

Week 10

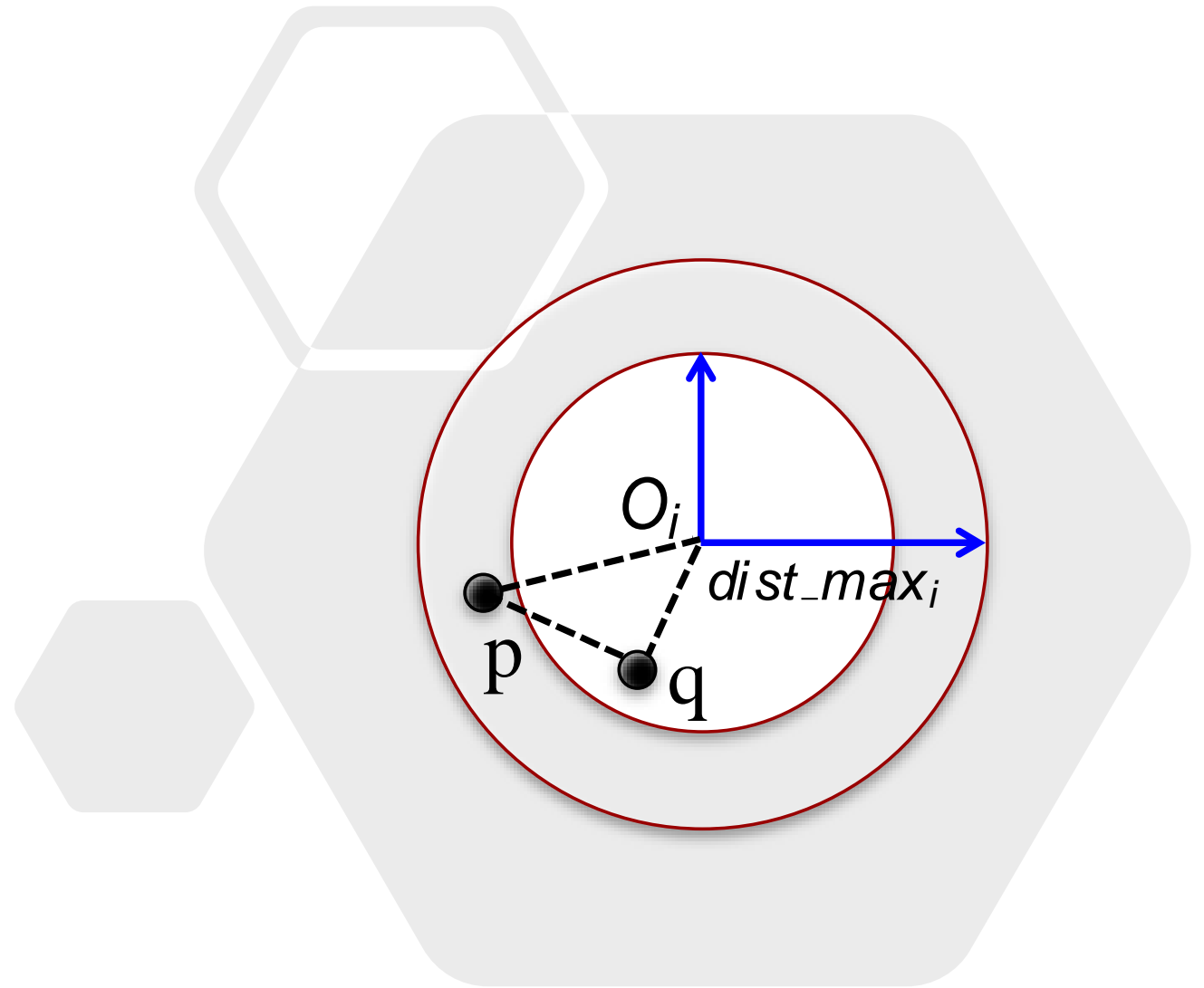
Tutorial – Multimedia Database

Fengmei Jin

Email: fengmei.jin@uq.edu.au

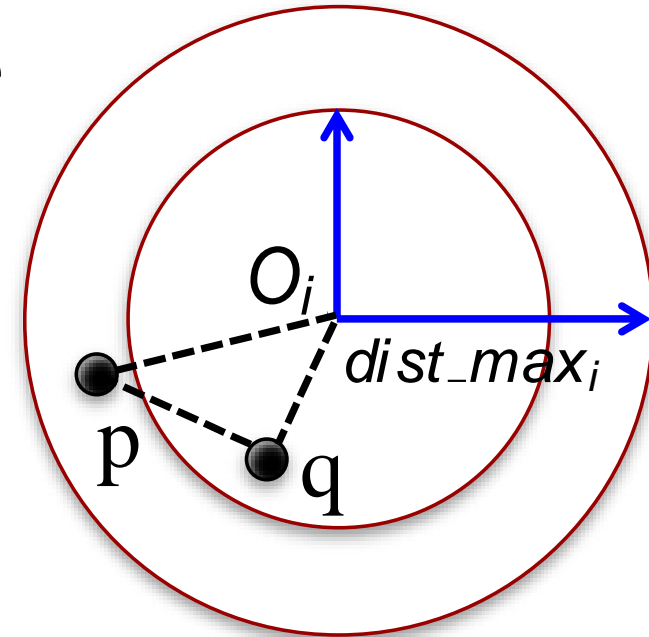
Question 1

- How is the triangle inequality is used to determine the search space in the *iDistance*?



Question 1 – example (1)

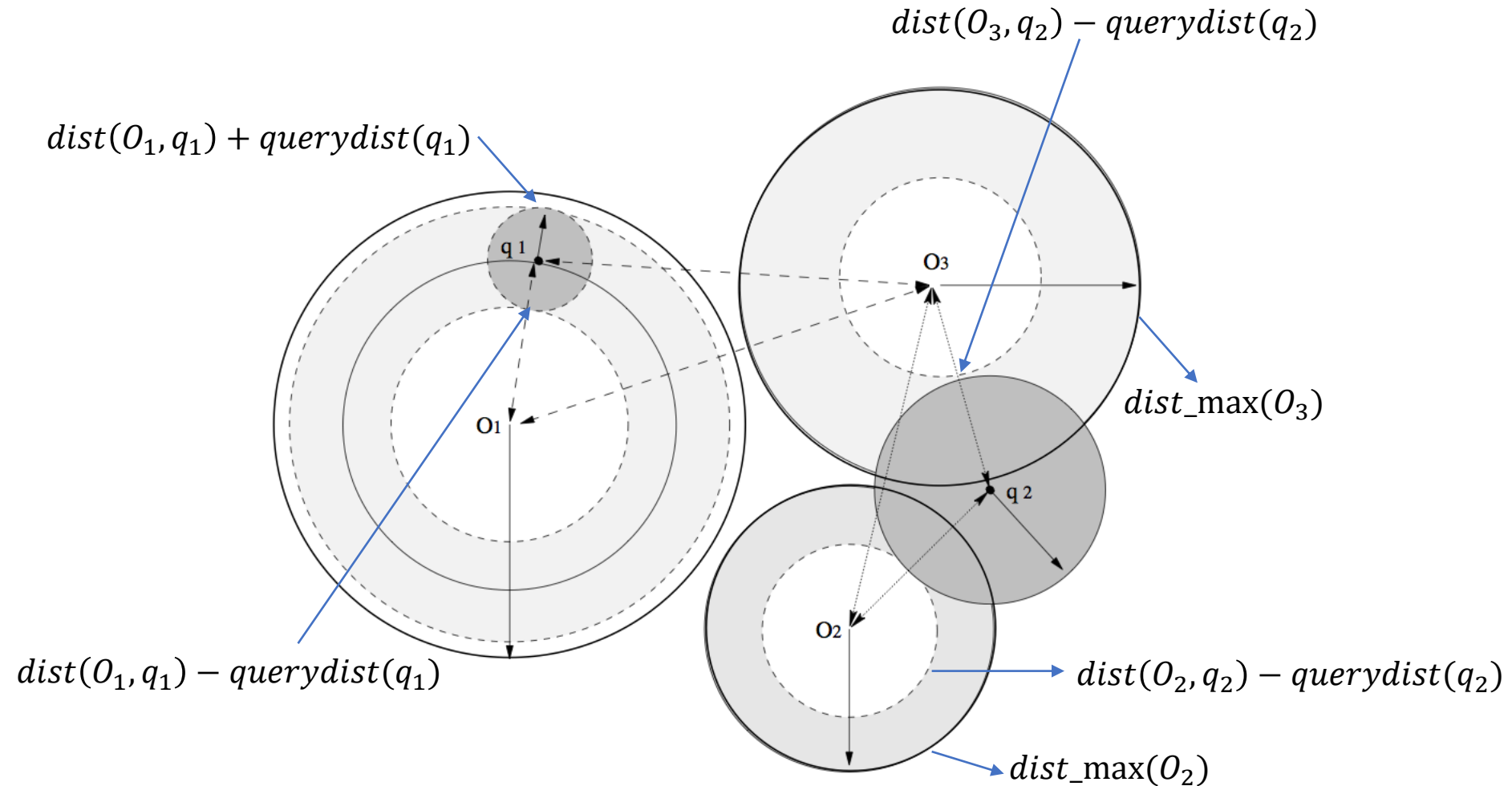
- As we can see from the example, the cluster reference point O , the query point q , and the result p form a triangle.
- Because the data are organized with their distance to O , so we have to know **the range of $dist(O_i, p)$** to determine the search space:
 - The **lower bound** of the search space (the inner radius) is $dist(O_i, q) - querydist(q)$,
 - The **upper bound** of the search space (the outer radius) is the smaller one of $dist_max(O_i)$ and $dist(O_i, q) + querydist(q)$.



Question 1 – example (2)

- As we can see in another example, for query point q_1 with query range $querydist(q_1)$
 - the **lower** bound of the query result is $dist(O_1, q_1) - querydist(q_1)$,
 - the **upper** bound of the query result is $dist(O_1, q_1) + querydist(q_1)$.
 - The result search space is a ring around O_1 .
- Query point q_1 does not need to visit cluster 2 and cluster 3
 - its lower bound to O_2 is $dist(O_2, q_1) - querydist(q_1) > dist_max(O_2)$;
 - O_3 is similar;
 - Therefore, O_2 and O_3 are pruned.

Question 1 – example (2)

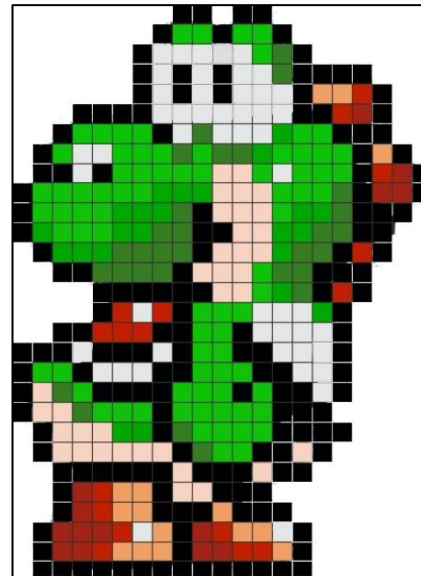


Question 1 – example (2)

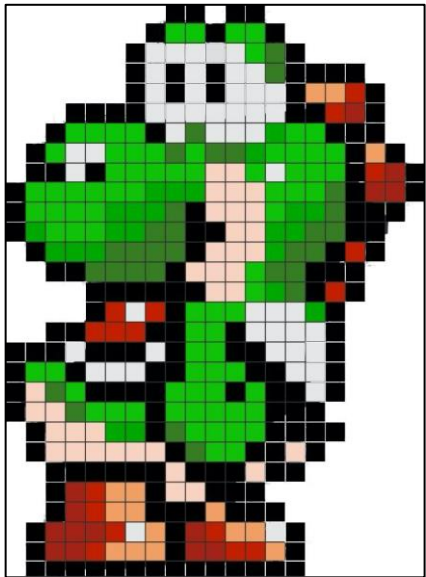
- For query point q_2 :
 - its **lower** bounds to O_2 and O_3 are $dist(O_2, q_2) - querydist(q_2)$ and $dist(O_3, q_2) - querydist(q_2)$.
 - Their **upper** bounds are all their maximum distance, because their maximum distances are all smaller than $dist(O_i, q_1) + querydist(q_1)$.
- q_2 does not need to visit cluster 1
 - its distance lower bound to O_1 is $dist(O_1, q_2) - querydist(q_2) > dist_max(O_1)$.











Question 2

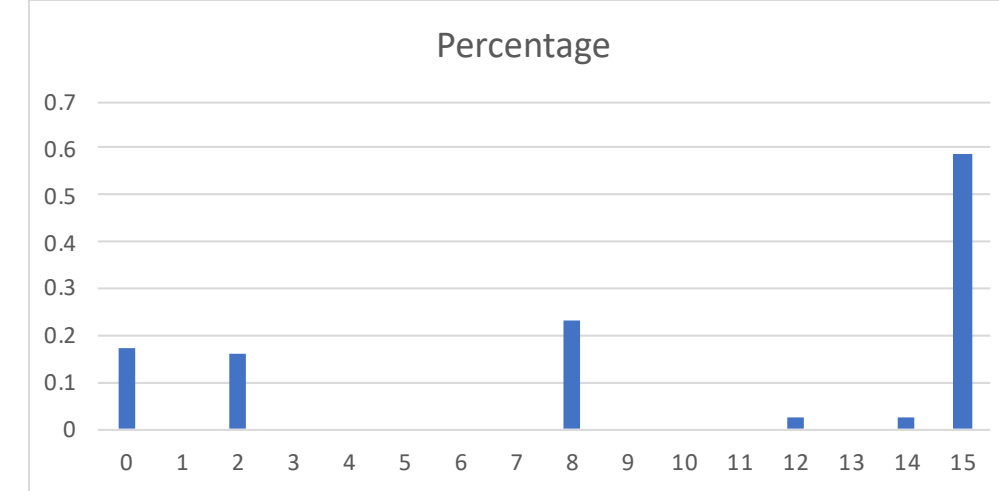
- Suppose we are using 24-bit RGB model for image representation, and extract colour features using the colour histogram by 16 bins: 4-quantile for Red, 2-quantile for green and 2-quantile for blue.
- Given a picture of the Yoshi (21 × 29 bit), what is the colour distribution of it?



Question 2 – answers



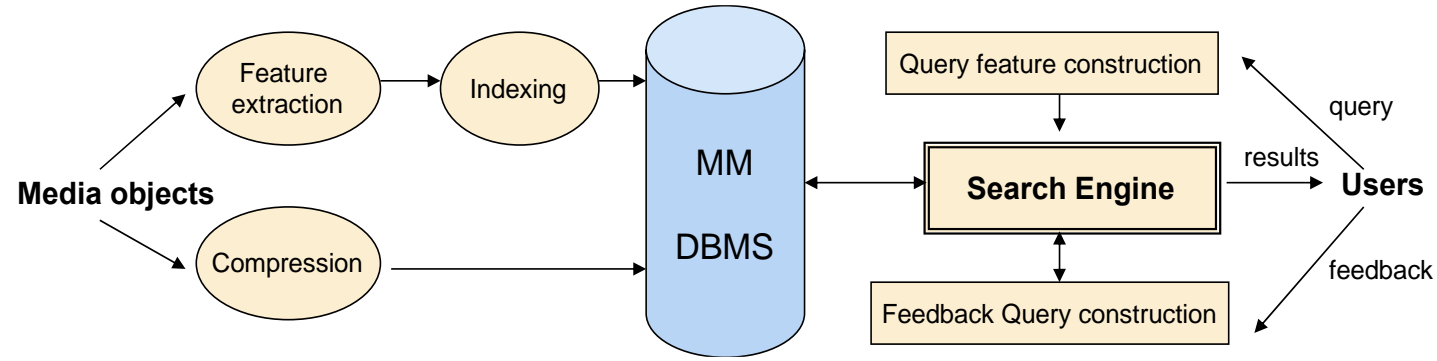
Colour	# Pixels	R,G,B
 (22, 191, 9)	78	(0,1,0)
 (0, 169, 0)	22	(0,1,0)
 (57, 125, 39)	30	(0,0,0)
 (246, 211, 193)	35	(3,1,1)
 (239, 160, 101)	15	(3,1,0)
 (193, 32, 1)	17	(3,0,0)
 (163, 35, 23)	14	(2,0,0)
 (0, 0, 0)	75	(0,0,0)
 (229, 230, 231)	46	(3,1,1)
 (255,255,255)	277	(3,1,1)



Bin	R,G,B	# Pixels	Percentage
0	0,0,0	105	0.172
1	0,0,1	0	0
2	0,1,0	100	0.164
3	0,1,1	0	0
4	1,0,0	0	0
5	1,0,1	0	0
6	1,1,0	0	0
7	1,1,1	0	0
8	2,0,0	14	0.23
9	2,0,1	0	0
10	2,1,0	0	0
11	2,1,1	0	0
12	3,0,0	17	0.028
13	3,0,1	0	0
14	3,1,0	15	0.025
15	3,1,1	358	0.588

Question 3

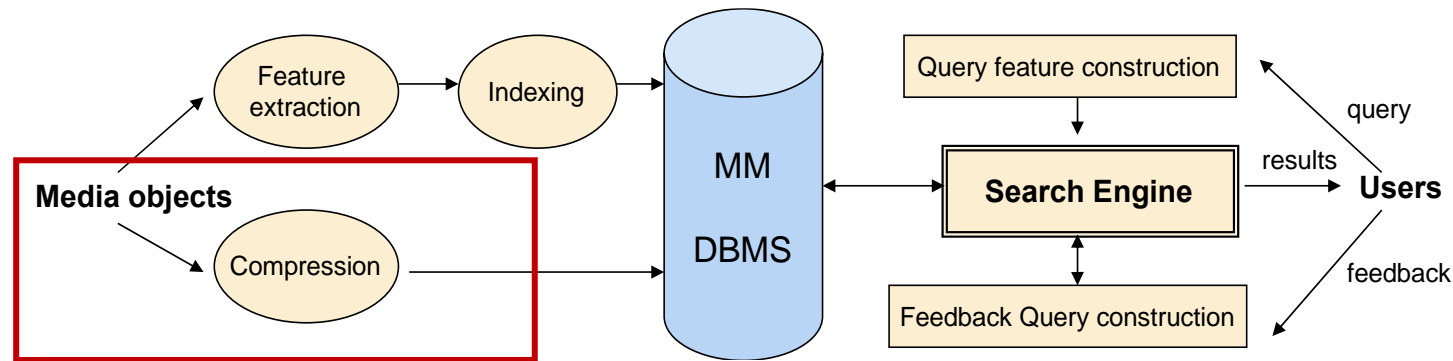
- Discuss the architecture of the multimedia database system.



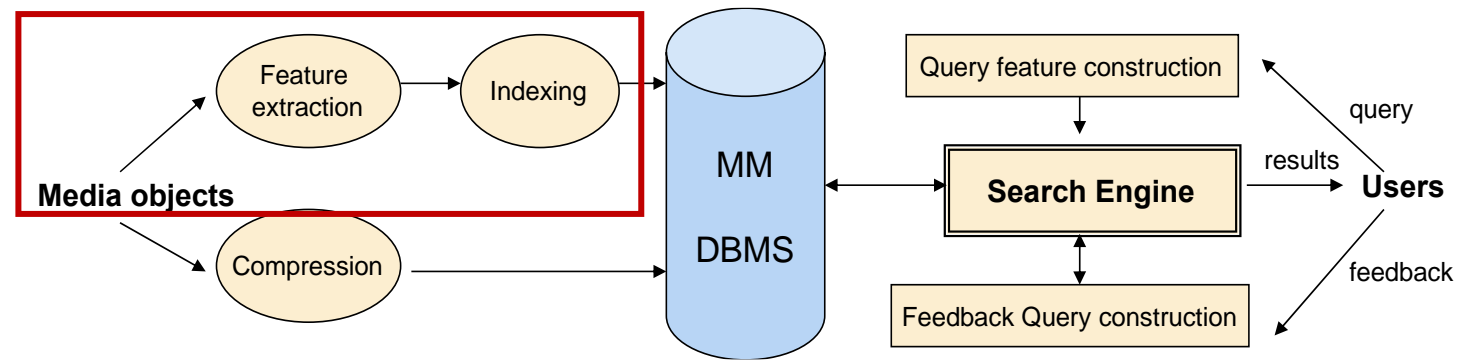
- On the left-hand side is the **storage/index** component, and on the right-hand side is the **query** component.

Question 3 – storage in MMDBMS

- For each multimedia object to be stored in the MMDBMS, we store its **compressed** version in disk because most of the MM data are very big. This compressed data is only used to return as the final result, and it will not be used during the search.

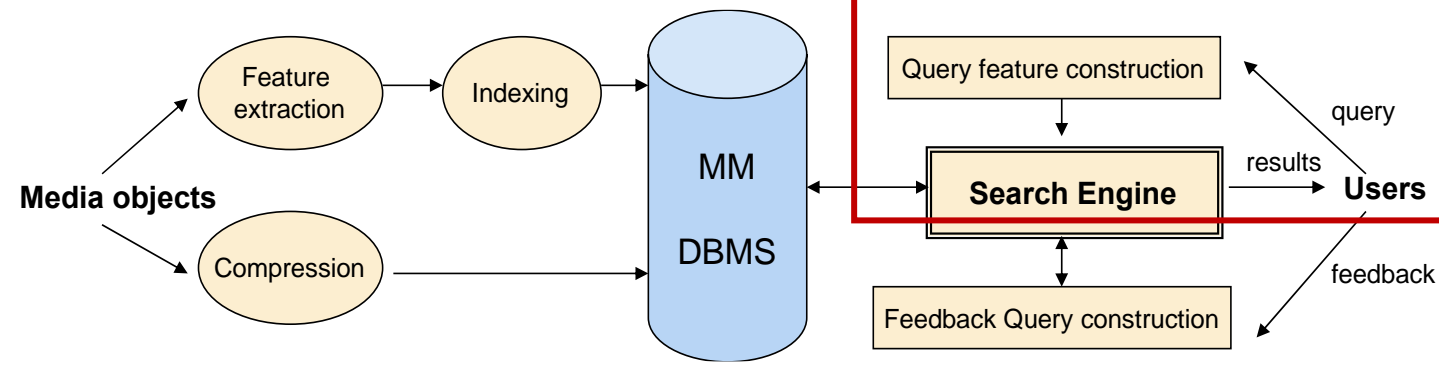


Question 3 – index in MMDDBMS



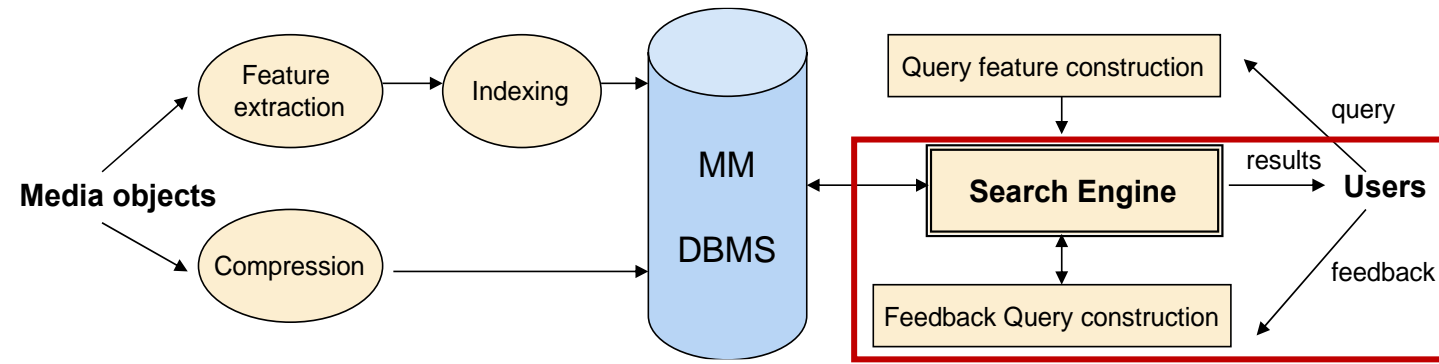
- Therefore, in order to support query/search, we need to extract features from the MM object and organize these features. This process is called **feature extraction**.
 - For most of the cases, the extracted features have a large dimensional number, so we treat them as high dimensional data and organize them with the **high dimensional indexes**.
 - Once again, the index is built on the features, not on the original data.
- Each original MM object is represented by a point in the **feature space**. How the features are abstracted is also part of the MMSDMBS design, so all the data stored in it share the same feature space.

Question 3 – query in MMDBMS



- During the search, user provides a **query object**.
- MMDBMS does not use this query object to search directly. Instead, it first abstracts the **features** of the query object with exactly the same process when it abstracts the data it stores.
- Then it uses the query features to **search in the feature space** with the help of high dimensional index.
- With the algorithm of kNN, it finds a list of points that are similar to the user's query object, and retrieves the actual data stored in the database, decompresses them, and returns them to the user.

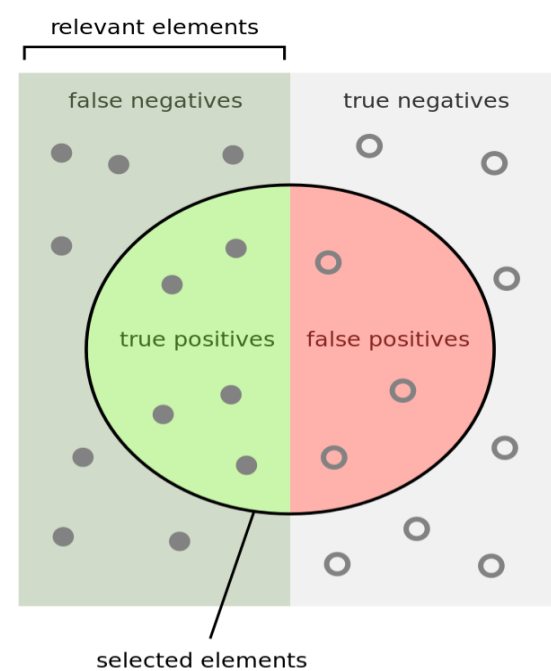
Question 3 – query in MMDBMS



- There is feedback loop for user to evaluate the query result and let the database refine the result.
 - Because all the results are **feature similarity-based**, there is no guarantee the results can satisfy the user's need.
- Therefore, this feedback loop can help identify which feature the user cares more and assigns higher weights to them during the search.
 - For example, if the user cares more about the colour, then in the next iteration, the search gives higher weight on the colour feature and lower weights on the other features like shape and texture.

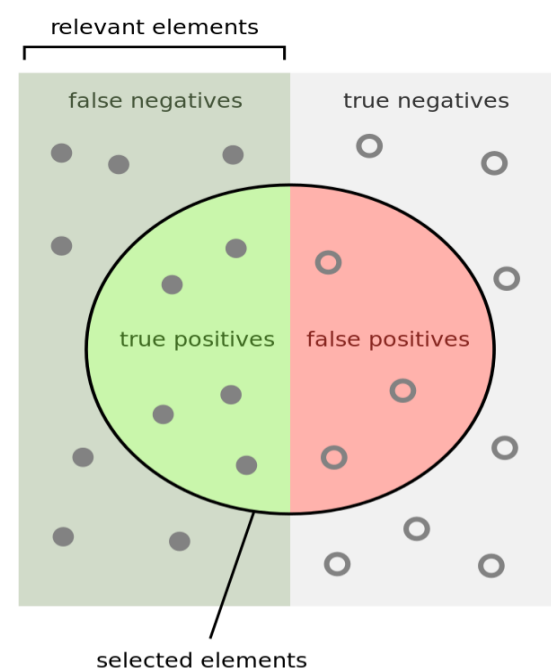
Question 4

- Q: Suppose we have 100 data in the database, 60 of them are positive, 40 of them are negative, and we want to retrieve all the positive ones. However, the system returns 50 results, and 40 of them are positive. What are the precision and recall of this result?
- As shown in the plot, the overall rectangle contains the **entire data set: [100]**
 - The dots (left half) represent the positive data **[60]**
 - The circles (right half) represent the negative data **[40]**



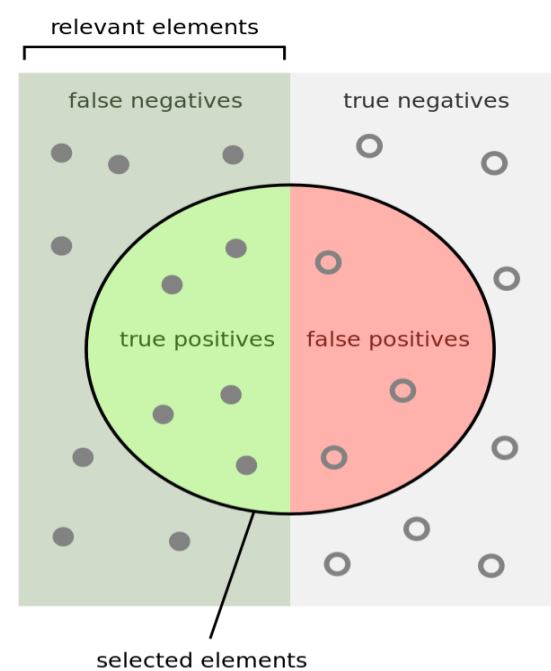
Question 4

- A **query** is finding all the positive data. The big circle is the result of query. [50]
- The dots in the green half are the correct ones, so they are call **true positive**: [40]
 - *true* because of correct
 - *positive* because the query asks for positive



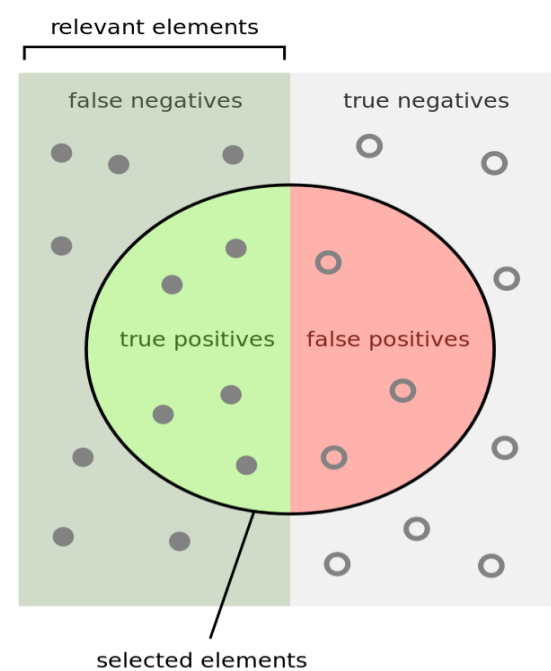
Question 4

- A **query** is finding all the positive data. The big circle is the result of query. [50]
- The dots in the green half are the correct ones, so they are call **true positive**: [40]
 - *true* because of correct
 - *positive* because the query asks for positive
- The circles in red half appear in the positive results, but they are actually negative, so they are not correct. We call them **false positive**: [10]
 - *false* because they are incorrect
 - *positive* because the query asks for positive



Question 4

- For all the data outside the result circle, the query result regards them as **negative**.
- However, on the left-hand side dots, they are positive, so this negative is not correct, and we call them **false negative**. $[60-40=20]$
- Similarly, the right circles are actually negative, so we call them **true negative**. $[40-10=30]$



Question 4 – answer

- The **precision** asks for the ratio of the correctly returned ones in the returned results:

$$\textit{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}} = \frac{40}{50} = 0.8.$$

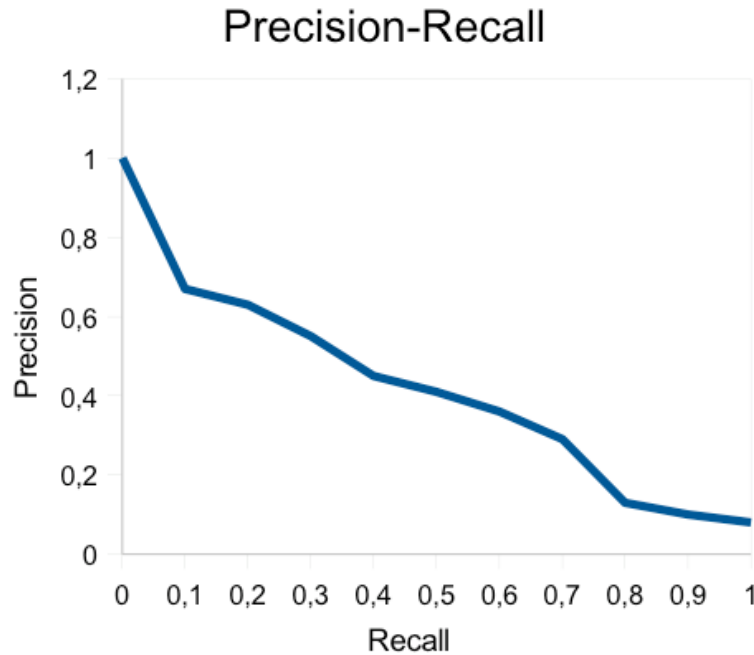
- The **recall** asks for the ration of the correctly returned ones in all the correct ones:

$$\textit{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False negative}} = \frac{40}{60} = 0.667.$$

Question 5

- We always want to achieve high precision and high recall in a system, but it is not always achievable. Please discuss the relation between the precision and recall.

+ Precision vs. Recall



$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

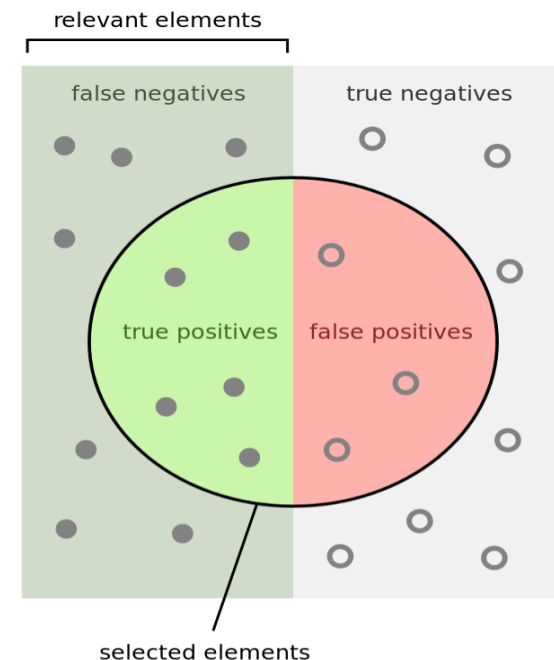
$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

■ Higher Recall

- Return more retrieved files, but may suffer from lower precision

■ Higher Precision

- May be increased by reducing the number of retrieved files, but may suffer from lower recall



Question 5 – answer

- Often, there is an inverse relationship between precision and recall, where it is possible to increase one **at the cost of** reducing the other.
- Brain surgery provides an illustrative example of the **trade-off**.
- Consider a brain surgeon tasked with removing a cancerous tumour from a patient's brain.
 - The surgeon needs to remove all of the tumour cells since any remaining cancer cells will regenerate the tumour.
 - Conversely, the surgeon must not remove healthy brain cells since that would leave the patient with impaired brain function.

Question 5 – answer

- The surgeon may be more liberal in the area of the brain she removes to ensure she has extracted all the cancer cells. **This decision increases recall but reduces precision.**
- On the other hand, the surgeon may be more conservative in the brain she removes to ensure she extracts only cancer cells. **This decision increases precision but reduces recall.**

Question 5 – answer

- That is to say, **greater recall** increases the chances of removing healthy cells (negative outcome) and increases the chances of removing all cancer cells (positive outcome).
- **Greater precision** decreases the chances of removing healthy cells (positive outcome), but also decreases the chances of removing all cancer cells (negative outcome).

Question 5 – answer

- Therefore, another performance measure, **F-measure**, is proposed to combine precision and recall by their harmonic mean, as shown below:

$$F = 2 \times \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}$$