SYSTEM-R について

鷹尾 洋一
（日本アイ・ビー・エム（株） サイエンティフィック・センター）

1. はじめに

理論的モデルとして E. F. Codd により提案された関係形式データ・ベース（Relational Database）は、多くの反響を呼び、その後、理論的検討はもとより実動化の試みも多くなされてきている。現在すでに、いくつかの大学や研究所では、中小規模の実験システムを開発し、その試用段階に入っている。また、最近になって、データ・ベース管理システムとしての種々の要件を考慮に入れないわば大規模実験システムを開発しようとする動きが起こっており、本稿で紹介する System－R もその中の1つである。

System－R は、IBM San Jose 研究所において研究開発中の関係形式主体のデータ・ベース管理システムである。研究実験を目的としたプロトタイプ・システムであるが、実用的な環境の下で実用的な規模での性能検討をめざしており、その設計目標としては下記の項目が挙げられる。

- 高級データ言語
  データ言語として非手順的な SEQUEL を提供するが、これは独立言語方式だけでなく、親言語方式でも使用できる。この事は、システムによる自動的な検索最適化とあいまって、プログラムの開発保守を容易にする。
- データ独立
  物理構造・スキーマ・サブスキーマの3水準でのデータ定義を可能にし、それぞれをできるだけ独立にする。
- データ制御
  機密保護・保全管理・同時使用の制御、それに回復管理の機能を提

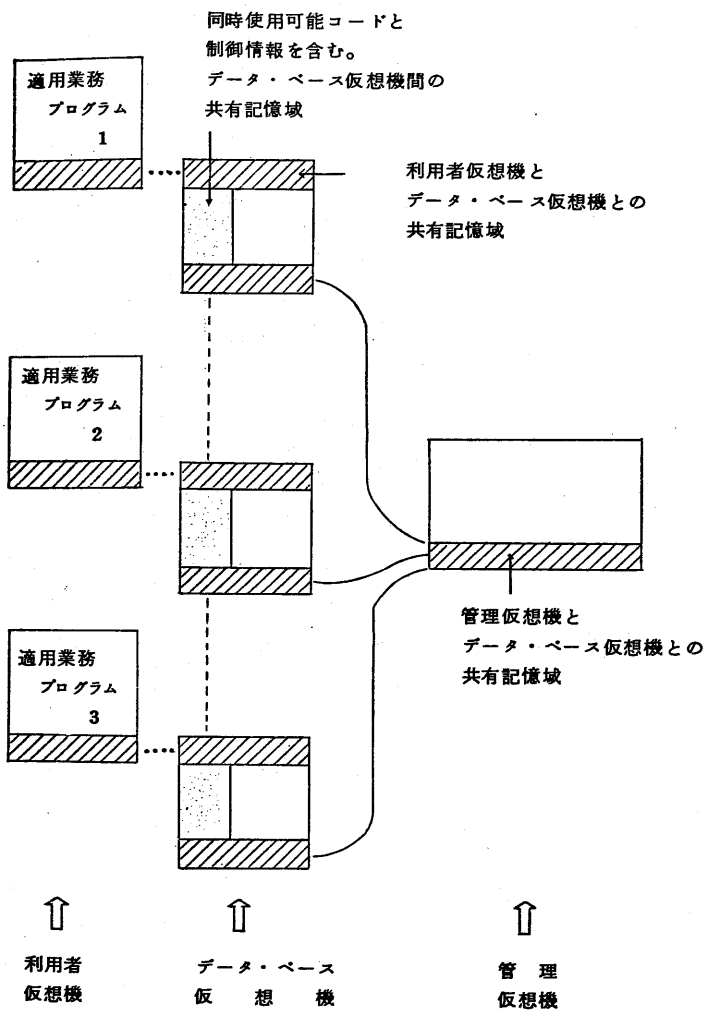

図1 System－Rの構成

供する。

○階層形式・網目形式の提供

関係形式を基礎として、階層形式（hierarchical model）および網目形式（network model）のシミュレーターを提供する。

2. システム構成

System－Rの構成は前頁図1の様になっており、これは仮想計算機（virtual machine）の機能を提供するオペレーティング・システムであるIBM VM/370の下で実働化されている。適用業務プログラムを実行する利用者仮想機には、それぞれ、専用のデータ・ベース仮想機が用意される。これは、セッション開始の要求に応じて管理仮想機が創設するものである。利用者仮想機とデータ・ベース仮想機との間の通信は共有記憶域を通してなされる。また、すべてのデータ・ベース仮想機は、1つの同時使用可能コードと同期の為の制御情報が入った1つの記憶域を共有している。利用者毎にデータ・ベース仮想機を設ける事の利点としては、

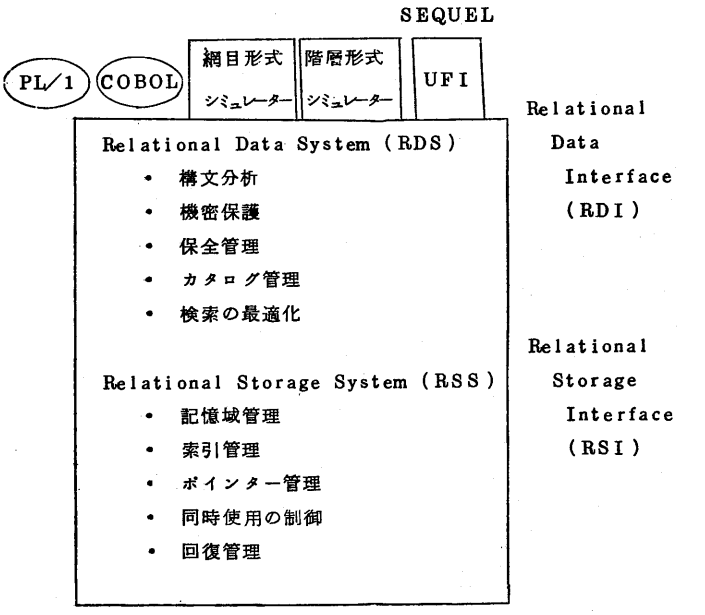

図2　System－Rの機能構成

○独自の並列処理が不要になり、論理構造が単純になる。

○仮想記憶におけるページ不在の影響が最小限になる。

○多重プロセッサを有効に利用できる。

等が挙げられる。なお、管理仮想機は、すべての利用者のセッション開始・終了を管理すると同時に、チェックポイントの管理や会計あるいは効率評価のための統計情報の収集管理も行なう。

個々の利用者から見たSystem－Rの機能構成は図2の様になっている。RSIは、n-tuple単位の操作を行なう為の低水準インターフェイスである。これを提供するRSSは、主に物理的資源の管理、すなわち、緩衝記憶域、補助記憶域、索引、ポインター連鎖、障害時の回復等の管理をするが、この他、同時使用制御の為のロックの管理やデッドロックの検出回避も行なう。

一方、RDIは適用業務プログラムの為の高水準インターフェイスであり、データ言語としてSEQUELを提供する。RDSは、RDIを介して利用者にデータ操作、データ定義、データ制御の機能を提供する他、カタログの保守管理やデータ・ベース検索の最適化も行なう。RDIは、PL/Iなどのプログラムから呼び出される親言語方式のインターフェイスだが、その他、SEQUELを独立言語として端末利用者に提供するUFI（User Friendly Interface）も用意されている。

本稿では、以下、RDIを通して利用者に提供されるSystem－Rの機能、すなわちSEQUELの機能を中心に述べてゆく。

3. データ照会

SEQUELの照会機能について、次のデータベース例を用いて説明してゆく。

EMP (ENO, ENAME, DNO, JOB, SAL, MGR)
DEPT (DNO, DNAME, LOC, NEMPS)

関係形式 EMP は、すべての社員について社員番号 (ENO)、名前 (ENAME)、所属部門コード (DNO)、職種 (JOB)、給料 (SAL)、上司の社員番号 (MGR) を与える。また、DEPT は、各部門の部門コード (DNO)、部門名 (DNAME)、所在地 (LOC)、所属社員数 (NEMPS) を与える。

まず、最も簡単な照会例を示す。

Q1. 「NEWBERG にある部門のコードと名前は？」

```
SELECT   DNO, DNAME
FROM     DEPT
WHERE    LOC = 'NEWBERG'
```

一般に、SELECT で取り出すべき属性 (フィールド) を、FROM で検索すべき関係形式を、WHERE で検索条件を指定する。検索条件は、AND, OR で組み合わせる事ができ、またすべての属性値が欲しい時には SELECT の後に * を書けば良い。

Q2. 「部門 19 に属し、給料が 20,000 より多い社員に関する全データは？」

```
SELECT   *
FROM     EMP
WHERE    DNO = 19   AND   SAL > 20000
```

SELECT, FROM, WHERE からなる照会ブロックは入り子にもできる。

Q3. 「EVANSTON にある部門に所属する社員の名前は？」

```
SELECT   ENAME
FROM     EMP
WHERE    DNO  IS IN   SELECT DNO
                      FROM   DEPT
                      WHERE  LOC = 'EVANSTON'
```

結果を特定の属性値について昇順 (ASC) あるいは降順 (DESC) に並べるには、ORDER BY でその属性を指定すれば良い。

Q4. 「部門 19 に属する社員のデータを、給料の昇順に示せ。」

```
SELECT   *
FROM     EMP
WHERE    DNO = 19
ORDER BY   SAL  ASC
```

組変数 (tuple variable) の利用も可能で、次の場合これが不可欠となる。

Q5. 「給料が上司の給料より高い社員の名前は？」

```
SELECT   X.ENAME
FROM     EMP X, EMP Y
WHERE    X.MGR = Y.ENO    AND    X.SAL > Y.SAL
```

集合演算としては、積（INTERSECT）、和（UNION）、差（MINUS）が利用できる。

Q6.「所属社員のいない部門のコードは？」

```
SELECT  DNO
FROM    DEPT
MINUS
SELECT  DNO
FROM    EMP
```

検索結果を、ある属性値に基づいてグループ分けする事もできる。

Q7.「各部門のコードと平均給料は？」

```
SELECT  DNO, AVG(SAL)
FROM    EMP
GROUP BY  DNO
```

集合関数（aggregate function）としては、平均（AVG）の外に、合計（SUM）、最大（MAX）、最小（MIN）、それにn-tuple組の数を与えるもの（COUNT）がある。グループ分けをした時、グループに対する条件はHAVINGによって指定する。

Q8.「所属社員数が50より多の部門のコードは？」

```
SELECT  DNO
FROM    EMP
GROUP BY  DNO
HAVING    COUNT(*) > 50
```

集合としての比較は、＝, ¬＝の外に、IS IN, IS NOT IN, CONTAINS, DOES NOT CONTAIN なる演算子で行なう事ができる。

Q9.「すべての職種の社員がいる部門のコードは？」

```
SELECT  DNO
FROM    EMP
GROUP BY DNO
HAVING SET(JOB)   CONTAINS   SELECT JOB
                             FROM   EMP
```

4. データ操作

SEQUELは親言語方式で使う事ができるが、このとき、SEQUEL文を直接PL/Iプログラムの中に書く事を可能にしている。これらは、プリプロセッサーによって通常のCALL文に変換される。適用業務プログラムとRDSの間のデータのやりとりは、カーソル（cursor）によって制御される。カーソルは、処理対象となるn-tuple組の集合（active set）を規定すると同時に、その中での現在のn-tuple組の位置（current position）をも表わす。n-tuple組の集合はSEQUEL照会文によって表わされ、例えば、カーソルC1の定義は次の様に行なう事ができる。

```
LET  C1  BE  SELECT  ENAME  INTO X
             FROM    EMP
             WHERE   JOB = Y ;
```

ここに、X,YはPL/I変数名であり、したがって処理対象は変数Yの値に基づいて限定される事になる。このカーソルC1を用いた処理は次の様になる。

```
OPEN C1;
DO WHILE (CODE=OK);
   FETCH C1;
   /* process as required */
 END;
CLOSE C1;
```

3行目のFETCH C1により、検索結果が1つづつ変数Xに入れられる。
なお、PL/I変数名がデータ・ベース内の属性名と一致した場合に曖昧さが生じ得るが、このような時、System-Rでは、PL/I変数名と解釈する事にしている。
前の例では、照会文を直接プログラム中に書いていたが、照会文をPL/I文字列変数に代入して、それに基づいてカーソルを定義する事もできる。例えば、照会文が文字列変数QSTRINGに代入されるとすれば、それに基づいたカーソルC2は、

```
LET C2 BE EXECUTE QSTRING;
```

によって定義される。ただし、この様な場合には、取り出すnケ組の次数とデータ・タイプが未知であり、次の様にして、それを知る必要がある。

```
DESCRIBE C2 INTO PTR1;
```

このとき、PTR1は、システムが用意する属性の記述ブロックへのポインターとなる。したがって、ここで与えられる属性およびそのデータ・タイプの記述に基づいて入力域を用意して、変数PTR2にその入力域へのポインターを代入しておけば、

```
FETCH C2 INTO PTR2;
```

によって、以後、nケ組を次々に読み込む事ができる。
なお、カーソルとデータ・ベースとの結合は、OPENによって行なわれ、CLOSEによって解除される為、同一のカーソルを次々とデータベースの異なる部分集合と結びつけて使用する事が可能である。

データの更新、削除、追加も、検索と同様にSEQUEL文で行なうが、この時、複数のnケ組を同時に処理する方式と、nケ組を1個づつ処理する方式とがある。

更新はUPDATE文で行ない、WHEREで更新するnケ組が満たすべき条件を指定し、SETで更新内容を指定する。2つの方式による更新の例を、U1, U2で示す。

U1. 「部門50の社員の給料を一率に10%高くせよ。」

```
UPDATE  EMP
SET     SAL = SAL X 1.1
WHERE   DNO = 50
```

U2. 「カーソルC3が指すnケ組のSALを変数NSALの値で置き換えよ。」

```
UPDATE  EMP
SET     SAL = NSAL
WHERE   CURRENT TUPLE OF C3
```

n-tuple の削除は DELETE 文で行ない WHERE で削除する n-tuple が満たすべき条件を指定する。削除の例を、D1, D2 で示す。

D1. 「部門 50 に属する社員に関する全データを削除せよ。」

```
DELETE  EMP
WHERE   DNO = 50
```

D2. 「カーソル C4 が指す EMP の n-tuple を削除せよ。」

```
DELETE  EMP
WHERE   CURRENT TUPLE OF C4
```

n-tuple の追加は INSERT 文で行なう。追加の例を、I1, I2 で示す。

I1. 「職種が SECRETARY である人に関するデータを EMP から取り出して、別の関係形式 EMP1 に追加せよ。」

```
INSERT  INTO EMP1 :  SELECT  *
                     FROM    EMP
                     WHERE   JOB = 'SECRETARY'
```

I2. 「ENO = 380, ENAME = 'JONES', DNO = 5, MGR = 135 なる n-tuple を EMP に追加せよ。」

```
INSERT  INTO EMP(ENO, ENAME, DNO, MGR) :
                < 380, 'JONES', 5, 135 >
```

I2 では、JOB と SAL の値を指定しておらず、これらには空値 (null) が与えられる。なお、System-R では、空値を表わす予約語 NULL が用意されており、例えば JOB = NULL なる条件で、JOB の値が空値である n-tuple の選択ができる。

SEQUEL には、その他に代入文があり、これは検索結果を新しい関係形式に複写し保存する機能を持つ。例えば、下の A1 の様に用いる。

A1. 「職種が PROGRAMMER で給料が 10,000 以下の社員の名前と給料とを、新らしい関係形式 UNDERPAID を作り、そこに入れよ。」

```
ASSIGN TO UNDERPAID :  SELECT  ENAME, SAL
                       FROM    EMP
                       WHERE   JOB = 'PROGRAMMER'
                           AND SAL < 10000
```

5. データ定義

System-R において スキーマを構成するのは、物理データに対応している基底関係形式 (base relation) であり、その定義、削除は CREATE TABLE, DROP TABLE なる SEQUEL 文で行なう。例として、EMP の定義文を示す。

```
CREATE TABLE EMP
 ( ENO    ( DECIMAL(6), NONULL),
   ENAME  ( CHAR(20) VAR),
   DNO    ( DECIMAL(5)),
   JOB    ( CHAR(10)),
   SAL    ( DECIMAL(5,2)),
   MGR    ( DECIMAL(6)) )
```

基本的には、各属性の名前とデータ・タイプを指定するが、空値を許さない属性については、特にNONULLでそれを指定する。

また、既存の基底関係形式に新らしい属性を追加して拡大する事もでき、例えば、EMPに新らしくSOCSECNOなる属性を追加するのは、次の様になる。

```
EXPAND TABLE EMP
    ADD COLUMN SOCSECNO ( DECIMAL(9))
```

この拡大操作は、データ・ベース記述部分を変更するだけで、物理的再編成等は一切必要なし。なお、既存のnケ組は、追加された属性については空値を持つものとして取り扱われ、以後の更新で実際の値が入れられる事になる。

個々の適用業務のためのスキーマは視点(view)という形で定義する。これは、基底関係形式に対する様々な見方を提供する仮想的な関係形式である。すなわち、実際に格納されるのはその定義式だけであり、参照された時点にその定義式が評価され具現化される。視点の定義例を下に示す。

```
DEFINE VIEW PEMP AS SELECT ENO, SAL, MGR
                    FROM   EMP
                    WHERE  JOB = 'PROGRAMMER'

DEFINE VIEW SDEPT ( DNO, JOB, NEMPS) AS
                    SELECT DNO, JOB, COUNT(*)
                    FROM   EMP
                    GROUP BY DNO, JOB
```

このように、一般のSEQUEL照会文が視点の定義式であり、その自由度は大きい。視点の導入は、スキーマとプログラムの分離、すなわち、論理的データ独立性を可能にする。なお、視点に対する更新は、基底関係形式に対する更新として処理されるが、このとき、視点のnケ組と基底関係形式のnケ組とが1対1に対応していなければ、更新は無効となる。

物理的データ構造の管理はすべてRSSで行なうが、索引とポインター連鎖については、その生成、削除をSEQUEL文で指定する。ただし、これらはあくまでも検索効率だけに影響するもので、利用者はその存在を一切考慮する必要がない。すなわち、その保守管理はもとより、その利用もシステムが検索の最適化という事で自動的に行なう。

System-Rの索引はimageと呼ばれ、任意の基底関係形式の任意の属性あるいは属性の組について、これを作ることができる。例えば、EMPの属性DNOについてIMG1なる名前の昇順索引を生成するには、次の指定をする。

```
CREATE IMAGE IMG1 ON EMP (DNO ASC)
```

このimageを利用すると、例えばDNO=50なる条件を満たすnケ組を直接取り出す事が可能になる。なお、imageの物理的構造は、B-treeに類似した釣合い木(balanced tree)になっている。

System-Rにおけるポインター連鎖はlinkと呼ばれ、2つの関係形式の結合(join)を効率良く行なうためのものである。すなわち、linkは共通の属性をもつ2つの基底関係形式の間で定義され、共通属性の値が一致するnケ組がポインターによって連鎖される。例えば、EMPとDEPTとを属性DNOについて、

結合する要求が多いとすれば、次の様なLNK1なるlinkを作る事になる。

```
CREATE LINK LNK1
  FROM DEPT (DNO)
  TO   EMP (DNO)
  ORDER BY ENO ASC
```

一般に、FROMで指定した関係形式のカケ組とTOで指定したそれとの間には1対1もしくは1対多の対応がなければならず、これは階層形式の親子関係あるいは網目形式のセット関係と同等である。

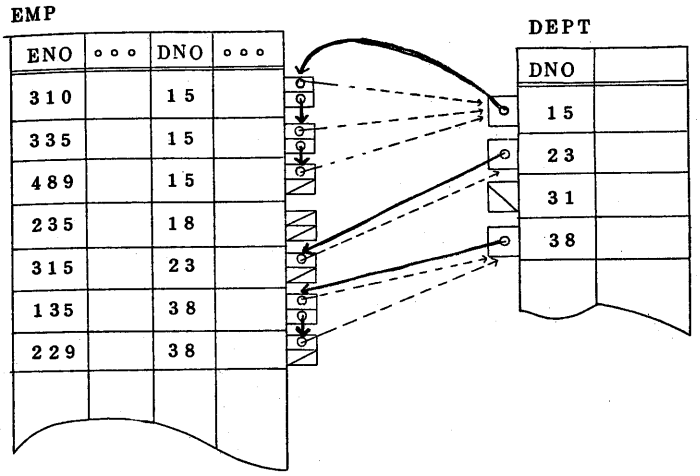

図3. linkの物理的構造

なお、上例のORDER BY ENO ASCは、DEPTの1つのカケ組から連鎖されるEMPのカケ組が、ENOの昇順に並べられる事を指定しており、物理的には、図3の様な構造となる。

6、機密保護

System-Rの機密保護は、個々の関係形式の創設者が必要に応じて他の利用者に利用認可（authorization）を与えるという方式に基づいている。利用認可および取り消しは、GRANT, REVOKEなるSEQUEL文が行なう。また、利用権限は、次の様な分類に基づいて指定する。

READ　：照会、検索
INSERT　：カケ組の追加
DELETE　：カケ組の削除
UPDATE　：カケ組の更新、ただし更新可能な属性を指定
DROP　：関係形式全体の削除
EXPAND　：関係形式の拡大
IMAGE　：imageの生成、削除
LINK　：linkの生成、削除
CONTROL：assertion, trigger（後述）の定義、削除

この他に、利用認可を受けた人が更に他の人に利用認可を与える事を可能にするGRANT OPTIONがある。なお、すべての権限を与える場合にはALL RIGHTSと指定すれば良く、また、すべての人に権限を与える場合には、PUBLICと指定すれば良い。ここで例を示そう。

```
GRANT  READ, UPDATE (DNO, JOB)  ON EMP  TO JONES
GRANT  ALL RIGHTS  ON EMP  TO MILLER  WITH GRANT OPTION
GRANT  READ  ON DEPT  TO PUBLIC
```

前節では、視点（view）がデータ独立性の観点から重要である事にふれたが、同時に、機密保護という面でも重要である。すなわち、ある関係形式の一部だけを見せたい場合には、その様な視点を定義し、これに対する利用権限を与えれば

よい。例えば、各上司に自分の部下に関するデータだけを見せる視点は、

```
DEFINE VIEW SEMP AS SELECT *
                    FROM   EMP
                    WHERE  MGR = USER
```

となる。ここに、USERは予約語で、参照時に利用者識別名で置き換えられる。

7. 同時使用の制御

System-Rにおける同時使用の制御（concurrency control）は、トランザクションを単位として行なわれる。ここに、トランザクションとは、利用者にとって意味のある最小の処理単位であり、具体的には一連のRDI callからなるプログラムである。一般に、トランザクションは、データ・ベースをある一貫した状態から他の一貫した状態へ変換するものと考えられるが、その処理途中では、一貫性を破る事があり得る。例えば、データ項目A，B間にA＝Bなる条件がある時、

```
A = A + 100 ;
B = B + 100 ;
```

なる処理をするトランザクションを考えると、最初の操作の直後ではA>Bと成っている。したがって、一般に、あるトランザクションによって変更されたデータは、それが終了するまでは矛盾を含むと考えるのが最も安全であり、System-Rでは、この考えに基づいて排他制御を自動的に行なっている。理想的には、すべてのトランザクションがあたかも単独で動いている様な制御が望ましいが、それでは処理の多重化が困難になり効率が悪くなる。そこで、System-Rでは、一貫性の水準（consistency level）を3段階用意し、個々のトランザクションは、必要に応じてこれを設定できる様にしている。ここに、3つの水準とは、

水準3：データの一貫性と再現性を要求，
水準2：データの一貫性だけを要求，
水準1：要求なし，

である。一貫性を要求すると、他のトランザクションによって変更されたデータの読み込みは、それが終って一貫性が保障されるまで待たされる。また、再現性を要求すると、一度読んだデータに対する他からの更新要求を、自分が終了するまで待たせる事になる。したがって、この両方を要求する水準3では、完全に単独で動いているのと同等である。なお、水準1では、矛盾を含むデータを見る事もあり得るが、統計データ収集等にはそれで十分であろう。今、例として、データXを2度読むトランザクションAと、Xを更新するトランザクションBを考える。図4は、このとき、Aの要求する一貫性の水

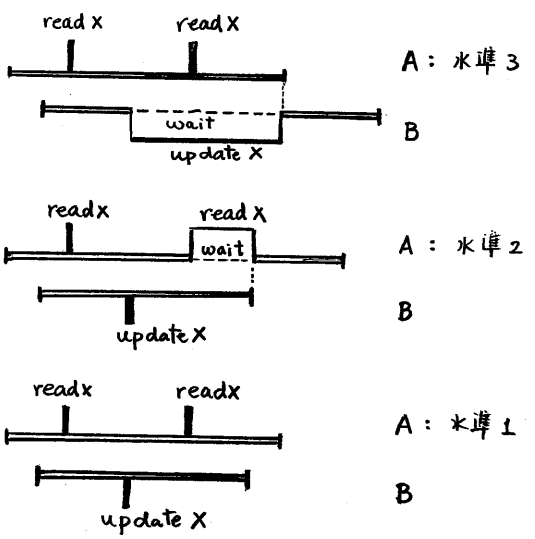

図4. 一貫性の水準と同時処理

導が、AとBの同時処理にどの様な影響を及ぼすかを示したものである。
System-Rでは BEGIN TRANSACTION, END TRANSACTION によってトランザクションの開始、終了を宣言する。その他、トランザクション制御の機能として、復帰点(save point)を設定するSAVE文と、特定の復帰点の状態にデータベース等を戻すためのRESTORE文とがある。

8. 保全管理

データの保全性(integrity)の管理は、従来個々の適用業務プログラムに任されており、プログラマーは、更新の際にデータの正当性を調べデータベース内に矛盾が生じない様に検査する手続きを作成する必要があった。しかし、この保全管理はデータベース管理機能の一部にする事が望ましく、System-Rでは、これをassertionおよびtriggerという形で実現している。

assertionは、個々のデータあるいはデータの集合が満たすべき条件を指定するものである。すべての更新操作は、この条件を満たすか否か検査がなされ、もし違反していれば、それは無効となる。assertionはSEQUELで表現した命題で定義され、大別して、常に満たされるべき「状態」に関する条件と、更新によるデータベースの「遷移」に関する条件とがある。ここで、例を示す。

```
ASSERT ASRT1 ON EMP : SAL < 50000
ASSERT ASRT2 ON EMP :
    ( SELECT MAX( COUNT(*)) FROM EMP GROUP BY MGR ) < 50
ASSERT ASRT3 ON UPDATE TO EMP(SAL) : NEW SAL > OLD SAL
```

一般に、assertionの検査は、トランザクション終了時に行なわれるが、処理途中でも、ENFORCE INTEGRITYなる命令で検査を要求できる。また、IMMEDIATEという属性をもつassertionは、常に更新操作のたびごとに検査が行なわれる。

データベースの目的の1つは、データの重複を避ける点にあるが、種々の適用業務の要求、特に効率面での要求を満たす為に、互いに依存したデータ項目、すなわち、一方から他方が導出できる様なものを重複して持つ事が起こり得る。この場合、一方に変更が生じたなら、それに依存しているデータ項目も同時に変更する必要があり、System-Rでは、この手続きをtriggerとして登録する事ができる。例えば、今まで用いてきたデータベース例において、DEPTの中のNEMPSは、EMPから導出できるものである。したがって、矛盾が生じない様にするには、以下の様な3種類のtriggerを登録しておく必要がある。

```
DEFINE TRIGGER EMPINS ON INSERTION OF EMP :
        ( UPDATE DEPT
          SET    NEMPS = NEMPS + 1
          WHERE  DNO = NEW EMP.DNO )
DEFINE TRIGGER EMPDEL ON DELETION OF EMP :
        ( UPDATE DEPT
          SET    NEMPS = NEMPS - 1
          WHERE  DNO = OLD EMP.DNO )
```

```
DEFINE TRIGGER EMPUPD ON UPDATE OF EMP(DNO):
    ( UPDATE DEPT
      SET    NEMPS = NEMPS - 1
      WHERE  DNO = OLD EMP.DNO ;
      UPDATE DEPT
      SET    NEMPS = NEMPS + 1
      WHERE  DNO = NEW EMP.DNO )
```

9. おわりに

本稿では、RDI を通して利用者に提供される System－R の機能、すなわち SEQUEL 言語の機能を中心に、いわばその外部仕様を述べるにとどめた。しかし、System－R については まだ多くの興味ある話題が残っている。例えば、データベース検索の最適化、階層形式や網目形式のシミュレーション、障害回復、それに排他制御の具体的実現方法等々である。これらに関しては、最後に掲げる参考文献を参照して頂きたい。

参考文献

1) Boyce, R. F.; Chamberlin, D. D.; King, W. F.; Hammer, M. M. : "Specifying Queries as Relational Expressions : SQUARE", IBM Research Report RJ1291 (1973), also available as, Data Base Management (Proc. IFIP Working Conference, Corsica), North-Holland (1974)

2) Boyce, R. F.; Chamberlin, D. D. : "Using a Structured English Query Language as a Data Definition Facility", IBM Research Report RJ1318 (1973)

3) Chamberlin, D. D.; Boyce, R. F.; Traiger, I. L. : "A Deadlock-free Scheme for Resource Locking in a Data-base Environment", IBM Research Report RJ1329 (1973), also available as, Information Processing 74 (Proc. IFIP Congress, Stockholm), North-Holland (1974)

4) Chamberlin, D. D.; Boyce, R. F. : "SEQUEL : A Structured English Query Language", IBM Research Report RJ1394 (1974), also available as, Proc. of ACM-SIGFIDET workshop, Ann Arbor (1974)

5) Lorie, R. A. : "XRM - An Extended (n-ary) Relational Memory", IBM Cambridge Scientific Center Report G320-2096 (1974)

6) Astrahan, M. M.; Lorie, R. A. : "SEQUEL-XRM : A Relational System", Proc. ACM Pacific Conference, San Francisco (1975)

7) Astrahan, M. M.; Chamberlin, D. D. : "Implementation of a Structured English Query Language", Comm. ACM, Vol. 18, No. 10 (1975)

8) Reisner, P.; Boyce, R. F.; Chamberlin, D. D. : "Human Factors Evaluation of Two Data Base Query Languages : SQUARE and SEQUEL", IBM Research Report RJ1478 (1974), also available as, Proc. AFIPS National Computer Conference, Anaheim (1975)

9) Chamberlin, D. D.; Gray, J. N.; Traiger, I. L. : "Views, Authorization, and Locking in a Relational Data Base System", IBM Research Report RJ1486 (1974), also available as, Proc. AFIPS National Computer Conference, Anaheim (1975)

10) Eswaran, K. P.; Gray, J. N.; Lorie, R. A.; Traiger, I. L. : "The Notions of Consistency and Predicate Locks in a Database System", Comm. ACM, Vol. 19, No. 11 (1976)

11) Eswaran, K. P.; Chamberlin, D. D. : "Functional Specification of a Subsystem for Data Base Integrity", IBM Research Report RJ1601 (1975), also available as, Proc. International Conference on Very Large Data Bases, Framingham (1975)

12) Gray, J. N.; Lorie, R. A.; Putzolu, G. R.; Traiger, I. L. : "Granularity of Locks and degrees of Consistency in a Shared Data Base", Proc. International Conference on Very Large Data Bases, Framingham (1975)

13) Reisner, P. : "Use of Psychological Experimentation as an Aid to Development a Query Language", IBM Research Report RJ1707 (1976), also available as, IEEE Trans. Software Engineering, Vol. 3, No. 3 (1977)

14) Griffiths, P. P.; Wade, B. W. : "An Authorization Mechanism for a Relational Data Base System", IBM Research Report RJ1721 (1976), also available as, ACM Transactions on Database Systems, Vol. 1, No. 3 (1976)

15) Blasgen, M. W.; Eswaran, K. P. : "A Comparison of Four Methods for the Evaluation of Queries in a Relational Data Base System", IBM Research Report RJ1726 (1976)

16) Macleod, D. J. : "SEQUEL and QUERY BY EXAMPLE : Translation and Compatibility", IBM Research Report RJ1730 (1976)

17) Astrahan, M. M.; Blasgen, M. W.; Chamberlin, D. D.; Eswaran, K. P.; Gray, J. N.; Griffiths, P. P.; King, W. F.; Lorie, R. A.; McJones, P. R.; Mehl, J. W.; Putzolu, G. R.; Traiger, I. L.; Wade, B. W.; Watson, V. : "System R : A Relational Approach to Data Base Management", IBM Research Report RJ1738 (1976), also available as, ACM Transactions on Database Systems, Vol. 1, No. 2 (1976)

18) Blasgen, M. W.; Eswaran, K. P. : "On the Evaluation of Queries in a Relational Data Base System", IBM Research Report RJ1745 (1976)

19) Blasgen, M. W.; Casey, R. G.; Eswaran, K. P. : "An Encoding Method for Multi-field Sorting and Indexing", IBM Research Report RJ1753 (1976)

20) Lorie, R. A. : "Physical Integrity in a Large Segmented Data Base", IBM Research Report RJ1767 (1976)

21) Chamberlin, D. D.; Astrahan, M. M.; Eswaran, K. P.; Griffiths, P. P.; Lorie, R. A.;, Mehl, J. W.; Reisner, P.; Wade, B. W. : "SEQUEL2 : A Unified Approach to Data Definition, Manipulation and Control", IBM Research Report RJ1798 (1976), also available as, IBM J. Res. Develop., Vol. 20, No. 6 (1976)

22) Eswaran, K. P. : "Specifications, Implementations and Interactions of a Trigger Subsystem in an Integrated Database System", IBM Research Report RJ1820 (1976)