Sure, let's create schedules for each scenario:

(a) Write-Read Conflict:

Transaction T1 reads object X, Transaction T2 reads objects X and Y, Transaction T1 writes object X, and finally Transaction T2 reads object X.

Schedule:

```
T1: Read(X)
T2: Read(X)
T2: Read(Y)
T1: Write(X)
T2: Read(X)
```

In this schedule, Transaction T1 writes to object X after Transaction T2 has read object X but before Transaction T2 reads it again. This constitutes a write-read conflict.

(b) Read-Write Conflict:

Transaction T1 reads objects X and Y, Transaction T2 reads objects X and Y, Transaction T1 writes object X, and finally Transaction T2 writes object Y.

Schedule:

```
T1: Read(X)
T2: Read(X)
```

T1: Write(X)

T2: Write(Y)

In this schedule, Transaction T2 writes to object Y after Transaction T1 has read object Y but before Transaction T1 writes to it. This constitutes a read-write conflict.

(c) Write-Write Conflict:

Transaction T1 reads objects X and Y, Transaction T2 reads objects X and Y, Transaction T1 writes object X, and finally Transaction T2 writes object X.

Schedule:

Sure, let's create schedules for each scenario:

(a) Write-Read Conflict:

Transaction T1 reads object X, Transaction T2 reads objects X and Y, Transaction T1 writes object X, and finally Transaction T2 reads object X.

Schedule:

```
css
Copy code
T1: Read(X)
T2: Read(X)
T2: Read(Y)
T1: Write(X)
T2: Read(X)
```

In this schedule, Transaction T1 writes to object X after Transaction T2 has read object X but before Transaction T2 reads it again. This constitutes a write-read conflict.

(b) Read-Write Conflict:

Transaction T1 reads objects X and Y, Transaction T2 reads objects X and Y, Transaction T1 writes object X, and finally Transaction T2 writes object Y.

Schedule:

```
css
Copy code
T1: Read(X)
T2: Read(X)
T2: Read(Y)
T1: Write(X)
T2: Write(Y)
```

In this schedule, Transaction T2 writes to object Y after Transaction T1 has read object Y but before Transaction T1 writes to it. This constitutes a read-write conflict.

(c) Write-Write Conflict:

Transaction T1 reads objects X and Y, Transaction T2 reads objects X and Y, Transaction T1 writes object X, and finally Transaction T2 writes object X.

Schedule:

```
css
Copy code
T1: Read(X)
T2: Read(X)
T2: Read(Y)
T1: Write(X)
T2: Write(X)
```

In this schedule, Transaction T2 writes to object X after Transaction T1 has written to it. This constitutes a write-write conflict.