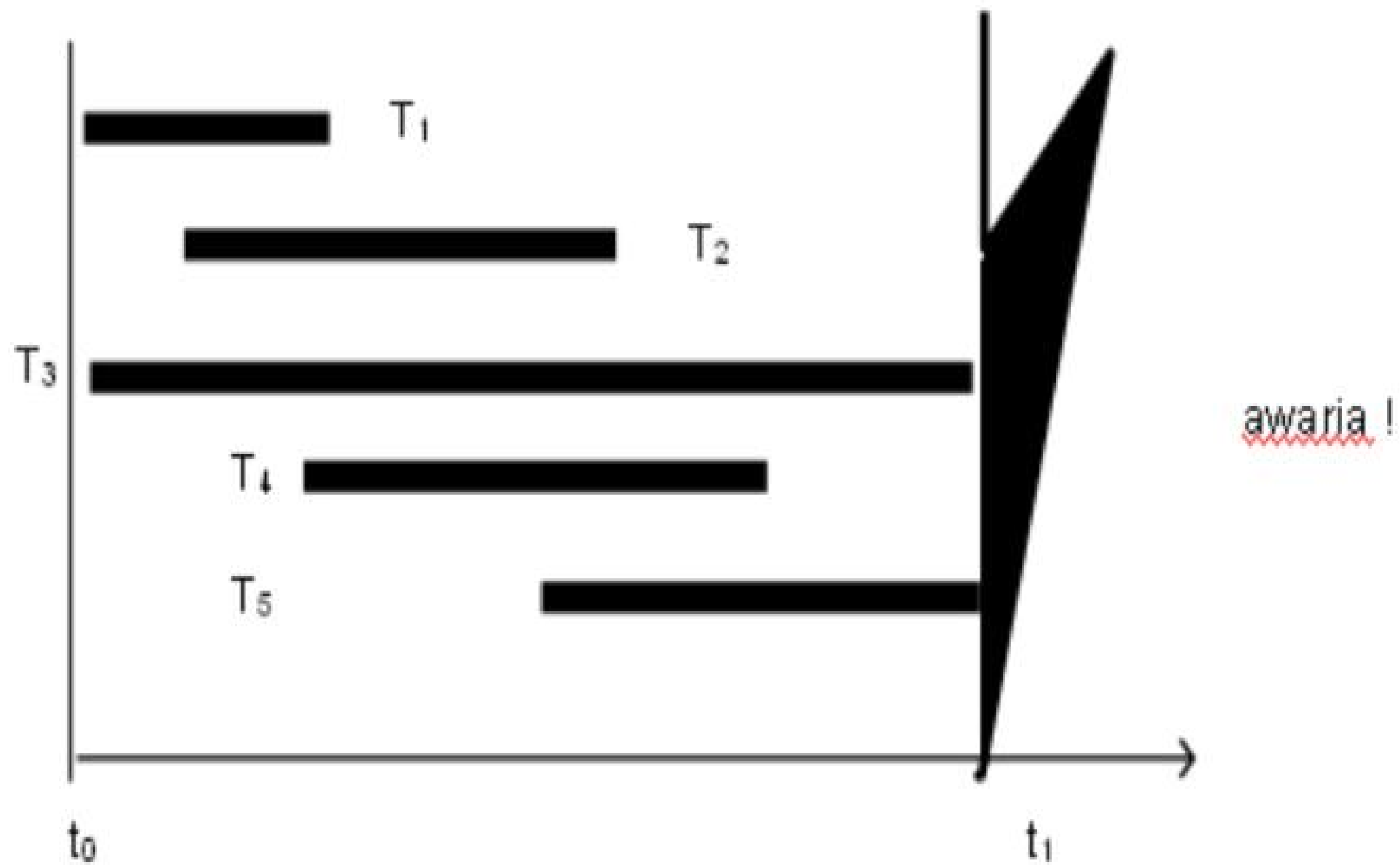


Wykład 9

- **Niezawodność bazy danych**
- **Odtwarzanie spójnego stanu bazy**
- **Odtwarzanie stanu bazy na podstawie dziennika transakcji**
- **Odtwarzanie nośników**

Bardzo ważną funkcją systemu zarządzania bazą danych jest zapewnienie jej spójności po awariach sprzętowych i programowych. Nie wszystkie transakcje kończą się w sposób normalny. Niektóre transakcje są wycofane na skutek decyzji inicjującego je użytkownika bazy danych, albo systemu zarządzania bazą danych (np. na skutek konfliktu dostępu do tych samych danych przez wiele transakcji). Po wycofaniu transakcje są zwykle inicjowane na nowo. Szczególną grupę stanowią transakcje, których wykonywanie zostało przerwane awarią systemu komputerowego. One również powinny zostać wycofane w sposób automatyczny przez system zarządzania bazą danych i ewentualnie zainicjowane ponownie po usunięciu awarii i odtworzeniu spójnego stanu bazy danych.

Rozważmy przykładowy przebieg realizacji pięciu transakcji (rys.8.1). W przedziale czasu od chwili t_0 do momentu awarii w chwili t_1 transakcje T_1, T_2, T_4 zostały zakończone osiągając swoje punkty akceptacji. Oznacza to, że wprowadzone przez nie zmiany do bazy danych są trwałe. Jeśli z określonych powodów, np. awarii urządzenia zewnętrznego, zmiany te zostały całkowicie lub częściowo utracone - transakcje T_1, T_2 i T_4 muszą być odtworzone (ang. *recovered*).



Rys.8.1 Przykład realizacji kilku transakcji

Transakcje *T3* i *T5* nie zostały zakończone przed nastąpieniem awarii, nie osiągnęły swoich punktów akceptacji. Wprowadzone przez nie zmiany do bazy danych są niekompletne i muszą zostać z niej wycofane w całości (ang. *roll - back*).

Mechanizmy odtwarzania spójnego stanu bazy sprzed awarii muszą uwzględniać:

1. Różne typy pamięci, w których przechowuje się dane:

- pamięć operacyjną,
- pamięć zewnętrzną o dostępie bezpośrednim (np. pamięć dyskową),
- pamięć zewnętrzną do archiwizacji (np. pamięć taśmową).

2. Różne typy awarii: awarie, po których można odtworzyć stan poprzedni i awarie powodujące nieodwracalne zmiany:

- awaria komputera,**
- awaria urządzenia zewnętrznego,**
- awaria programu użytkowego.**

3. Poszczególne poziomy zabezpieczeń awaryjnych, np. poziom fizyczny - pełne strony danych przechowywane w pamięciach dyskowych, poziom logiczny - stany bazy i zmieniające je transakcje.

4. Metody przywracania stanu bazy danych po awarii:

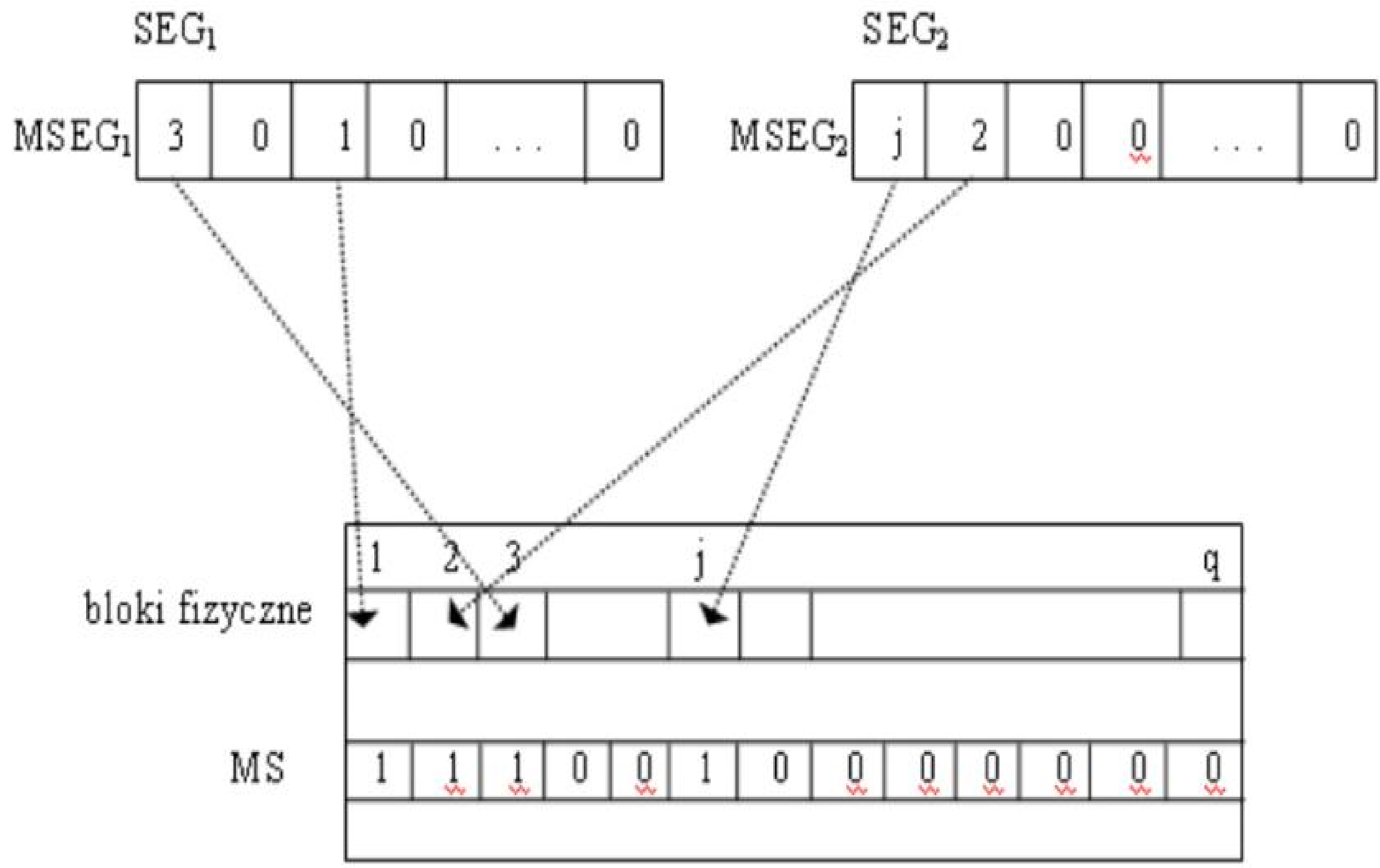
- obrazy przed i po transakcji,**
- kronikowanie wprowadzonych modyfikacji, tworzenie kopii zapasowych przechowywanych w pamięciach dyskowych lub archiwizacja (ang. *back - up*) na taśmie magnetycznej,**
- aktualizacja odroczone,**
- okresowe składowanie bazy danych.**

W systemach zarządzania bazami danych istnieją dwa rodzaje mechanizmów ochrony danych przed skutkami awarii. Pierwszy rodzaj to taki, który nie uwzględnia pojęcia transakcji. Inaczej - są to sprzętowe mechanizmy zapewnienia niezawodności bazy.

Baza danych jest przechowywana na dysku w tzw. *segmentach* SEGn, będących obszarami wirtualnymi. Liczba takich obszarów jest określana podczas tworzenia bazy danych. Ich rozmiar zmienia się dynamicznie w czasie użytkowania bazy. Każdy z segmentów składa się ze *stron* Si, które są jednostkami przydziału pamięci dyskowej oraz jednocześnie jednostkami transmisji informacji do/z jednostki centralnej. Strony są ponumerowane od 1 do q. Zbiór stron przydzielonych do segmentu Si przedstawia *mapa segmentu* MSEG_i.

Mapa i-tego segmentu zawiera fizyczne numery stron. Na dysku jest również przechowywana *mapa stron* MS składająca się z q elementów (po jednym elemencie dla każdej strony dysku. Mapa stron zawiera informację o tym, czy dana strona jest wolna (0) czy przydzielona (1) do jakiegoś segmentu.

Rys.8.2 przedstawia dwa przykładowe segmenty SEG1 oraz SEG2. Segment SEG1 jeden składa się ze stron 1 i 3, a segment SEG2 - ze stron o numerach j oraz 2. Fakt przydzielenia stron 1, 2, 3 oraz j jest odnotowany w mapie stron MS przez wpisanie jedynek do odpowiednich jej elementów.



Rys.8.2. Przykładowy przydział bloków na dysku

Mapy segmentów, tak jak inne dane, są pamiętane na stronach, które w miarę potrzeby są wprowadzane do pamięci operacyjnej. W pamięci operacyjnej można wyróżnić dwa szczególne bufory:

- bufor stron map (BSM),**
- bufor stron danych (SD).**

W celu sprowadzenia do pamięci operacyjnej strony i segmentu SEGk należy wykonać następujące operacje:

- sprowadzić do bufora BSM stronę mapy segmentu MSEGk, która zawiera i-ty element (jeżeli ta strona nie została wcześniej sprowadzona do BSM),**
- określić adres j i-tej strony segmentu SEGk, sprowadzić do bufora BSD stronę o numerze j.**

Przetwarzanie danych jest realizowane przy następujących założeniach:

zanim przystąpi się do przetwarzania danych z segmentu, segment należy otworzyć,

segment zostaje zamknięty, jego zawartość jest zapisywana w pamięci dyskowej, jakiegokolwiek przetwarzanie danych zawartych w tym segmencie jest niemożliwe aż do chwili jego ponownego otwarcia,

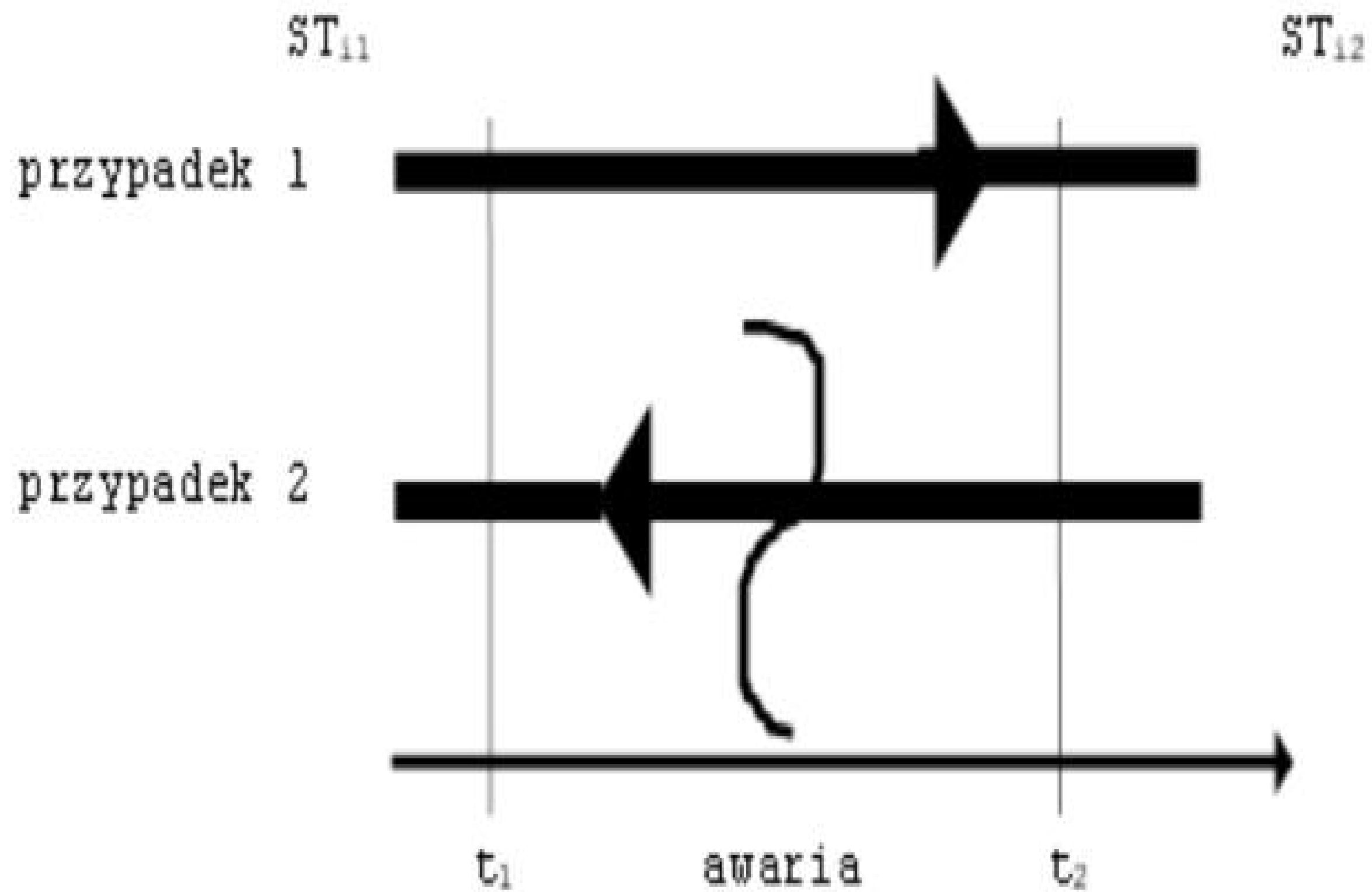
dana strona może być przeczytana, w tym celu jest ona sprowadzana do pamięci operacyjnej, a następnie odpowiednie dane są czytane lub modyfikowane.

Zarządzanie buforami BSM i BSD może być realizowane różnymi metodami. Jeżeli do pamięci operacyjnej sprowadzana jest nowa strona, to zwykle należy z niej usunąć inną stronę i ewentualnie zaktualizować ją na dysku (jeżeli ustawiony jest odpowiedni wskaźnik zajętości). Stroną usuwaną może być np. strona, do której zrealizowano najmniej dostępow lub strona, która była wykorzystywana najdawniej (algorytm LRU -Least Recently Used).

Jak zatem działa przedstawiony wyżej mechanizm sprowadzania stron określonych segmentów w przypadku wystąpienia awarii ?

Jeżeli do chwili awarii mapy segmentów nie znalazły się w całości na dysku, powstaje niespójność między informacją zawartą w tych mapach i mapą stron MS. Podobnie, jeżeli niektóre strony danych z bufora BSD o ustawionym wskaźniku zmian nie zostały uaktualnione na dysku - zostaną utracone zmiany wprowadzone przez użytkowników bazy danych. Muszą zatem istnieć odpowiednie mechanizmy umożliwiające odtworzenie stanu bazy sprzed awarii.

Założmy, że segment $SEGi$ w chwili czasu t_1 był w stanie STi_1 , a w chwili $t_2 > t_1$ w stanie STi_2 (innym niż STi_1). Jeżeli w czasie między t_1 i t_2 nie nastąpiła awaria (przypadek 1 na rys.8.3) - stan STi_1 staje się nieistotny i zostaje zastąpiony stanem STi_2 . Jeżeli wystąpiła awaria, rozważany segment $SEGi$ musi wrócić do stanu STi_1



Rys.8.3. Przejście między stanami segmentu SEGi

Wprowadzone poprzednio trzy operacje na segmentach rozszerzamy o dwie nowe:

- ***uaktualnianie*** stanu segmentu, polegające na uznaniu

 - nowego stanu segmentu,

- ***odtworzenie*** stanu segmentu, polegające na powrocie

 - do stanu poprzedniego.

Zastąpienie jednego stanu segmentu drugim stanem jest realizowane ***atomowo*** tzn. jest wykonywane w całości lub wcale.

Każdemu segmentowi SEGi odpowiadają dwie mapy segmentów MSEGi1 i MSEGi2 odpowiednio z chwil czasu t1 i t2. Podobnie istnieją dwie mapy stron MS1 oraz MS2. Wprowadzamy ponadto wektor STAN o liczbie elementów zgodnej z liczbą segmentów. W poszczególnych składowych tego wektora kodowany jest aktualny stan odpowiedniego segmentu, np. otwarty, zamknięty. Wektor STAN jest zapisywany dysk bezpośrednio po każdym jego uaktualnieniu.

Przebieg uaktualniania segmentów czyli ich przechodzenia do nowych stanów jest następujący:

- otwarcie segmentu i ustalenie nowej wartości odpowiadającego mu elementu wektora STAN oraz przepisanie jego mapy stron MSEG_{i1} do MSEG_{i2},**
- jeżeli zależy nam na modyfikacji strony j segmentu i , to należy ją najpierw sprowadzić do pamięci operacyjnej na podstawie informacji zawartej w mapie segmentu; niech adres tej strony będzie równy k ,**
- po dokonaniu zmian na stronie jej nowa wersja k' jest zapisywana na dysku, a stara wersja jest nadal przechowywana; pierwsza z dwóch map segmentu wskazuje na starą wersję strony, a druga - na nową.**

Poszukując nowej strony korzystamy z mapy stron MS2 odpowiednio ją uaktualniając. Mapa MS1 nie ulega zmianie.

Opisana wyżej struktura danych i procedury uaktualniania umożliwiają zarówno uaktualnienie stanu segmentu jak i powrót do stanu poprzedniego, w przypadku wystąpienia większości typowych awarii systemu.

Drugi rodzaj mechanizmów zabezpieczających bazę danych przed skutkami awarii uwzględnia fakt współbieżnego wykonywania transakcji. Jeśli kilka transakcji aktualizowało pewną relację z określonego segmentu, to powrót do poprzedniego stanu wymaga wycofania wszystkich zmian wprowadzonych przez te transakcje, a przejście do nowego stanu - z ich potwierdzeniem.

Problem ten rozwiązuje się zwykle za pomocą tzw. ***dziennika transakcji*** (ang. transaction log file), w którym odnotowuje się wszystkie operacje zrealizowane przez współbieżnie wykonywane transakcje.

Dziennik zawiera zwykle informacje o następujących zdarzeniach:

- początek nowej transakcji,
- koniec transakcji - potwierdzenie aktualizacji,
- anulowanie transakcji.

W przypadku każdej aktualizacji zapamiętuje się:

- identyfikator transakcji, która dokonuje aktualizacji,
- identyfikator modyfikowanej krotki,
- stare wartości krotki,
- nowe wartości krotki.

Aby uniknąć utraty danych z bazy w wyniku awarii pamięci dyskowej, dokonuje się co pewien czas przepisania bazy danych na inny nośnik (ang. back copies). Informacja o tym zdarzeniu jest również wpisywana do dziennika. Każda z transakcji osiąga punkt potwierdzenia lub jest anulowana. Aby nie utracić w wyniku awarii zmian dokonanych przez transakcję, która osiągnęła punkt akceptacji, stosuje się następujące zabezpieczenia: dziennik transakcji przepisuje się na dysk zanim zniszczy się stary stan bazy, w dzienniku rejestruje się koniec danej transakcji, a następnie przepisuje się na dysk wszystkie informacje z dziennika, które dotyczą danej transakcji.

Odtwarzanie stanu bazy na podstawie dziennika transakcji

System musi być przygotowany do odtworzenia nie tylko z powodu czysto lokalnych błędów typu wystąpienia niedozwolonej operacji arytmetycznej w pojedynczej transakcji, ale także z powodu „globalnych” awarii, takich jak brak zasilania CPU (jednostki centralnej komputera).

Z definicji lokalny błąd dotyka jedynie tę transakcję, w której błąd ów wystąpił.

Awaria globalna dotyka wszystkie transakcje dokonywane w chwili wystąpienia, ma zatem znaczące skutki dla całego systemu.

Awarie można podzielić na dwie kategorie:

- ***Awarie systemowe*** (np. awaria zasilania), które dotyczą aktualnie wykonywa-nych transakcji, ale nie uszkadzają fizycznej bazy danych. Awarie systemowe cza-sami nazywa się ***miękkimi awariami*** (*soft crash*).
- ***Błędy nośników*** (np. uszkodzenie dysku), które powodują uszkodzenia bazy danych lub jej części i mają wpływ na wyniki przynajmniej tych transakcji, które z tej części ko-rzystają. Takie awarie nazywane są czasami ***twardymi awariami*** (*hard crash*).

Zasadniczą sprawą w przypadku awarii systemu jest *utrata zawartości pamięci operacyjnej* (w szczególności tracone są bufory bazy danych). Oznacza to, że nie jest znany dokładny stan żadnej trwającej w tym czasie transakcji. Transakcja taka nie może zatem zakończyć się pomyślnie i trzeba ją *robić od początku* po restarcie systemu. Co więcej, może okazać się konieczne powtórzenie przy restarcie pewnych transakcji, które zakończyły się pomyślnie, zanim wystąpiła awaria, jednak ich aktualizacje nie zostały w porę przeniesione z buforów bazy danych do pamięci fizycznej.

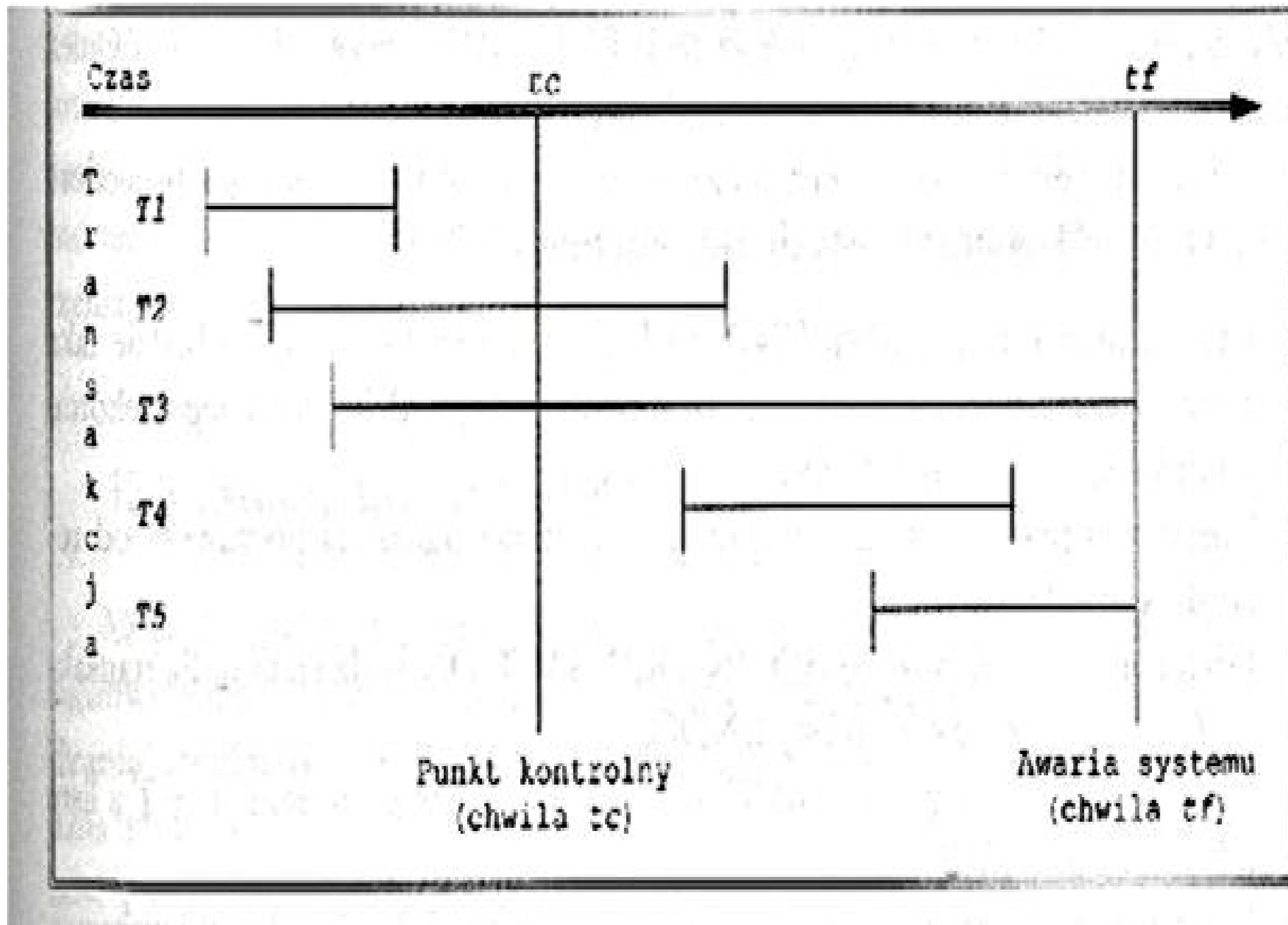
Skąd system "wie" w chwili restartu, które operacje należy wykonać ponownie, a których skutki cofnąć?

Co pewien określony czas - zwykle co ileś zadanych z *góry* encji, zapisanych do dziennika transakcji — system automatycznie ustala tzw. ***punkt kontrolny*** (ang. *check point*).

Ustalenie punktu kontrolnego obejmuje :

- a) zapisanie zawartości buforów bazy danych do bazy na dysku,
- b) zapisanie specjalnego rekordu kontrolnego do fizycznego dziennika transakcji w pamięci dyskowej.

Rekord kontrolny zawiera spis wszystkich transakcji będących w toku w chwili ustalenia punktu kontrolnego.



Rys.8.4 Przykład kilku transakcji, punktu kontrolnego i momentu wystąpienia awarii

Przykład – rysunek:

- W czasie t_f nastąpiła awaria systemu.
- Najbardziej aktualny punkt kontrolny przed chwilą t_f został zapisany w chwili t_c .
- Transakcja $T1$ zakończyła się (pomyślnie) przed chwilą t_c .
- Transakcja $T2$ rozpoczęła się przed chwilą t_c i zakończyła się (pomyślnie) po chwili t_c , ale przed chwilą t_f .
- Transakcja $T3$ także rozpoczęła się przed chwilą t_c , ale nie zakończyła się przed chwilą t_f ,
- Transakcja $T4$ rozpoczęła się po chwili t_c , zakończyła się (pomyślnie) przed chwilą t_f .
- Transakcja $T5$ także rozpoczęła się po chwili t_c , ale nie zakończyła się przed chwilą t_f .

Kiedy system restartuje, należy cofnąć skutki wykonywania transakcji $T3$ i $T5$, zaś transakcje $T2$ i $T4$ należy wykonać ponownie. Zwróćmy uwagę na to, że transakcja $T1$ w ogóle nie bierze udziału w procesie restartu, ponieważ jej aktualizacje zostały fizycznie dokonane w chwili t_c jako część procesu tworzenia punktu kontrolnego.

Transakcje, które zakończyły się niepomyślnie (tj. z instrukcją ROLLBACK) przed chwilą t_f , również wcale nie wchodzą do procesu restartu.

W chwili restartu system zatem najpierw musi wykonać następującą procedurę w celu zidentyfikowania wszystkich transakcji *T1-T5*:

- 1. Utwórz dwie listy transakcji UNDO i REDO. Niech lista UNDO będzie taka sama jak lista wszystkich transakcji uzyskanych z najbardziej aktualnego rekordu kontrolnego. Niech lista REDO będzie pusta.**
- 2. Rozpocznij przeszukiwanie (do przodu) dziennika transakcji począwszy od rekordu kontrolnego.**
- 3. Jeżeli napotkasz na pozycję, od której rozpoczyna się transakcja (*BEGIN TRANSACTION*) w dzienniku dla transakcji T, dodaj transakcję T do listy UNDO.**
- 4. Jeżeli znajdziesz pozycję COMMIT dla transakcji T, przesuń transakcję T z listy UNDO na listę REDO.**
- 5. Kiedy napotkasz na znak końca dziennika, wtedy listy UNDO i REDO wskazują odpowiednio transakcje typu *T3* i *T5* oraz transakcje typu *T2* i *T4*.**

Następnie system ponownie przegląda dziennik, ale tym razem wstecz, cofając skutki transakcji z listy UNDO. Potem posuwa się znów do przodu i ponownie wykonuje transakcje z listy REDO.

Odtwarzanie bazy danych do właściwego stanu przez cofanie skutków rozpoczętych transakcji nazywa się niekiedy *odtworzeniem wstecznym (backward recovery)*. Podobnie, proces odtwarzania poprawnego stanu przez ponowne wykonywanie pracy czasem zwie się *odtworzeniem w przód (forward recovery)*.

Wreszcie, kiedy zostaną zakończone wszystkie te czynności związane z odtwarzaniem (i tylko wtedy), system jest gotowy do zaakceptowania następnej pracy.

Odtwarzanie nośników

Zagadnienie odtwarzania nośników jest nieco innego rodzaju niż problemy związane z transakcjami czy odtwarzaniem systemu.

Błąd danych jest to błąd spowodowany awarią dysku lub awarią kontrolera dysku w wyniku czego część bazy danych ulega fizycznemu uszkodzeniu. Odzyskiwanie danych po takiej awarii polega na ponownym załadowaniu (lub *odtworzeniu*) bazy z zapasowej kopii (*dump*), a następnie na wykorzystaniu dziennika transakcji — zarówno części aktywnej, jak i archiwum — do ponownego wykonania wszystkich operacji, które zakończyły się od czasu wykonania tej kopii. Nie ma potrzeby cofania transakcji będących w toku w czasie powstania awarii, ponieważ z definicji wszystkie aktualizacje tych transakcji zostały i tak „wycofane” (w rzeczywistości utracone).

Ponieważ potrzebna jest możliwość odzyskiwania nośników, to potrzebny jest program narzędziowy do robienia *kopii/odtworzenia* (*unload/reload*). Część związaną z zapisem stosuje się do robienia na żądanie kopii bazy danych (kopie te mogą być przechowywane na taśmie bądź innej pamięci archiwalnej, niekoniecznie na nośnikach z dostępem bezpośrednim). W razie wystąpienia błędu nośnika część programu narzędziowego związana z odtwarzaniem jest stosowana do ponownego tworzenia bazy danych z określonej kopii zapasowej.