STMIK
Amik Riau

# MATAKULIAH
# BASIS DATA TERDISTRIBUSI

Rahmaddeni, M.Kom
NIDN : 1007128301


## Pertemuan 7 – Manajemen Deadlock

STMIK
Amik Riau

# #01

## DEADLOCK

# DEFENISI dan LATAR BELAKANG

**Deadlock** adalah keadaan dimana dua program memegang kontrol terhadap sumber daya yang dibutuhkan oleh program yang lain. Tidak ada yang dapat melanjutkan proses masing-masing sampai program yang lain memberikan sumber dayanya, tetapi tidak ada yang mengalah.

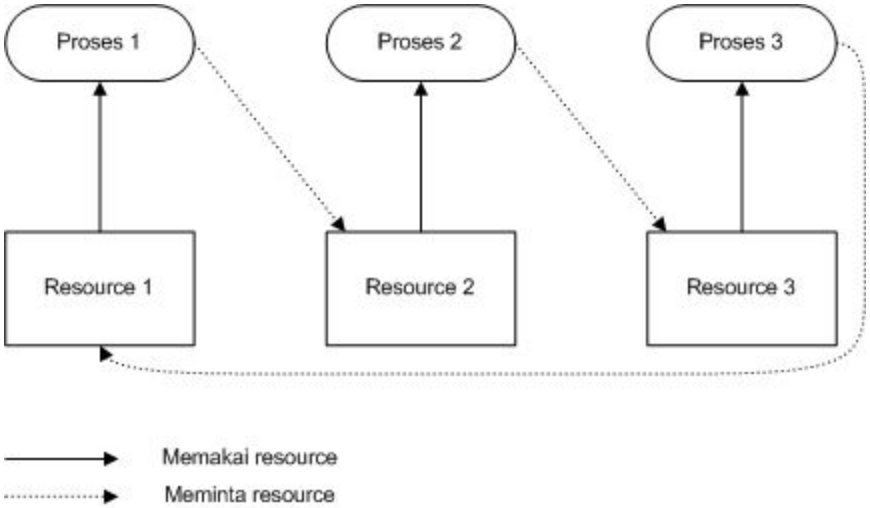
- Analoginya seperti pada kondisi jalan raya dimana terjadi kemacetan parah
- Deadlock adalah efek samping dari sinkronisasi, dimana satu variabel digunakan oleh 2 proses

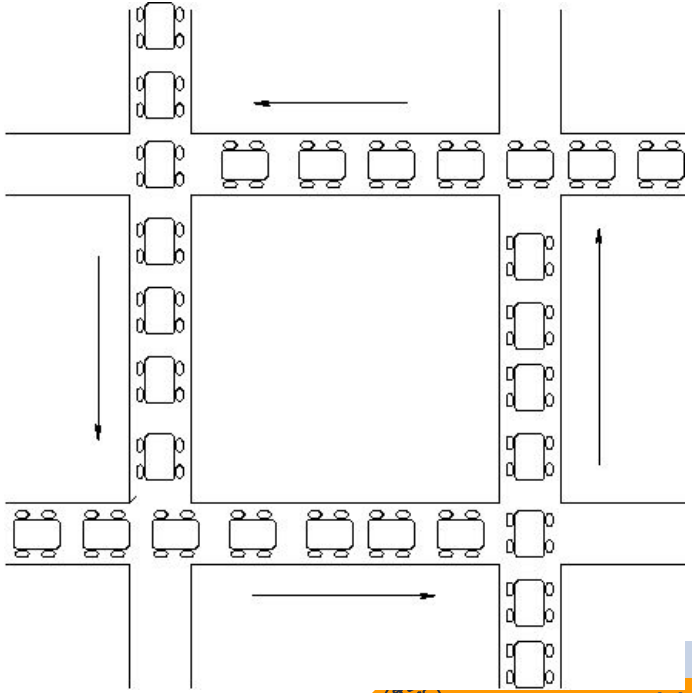
# DEFENISI dan LATAR BELAKANG

- Deadlock dalam arti sebenarnya adalah kebuntuan dalam sistem operasi adalah kebuntuan proses. Jadi Deadlock ialah suatu kondisi dimana proses tidak berjalan lagi atau pun tidak ada komunikasi lagi antar proses.

- Deadlock disebabkan karena proses yang satu menunggu sumber daya yang sedang dipegang oleh proses lain yang sedang menunggu sumber daya yang dipegang oleh proses tersebut..

# MODEL DEADLOCK



# PRINSIP DEADLOCK

## Prinsip Deadlock

### Situasi

- Proses diberi akses eksklusif ke device.
- Device ini disebut sebagai resources/sumber daya.

### Resources

- Preemptable
  Contoh : Memori Utama.
  Memori dipakai bergantian tanpa menimbulkan fail.
- Non Preemptable
  Contoh : Printer
  Proses printing tidak dapat diganggu.

### Penggunaan Resources

1. Request.
2. Use.
3. Release.

### Penyebab

1. Mutual Exclusion.
2. Hold and Wait.
3. No Preemption.
4. Circular Wait.

# PENYEBAB DEADLOCK

- Mutual Exclusion
- Hold and Wait
- No Preemption
- Circular Waiting

Deadlock harus memenuhi semua syarat diatas

STMIK
Amik Riau

# #02

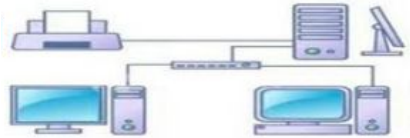
# MUTUAL EXCLUSION

# DEFENISI

Kondisi yang pertama adalah **mutual exclusion** yaitu proses memiliki hak milik pribadi terhadap sumber daya yang sedang digunakannya. Jadi, hanya ada satu proses yang menggunakan suatu sumber daya. Proses lain yang juga ingin menggunakannya harus menunggu hingga sumber daya tersebut dilepaskan oleh proses yang telah selesai menggunakannya.

Ilustrasinya :

- Pada Printer Daemon
- Aplikasi Tabungan



STMIK
Amik Riau

# #03

## HOLD and WAIT

# DEFENISI

Kondisi yang kedua adalah **hold and wait** yaitu beberapa proses saling menunggu sambil menahan sumber daya yang dimilikinya. Suatu proses yang memiliki minimal satu buah sumber daya melakukan request lagi terhadap sumber daya. Akan tetapi, sumber daya yang dimintanya sedang dimiliki oleh proses yang lain.

STMIK
Amik Riau

# #04

## NO PREEMPTION

# DEFENISI

Kondisi yang selanjutnya adalah **no preemption** yaitu sebuah sumber daya hanya dapat dilepaskan oleh proses yang memilikinya secara sukarela setelah ia selesai menggunakannya. Proses yang menginginkan sumber daya tersebut harus menunggu sampai sumber daya tersedia, tanpa bisa merebutnya dari proses yang memilikinya.

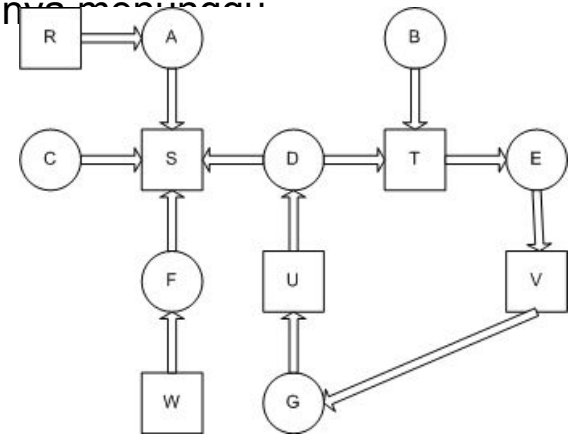
STMIK
Amik Riau

# #05

## CIRCULAR WAITING

# DEFENISI

Kondisi yang terakhir adalah **circular wait** yaitu kondisi membentuk siklus yang berisi proses-proses yang saling membutuhkan. Proses pertama membutuhkan sumber daya yang dimiliki proses kedua, proses kedua membutuhkan sumber daya milik proses ketiga, dan seterusnya sampai proses ke n-1 yang membutuhkan sumber daya milik proses ke n. Terakhir, proses ke n membutuhkan sumber daya milik proses yang pertama. Yang terjadi adalah proses-proses tersebut akan selamanya menunggu.



# STRATEGI MENGATASI DEADLOCK

- Prevention : memastikan paling sedikit satu penyebab Deadlock tidak berlaku
- Avoidance : sistem menolak request terhadap resource yang berpotensi deadlock, Algoritma Banker
- Detection and Recovery : membiarkan Deadlock terjadi, lalu mendeteksinya, kemudian melakukan recovery, Algoritma Ostrich

STMIK
Amik Riau

# #06

# ALGORITMA BANKER

# DEFENISI

Algoritma Banker adalah suatu metode untuk mengatasi deadlock yang dikemukakan oleh W.Dijkstra. Disebut banker karena memodelkan banker di sebuah kota kecil yang berhubungan dengan sekumpulan nasabah yang memohon pinjaman/kredit yang dapat dianalogikan sebagai berikut :

- Sistem operasi sebagai bank
- Resource sebagai uang
- Proses sebagai nasabah

# DEFENISI

Secara umum algoritma bankir dapat dibagi menjadi 4 struktur data:

1. Tersedia: jumlah sumber daya/dana yang tersedia
2. Maksimum: jumlah sumber daya maksimum yang diminta oleh setiap proses
3. Alokasi: jumlah sumber daya yang dibutuhkan oleh setiap proses
4. Kebutuhan: sumber daya yang sedang dibutuhkan oleh setiap proses

# Analogi Kerja Algoritma Banker


Cara kerja algoritma banker adalah dengan mempertimbangkan permintaan pinjaman yang diajukan oleh nasabah tersebut sesuai dengan uang yang dimiliki oleh bank, sekaligus mempertimbangkan jumlah pinjaman yang akan diajukan lagi oleh nasabah tersebut/nasabah lain.

Setiap nasabah memiliki batas kredit dan apabila nasabah telah mencapai batas kredit pinjaman, maka diasumsikan bahwa nasbah tersebut telah menyelesaikan bisnisnya dan dapat mengembalikan pinjamannya kepada bank.

Jangan sampai dana yang ada pada bank habis dan tidak dapat melakukan pinjaman lagi atau yang disebut deadlock.

# Analogi Kerja Algoritma Banker

## Istilah dalam Algoritma Banker

**Allocation**

Jumlah resource yang dibutuhkan oleh setiap proses.

**Max**

Jumlah resource maksimal yang diminta oleh setiap proses.

**Available**

Jumlah resource yang tersedia.

**Need**

Jumlah resource yang dibutuhkan oleh setiap proses.

# CONTOH KASUS

| | ALLOCATION | | | | MAX | | | | AVAILABLE | | | | NEED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| S | | | | | 4 | 7 | 5 | 3 | 3 | 0 | 7 | 6 | 0 | 0 | 1 | 2 |
| I | | | | | 5 | 4 | 8 | 9 | | | | | 2 | 0 | 4 | 5 |
| S | | | | | 8 | 4 | 9 | 9 | | | | | 6 | 2 | 1 | 2 |
| T | | | | | 7 | 4 | 3 | 2 | | | | | 3 | 4 | 2 | 0 |
| E | | | | | 5 | 5 | 6 | 8 | | | | | 0 | 0 | 1 | 4 |
| M | | | | | 8 | 3 | 1 | 9 | | | | | 2 | 2 | 0 | 5 |
| O | | | | | 7 | 9 | 2 | 3 | | | | | 3 | 0 | 0 | 2 |
| P | | | | | 2 | 4 | 8 | 7 | | | | | 2 | 0 | 7 | 6 |
| E | | | | | 6 | 3 | 4 | 7 | | | | | 4 | 0 | 2 | 5 |
| R | | | | | 9 | 2 | 4 | 3 | | | | | 4 | 1 | 2 | 0 |
| A | | | | | 4 | 7 | 4 | 5 | | | | | 2 | 0 | 4 | 3 |
| S | | | | | 3 | 3 | 2 | 9 | | | | | 2 | 0 | 2 | 5 |
| I | | | | | 7 | 7 | 5 | 3 | | | | | 5 | 4 | 0 | 2 |
| TOTAL RECOURCES | | | | | | | | | | | | | | | | |

1. Bagaimana isi dari Allocation Matrix?
2. Berapa total sumber daya dari A, B, C, dan D
3. Apakah system dalam keadaan aman (Safe State)? Jika iya, temukan urutan amannya (Safe Sequence).

# JAWABAN

1. Untuk mencari Allocation = Max – Need
   Mis : Untuk Proses S
   **Allocation** (S) = Max – Need = 4 7 5 3 – 0 0 1 2 = **4 7 4 1** , Begitu seterusnya sampai proses I

| | ALLOCATION | | | | MAX | | | | AVAILABLE | | | | NEED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| S | 4 | 7 | 4 | 1 | 4 | 7 | 5 | 3 | 3 | 0 | 7 | 6 | 0 | 0 | 1 | 2 |
| I | 3 | 4 | 4 | 4 | 5 | 4 | 8 | 9 | | | | | 2 | 0 | 4 | 5 |
| S | 2 | 2 | 8 | 7 | 8 | 4 | 9 | 9 | | | | | 6 | 2 | 1 | 2 |
| T | 4 | 0 | 1 | 2 | 7 | 4 | 3 | 2 | | | | | 3 | 4 | 2 | 0 |
| E | 5 | 5 | 5 | 4 | 5 | 5 | 6 | 8 | | | | | 0 | 0 | 1 | 4 |
| M | 6 | 1 | 1 | 4 | 8 | 3 | 1 | 9 | | | | | 2 | 2 | 0 | 5 |
| O | 4 | 9 | 2 | 1 | 7 | 9 | 2 | 3 | | | | | 3 | 0 | 0 | 2 |
| P | 0 | 4 | 1 | 1 | 2 | 4 | 8 | 7 | | | | | 2 | 0 | 7 | 6 |
| E | 2 | 3 | 2 | 2 | 6 | 3 | 4 | 7 | | | | | 4 | 0 | 2 | 5 |
| R | 5 | 1 | 2 | 3 | 9 | 2 | 4 | 3 | | | | | 4 | 1 | 2 | 0 |
| A | 2 | 7 | 0 | 2 | 4 | 7 | 4 | 5 | | | | | 2 | 0 | 4 | 3 |
| S | 1 | 3 | 0 | 4 | 3 | 3 | 2 | 9 | | | | | 2 | 0 | 2 | 5 |
| I | 2 | 3 | 5 | 1 | 7 | 7 | 5 | 3 | | | | | 5 | 4 | 0 | 2 |
| TOTAL RECOURCES | | | | | | | | | | | | | | | | |

# JAWABAN

| | ALLOCATION | | | | MAX | | | | AVAILABLE | | | | NEED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| S | 4 | 7 | 4 | 1 | 4 | 7 | 5 | 3 | 3 | 0 | 7 | 6 | 0 | 0 | 1 | 2 |
| I | 3 | 4 | 4 | 4 | 5 | 4 | 8 | 9 | | | | | 2 | 0 | 4 | 5 |
| S | 2 | 2 | 8 | 7 | 8 | 4 | 9 | 9 | | | | | 6 | 2 | 1 | 2 |
| T | 4 | 0 | 1 | 2 | 7 | 4 | 3 | 2 | | | | | 3 | 4 | 2 | 0 |
| E | 5 | 5 | 5 | 4 | 5 | 5 | 6 | 8 | | | | | 0 | 0 | 1 | 4 |
| M | 6 | 1 | 1 | 4 | 8 | 3 | 1 | 9 | | | | | 2 | 2 | 0 | 5 |
| O | 4 | 9 | 2 | 1 | 7 | 9 | 2 | 3 | | | | | 3 | 0 | 0 | 2 |
| P | 0 | 4 | 1 | 1 | 2 | 4 | 8 | 7 | | | | | 2 | 0 | 7 | 6 |
| E | 2 | 3 | 2 | 2 | 6 | 3 | 4 | 7 | | | | | 4 | 0 | 2 | 5 |
| R | 5 | 1 | 2 | 3 | 9 | 2 | 4 | 3 | | | | | 4 | 1 | 2 | 0 |
| A | 2 | 7 | 0 | 2 | 4 | 7 | 4 | 5 | | | | | 2 | 0 | 4 | 3 |
| S | 1 | 3 | 0 | 4 | 3 | 3 | 2 | 9 | | | | | 2 | 0 | 2 | 5 |
| I | 2 | 3 | 5 | 1 | 7 | 7 | 5 | 3 | | | | | 5 | 4 | 0 | 2 |
| TOTAL RECOURCES | | | | | | | | | | | | | | | | |

2. Mengecek proses apakah proses pada keadaan aman (Safe State) atau Unsafe State
Jika Need <= Available maka proses aman

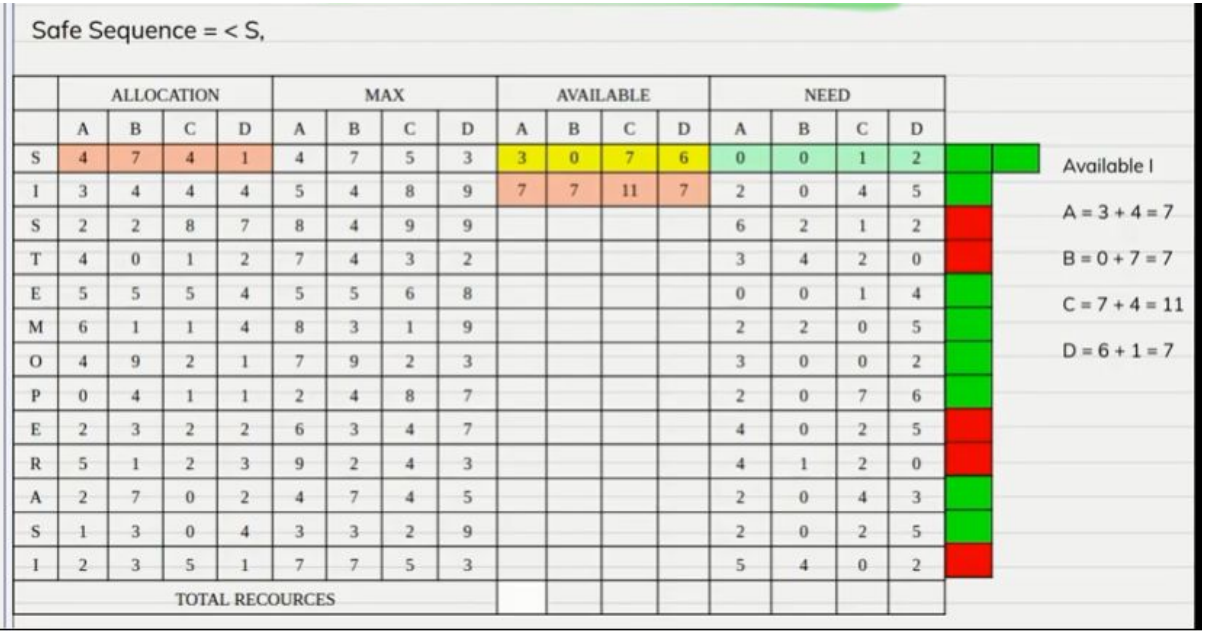
Proses S : Need = 0012 dan Available 3076, karena Need <= Available = Safe Sate ,Jadi S = **Safe State**

Proses T : Need = 3420 dan Available 3076, karena Need <= Available = **Un safe State** disebabkan karena di B (Need=4) sementara (Available=0)

Safe State

Unsafe State

# JAWABAN



Safe Sequence = < S,

| | ALLOCATION | | | | MAX | | | | AVAILABLE | | | | NEED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| S | 4 | 7 | 4 | 1 | 4 | 7 | 5 | 3 | 3 | 0 | 7 | 6 | 0 | 0 | 1 | 2 |
| I | 3 | 4 | 4 | 4 | 5 | 4 | 8 | 9 | 7 | 7 | 11 | 7 | 2 | 0 | 4 | 5 |
| S | 2 | 2 | 8 | 7 | 8 | 4 | 9 | 9 | | | | | 6 | 2 | 1 | 2 |
| T | 4 | 0 | 1 | 2 | 7 | 4 | 3 | 2 | | | | | 3 | 4 | 2 | 0 |
| E | 5 | 5 | 5 | 4 | 5 | 5 | 6 | 8 | | | | | 0 | 0 | 1 | 4 |
| M | 6 | 1 | 1 | 4 | 8 | 3 | 1 | 9 | | | | | 2 | 2 | 0 | 5 |
| O | 4 | 9 | 2 | 1 | 7 | 9 | 2 | 3 | | | | | 3 | 0 | 0 | 2 |
| P | 0 | 4 | 1 | 1 | 2 | 4 | 8 | 7 | | | | | 2 | 0 | 7 | 6 |
| E | 2 | 3 | 2 | 2 | 6 | 3 | 4 | 7 | | | | | 4 | 0 | 2 | 5 |
| R | 5 | 1 | 2 | 3 | 9 | 2 | 4 | 3 | | | | | 4 | 1 | 2 | 0 |
| A | 2 | 7 | 0 | 2 | 4 | 7 | 4 | 5 | | | | | 2 | 0 | 4 | 3 |
| S | 1 | 3 | 0 | 4 | 3 | 3 | 2 | 9 | | | | | 2 | 0 | 2 | 5 |
| I | 2 | 3 | 5 | 1 | 7 | 7 | 5 | 3 | | | | | 5 | 4 | 0 | 2 |
| TOTAL RECOURCES | | | | | | | | | | | | | | | | |

Available I

A = 3 + 4 = 7

B = 0 + 7 = 7

C = 7 + 4 = 11

D = 6 + 1 = 7

3. Mencari Safe Sequence dengan cara membandingkan Need dan juga Available Serta mencari Available selanjutnya 
Available = Available + Allocation

Misal : Proses S  Need  = 0012 dan Available = 3076 , sehingga S dikatakan dalam keadaan Safe State
Dan kita masukkan proses S pada Safe Sequence yang pertama

Untuk Availale selanjutnya yaitu Available I caranya = Available pada proses S + Allocation pada proses S
Yaitu 3 0 7 6 + 4 7 4 1 = **7 7 11 7**

Safe State

Unsafe State

# JAWABAN

Safe Sequence = < S, I, S, T, E, M,

| | ALLOCATION | | | | MAX | | | | AVAILABLE | | | | NEED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| S | 4 | 7 | 4 | 1 | 4 | 7 | 5 | 3 | 3 | 0 | 7 | 6 | 0 | 0 | 1 | 2 |
| I | 3 | 4 | 4 | 4 | 5 | 4 | 8 | 9 | 7 | 7 | 11 | 7 | 2 | 0 | 4 | 5 |
| S | 2 | 2 | 8 | 7 | 8 | 4 | 9 | 9 | 10 | 11 | 15 | 11 | 6 | 2 | 1 | 2 |
| T | 4 | 0 | 1 | 2 | 7 | 4 | 3 | 2 | 12 | 13 | 23 | 18 | 3 | 4 | 2 | 0 |
| E | 5 | 5 | 5 | 4 | 5 | 5 | 6 | 8 | 16 | 13 | 24 | 20 | 0 | 0 | 1 | 4 |
| M | 6 | 1 | 1 | 4 | 8 | 3 | 1 | 9 | 21 | 18 | 29 | 24 | 2 | 2 | 0 | 5 |
| O | 4 | 9 | 2 | 1 | 7 | 9 | 2 | 3 | 27 | 19 | 30 | 28 | 3 | 0 | 0 | 2 |
| P | 0 | 4 | 1 | 1 | 2 | 4 | 8 | 7 | | | | | 2 | 0 | 7 | 6 |
| E | 2 | 3 | 2 | 2 | 6 | 3 | 4 | 7 | | | | | 4 | 0 | 2 | 5 |
| R | 5 | 1 | 2 | 3 | 9 | 2 | 4 | 3 | | | | | 4 | 1 | 2 | 0 |
| A | 2 | 7 | 0 | 2 | 4 | 7 | 4 | 5 | | | | | 2 | 0 | 4 | 3 |
| S | 1 | 3 | 0 | 4 | 3 | 3 | 2 | 9 | | | | | 2 | 0 | 2 | 5 |
| I | 2 | 3 | 5 | 1 | 7 | 7 | 5 | 3 | | | | | 5 | 4 | 0 | 2 |
| TOTAL RECOURCES | | | | | | | | | | | | | | | | |

Available O

A = 21 + 6 = 27

B = 18 + 1 = 19

C = 29 + 1 = 30

D = 24 + 4 = 28

3. Mencari Safe Sequence dengan cara membandingkan Need dan juga Available Serta mencari Available selanjutnya □
Available = Available + Allocation

Misal : Proses M □ Need = 2205 dan Available = 21 18 29 24 , sehingga M dikatakan dalam keadaan Safe State
Dan kita masukkan proses M pada Safe Sequence yang Keenam

Untuk Availale selanjutnya yaitu Available O
caranya = Available pada proses M + Allocation pada proses M
Yaitu 21 18 29 24 + 6 1 1 4 = **27 19 30 28**

Safe State

Unsafe State

# JAWABAN

Safe Sequence = < S, I, S, T, E, M, O, P, E, R, A, S, I >

| | ALLOCATION | | | | MAX | | | | AVAILABLE | | | | NEED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| S | 4 | 7 | 4 | 1 | 4 | 7 | 5 | 3 | 3 | 0 | 7 | 6 | 0 | 0 | 1 | 2 |
| I | 3 | 4 | 4 | 4 | 5 | 4 | 8 | 9 | 7 | 7 | 11 | 7 | 2 | 0 | 4 | 5 |
| S | 2 | 2 | 8 | 7 | 8 | 4 | 9 | 9 | 10 | 11 | 15 | 11 | 6 | 2 | 1 | 2 |
| T | 4 | 0 | 1 | 2 | 7 | 4 | 3 | 2 | 12 | 13 | 23 | 18 | 3 | 4 | 2 | 0 |
| E | 5 | 5 | 5 | 4 | 5 | 5 | 6 | 8 | 16 | 13 | 24 | 20 | 0 | 0 | 1 | 4 |
| M | 6 | 1 | 1 | 4 | 8 | 3 | 1 | 9 | 21 | 18 | 29 | 24 | 2 | 2 | 0 | 5 |
| O | 4 | 9 | 2 | 1 | 7 | 9 | 2 | 3 | 27 | 19 | 30 | 28 | 3 | 0 | 0 | 2 |
| P | 0 | 4 | 1 | 1 | 2 | 4 | 8 | 7 | 31 | 28 | 32 | 29 | 2 | 0 | 7 | 6 |
| E | 2 | 3 | 2 | 2 | 6 | 3 | 4 | 7 | 31 | 32 | 33 | 30 | 4 | 0 | 2 | 5 |
| R | 5 | 1 | 2 | 3 | 9 | 2 | 4 | 3 | 33 | 35 | 35 | 32 | 4 | 1 | 2 | 0 |
| A | 2 | 7 | 0 | 2 | 4 | 7 | 4 | 5 | 38 | 36 | 37 | 35 | 2 | 0 | 4 | 3 |
| S | 1 | 3 | 0 | 4 | 3 | 3 | 2 | 9 | 40 | 43 | 37 | 37 | 2 | 0 | 2 | 5 |
| I | 2 | 3 | 5 | 1 | 7 | 7 | 5 | 3 | 41 | 46 | 37 | 41 | 5 | 4 | 0 | 2 |
| TOTAL RECOURCES | | | | | | | | | 43 | 49 | 42 | 42 | | | | |

Total Resources

A = 41 + 2 = 43

B = 46 + 3 = 49

C = 37 + 5 = 42

D = 41 + 1 = 42

4. Mencari Total Resources
Setelah selesai Proses pengisian Available I yang terakhir, maka untuk menghitung Total Resources nya adalah

Available I + Allocation I
41 46 37 41 + 2 3 5 1 = **43 49 42 42**

Dan Safe Sequence yang terbentuk adalah

**< S, I, S, T, E, M, O, P, E, R, A, S, I>**

Safe State

Unsafe State

# CONTOH BERIKUTNYA

Terdapat 5 proses P0 sampai P4, 3 tipe sumber daya yaitu :
A (10 Anggota), B (5 Anggota), C (7 anggota)

| Process | Allocation | Max | Available | Need |
|---|---|---|---|---|
| | A B C | A B C | A B C | A B C |
| P0 | 0 1 0 | 7 5 3 | | |
| P1 | 2 0 0 | 3 2 2 | | |
| P2 | 3 0 2 | 9 0 2 | | |
| P3 | 2 1 1 | 2 2 2 | | |
| P4 | 0 0 2 | 4 3 3 | | |

# JAWABANNYA

| Process | Allocation | Max | Available | Need |
|---|---|---|---|---|
| | A B C | A B C | A B C | A B C |
| P0 | 0 1 0 | 7 5 3 | **3 3 2** | **7 4 3** |
| P1 | 2 0 0 | 3 2 2 | | **1 2 2** |
| P2 | 3 0 2 | 9 0 2 | | **6 0 0** |
| P3 | 2 1 1 | 2 2 2 | | **0 1 1** |
| P4 | 0 0 2 | 4 3 3 | | **4 3 1** |

Mendapatkan Nilai Available **3 3 2** itu caranya adalah
Allocation A yaitu 0 + 2 + 3 + 2 + 0 = 7 sementara anggota A 10 , sehingga ditambah **3**
Allocation B yaitu 1 + 0 + 0 + 1 + 0 = 2 sementara anggota B 5 , sehingga ditambah **3**
Allocation C yaitu 0 + 0 + 2 + 1 + 2 = 5 sementara anggota C 7 , sehingga ditambah **2**

Mendapatkan Nilai Need dari Proses P0 sampai P4 caranya adalah
P0 = Max – Allocation = 7 5 3 – 0 1 0 = **7 4 3**
P1 = 3 2 2 – 2 0 0 = **1 2 2**
Dst.

# JAWABANNYA

## Algoritma Safety pada Banker Algoritm

If

need ≤ available

Then

execute Process

new available = available + allocation

Else

do not execute go forward

# JAWABANNYA

| Process | Allocation | Max | Available | Need |
|---|---|---|---|---|
| | A B C | A B C | A B C | A B C |
| P0 | 0 1 0 | 7 5 3 | 3 3 2 | 7 4 3 |
| P1 | 2 0 0 | 3 2 2 | 5 3 2 | 1 2 2 |
| P2 | 3 0 2 | 9 0 2 | 7 4 3 | 6 0 0 |
| P3 | 2 1 1 | 2 2 2 | 7 4 5 | 0 1 1 |
| P4 | 0 0 2 | 4 3 3 | 7 5 5 | 4 3 1 |
| | | | 10 4 7 | |

P0 → need ≤ available

743 ≤ 332 ✗

do not execute P0 go forward

P1 → 122 ≤ 332 ✓

Excecute P1

New available = available + Allocation

New available = 332 + 200 → 532

P2 → 600 ≤ 532 ✗

do not execute P2 go forward

P3 → 011 ≤ 532

Excecute P3 ✓

New available = 532 + 211 → 743

P4 → 431 ≤ 743

Excecute P4 ✓

New available = 743 + 002 → 745

P0 → 743 ≤ 745

Excecute P0 ✓

New available = 745 + 010 → 755

P2 → 122 ≤ 755

Excecute P2 ✓

New available = 755 + 302 → 10 5 7

Urutan Eksekusi Proses < P1, P3, P4, P0, P2 >

Sistem dalam keadaan state selamat dan memenuhi kriteria **Safety**

# KESIMPULAN

- Deadlock adalah situasi dimana 1 atau lebih proses tidak akan pernah selesai tanpa adanya recovery
- Empat kondisi penting untuk deadlock : mutual exclusion, hold and wait, circular wait, and no preemption
- Deadlock bisa diatasi oleh berbagai strategi : prevention, avoidance, detection and recovery

STMIK Amik Riau

# THANKS!

Ada Pertanyaan?
Boleh juga ke
WA
FB
IG