**CS623 Project - Practical**

**Data Source**

[**https://hub.arcgis.com/**](https://hub.arcgis.com/)

[**https://www.arcgis.com/home/item.html?id=3a76c18b9d524b89b6b393afa9c51643**](https://www.arcgis.com/home/item.html?id=3a76c18b9d524b89b6b393afa9c51643)

CREATE EXTENSION postgis;

SELECT postgis\_version();

SELECT postgis\_full\_version();

-- RETREIEVE FEATURE LOCATIONS

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1a\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941');

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1b\_v

WHERE ST\_Equals(geom, '0106000020110F0000010000000103000000010000000A0000005FE176F4A21A1F418516C9AF24E259415650B3F03F1C1F41D9B71EB51BE259419907D6A1381C1F414B6142631AE25941D9084812FC1B1F4180FE27B41BE25941109984AC671B1F410017AEED1EE25941E06BD425651B1F41043FB0F91EE2594109392036ED1A1F41233DA49121E25941BE4402A99E1A1F415BBE714923E259416058B7BC9B1A1F4101A5BC6123E259415FE176F4A21A1F418516C9AF24E25941');

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1c\_v

WHERE ST\_Equals(geom

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1d\_v

WHERE ST\_Equals(geom, '0106000020110F00000100000001030000000100000006000000508EC412AAE51E41A3AFB53487E059418DEEFA1E0EE41E411EAF31D87CE05941320DA0850DE41E413D9B58D47CE059417EF2D7E5FAE31E4158F514B47FE05941F76C1FEAC5E51E4160049F3A8BE05941508EC412AAE51E41A3AFB53487E05941');

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_2\_v

WHERE ST\_Equals(geom, '0106000020110F0000010000000103000000010000000B000000B57C1A9C9D551F41BCECF01E49E559419194527DB5541F410A6BFC3E53E55941F94B4F3682541F414BF4C47E55E55941EFA1B81580541F4162E16F9755E55941898900528E541F41D752AD4757E5594104B78EF292541F41E59771D357E5594160DD71D2C7541F419A3D197055E55941003FA70DBB551F4133D004164BE5594183E2DE1DF2551F41CA0577CE48E5594147707EF0DA551F419EF23A9F46E55941B57C1A9C9D551F41BCECF01E49E55941');

-- INTERSECTIONS

-- POINTS OF INTERSECTION IN categorie1A

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1a\_v

WHERE ST\_Intersects(geom, ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941'), 3857));

-- POINTS OF INTERSECTION BETWEEN categorie1A AND categorie1B

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1b\_v

WHERE ST\_Intersects(geom, ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941'), 3857));

-- POINTS OF INTESECTION BETWEEN categorie1A AND categorie1D

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1d\_v

WHERE ST\_Intersects(geom, ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941'), 3857));

-- POINTS OF INTESECTION IN categorie2

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_2\_v

WHERE ST\_Intersects(geom, ST\_GeomFromText(ST\_AsText('0106000020110F0000010000000103000000010000000B000000B57C1A9C9D551F41BCECF01E49E559419194527DB5541F410A6BFC3E53E55941F94B4F3682541F414BF4C47E55E55941EFA1B81580541F4162E16F9755E55941898900528E541F41D752AD4757E5594104B78EF292541F41E59771D357E5594160DD71D2C7541F419A3D197055E55941003FA70DBB551F4133D004164BE5594183E2DE1DF2551F41CA0577CE48E5594147707EF0DA551F419EF23A9F46E55941B57C1A9C9D551F41BCECF01E49E55941'), 3857));

-- FEATURES WITHIN DISTANCE -> 500meters

-- FEATURES in categorie1a

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1a\_v

WHERE ST\_DWithin(

geom,

ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941'), 3857),

500

)

ORDER BY id

LIMIT 3;

-- FEATURES IN categorie1b

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1b\_v

WHERE ST\_DWithin(

geom,

ST\_GeomFromText(ST\_AsText

500

)

ORDER BY id

LIMIT 3;

-- FEATURES IN categorie1c

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1c\_v

WHERE ST\_DWithin(

geom,

ST\_GeomFromText(ST\_AsText

500

)

ORDER BY id

LIMIT 3;

-- FEATURES IN categorie1d

SELECT \* FROM categorie\_1d\_v;

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1d\_v

WHERE ST\_DWithin(

geom,

ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000060000001BE91EA240091F418611E7012BE15941586AC4400F091F41F7E6456B27E159416F2848C9ED081F41EA4CBB3929E15941819EA03207091F4146EB23132BE159410BD4E5191F091F41AE0B49D12CE159411BE91EA240091F418611E7012BE15941'), 3857),

500

)

ORDER BY id DESC;

-- FEATURES IN categorie2

SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_2\_v

WHERE ST\_DWithin(

geom,

ST\_GeomFromText(ST\_AsText('0106000020110F0000010000000103000000010000000B000000B57C1A9C9D551F41BCECF01E49E559419194527DB5541F410A6BFC3E53E55941F94B4F3682541F414BF4C47E55E55941EFA1B81580541F4162E16F9755E55941898900528E541F41D752AD4757E5594104B78EF292541F41E59771D357E5594160DD71D2C7541F419A3D197055E55941003FA70DBB551F4133D004164BE5594183E2DE1DF2551F41CA0577CE48E5594147707EF0DA551F419EF23A9F46E55941B57C1A9C9D551F41BCECF01E49E55941'), 3857),

500

)

ORDER BY id DESC;

-- CALCULATING AREAS

-- AREA IN categorie1a

SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_1a\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941');

-- AREA OF LOCATION IN categorie1b

SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_1b\_v

WHERE ST\_Equals(geom

-- AREA OF LOCATION IN categorie1c

SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_1c\_v

WHERE ST\_Equals(geom

-- AREA OF LOCATION IN categorie1d

SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_1d\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000060000001BE91EA240091F418611E7012BE15941586AC4400F091F41F7E6456B27E159416F2848C9ED081F41EA4CBB3929E15941819EA03207091F4146EB23132BE159410BD4E5191F091F41AE0B49D12CE159411BE91EA240091F418611E7012BE15941');

-- AREA OF LOCATION IN categorie2

SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_2\_v

WHERE ST\_Equals(geom, '0106000020110F0000010000000103000000010000000B000000B57C1A9C9D551F41BCECF01E49E559419194527DB5541F410A6BFC3E53E55941F94B4F3682541F414BF4C47E55E55941EFA1B81580541F4162E16F9755E55941898900528E541F41D752AD4757E5594104B78EF292541F41E59771D357E5594160DD71D2C7541F419A3D197055E55941003FA70DBB551F4133D004164BE5594183E2DE1DF2551F41CA0577CE48E5594147707EF0DA551F419EF23A9F46E55941B57C1A9C9D551F41BCECF01E49E55941');

-- AREA OF LOCATION IN categorie2a

SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_2a\_v

WHERE ST\_Equals(geom

-- FINDING DISTANCES

-- DISTANCE OF A POINT IN categorie1a

SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_1a\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941');

-- DISTANCE OF A POINT IN categorie1b

SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_1b\_v

WHERE ST\_Equals(geom

-- DISTANCE OF A POINT IN categorie1c

SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_1c\_v

WHERE ST\_Equals(geom

-- DISTANCE OF A POINT IN categorie1d

SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_1d\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000060000001BE91EA240091F418611E7012BE15941586AC4400F091F41F7E6456B27E159416F2848C9ED081F41EA4CBB3929E15941819EA03207091F4146EB23132BE159410BD4E5191F091F41AE0B49D12CE159411BE91EA240091F418611E7012BE15941');

-- DISTANCE OF A POINT IN categorie2

SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_2\_v

WHERE ST\_Equals(geom, '0106000020110F0000010000000103000000010000000B000000B57C1A9C9D551F41BCECF01E49E559419194527DB5541F410A6BFC3E53E55941F94B4F3682541F414BF4C47E55E55941EFA1B81580541F4162E16F9755E55941898900528E541F41D752AD4757E5594104B78EF292541F41E59771D357E5594160DD71D2C7541F419A3D197055E55941003FA70DBB551F4133D004164BE5594183E2DE1DF2551F41CA0577CE48E5594147707EF0DA551F419EF23A9F46E55941B57C1A9C9D551F41BCECF01E49E55941');

-- DISTANCE OF A POINT IN categorie2a

SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_2a\_v

WHERE ST\_Equals(geom

-- QUERY OPTIMIZATION -> table index

CREATE INDEX cat\_1a\_idx

ON categorie\_1a\_v

USING GIST(geom);

CREATE INDEX cat\_1b\_idx

ON categorie\_1b\_v

USING GIST(geom);

CREATE INDEX cat\_1c\_idx

ON categorie\_1c\_v

USING GIST(geom);

CREATE INDEX cat\_1d\_idx

ON categorie\_1d\_v

USING GIST(geom);

CREATE INDEX cat\_2\_idx

ON categorie\_2\_v

USING GIST(geom);

CREATE INDEX cat\_2a\_idx

ON categorie\_2a\_v

USING GIST(geom);

-- QUERY ANALYSIS

-- EXPLAIN SELECT

EXPLAIN SELECT \*

FROM categorie\_1a\_v

ORDER BY id DESC

LIMIT 10;

-- EXPLAIN RETRIEVE FEATURE LOCATION

EXPLAIN SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1a\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941');

-- EXPLAIN INTERSECTIONS

EXPLAIN SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1a\_v

WHERE ST\_Intersects(geom, ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941'), 3857));

-- EXPLAIN SELECT FEATURES WITHIN DISTANCE

EXPLAIN SELECT id, categorie, ST\_AsText(geom)

FROM categorie\_1a\_v

WHERE ST\_DWithin(

geom,

ST\_GeomFromText(ST\_AsText('0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941'), 3857),

500

)

ORDER BY id

LIMIT 3;

-- EXPLAIN CALCULATING AREA

EXPLAIN SELECT id, categorie, ST\_Area(geom) AS area

FROM categorie\_1a\_v

WHERE ST\_Equals(geom, '0106000020110F000001000000010300000001000000090000009C81D87B761B1F41DA7B4B9E21E2594171E6EC79471C1F41ABF584111DE259415650B3F03F1C1F41D9B71EB51BE259415FE176F4A21A1F418516C9AF24E259413602D9B7AA1A1F413FDB431726E259414C7CD972AD1A1F418D424FF325E25941B723A540731B1F416D22C3A921E25941475D4864761B1F4147905E9A21E259419C81D87B761B1F41DA7B4B9E21E25941');

-- EXPLAIN FINDING DISTANCE

EXPLAIN SELECT id, categorie, ST\_Distance(

ST\_GeomFromText(ST\_AsText(geom)),

ST\_GeomFromText('LINESTRING(2 2, -2 2)')

) AS distance

FROM categorie\_2a\_v

WHERE ST\_Equals(geom

**CS623 Project - Theory**

**1.**

Number of pages per pass

6 -1 = 5

log 5 (N) = log (N) / log (5)

log 5 (1000000) = log (1000000) / log (5)

log (1000000) / log (5) = 8.5840

**9 Passes**

**2.**

Beginning at the root node, every key in the left subtree of the root (10) is less than 9\* in size. Therefore, navigate to the relevant subtree. The right subtree's initial node, 20, has no keys between 9\* and 19\*, hence no need to navigate to the root's subtree. Since the second child of the root (30) is bigger than 19\*. The left child of the root (10 is the sole node that has keys between 9\* and 19\*).

One pointer from the root to the leftmost child

(20): 1 sibling-to-sibling pointer from the left kid to its right sibling

1 parent-to-child pointer is present from 20 to its left child (11).

One pointer connecting siblings from 11 to its rightful sibling (12).

1 sibling-to-sibling pointer is present from 12 to its right sibling (13).

**5 pointers**

**3.**

**25**

The number of primary pages rises by one whenever a key that causes a split is entered. There are four primary pages with a capacity of three each at level 0. Therefore, the most keys that may be put without a split are 12 (4 pages x 3 keys each page). Using h1(x), 25 has a hash value of 001 and h0(x), a value of 01. As a result, the bucket that now contains 9, 25, and 5 would be given the number 25. An error will occur when entering a key into this bucket. As a result, the largest key now in the bucket, which is 25, will be the largest key less than 25 to induce a split.

**4.**

Level 1: Keyed nodes (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), (12), (13), (14), (15), (17), (18), and (19)

Level 2: Key Nodes (5, 9, 13, and 17)

Level 3: Keyed root node (9, 17)

**15 nodes**

**5.**

Plan I:

1. Join R and S on b: R⋈b=b (S)
2. Select rows where c=3: σc=3(R ⋈b=b (S))
3. Project only column a: πa(σc=3(R ⋈b=b (S)))

Plan II:

1. Select rows from S where c=3: σc=3(S)
2. Join R with the result of step 1 on b: R⋈b=b σc=3(S)
3. Project only column a: πa(R⋈b=b σc=3(S))

Plan II is more effective because, before conducting the join operation, it selects rows from S based on the criteria c=3, which decreases the size of the intermediate result that needs to be combined with R. Before filtering out rows where cis not equal to 3, Plan I performs the join, which can produce a bigger intermediate result.

**6.**

**True**

By reducing the overhead of function calls and control flow, the vectorized processing paradigm enables processing to be done more effectively. Each operator needs multi-threaded execution when receiving input from numerous children in order to produce the subsequent output tuples from each child.

**7.**

**Parallelism:** The hash join technique can be parallelized inherently since each split can be handled separately. The method makes use of the parallel processing capabilities of computers by using a large number of threads or processes to process the partitions.

**Memory Management:** The hash tables used in the hash join strategy must be built using a significant amount of RAM. Performance can be improved by using a multi-pass mechanism that spills intermediate results to disk when memory utilization exceeds a certain threshold.

**Caching:** Performance can be improved by decreasing the need to recompute intermediate results. The hash table produced from one input table can be cached and used again when connecting several tables.

**Partitioning:** Speed can be improved by breaking up the input data into smaller pieces by reducing memory use and raising cache locality. The input data may be divided using the join key, and each division can be handled separately.

**8.**

**3720 page I/Os.**

The number of page I/Os necessary to execute each operation can be used in estimating the cost of the query plan.

**i).** File scan on schools - Schools has 10 pages, the cost is 10 page I/Os.

**ii).** Selection - Since there are 100 schools and we assume a uniform distribution of srank values, about 10 schools will have srank < 10. Therefore, the cost of this operator is about 10 page I/Os.

**iii).** Sort-merge join - This join will use a merge join algorithm. Since there are 10 qualifying schools and 200 applicants who want to major in CSE, we will need to read each of the 10 pages of Schools once and each of the 200 pages of Applicants once. Therefore, the cost of this operator is about 210 page I/Os.

**iv).** Index nested loop join - Since there are 3000 applicants who want to major in CSE and there is an index on Major.id, we will need to read each of the 3000 index pages and each of the 100 pages of Applicants once. Therefore, the cost of this operator is about 3100 page I/Os.

**v).** Selection - Since there are 3000 applicants who want to major in CSE and we assume a uniform distribution of majors, about 300 applicants will want to major in CSE. Therefore, the cost of this operator is about 300 page I/Os.

**vi).** Projection - Only the name attribute is being projected, read each of the 300 pages of the result of the previous operator once. Therefore, the cost of this operator is 300 page I/Os.

**9.**

**a)**

**External Hashing** **-** Executing hash-based operations on data that is too big to fit in main memory requires the use of external hashing. The data must first be divided into memory-sized chunks, and each of those blocks must then be individually hashed. This reduces the amount of information that has to be processed all at once and guarantees that the hash function is used consistently.

**b)**

R: 2,400 pages \* 1 block per page = 2,400 blocks

S: 1,200 pages \* 10 blocks per page = 12,000 blocks

2,400 \* 12,000 = 28,800,000

2,400 blocks \* 1 write per block = 2,400

28,800,000 + 2,400 + 2,400 = 28,804,800

**cost = 28,804,800**

**10.**

**n + 1 leaf nodes**

number of leaf nodes = number of total nodes - number of internal nodes

(2n + 1) - (2n)

**11.**

**a.) Insert 12, Insert 13**