Rid) AND (B.Eid=E.Eid)
        AND (E.Eid=S.Eid) AND (S.Pid=P.Pid)
GROUP BY R.Rid
HAVING (0 BETWEEN MIN(P.x) AND MAX(P.x))
        AND (0 BETWEEN MIN(P.y) AND (P.y))
```





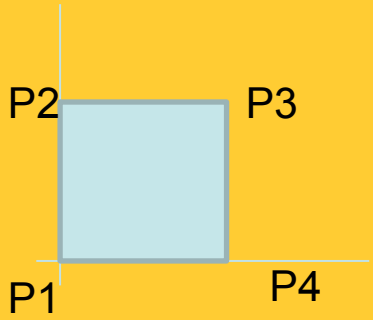

# Semantic Gap, Impedance Mismatch between Old SQL and Spatial Applications!

If you found last query hard to code in SQL,

- You have just experienced the pain many felt in last century
- And forced SQL to support user-defined data-types!

Simpler code with user-defined spatial data-types

```
SELECT Rid, Rname
FROM Rectangle R
WHERE within ( Point(0,0), R.Shape_Polygon  )
```




- New Table Design
  - Rectangle( Rid, RName, Shape_Polygon )
  - Point (Pid, x, y)