- To increase concurrency
  - Commit soon
- To protect against failures
  - Commit as late as possible

Analogy

Real life actions | system transactions

Goal is to:

- Maximize commitment of actions/transactions
- Minimize rollback
- Maximize concurrency
- Minimize blocking

- Obstacles
  - Transmission delays
  - Communication failure (network partition)
  - Site failures
  - Long and short transactions
  - Nested transactions

Possible solution:

- Assign a degree of {commitment, importance, success} to a transaction

Many ideas were originally mentioned by:

C.T. Davies	-	ACM Conf., 1972
L.A. Bjor	-	ACM Conf., 1972
S.K. Srivastava	-	Symp. of Reliability in Dist. Soft/OB, 1982

Degree of commitment (for a single transaction in the system)

= 0 when transaction arrives in the system

= 1 when

(a) transaction has left the system permanently

(b) transaction can no longer be rolled back

- resources necessary to back out are lost
- some undoable action has been performed

(when other transactions are involved)

= 1 (a) when transaction has given results to other transactions who have a degree of comm = 1

(b) dependency information among transactions has been lost

Dependency Graph

GD = (V,E)

V: set of nodes representing transactions (T)

E: set of edges representing the dependency relation among T

Types of dependency relations:

(a) Concurrency control dependency

$T_i \rightarrow T_j$ if R-W or W-W conflict

T_i should commit before T_j

(b) User defined dependencies

(i) Ti → Tj if Ti should commit before Tj

(ii) Ti ↔ Tj if Tj should commit simultaneously

(iii) Ti ↔ Tj if either T_i or T_j should commit, but not both

Sphere of Dependency

Ancestor: $A(T)$

Successor: $S(T)$
(supporter)

Competitor: $C(T)$

$$A(T) = \{T' \mid T' \xrightarrow{*} T \text{ in } G_D\}$$

$$S(T) = \{T'' \mid T \xrightarrow{*} T'' \text{ in } G_D\}$$

$$C(T) = \{T''' \mid T \overset{*}{\leftrightarrow} T''' \text{ in } G_D\}$$

Degree of commitment: $D(T_i)$

$$D(T_i) > D(T_j) \text{ if } W(S(T_i) > W(S(T_j))$$

weight function

If T is aborted

then S(T) is aborted

Loss = W(S(T))

$S_{cycle}(T_1) = \{T' / T' \in S(T_1)$ and

$T' \in A(T_1)\}$

If $W(S_{cycle}(T_1)) > W(T_1)$

then abort T_1