

CODASYL データベース用 簡易形データベース言語 - DQL -

逆井義文 益井清紀 大沼幸平 清水信昭
(電電公社 横須賀通研)

1. まえがき

データベースをより普及させるためには、利用者層の拡大や処理の多様化に対処する必要があり、プログラミング技術をもたないユーザや非定形的な業務にも適用可能な非手順型のデータベース言語の実現が必須である。

公社のDIPS⁽¹⁾においては、CODASYL仕様⁽²⁾に準拠したDBMSであるDEIMS⁺を既に実用化している⁽³⁾。しかし、一般にDBMSを利用してデータベースを操作するには手順型のデータベース言語を用いて業務プログラム(AP^{tt})を作成しなければならなかった。このため、これまでのDBMSは、高度なプログラミング技術をもつユーザや定形的な業務(処理内容に変動がなく、APの修正が不要な業務)での利用に限られていた。

そこで、DEIMSが管理するデータベースを簡単に操作できる機能をもつ簡易形データベース言語(DQL^{ttt})を開発した。

DQLは以下のような特徴をもつ。

- (1) 木構造、ネットワーク構造をもつCODASYL形の複雑なデータベースを、ドメインおよびタフルの2要素による表形式のリレーションで表現することにより単純化している。
- (2) APを作成することなく、端末から会話的にデータベースの操作かできるコマンド形式のデータベース言語である。
- (3) 1質問文で複数レコードを検索または更新できる高水準のアクセス機能をもつ。

本報告では、章2でリレーションの実現方式、章3では言語機能の概要、章4で質問文の解析および最適化の処理方式について述べ、章5で簡易化の効果等について考察する。

2. リレーションの実現方式

DQLは、CODASYLデータモデルに基づくデータベースを、エンドユーザに対してリレーショナルデータモデル⁽²⁾に準拠したデータベースとして表現する機能およびその操作機能を実現する。

本章では、DQLの操作対象であるリレーションの概念および具体的な実現方式について述べる。

2.1 リレーションの概念

CODASYLデータモデルでは、レコードとレコードの関係をセットという概念で陽に表現し、これによって、順、木、ネットワークというデータ間の構造(データ構造)を表し、このデータ構造にそってデータベースの操作を行う。

これに対し、リレーショナルデータベースモデルでは上記のような構造を陽には表現せず、データベースアクセス時にデータの内容に基づいてダイナミックに関係を決定して操作を行う。

次に、リレーショナルデータモデルに準拠したデータベースであるDQLのリレーションについて述べる。

DQLでは、リレーションの利用特性に応じて以下の3種類のリレーションを設定している。

① ベースリレーション

CODASYL形データベースのレコードタイプと直接対応するリレーションであり、DQLの操作対象

† Dendenkosha Information Management System

tt Application Program

ttt Dendenkosha database Query Language

の基本となるリレーションで検索および更新の両操作が可能である。

② 仮想リレーション

ベースリレーションに対してあらかじめ関係演算等を定義しておき、あたかも業務に対応した別のベースリレーションが存在しているように見せる仮想的なリレーション。

③ 一時リレーション

検索の中間結果を保持するために一時的に作成されるリレーションであり、データベースに対する種々の操作間でのデータ授受を行うのに必要なリレーション。

2.2 ベースリレーションの実現方式⁽³⁾

(1) CODASYL形データ構造とのマッピング方式

CODASYL形のデータ構造の構成要素としてレコード間の関係を示す「セット」、個々のレコード実体(すなわちレコードオカレンス)の格納場所を示す「レルム」がある。これらの要素はリレーショナル形では存在しないため、その要素がもつ情報を失うことなく概念のみを隠ぺいすることが必要となる。

CODASYL形のデータ構造をベースリレーションにマッピングする方式は、最も重要な構成要素であるセットに着目して考えると次の2つがある。

(a) 統合リレーション方式

セットで関係づけられている全てのレコードタイプを統合して一つのベースリレーションとする方式。すなわち、相互に関係のあるレコードオカレンスを結合させてベースリレーションの1ファルとすることによりセットの情報を隠ぺいする方式。

(b) 単位リレーション方式

各レコードタイプを個々の独立したベースリレーションとし、セットの情報を保持するために特別なドメ

イン(これを付加ドメインと呼ぶ)を新たに追加する方式。

この2方式の概念図を図1に、比較を表1に示す。

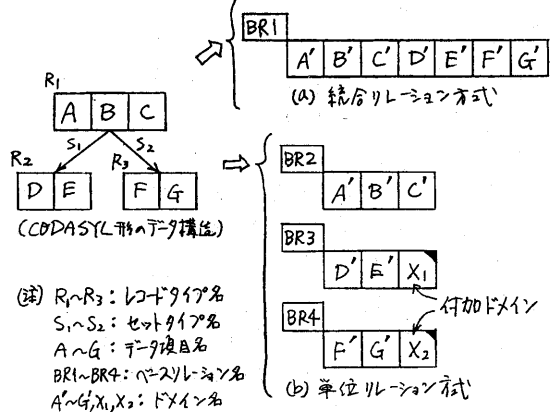


図1 CODASYL形データ構造からベースリレーションへのマッピング方式

表1 マッピング方式の比較

方式 比較項目	統合 リレーション 方式	単位 リレーション 方式	説 明
リレーション数	少	多	単位リレーション方式ではレコードタイプと同数が必要である。
関係づけの 必要性 (操作時)	小	大	統合リレーション方式では、セットで関係づけられている部分については操作時の指定は不要となる。 単位リレーション方式では、複数リレーションと操作时必须に結合演算により関係づけが必要である。
表示法の 容易性	困難	容易	統合リレーション方式では、簡單なネットワーク構造でもオカレンス間の表示法が極めて困難である。 また、ファル長が大きい出入力の操作性が低下する。
更新時の色相 性への対応	困難	比較的 容易	1ファルは複数のレコードタイプのオカレンスを含み、統合リレーション方式では、ドメインの追加、削除が必要となり対応が困難。 単位リレーション方式では、1つのレコードタイプは着目できるので、対応が容易である。
マッピングアル ゴリズムの容易性	困難	比較的 容易	統合リレーション方式では、全てのレコードタイプおよびセットタイプが対象となるので、アルゴリズムが複雑となる。 単位リレーション方式では、当該リレーションに対応するレコードタイプと関係するレコードタイプおよびそのセットタイプに着目すればよい。
総合評価	△	○	

統合リレーション方式は、リレーション数が少ない、セットの情報からリレーション自体に包含されるため操作時の関係づけ指定が不要になる等の長所をもつ反面、データ構造が複雑になるにつれて出力時の表示法、マッピング処理への対応が困難となる。

一方、単位リレーション方式はレコードタイプ数と同数のリレーションが必要となるが、データ構造が複雑になっても付加ドメインの追加のみでリレーションへのマッピングが可能であり汎用性が高い。また、関係づけ指定はリレーションモデルにおける基本操作であるので操作性上は特に問題とはならない。

以上のことから、DQLでは単位リレーション方式を基本方式として採用し、更に文献データベースなどの比較的単純な木構造の場合に長所を活かすことのできる統合リレーション方式も制限付きでサポートすることとした。

(2) 付加ドメイン

単位リレーション方式で必要となる付加ドメインの実現方式について述べる。

セットには、その属性としてオーナのレコードオカレンスを特定するための手段を示すセットセレクションがあり、その種類としてはSYSTEM, KEY, APPLICATIONの3種がある。表形式のリレーションに対するアクセス処理を効率よく行うためには、個々のタプルを一意に識別できるキー情報が必要となる。しかし、セットセレクションがAPPLICATIONの場合はナビゲーション処理を目的としているため、キー情報が無いので対応困難である。また、SYSTEMの場合はデータベースモニターが対象レコードを一意に決定するのでキーとなる情報は不要であり、セットの情報をベースリレーションにとり込む必

要はない。

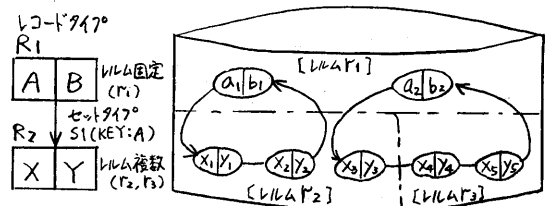
したがって、セットセレクションがKEYの場合のみを対象とし、「関係キードメイン」の中に対応するオーナーレコードオカレンス内のキーデータ項目値を格納することによりセット情報を保持している。

レルムについては、オーナーレコードおよびメンバレコードが複数レルムに関与する場合のみ、それぞれのレルム名を「関係レルムドメイン」、「自己レルムドメイン」に格納することによりレルムの情報を保持している。

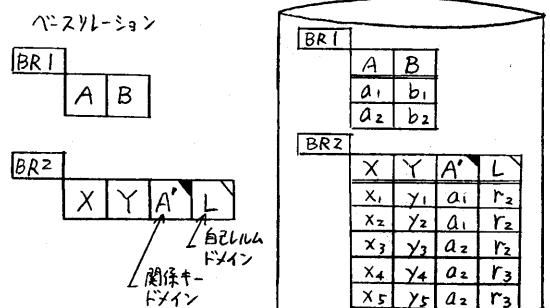
付加ドメインの種類と設定条件を表2に、具体例を図2に示す。

表2 付加ドメインの種類と設定条件

付加ドメイン名(略号)	説 明	設定条件
関係キードメイン (RKD)	オーナーレコード内のキー値を保持するドメイン	当該レコードメンバレコードの場合
関係レルムドメイン (RRD)	オーナーレコードの各オカレンスごとのレルムを決定するドメイン	オーナーレコードの関与するレルムが複数の場合
自己レルムドメイン (SRD)	当該レコードの各オカレンスごとのレルムを決定するドメイン	当該レコードの関与するレルムが複数の場合



(a) CODASYL形のデータベース



(b) DQLのベースリレーション

図2 付加ドメインの具体例

2.3 仮想リレーションの実現方式

業務対応の処理操作の簡易化、効率化のためには、それぞれの業務に必要なドメインまたはタプルのみをベースリレーションから抽出したり、複数のベースリレーションを種々の条件で結合することによりあらかじめ当該業務対応のベースリレーションが存在するように見せる機能が有効である。

これを実現するため、DQLでは仮想リレーションの定義機能を導入した概念図を図3に示す。

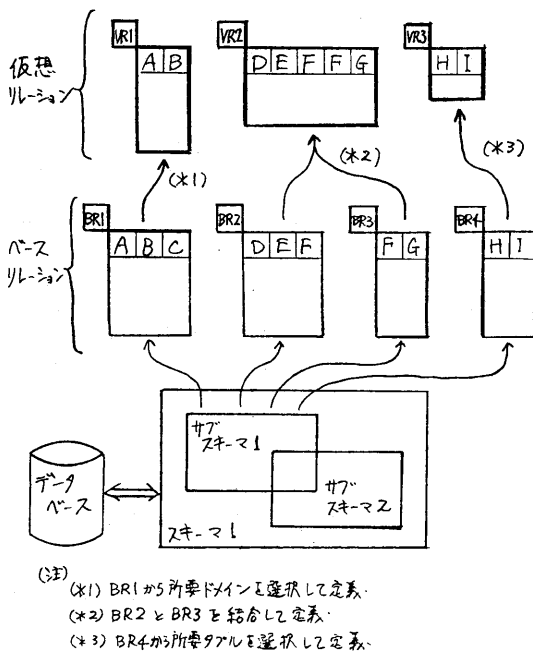


図3 仮想リレーションの定義機能の概念

仮想リレーション定義機能を実現する際に重要となる定義情報の保持方式としては、定義内容をそのままソース形式で格納しておく方式を採用した。このソース格納方式は格納域が小さく、定義情報の案内がユーザが指定した形式のままで出力できる等ユーザ自身にとって望ましい特徴を有する。

3. DQLの概要

3.1 言語形式

言語形式は一般に手順形と非手順形に大分類できる。DQLでは初心者でも容易に使用できる言語とすることから非手順形を採用することとした。さらに、DQL操作を依頼するための質問文（DQL文）の具体的な指定形式について表3に示す4つのタイプを考え、①ノンプログラマユーザの理解のし易さ、②質問内容の表現のし易さ、③表現の明確さ、④機能の拡張性、の観点から比較評価した。

その結果、表3のうち「代数形」と「述語形」はともに表現形式が特殊であり、他の2つのタイプに比べ①および②の評価項目で劣り、DQLとしては望ましくないと判断した。

「流れ形」と「構造型」はいずれも①～④を満足しており、かつ同程度で優劣がつけ難い。表4の相違点に示すように流れ形の方が日本人の思考過程にマッチしていること、エンドユーザを対象とした調査において流れ形がより使い易いとの結果が得られたことから、DQLでは流れ形を採用した。

表3 非手順形の分類

項番	形式名	概要	表現例(注)
1	流れ形	機能対応の質問文の指定順序により処理内容を表わす。	GET EMP GROUP DNO FOR AVG(SAL) >= 20 SELECT DNO
2	構造型	質問文に階層的に指定して処理内容を表わす	SELECT DNO FROM EMP GROUP DNO HAVING AVG(SAL) >= 20
3	代数形	代数文により処理内容を表わす	EMP(AVG(SAL BY DNO) >= 20) (DNO)
4	述語形	述語論理式により処理内容を表わす。	RANGE OF E IS EMP RETRIEVE INTO W(E,DNO) WHERE AVG(E.SAL BY E.DNO) >= 20

(注) 表現例の質問内容およびリレーションの構成は次のとおりである

<質問内容>

社員の平均給料が20以上の部門を求めよ。

<リレーション構成>

EMP						
社員No	社員名	部門名	給料	上司ID	仕事	
EMPNO	NAME	DNO	SAL	MGR	JOB	

表4 構造型言語と流れ形言語との相違点

項番	比較項目	表現形式	
		構造型	流れ形
1	質問文を指定する順序	1. 出力項目 2. 検索対象 3. 検索条件 (実際の処理手順と逆の順序で指定する)	1. 検索対象 2. 検索条件 3. 出力項目 (実際の処理手順と同じ順序で指定する)
2	アロツク間のバグの受け渡し方法	質問アロツク間のネスト関係によりインプリメントに指定する。	一明リレーションの作成を陽に指定する。
3	言語の類似性	英語的 (関係代名詞の概念がある ※使用しては、語に付する修飾を後で行う)	日本語的 (語に対する修飾は前に行われ、その結果に基づいて以降の記述を行う)
4	処理手法	トップダウン的 (お目的を先に述べ、その後個々の記述を行う)	ボトムアップ的 (お目的の記述を行、その後これらと組合せて目的を得る)

3.2 言語構成

操作性向上のためには、データベースの操作内容を指示する言語とその実行を指示する言語とを明確に分離して階層構成にする必要がある。このため、DQLでは前者をDQL文、後者をサアコマンドとして分離し実現している。

DQLの言語構成は以下の4つに階層化される。

- ① DQLコマンド
DQLを起動するためのシステムコマンド名(DQL)を入力する。
- ② コマンドパラメータ
DQLでの処理全体を通じて必要な制御情報を入力する。
- ③ サアコマンド
実行開始、ベースリレーションの定義・案内、カタログされたDQL文の呼出しなどの機能をサアコマンド名で指定する。
- ④ DQL文
データベースの操作内容をDQL文の組合せにより指定する。
DQLの言語構成を図4に示す。

3.3 機能概要

節3.2の言語構成で述べたようにデータベース自体に対する操作機能をD

```

/D001/ DQL -----DQLコマンド
DQLパラメータ: RDI, U -----DQLパラメータ

#000010 GET RI
#000020 FOR KA="イトウ"
#000030 PRINT NAME, NENREI
#000040 /RUN -----サマコマンド

```

処理結果

```

*** XXテン シュツリョウシメシタ -----通知メッセージ

#000010 MODIFY RI
#000020 FOR KA="イトウ"
#000030 SET KA="イトウ1"
#000040 /RUN

*** XXテン コウシンシメシタ

#000010

#000040 /RUN

#000040 /END -----サマコマンド
(DQL終了指示)

```

図4 DQLの言語構成

QL文で、その他の機能をサアコマンドで分担させている。DQL文、サアコマンドの機能をそれぞれ表5、表6に示す。

本節では、これらのうちデータベース操作の主要機能について述べる。

(1) 検索機能

(a) 高水準検索機能

データベースから目的とするデータを効率よく、かつ少ないインタラクシオン量で検索するため高水準検索機能を実現している。具体的には、大小比

表5 DQL文の機能

分類	DQL文	内容
検索機能	GET	検索対象リレーションの指定
	PRINT	検索結果の定型出力
	SELECT	検索結果の一時保存
更新機能	INSERT	ダブル単位の追加
	DELETE	ダブル単位の削除
	REPLACE	ダブル単位の置換
	MODIFY	ドメイン単位の加工
	SET	ドメインの加工方法の指定
共通機能	DECLARE	一時変数ドメインの定義
	FOR	条件式の指定
	GROUP	ダブルのグループ化(群別化)
	SORT	ダブルのソート

表6 サブコマンドの機能

分類	サブコマンド名	内 容
デレトリ 管理	/BRELATION	ヒストリシオン(BR)の定義及び削除を行う
	/VRELATION	仮想ヒストリシオン(VR)の定義及び削除を行う
	/GUIDE	BR及びVRの定義情報を出力する
実行制御	/RUN	DQL実行プログラムの処理を開始する
	/END	DQLコマンドの処理を終了する
	/SKIP	DQL文の取消しを行う
	/EOD	入カデータの終了を指示する
	/CONTINUE	割込後の実行継続を指示する
	/DISCONTINUE	割込後の実行中止を指示する
ファイル入力	/FINPUT	ファイル入力状態へ切替える
	/TERMINAL	一時記憶入力状態へ切替える
	/PRETURN	ファイル入力状態へ戻る
	/FSTOP	ファイル入力を中止し、諸入力状態へ戻る

較演算子、論理演算子などの各種演算子を用いた条件式の指定により、その条件を満足する複数のタプルを同時に検索、出力することが出来る。

(b) 関係演算機能

リレーショナルデータモデルの基本操作機能によって、ほとんどのCODASYL形データ構造に制約されないレコード間の動的な関連づけが可能となり処理の柔軟性が向上する。

(2) 更新機能

基本機能として、タプル単位の追加置換、削除およびドメイン単位の加工がある。これらの更新処理においても前項で述べた各機能によって処理対象タプルを選択することが出来るため、高水準な操作が可能である。

4. 処理方式

4.1 モニタとのインタフェース

DQLは、リレーションに対する検索、更新等の処理を行う際、データ実体としてCODASYL形データベースにDEIMSのモニタを介してアクセスする。

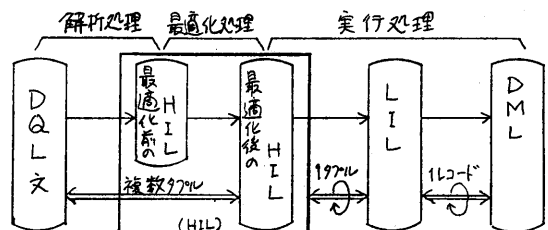
DQLとDEIMS間のインタフェースとしては、データベース操作命令(DML)を用いる方式を採用した。DMLはAPに開放されている命令で

† Database Manipulation Language

あるので、DMLインタフェースとすることによりDQLをDEIMSモニタから独立したプログラムとすることが可能となり、モニタの内部処理の変更、機能拡張による影響を少なくできる。

4.2 中間言語構成

DQLは単一レコードアクセス機能をもつDMLを利用して、エンドユーザに対しては複数レコード(すなわち複数タプル)を同時に操作できる集合アクセス機能を提供している。したがって、DQLは両機能のマッピングを行う必要があるが、これを中間言語によって実現している。中間言語の構成は、DQLの機能拡張やDMLの仕様変更等に対して柔軟な対応が出来ること、最適化処理およびリレーションの演算処理が簡潔に行えることを重視し、図5に示すような2階層構成とした。



(注) HIL:高水準中間言語 ⇔:データの複合単位
LIL:低水準中間言語 ⇔:くり返し要素
→:各言語間の変換過程

図5 2階層の中間言語構成

(2) 高水準中間言語(HIL^{††})

HILは関係演算等の集合アクセス機能をもつ中間言語であり、DQL文とほぼ1対1に対応した木構造で表現される。HILはDMLとのインタフェースを直接にはもっていない。したがって、DMLの機能を意識せずに木構造の入替えのみで最適化が可能であり、最適化処理を簡潔にすることが出来る。図6に最適化の例を示す。

†† High Level Intermediate Language

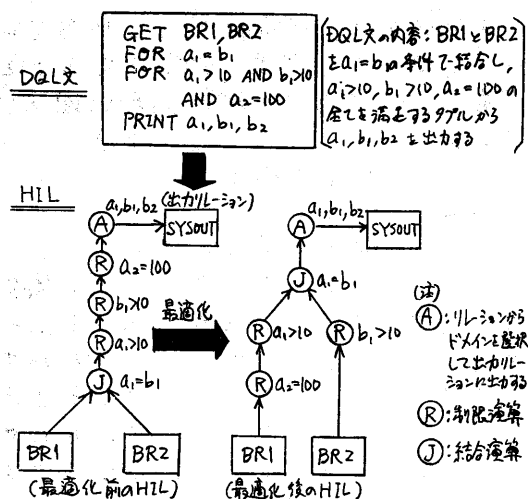


図6 最適化

(b) 低水準中間言語 (LIL[†])

LILは単一アクセス機能をもつ中間言語であり、最適化されたHILから変換される。LILは、タプル(付加ドメインを含む)に対するアクセス処理をCOBOL DASYL形のレコードに対するアクセス処理にマッピングするものであり、実行順序を示す直線構造で表現される。

DQLでは、DMLとのインタフェースをLILにのみもたせているので、DMLの仕様変更等による影響範囲の局所化が可能である。

4.3 最適化

同時に複数のリレーションを操作対象として複合演算を行うプログラムでは、一般に処理時間が長くなる。したがって、処理効率の最適化を行うことが必須となる。

このため、DQLではHILの段階で無効演算の削除、制限演算および結合演算の実行順序の変更などの最適化処理を行いデータベースへのアクセス回数、内部演算回数の最小化を図って

† Low Level Intermediate Language

いる。

(1) 無効演算の削除

DQL文中の無意味な処理(例えば、保存要求のない一時リレーションの作成処理)、データ構造からみて不要な処理(例えば、レコードタイプ内のソートキーの順位および昇降順がユーザから指定されたソート処理と一致している場合)などを無効演算として削除する。

(2) 制限演算順序の変更

制限演算(条件に合致するタプルのみを選択する演算)の実行により一般にはタプル数が少なくなる。したがって、制限演算を先に行えばそれ以降のアクセス回数、演算回数を減らせることができる。

このためDQLでは、DQL文によって指定された操作内容の意味が変わらない範囲内で制限演算を優先させる。

(3) 結合演算順序の変更

結合演算は2個のリレーションを処理対象とするためデータベースへのアクセス回数が大幅に増加する。したがって、関係演算の中で最も処理効率が問題となる。このため、DQLでは結合演算についても制限演算と同様に、結合の結果によって得られるタプル数がより少なくなるものを優先するように演算順序を変更する。

制限演算と結合演算が混在している場合は、データベースへのアクセス回数の統計的な値に基づいてあらかじめ作成した総合順位表により、演算順序を決定する。

5. 考察

DQLの最も大きな特徴であるデータベース操作性の向上について具体例を基に考察する。

図7に示す検索例では、従来のAP作成による方法を用いると、ソースプログラムに約70ステートメントを要し、

さらにこれを③～⑥に示すコンパイル等の手順をふむ必要がある。

一方、DQLで同じ処理をするには4つのDQL文等を実行で済む。

DQLを利用することによる効果を以下に示す。

- (1) 同一処理に要するDQL文数、拡張COBOLのステートメント数の比は、1:10～1:20である。したがって、DQLを利用することによりステートメント数を大幅に減少で主作業工数の削減が図れる。
- (2) APを実行する場合、ソースプログラムから多くの手順をふむ必要があるのに対し、DQLでは端末からDQL文を入力するだけで直ちに結果が得られる。

6. むすび

本資料では、簡易形データベース言語(DQL)の特徴、実現方式、機能概要および簡易化の効果などについて述べた。

DQL技術を要約すると次のとおりである。

- (1) データ構造に関する情報を付加ドメインの内容として表現することにより、CODASYL形データベースを単純な表形式のデータベースにマッピングする方式を実現した。
- (2) 表形式のデータベースを多様に操作できる高水準の言語仕様を確立した。
- (3) 利用者の質問文から最適なデータベース操作命令(DML)群を効率よく生成する2階層の中間言語方式を実現した。

今後はDQLを公衆形サービス(DEMOSEなど)に適用することによりデータベース利用者層の拡大を図るとともに、運用経験をふまえてシステムの改良を進める予定である。

参考文献

- (1) CODASYL: CODASYL Data Description Language Committee, Journal of Development, 1978
- (2) D.D.Chamberlin and M.M.Astrahan et al.: SEQUEL 2; A Unified Approach to Data Definition, Manipulation and Control, IBM J. of Res. & Dev., 20, No.6, 1976.
- (3) 益井, 山下, 大沼: CODASYL形データベース操作の簡易化の手法, 信学会誌, No.1430, 1980.
- (4) 伊藤, 石塚他: 更新処理効率を重視したインベータシステムの構成法, 信学技報, EC81-14, 1981.

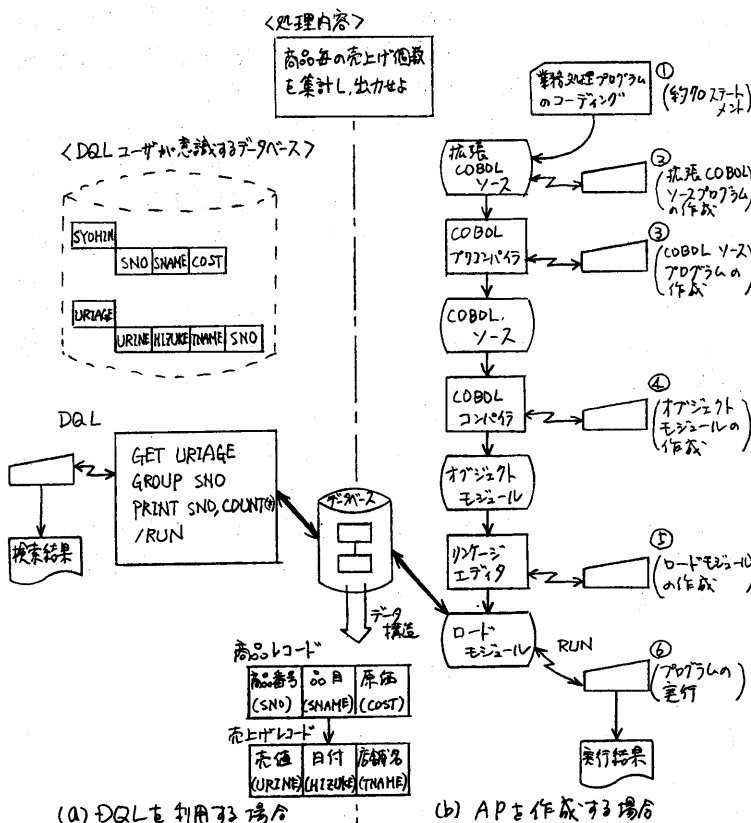


図7 操作性の比較