Precedence Graph For Testing Conflict Serializability

Precedence Graph / Serialization Graph

Precedence Graph or Serialization Graph is a directed Graph $G = (V, E)$ consisting of a set of nodes V and a set of directed edges E, having the following properties:

1) • Each node $v_i \in V$ represents a transaction T_i
2) • A directed edge $(v_i, v_j) \in E$ exists in G if
    - One of the operations in T_i appears in the schedule before some conflicting operation in T_j

- Precedence Graph is used commonly to test Conflict Serializability of a schedule
- A Schedule S is serializable if there is no cycle in the corresponding precedence graph

$G(V, E)$

$V: \{v_1, v_2, v_3\}$

$E: \{v_1 v_2, v_2 v_3\}$

$\{e_1, e_2\}$

Precedence Graph For Testing Conflict Serializability

- Algorithm to generate precedence graph
  - Create a node in the graph for each participating transaction in the schedule
  - *For conflicting operation R(X) and W(X)*: If a Transaction T_j executes a R(X) after T_i executes a $W(X)$, draw an edge from T_i to T_j in the graph
  - *For conflicting operation W(X) and R(X)*: If a Transaction T_j executes a W(X) after T_i executes a R(X), draw an edge from T_i to T_j in the graph
  - *For conflicting operation W(X) and W(X)*: If a Transaction T_j executes a W(X) after T_i executes a W(X), draw an edge from T_i to T_j in the graph
- The Schedule is serializable if there is no cycle in the corresponding precedence graph
- No cycle in the precedence graph implies that there exists a serial schedule conflict equivalent to the given schedule

$\overset{1}{T_i} \longrightarrow \overset{2}{T_j}$

$W(x) \quad R(x)$

$\overset{1}{T_i} \longrightarrow \overset{2}{T_j}$

$R(x) \quad W(x)$

$T_i \longrightarrow T_j$

$W(x) \quad W(x)$

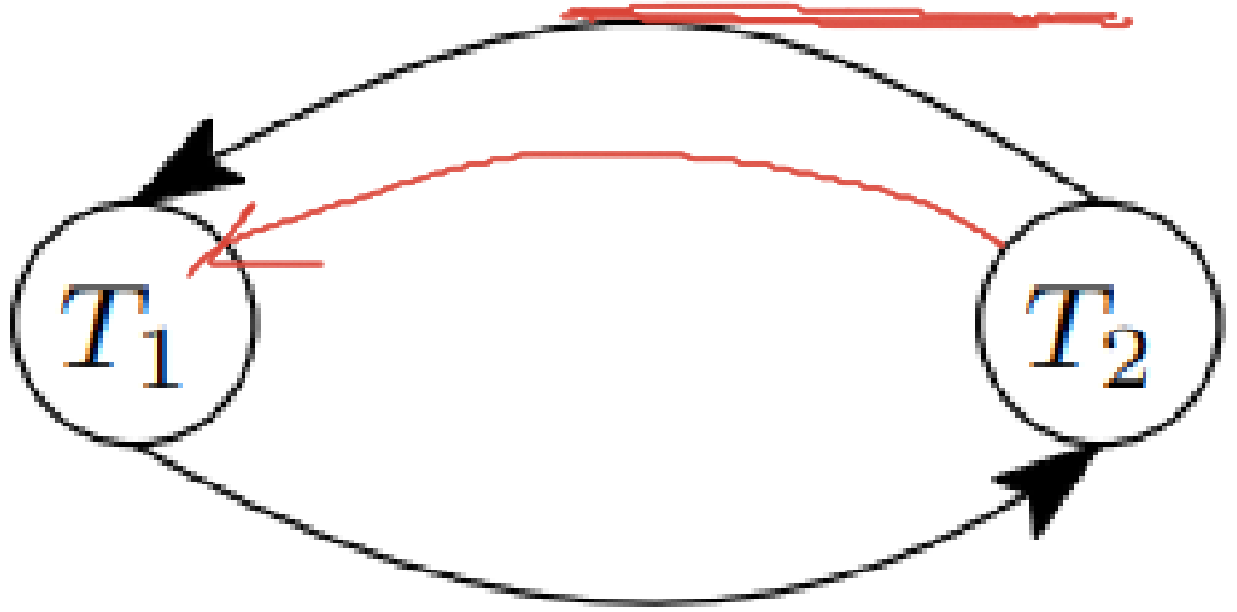
$T_1 \longrightarrow T_2$

Precedence Graph For Testing Conflict Serializability

Consider the following schedule: $S = R_1(x)R_1(y)W_2(x)W_1(x)R_2(y)$

T_2 T_1

$$R_1(x)R_1(y)W_2(x)W_1(x)R_2(y)$$



$$R_1(x)R_1(y)W_2(x)W_1(x)R_2(y)$$

Figure: Precedence Graph for schedule S

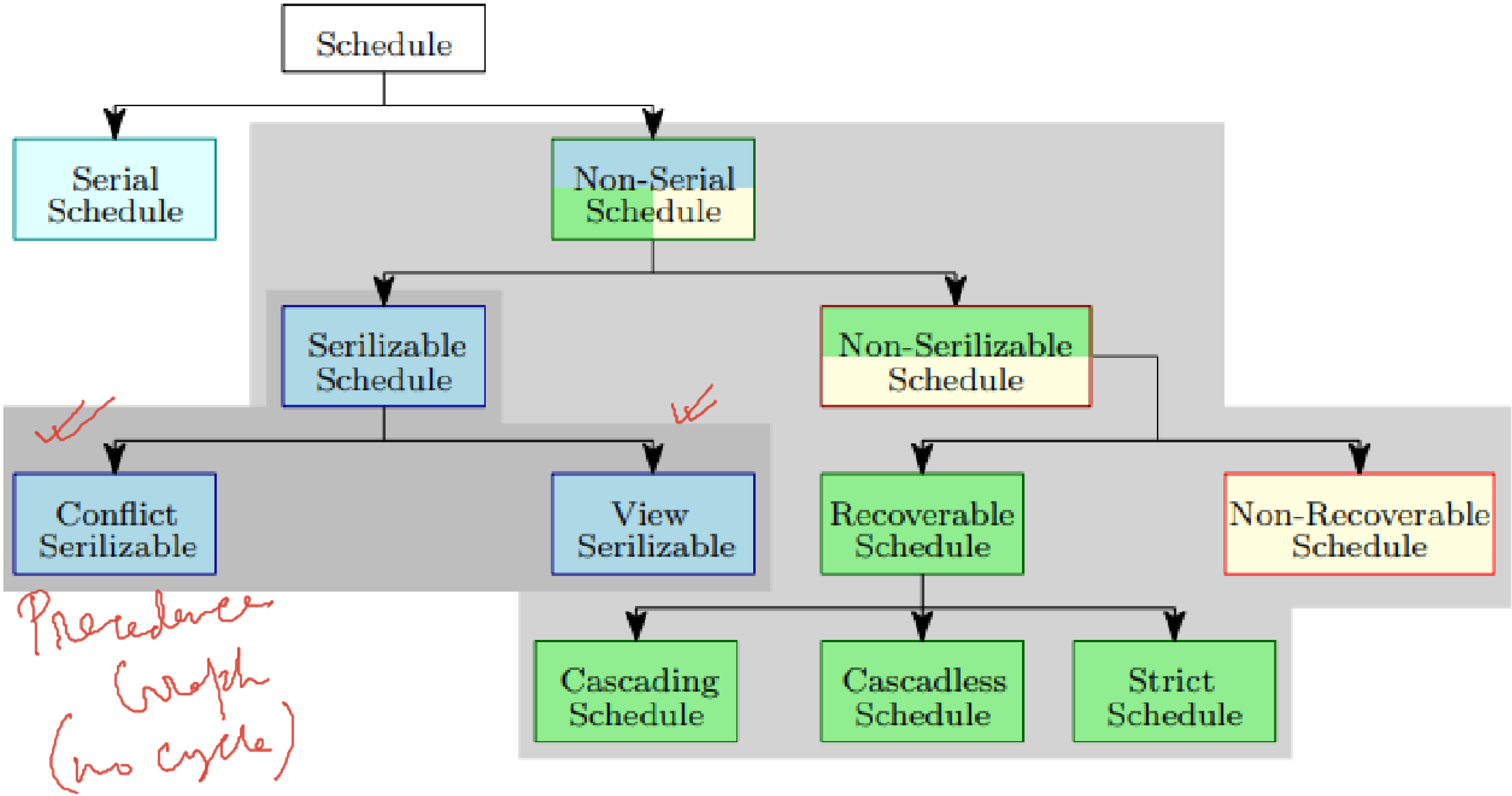
- Since the graph contains cycle, S is not conflict serializable

Problems

Precedence Graph

No cycle → conflict serializable

- $T_1 : R_1(A), W_1(A), R_1(B), W_1(B);$
  $T_2 : R_2(A), W_2(A), R_2(B), W_2(B)$
  $S_1 : R_2(A), W_2(A), R_1(A), W_1(A), R_1(B), W_1(B), R_2(B), W_2(B)$
  Find out whether S_1 is conflict serializable.
- $S_1 : R_2(A), W_2(A), R_2(B), W_1(A), R_1(B), W_1(B), R_1(A), W_2(B)$
  $S_2 : R_2(A), W_2(A), R_2(B), W_2(B), R_1(B), W_1(B), R_1(A), W_1(A)$
  Find out whether both the schedules are conflict equivalent.
- $S_1 : R_1(X)R_1(Y)R_2(X)R_2(Y)W_2(Y)W_1(X);$
  $S_2 : R_1(X)R_2(X)R_2(Y)W_2(Y)R_1(Y)W_1(X)$
  Find out whether S_1 and S_2 are conflict serializable schedules.
- $S_1 : R_1(x)R_3(y)W_1(x)W_2(y)R_3(x)W_2(x)$
  Find out whether S_1 is conflict serializable schedules.



Schedule
Serial Schedule
Non-Serial Schedule
Serilizable Schedule
Non-Serilizable Schedule
Conflict Serilizable
View Serilizable
Recoverable Schedule
Non-Recoverable Schedule
Cascading Schedule
Cascadless Schedule
Strict Schedule
Precedence Graph (no cycle)

View Serializable Schedule

View Equal

Two schedules S_1 and S_2 are said to be view equal if all the following conditions are satisfied :

1 • **Initial Read** : In S_1, if a transaction T_1 reading data item A from initial database; then in S_2 also T_1 should read A from initial database

T_1 R(A) T_2

2 • **Updated Read**: If T_i is reading A which is updated by T_j in S_1, then in S_2 also T_i should read A which is updated by T_j

W(A)

3 • **Final Write**: If a transaction T_1 updated A at last in S_1, then in S_2 also T_1 should perform final write operations W(A)

View Serializability

A Schedule is called view serializable if it is view equal to a serial schedule

Example of View Serializability

```
         S1                                S2
   Schedule 1                        Schedule 2
  --------------                   ---------------
   T1       T2                       T1       T2
  --------------                   ---------------
  r1(A)                            r1(A)
  A=A+10                           A=A+10
  w1(A)                            w1(A)
  r1(B)                                     r2(A)
  B=B*10                                    A=A+10
  w1(B)                                     w2(A)
           r2(A)                   r1(B)
           A=A+10                  B=B*10
           w2(A)                   w2(B)
           r2(B)                            r2(B)
           B=B*10                           B=B*10
           w2(B)                            w2(B)
```

Relationship between Conflict Serializability and View Serializability

CS → VS

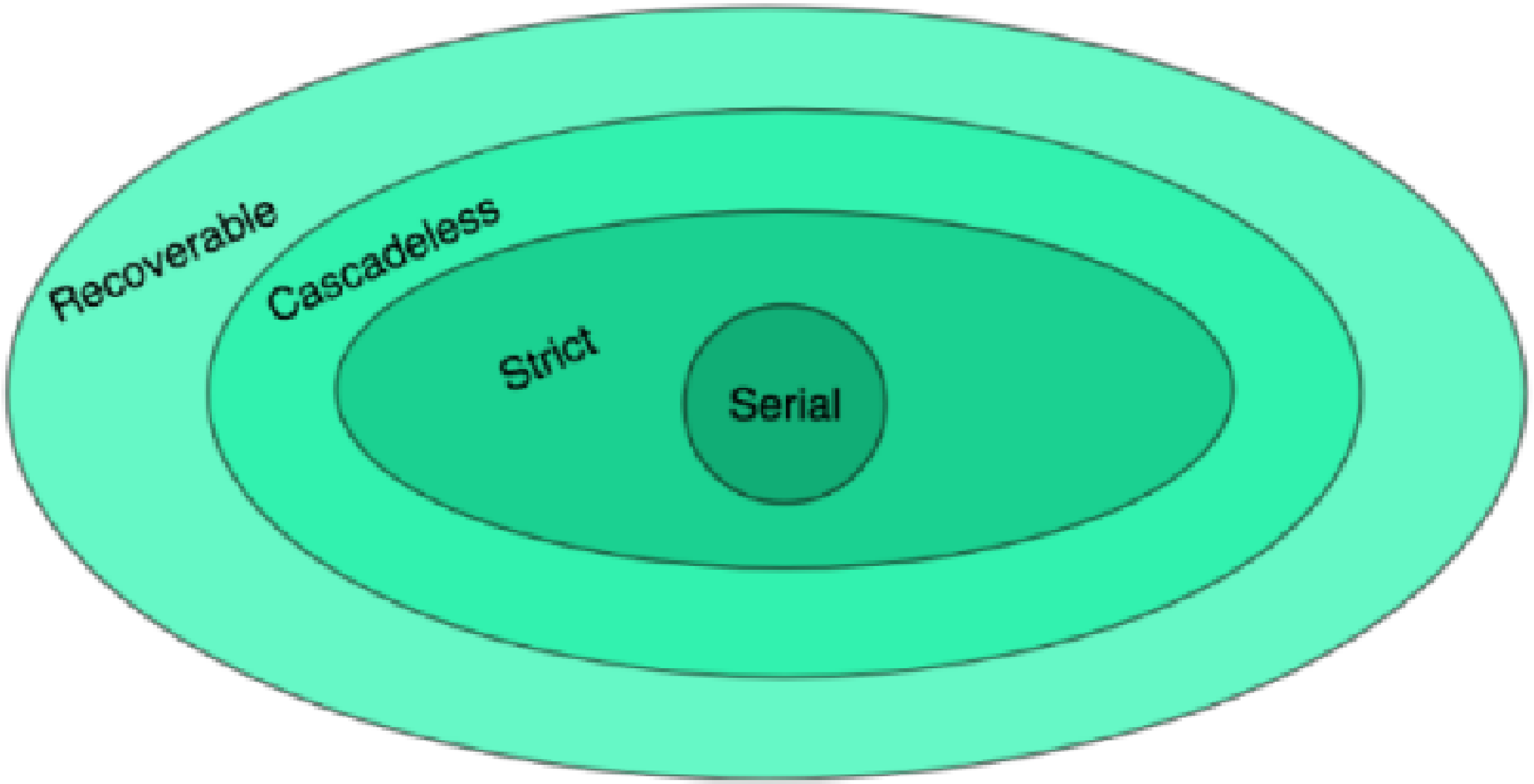
?

- A conflict serializable schedule is also view serializable
- A view serializable schedule may not be conflict serializable
  - Due to Blind Write (Write without Read)

T1	T2	T3
	$R_2(X)$	
$R_1(X)$		
		$W_3(X)$
	$W_2(X)$	

no $R_3(X)$ in T_3

- Cascadeless schedules are stricter than recoverable schedules or are a subset of recoverable schedules.
- Strict schedules are stricter than cascadeless schedules or are a subset of cascadeless schedules.
- Serial schedules satisfy constraints of all recoverable, cascadeless and strict schedules and hence is a subset of strict schedules.



Supervised ⟶ Lg. Reg.

Pred. ⟶ Lin R

✓ Supervised → KNN (in terms of classification)

✓ Unsupervised → K-means / K-mode ←

✓ Association Rule Mining

k-nearest neighbor (kNN)

$k = 3$

kNN – (i/p : k)

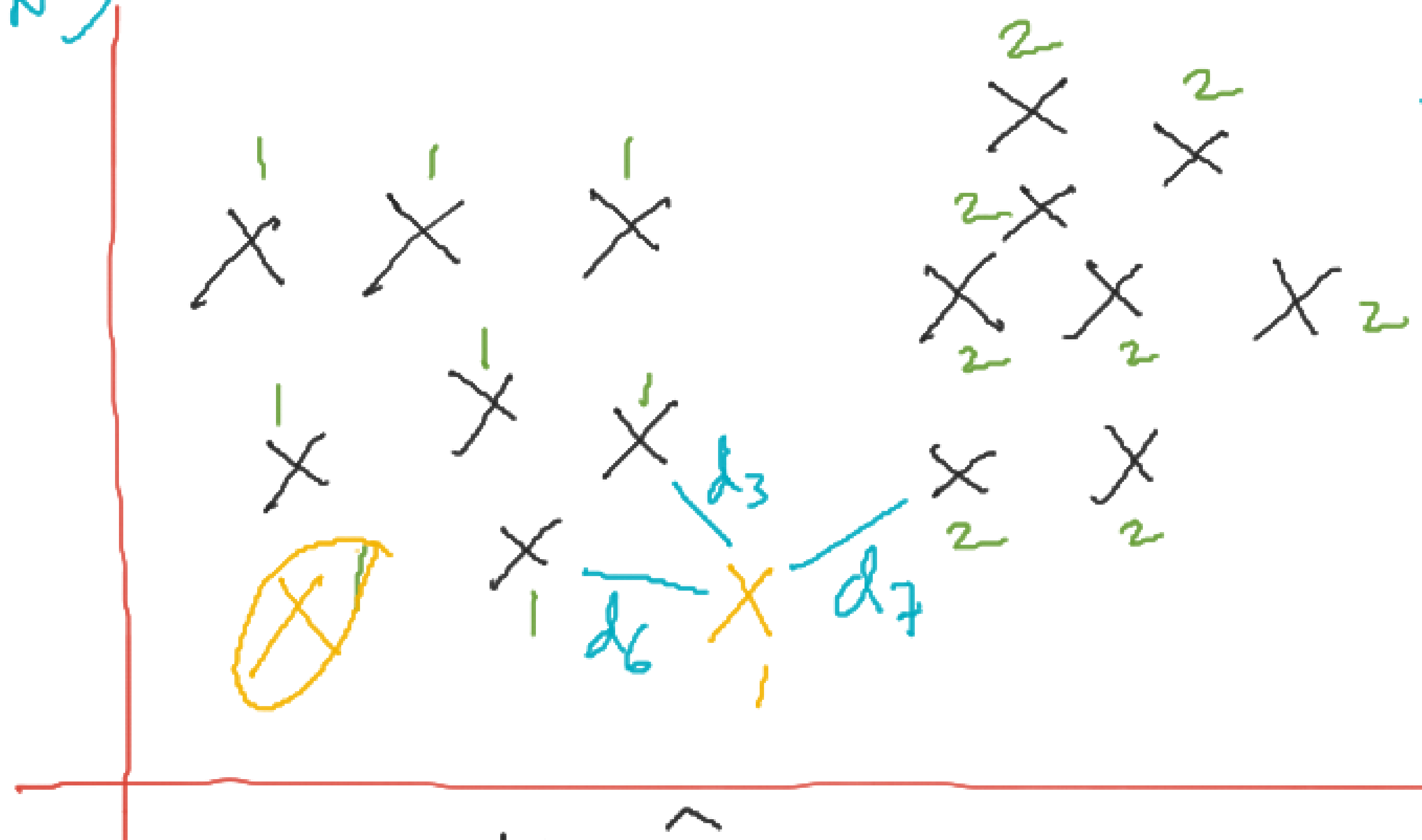
Supervised

$\{X^i, l^i\}$

$X^i \in R^n$

$\{x_1^i, x_2^i, \ldots, x_n^i\}$

x_j^i



3) label the new sample to majority class labelled nearest neighbors

for new sample $\hat{X}$

1) Calculate distances from $\hat{X}$ to X^i's

Distance vector D $\boxed{d_1 \mid d_2 \mid d_3 \mid \cdots \mid d_{15}}$

2) Sort D ascending order

low $\underbrace{\boxed{d_3 \mid d_6 \mid d_7} \boxed{d_8 \mid \cdots \mid \quad}}$ high

no. of sample = 15 $\quad i : 1 \to m$

features $\quad j : 1 \to n$

K-Nearest Neighbor (KNN):
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Look at k nearest neighbors of unknown sample

label the unknown sample to the class from which highest number of members we have in the k nearest neighbors

Empirically, choose k as odd number

How to calculate nearest neighbors:
- Calculate distances from the unknown sample to all other samples
- Sort the distances in ascending order

How to calculate distance: e.g., Euclidean distance

Limitation: you need to input K (no. of neighbors)
Here, K is a hyperparameter

https://www.youtube.com/watch?v=DIQIi0OCkf8

https://en.wikipedia.org/wiki/Chivukula_Anjaneya_Murthy