

データベース構築促進及び技術開発に関する報告書

建築CAD用拡張可能データベースの
プロトタイプ作成

平成3年3月

財団法人 データベース振興センター

委託先 三菱電機株式会社

本報告書は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて作成したものである。

序

データベースは、わが国の情報化の進展上、重要な役割を果たすものと期待されている。今後、データベースの普及により、わが国において健全な高度情報化社会の形成が期待される。さらに海外に対して提供可能なデータベースの整備は、国際的な情報化への貢献および自由な情報流通の確保の観点からも必要である。しかしながら、現在わが国で流通しているデータベースの中でわが国独自のものは3割にすぎないのが現状であり、わが国データベースサービスひいてはバランスある情報産業の健全な発展を図るためには、わが国独自のデータベースの構築およびデータベース関連技術の研究開発を強力に促進し、データベースの拡充を図る必要がある。

このような要請に応えるため、(財)データベース振興センターでは日本自転車振興会から機械工業振興資金の交付を受けて、データベースの構築および技術開発について民間企業、団体等に対して委託事業を実施している。委託事業の内容は、社会的、経済的、国際的に重要で、また地域および産業の発展の促進に寄与すると考えられているデータベースの構築とデータベース作成の効率化、流通の促進、利用の円滑化、容易化などに関係したソフトウェア技術・ハードウェア技術である。

本事業の推進に当って、当財団に学識経験者の方々に構築されるデータベース構築・技術開発促進委員会(委員長 山梨学院大学教授 蓼沼良一氏)を設置している。

この「建築CAD用拡張可能データベースのプロトタイプ作成」は平成2年度のデータベース構築促進および技術開発促進事業として、当財団が三菱電機㈱に対して委託実施した課題の一つである。この成果が、データベースに興味をお持ちの方々や諸分野の皆様方のお役に立てば幸いである。

なお、平成2年度データベースの構築促進および技術開発促進事業で実施した課題は次表のとおりである。

平成 3年 3月

財団法人 データベース振興センター

平成2年度 データベース構築促進・技術開発委託課題一覧

分野	課題名	委託先
社会	1 形態学的コメントを含む病理データベースのフィージビリティ調査	(株)エス・ピー・オー
	2 災害情報データベース支援環境の構築	(株)防災都市計画研究所
	3 AV/MARCのための分類索引データベース構築	(株)ダイソメディアサービス
	4 気候情報データベースの構築	(株)エムテーエス雪氷研究所
	5 健康の自己管理と病気予防データベースの構築	(株)コンピュータコンビニエンス
	6 シルバーエイジの実態及び生活に必要な情報のデータベース構築のための調査研究	美崎高齢者福祉互助会美崎生活館
	7 交通事故調査データのデータベース化に関する調査研究	(財) 日本自動車研究所
地域活性化 中小企業振興	8 アジア太平洋交流データベースの課題性の研究	(株)西日本新聞社
	9 戦略商圏レベルに細分化した地域データと分析・提案手法を統合化した企画支援システムデータベースの構築	パラシュート情報開発研究会 札幌凸版印刷(株)
	10 ネットワーク化された地域情報データベースの有効なマネジメントについての調査研究	セントラル開発(株)情報図書館 RUKIT
	11 徳島市中小商業振興データベースの構築	(株)ニューメディア徳島
	12 九州地域の人材情報データベース構築	(財) 九州産業技術センター
海外	13 海外向け国内先端技術分野中堅企業情報英文データベース構築	コムラインインターナショナル(株)
	14 海外規格 (ソ連邦国家規格) データベースの整備	日本電子計算(株)
	15 政府開発援助 (ODA) に関するデータベースの構築調査	(財) 日本国際協力システム
	16 専門用語データベースシステムの機能に関する調査研究	アイ・エヌ・エス(株)
	17 専門家データベース構築事業	(財) 海外貿易開発協会
技術	18 VAN用データベース管理システムの開発	シャープ(株)
	19 レコードマネジメント用辞書管理システムの開発研究	(株)オフィス総研
	20 建築CAD用拡張可能データベースのプロトタイプ作成	三菱電機(株)
	21 先進複合材料データベース・プロトタイプの作成	(財) 次世代金属・複合材料研究開発協会
	22 マイクロコンピュータのプロγραμμαブル周辺デバイスのデータベース化	(社) 日本システムハウス協会
	23 書誌データベース用ダイナミック・シソーラスの可能性調査と実験	(株)紀伊國屋書店

目 次

1	本委託開発の概要	1
1. 1	本委託開発の背景と目的	1
1. 2	本委託開発の作業概要	3
2	拡張可能データベースの設計	4
2. 1	拡張可能データベースの機能要件	4
2. 2	スキーマ定義言語の構成検討	7
2. 3	データベース操作言語の構成検討	14
2. 4	ユーザインタフェース機能の概要	28
3	データ構成	30
3. 1	建築CAD用のデータ構成	30
3. 2	属性データの構成	32
3. 3	形状データの構成	34
4	プロトタイプシステムの概要	36
4. 1	システムの概要	36
4. 2	開発環境とハードウェア構成	37
4. 3	ソフトウェア構成	37
4. 4	操作イメージ	38
4. 5	実現方式	39
5	実行方法	55
5. 1	実行方法の概略	55
5. 2	システム開始機能	56
5. 3	システム終了機能	57
5. 4	データベース初期化機能	58
5. 5	データベースオープン機能	61
5. 6	データベースクローズ機能	65
5. 7	オブジェクト生成機能	69
5. 8	オブジェクト格納機能	73

5. 9	オブジェクト削除機能	77
5. 10	オブジェクト検索機能	83
5. 11	オブジェクト定義機能	91
5. 12	オブジェクト属性表示機能	98
5. 13	形状表示機能	102
5. 14	リンク生成機能	110
5. 15	リンク削除機能	114
5. 16	リンク表示機能	118
5. 17	スキーマ更新機能	122
5. 18	操作履歴初期化機能	126
5. 19	操作履歴表示機能	129
6	今後の課題と展望	131

1. 本委託開発の概要

1. 1 本委託開発の背景と目的

技術情報やノウハウの共有化、標準化、蓄積を目的としているCAD/CAM応用システムは、一般機械設計に限らず、自動車、航空機、更に建築分野やVLSIに代表されるエレクトロニクス分野においても実用化されている。

適用分野の広がりに伴い、システムに対する要求の多様化、複雑化に対処するための柔軟かつ効果的な開発技法や要素技術の研究・開発が待たれている。

設計対象のモデリング手法やそれらの内部表現が性能を左右するCADシステムにおいては、特にデータベースに対する要求が増加してきている。

建築CADシステムを例にした場合、そのデータベースは、建築物の形状データと設計・管理情報としての属性データとを統合管理することや、形状の追加、削除、詳細化に伴いデータベースの構造を頻繁に変更できることが必要となる。

本委託開発の目的は、

- ・ 形状データと属性データの統合管理機能
- ・ 階層的データベース構造の拡張機能
- ・ 建築レイアウト図の表示機能

を特長とする建築CAD用拡張可能データベースのプロトタイプを作成し、建築CADデータベースに対する適用性を検討することである。

利用分野としては、建築CADを想定しているが、機能の実現に際しては、拡張可能データベースやオブジェクト指向データベース等の研究成果を鑑みつつ、建築CAD特有のものではなく、広く、エンジニアリングあるいはオフィス・オートメーション用のデータベースへの適用を目指す方針とした。

本委託開発においては、形状データと属性データを統合するため、データベース・スキーマ定義言語とデータベース操作言語を設計し、それらのプロトタイプシステムの試作を以下の2つの作業フェーズに分けて行った。

(1) 方式検討・詳細設計

方式検討フェーズにおいては、

- ・ 形状データと属性データの統合管理機能
- ・ 階層的データベース構造の拡張機能
- ・ 建築レイアウト図の表示機能

の実現を目標に、

- ・ データベーススキーマ定義言語の仕様検討
- ・ データベース操作言語の仕様検討
- ・ ユーザインタフェース機能の仕様検討

を中心とした設計作業を行った。

(2) プロトタイプシステムの製作

(1)で得られた結果をもとに、実用化への知見を得るためプロトタイプシステムの製作を行った。

2章においては、拡張可能データベースの設計作業として実施した、機能要件の明確化、データベーススキーマ定義言語やデータベース操作言語の構成、更にユーザインタフェース機能に関する検討結果を記述する。

3章においては、本システムが対象としている建築CAD用システムのデータ構成（属性データ及び形状データ中心）やそれらのスキーマ定義言語による記述例について記述する。

4章においては、作成したプロトタイプシステムの概要、開発環境とハードウェア構成やソフトウェア構成、実現方式について述べる。

5章においては、本システムの実行方法を記述する。最後に、6章において、今後の課題と展望を述べる。

1. 2 本委託開発の作業概要

本委託開発における拡張可能データベースのアーキテクチャを図1—1に示す。

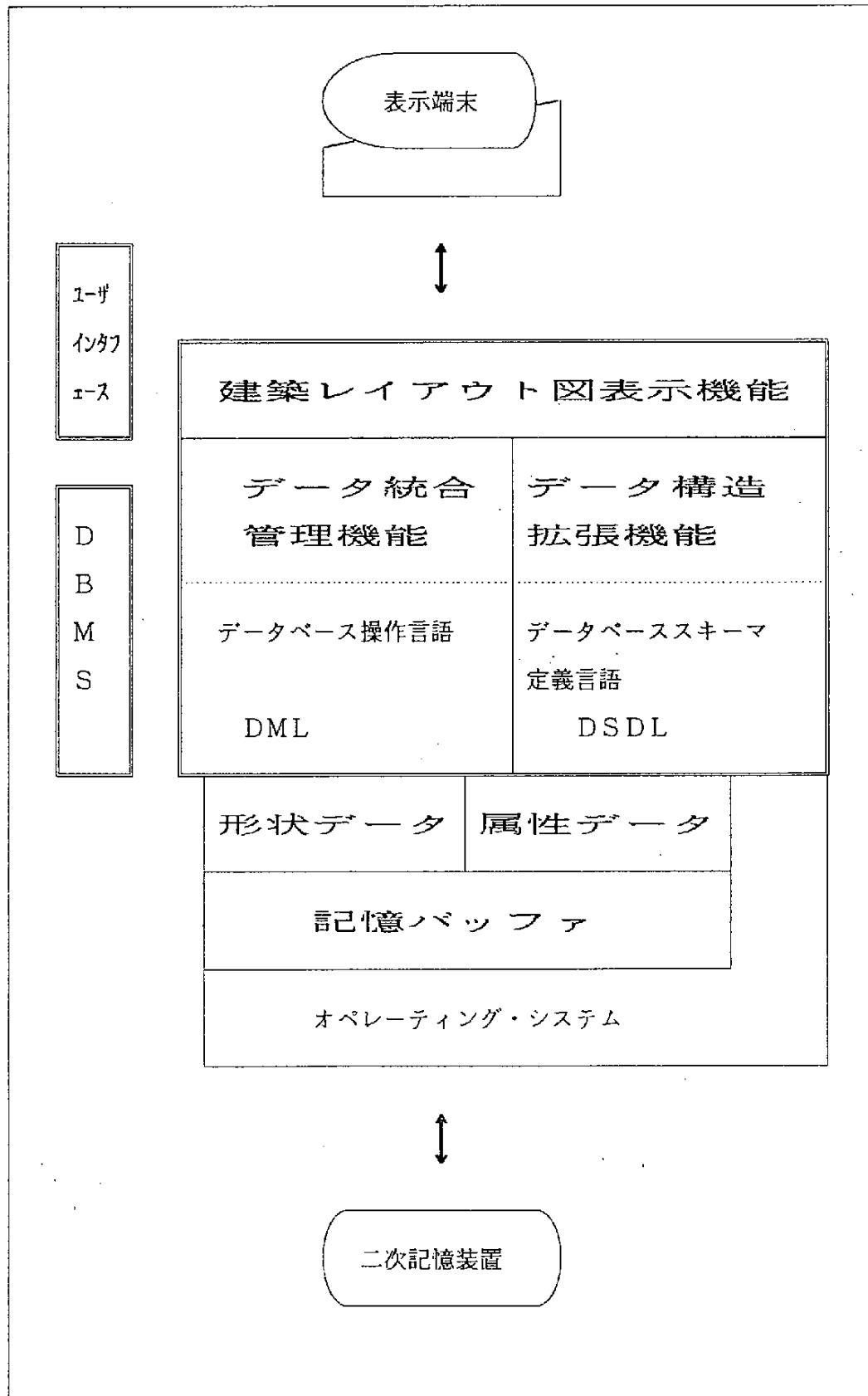


図1—1 拡張可能データベースのアーキテクチャ

2. 拡張可能データベースの設計

2. 1 拡張可能データベースの機能要件

本委託作業においては、

- ・ 形状データと属性データの統合管理機能
- ・ 階層的データベース構造の拡張機能
- ・ 建築レイアウト図の表示機能

の実現を目標としている。以下、各実現機能毎に機能要件の明確化を試みる。

(1) 形状データと属性データの統合管理機能

従来、主なるデータベースの適用分野であった事務処理において用いられるデータの構造は、文字や数字といった比較的構造を持たないものであった。しかし、CAD分野に留まらずOAの分野においても、複雑な構造を有するデータの取扱が要求され、形状データと属性データを統合して扱う機能が必須となりつつある。マルチメディア化は、こうした動きをさらに加速すると考えられる。これまでに、形状データと属性データの階層的表現とそれらの統合方法を開発する試みがいくつかなされてきている。

図2-1は、建築レイアウト図を表現するための形状データと属性データを統合した例である。

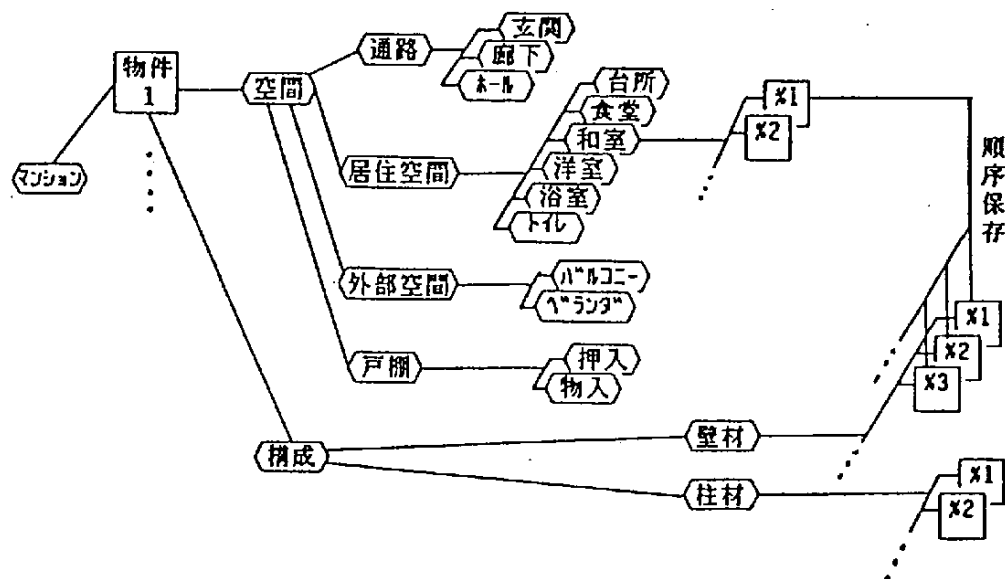


図2-1 形状データと属性データの統合例

本システムにおいても、何らかの形で、建築レイアウト図を表現するための形状データと属性データを統合する必要がある。

統合の内容・範囲についての設定は、いくつかのレベルが想定されるが、統合、管理機能に対する要件としては、形状（図形データ）から属性（文字データ）を参照でき、さらに、属性から形状を参照できることを目指すものとする。

(2) 断層的データ構造の拡張機能

CADやOAにおけるデータは、一度に完全な形でデータベースに入力されるわけではない。例えば、建築レイアウト図の場合、一本のエッジ（線分）から始まり、徐々にレイアウト図が作られていく。エッジは“壁”に対応し、エッジによって囲まれた領域は“部屋”に対応する。このように、レイアウト図が作られてゆく過程で、建築オブジェクトの階層構造を形成しなければならない。図2-2は、このような要求に答えるために開発した形状モデルの例を示している。

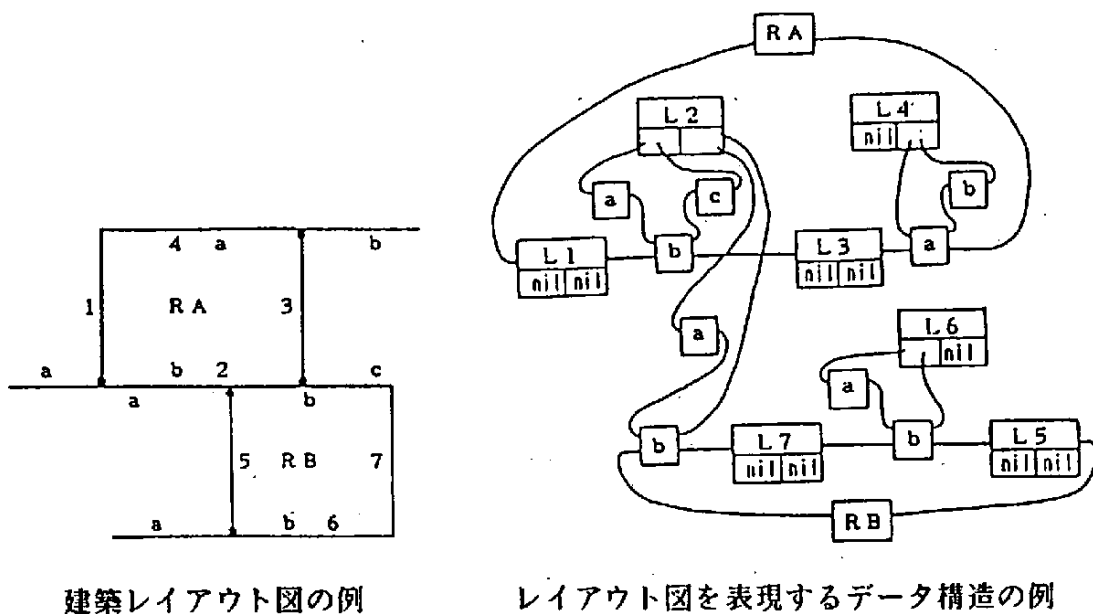


図2-2 形状モデル表現のためのデータ構造例

今回の委託開発においても、境界表現 (Boundary Representaion) モデルを操作する一連のコマンドを、上記(1)の機能を基盤として作成する必要がある。

具体的には、建築CADにおける階層構造の定義に際して、

- ・ 新しいクラスが定義できる。
- ・ クラス間の関係がリンクとして定義できる。
- ・ クラスの定義変更を動的 (スキーマ変更後、データベースの再構築を自動的に行う。) に行える。

等の機能を実現することにより、論理データに対する拡張可能性を実現することを目指す。

(3) 建築レイアウト図の表示機能

(1)、(2)で述べた“建築CAD用拡張可能データベース”の内容を確認するために、建築レイアウト図の図形表示機能が必要となる。

表示内容は、マンションを対象とした、2次元の建築レイアウト図であり、不動産分野のデータベース構造に関する設定の確認が可能なものとする。

表示機能の実現にあたって、論理的な視点 (ユーザが操作する視点) と物理的な視点 (データベース内部の構造) から以下の要件が設定された。

① 論理的な視点

- ・ マンション物件全体を表示できる。
- ・ 物件内の部屋毎に表示ができる。
- ・ 壁単位に表示ができる。

② 物理的な視点

- ・ クラス名、及び、オブジェクト名 (ID) で決定されるインスタンス単位に格納されている情報内容を表示できる。
- ・ データベースを構成している、インスタンス自身と他のインスタンスをつなぐリンク、及びそのリンクによってつながれた他のインスタンス全体の情報を併せて表示できる。

2. 2 データベーススキーマ定義言語の構成

データベースとしての仕様、および対象となるマッシュンデータの検討・モデル化の結果より、スキーマ定義言語としての機能要件を策定し仕様を検討した。以下、その経緯と結果について説明する。

2. 2. 1 スキーマ定義言語の仕様検討

基本構造については、以下の2つの案が検討された。

- ① C言語（以下Cと記す）のライブラリレベル。スキーマとCのプログラムは同じレベルであり、スキーマの変更はプログラムの再コンパイルによる方法。
- ② スキーマ作成モジュールを作成し、スキーマを別途作成させる。

一方、プログラムとデータ独立の観点から、スキーマのダイナミックな変更に対するデータベースの完全性についての検討を行った。

オブジェクト指向の反映形態・レベル（クラス定義、継承、e t c）については、仕様レベル、およびインプリメンテーションレベル各々場面において、どの程度「オブジェクト指向」の概念を反映させるかの検討を行った。

(1) 仕様検討の経緯

スキーマはデータベースの論理構造を決定付けるものであるが、このスキーマをどのような位置づけとするかは、以下の考え方がある。

- ① プログラムの中に直接記述する。
- ② スキーマをCのtypedef として宣言し、プログラムのコンパイル、リンクによりスキーマのアップデートを行う。
- ③ スキーマをファイルとして独立させ、スキーマを処理するモジュールを別途用意する。

①は拡張性がまったくないため、拡張可能データベースのスキーマとしては検討の対象外である。

②はスキーマをCの構造体として定義する方法である。したがって、スキーマの文法はCに準拠し、スキーマの解釈および作成はCのプリプロセッサおよびコンパイラによって行われる。この方法では、解釈系および生成系を作成する必要がないが、スキーマの変更はデータベースシステムそのものの再コンパイル、リンクにつながる。またCの複雑な文法規約をスキーマ作成者に要求することとなるうえ、Cが提供してい

るあらゆるデータタイプを許すこととなるので、データベースとしては不必要なデータタイプに対する対応を行う必要が生じる。スキーマの変更が多発しないのであればシステムとしての有為性があるが、プロトタイプシステムの様にスキーマ変更が頻繁に行われるようなシステムには適さない。

③はスキーマ定義のための文法を用意し、その文法に従って記述したスキーマファイルを解釈する処理系を用意する方法である。スキーマ処理系を用意しなければならない分システムとしては複雑となるが、スキーマ変更に対するシステムの再コンパイルおよびリンクは不要のため、拡張は容易である（データベースの完全性の保証は別の問題である。スキーマ変更に対するデータベース再構築に関しては2. 2. 1. 2 データベースの完全性 にて述べる）。また、文法は対象データベースを構築するうえで最低限度のものを用意すればよいので、今回のようなプロトタイプ作成に適している。アプリケーション（データベースを使用する側）からの要請により文法を拡張する必要が生じた場合でも、ある程度の範囲であれば、スキーマ処理系のみの拡張をおこなえばよい。

以上のような検討の結果、本プロトタイプにおいては③のスキーマファイルにより定義する方法を採用した。ただし、スキーマの文法としては

- ・クラスの定義
- ・親クラスの定義
- ・クラスのメンバ（属性）の定義

を行うのみとした。通常のスキーマ定義言語が提供しているような、探索キーの設定や親子集合の定義などはスキーマ定義言語としては特に用意しない。必要があれば各クラスの属性としてキーを定義したり、オブジェクト間に論理的な名前付でリンクをはるにより関係のあるオブジェクトを集合としてあつかうことができる機能をコマンド（データベース操作言語）として用意した。このような方式を採用することによりデータベースとして十分な拡張性を確保しつつ、スキーマ処理系の負荷を軽減させた。

(2) データベースの完全性

(1)で述べたように、本プロトタイプにおいてはスキーマファイルの変更によりデータベースの拡張を行うことができる。スキーマファイルが変更された場合、既存のデータ（オブジェクト）に対しても、新スキーマに適合するようにデータを更新する必要が生じる。これをデータベースの再構築と呼ぶが、データベースとしての完全性を保持したうえでデータベースの再構築を実行するのは困難である。従って、スキーマ変更に対してある程度

の制約をもうけ、その範囲内であるならばデータベースの完全性を保証して再構築を行えることを目標とした。

ここで、(1)で述べた文法を有するスキーマに対する変更要件としては以下のものが考えられる。

- ① 新規クラスの追加
- ② クラスの削除
- ③ クラス名称の変更
- ④ クラスの親クラス（スーパークラス）の変更
- ⑤ クラスのメンバの追加
- ⑥ クラスのメンバの削除
- ⑦ クラスのメンバの変更

これらの変更に対して、データベースの完全性を保証するためには以下の操作が必要である。

①は既存の他のクラスには影響を与えないため、問題はない。

②は削除対象のクラスに属するオブジェクトが全て削除される。さらに、削除されたオブジェクトに論理的にリンク付けられたオブジェクトについては、そのリンク関係を白紙化する必要がある。また、該当クラスをスーパークラスとして定義しているクラスに関しては、

- (i) スーパークラスの定義を無効にする（スーパークラスが存在しない）
- (ii) スーパークラスの削除時に同時に子クラスを全て削除する。

のいずれかの処理をおこなわなければならない。

子クラスは、(3)オブジェクト指向の反映形態 で述べるように、スーパークラスのメンバ（属性）を全て継承するので、(i) の方式の場合、該当クラスの全てのオブジェクトからスーパークラスの属性を削除しなければならないのでかなり困難である。

(ii)の方式は、処理としては1つのクラスの削除と同様の処理を繰り返すのみであるので比較的単純であるが、利用者側としては予期せぬクラスが削除されるので注意が必要である。逆にいえば、利用者が配慮して使用すれば問題は生じない。

以上から、②をサポートするとしたら、(ii)の方式が妥当である。

③クラス名称の変更はクラスに所属する全てのオブジェクトおよびスーパークラスを指定している全ての子クラスに影響する。さらにクラスの識別を行う唯一のキーであるのでこれを変更するともとのクラスとの同一性を認識できないのでクラス名の変更は全く別のクラスができる（前のクラスのインスタンスは全て消滅する）ことを意味する。

④スーパークラスの変更は、基本的には②と同じ問題が発生し、該当オブジェクト全てのスーパークラスから引き継いだ属性を削除し、新たなクラスから引き継いだ属性を設定しなければならない。これは②同様困難である。

⑤及び⑥はクラスのメンバの追加・削除は、各オブジェクトに新メンバ用の領域を確保するかデータごと削除するのみであるから、比較的容易である。

⑦クラスのメンバが変更になった場合、特にデータ長が変わるような変更の場合に、格納されているデータそのものをどうするか（新しい型に合わせて変更するか、データそのものを削除するか）が問題となる。

以上の検討から、③④⑦を除く変更ならばデータベースの安全性が保証できる仕様とした。

(3) オブジェクト指向の反映

本プロトタイプにおいて、オブジェクト指向の概念をどこまで反映させるかについての検討をおこなった。オブジェクト指向を特徴づける考え方として、クラス定義や継承などの概念があるがこれらをどの様に反映させるかについて述べる。なお、この検討結果は

2. 3 データベース操作言語の構成検討にもそのままあてはまる。

① データ単位としてのオブジェクト

データの単位として、リレーショナルデータベースにおけるレコードの代わりにオブジェクトという単位でデータを表現することを検討した。オブジェクトはインスタンス変数の値により一意に識別され、そのインスタンス変数はクラス定義において定義される。このインスタンス変数の定義にデータ型の概念を導入することで、データベースとして、様々なデータタイプを扱うことが可能となる。この考え方はリレーショナルデータベースにおけるレコードの概念と比較的近いうえ、後述のクラス階層や継承の概念を導入することでデータの表現を簡略化することができる。結合、分離などの関係演算もメソッドの記述により実現することは可能である。

② クラスと階層

データ型を定義するレコード型に相当するものとしてクラスを考える。クラス定義は上位クラスの指定とインスタンス変数の定義とからなる。レコード型でもレコード型間の関係を定義することは可能であったが、内部のデータ型を共有したり、継承したりすることはできない。クラス定義においては、上位クラスとして指定したクラスのインスタンス変数は自動的にすべて継承されるので、各クラスの定義が簡略化される。

2. 2. 2 スキーマ定義言語の文法

スキーマ定義言語とは、本プロトタイプで使用するオブジェクトのデータ型を定義するための言語である。ここで、スキーマとは各オブジェクトの型を規定するクラス定義の集まりであり、このスキーマ定義言語によってクラス定義を記述しているファイルをスキーマファイルとよぶ。

(1) 構成

クラスの記述はschema命令、super命令、member命令の組み合わせで記述される。各命令の出現順序はこの順序でなければならない。1命令は1行(80文字)以内に記述しなければならない。1行の最後には必ず改行コードを挿入しなければならない。また、行頭が#で始まる行又は各行の#以降はコメントとして無視される。

(2) schema命令

〔機能〕

この命令は、1つのクラス型ごとに最初に与える命令で、この中でクラス型の定義を行う。この命令に引き続き、super命令、member命令により、各クラスのidentityを定義する。

〔書式〕

schema schema_name

〔備考〕

schema_name は全クラスを通じて一意でなければならない。文字数は14文字以内とする。

(3) super命令

〔機能〕

この命令は、1つのクラスに対する上位クラスを定義するもので、schema命令に引き続いて記述する。上位クラスは必ず定義しなくてはならない。また、super命令は1つのschema命令の後に複数記述することができる。

また、すべてのクラスのsuper クラスとしてrootクラスが用意されている。rootクラス

には、データベース管理のためのインスタンス変数が定義されており、スキーマ定義ファイルの中で最低1つのschemaのsupere命令でroot クラスが指定されていなくてはならない。これにより、全てのクラスにデータベース管理のためのインスタンス変数が継承される。root schema に属するインスタンス変数は2. 3 で述べるデータベース操作言語では操作することはできない。

〔書式〕

```
super    super__name
```

〔備考〕

super__name はこの命令が記述されるより前にschema命令で記述されていなければならない。スーパークラスの宣言がループしてはならないが、複数のクラスが同一のクラスをスーパークラスとして宣言することはできる。

(4) member命令

〔機能〕

この命令は、schema命令、super命令に引き続き記述し、schema命令内部で使用される属性（インスタンス変数、つまりオブジェクト内部で使用される関数）を定義する。存在しない場合は定義は不用であり、また1つのschema命令に対して複数のmember命令を記述することができる。

また、一般的なオブジェクト指向パラダイムにおいて、インスタンス変数として要求されるような内部データの項目（ポインタなど）は本スキーマでは記述しない（あくまでデータベースの概念スキーマである）。このような項目はroot schema において記述し、各クラスに継承される。

〔書式〕

```
member  type variable__name  [initial__value]
```

〔備考〕

type は変数の型名であり、以下のいずれかを指定する。

- 1) char[バイト数]
- 2) short
- 3) cshort

- 4) long
- 5) clong
- 6) float
- 7) double
- 8) binary[バイト数]

cshort、clong は内容的にはデータの格納形式としてはそれぞれshort、longと同一であるが、表示時に3桁ごとにカンマが表示される。

variable __name は14文字以内で、このクラス定義のなかで一意的のもでなければならない（別のクラスのインスタンス変数との重複は許される）。又、memberの変数名として漢字（EUC漢字コードであること）を使用することができる。

オブジェクト間のリンクを表現するポインタはスキーマとしては表現されない。スキーマのインタプリタにより、リンク用ポインタの格納場所（root schemaにて定義される）はリンクが発生する度にダイナミックに生成される。

クラス内のインスタンス変数は全て固定長である。さらに、これらはハーフワードバウンダリを遵守しなければならない。

2. 3 データベース操作言語の構成検討

2. 3. 1 実現方式の検討

データベース操作言語の利用形式は以下の3つの形式が考えられる。

(1) ライブラリ形式

C言語等にリンクするライブラリ形式で実現し、上位のアプリケーションプログラム(CADなど)より呼び出す形態で利用する形式。

(2) コマンド形式

コマンドレベルでユーザがインタラクティブに指示・操作することにより利用する形式。

(3) 混合形式

(1)、(2)を組み合わせて利用する形式。

本プロトタイプでは、データベース定義言語およびデータベース操作言語に対する検討を目的としている。したがって、データベース管理システムとしての有為性とアプリケーション(データベース管理システム自体もアプリケーションの1つと考えるならば)に組み込む場合の有為性を考える必要がある。そこで、本プロトタイプにおいてはアプリケーションとしてデータベース管理システムを想定し、ユーザインタフェースを一般のC言語で記述し、データベース操作の為の関数をライブラリとして独立させた。これにより、データベース操作言語としての有為性とデータベースシステムとしての有為性の双方を検証することができる。

2. 3. 2 実現機能の種類検討

実現機能の種類検討に際しては、図2—3の機能分類・作成範囲を対象に作業を進めた。

機能大分類	機能中分類	機能小分類
属性・形状の統合	属性／形状インスタンスリンクの	A・U・D・S
拡張性	インスタンスが動的に	A・U・D・S
	リンク付けが動的に 物理リンク 論理リンク	A・U・D・S A・U・D・S A・U・D・S
	スキーマが動的に 基本型の組合せ 定義型の組合せ	A・U・D・S A・U・D・S A・U・D・S
	メソッドが動的に 変数宣言 Sequence if else do while 関数	A・U・D・S
入出力	図形表示 DBファイル OP（ログ）ファイル	初期化・表示 初期化・入出力 初期化・入出力

注）A：追加

U：変更

D：削除

S：検索

図2—3 機能分類・作成の検討範囲

2. 3. 3 実現機能の範囲・レベルの検討

本委託作業で実現する機能の範囲・レベルに対する検討の結果、以下の機能群を実現することとした。

- ・ 属性・形状の統合が可能
- ・ 新しいクラスが定義できる。
- ・ インスタンスが動的に作成できる。
- ・ クラス間の関係がリンクとして定義できる。
- ・ リンク付けが動的にできる。
- ・ クラスの定義変更を動的（スキーマ変更後、データベースの再構築を自動的に行う。）に行える。

(1) 属性・形状の統合機能

拡張可能データベースの機能要件として、属性（一般データ）と形状（幾何データ）の両方を統合して扱うことがあげられた。属性データと形状データを統合するためには、両方で共通の部分と異なる部分を明確に区別する必要がある。これについては、2. 2 スキーマ定義言語の構成検討 の節で述べたように、共通の部分を親クラスのインスタンス変数として記述し、属性、形状をその子クラスとして定義することで、統合機能については親クラスの機能を継承し、属性・形状固有の機能については各クラスで定義することで実現できる。

(1)-a 属性・形状インスタンスリンクの追加定義・変更・削除・検索

リンクに関しては、属性・形状のクラスでなく、2. 2 でのべたrootクラス（全てのクラスに共通の親クラス）において定義された変数を使用する。したがって、属性・形状の区別なくリンク操作を行うことができる。

追加定義に関しては、インスタンスをリンクするための領域をrootクラスのインスタンス変数として用意することで実現できる。このとき、

- ・ リンクできるインスタンス数の上限
- ・ 1つのインスタンスがリンクできるリンクの種類の上限

が問題となる。

インスタンスの数に関しては、CODASYL 型のデータベースに用いられた様なリンクのメンバ間でリンクポインタを保持する（各メンバが次のメンバへのポインタを持つ）ことで（メモリ等物理的な制約を除けば）上限は存在しない。リンク種類の数に関しては、同様にリンクの種類ごとにリンクポインタをもつことで上限をなくすることができるがインスタ

ンスとリンクで双方をリンクポイントにより実現するのはかなり複雑な処理となり、かつ必要性はあまりないと考え、リンクの種類には上限を設け、あらかじめそのための領域を確保する方式で実現することとした。

変更・削除機能に関しても、上で述べたようなリンクのメカニズムにより実現することができる。

検索機能はリンクの論理名を指定することでそのリンクに属するメンバを検索することができる。

(2) 拡張性

(2)-a インスタンスの動的な追加定義・変更・削除検索

インスタンスの動的な生成は、スキーマファイルによって定義されたクラスのインスタンス変数をもつ領域を動的に確保することで実現することができる。この領域は各インスタンスごとに独立にとられ、かつその大きさは(スキーマの動的な変更が行われない限り)固定であるのでその領域の先頭ポイントのみ管理していればよい。

動的なデータ変更はこの先頭ポイントから始まる固定長領域に対するデータ変更を行えばよい。

削除はこの先頭ポイントで示される領域を解放するのはもちろんであるが、リンクにより関係づけられたインスタンスのリンク情報を修正する必要がある。以下のような処理を行う必要がある。

- ・自分を親とするリンクに属するインスタンスの該当リンクに関するリンク情報の削除
- ・自分を子とするリンクに属するインスタンスの該当リンクに関するリンク情報の修正
(該当インスタンスをバイパスする)

検索はクラスに定義されたインスタンス変数の値により検索する機能のみ実現する。したがって、インスタンス変数を規定するためのクラスを指定しなければならない。これは検索の対象となるクラスの変数のみしか検索条件として指定できないことを意味する。そこで、検索の機能を拡張し検索対象のクラスと、検索条件のインスタンス変数が定義されるクラスとは異なっても可とすることとした。ただし、検索条件の定義されるクラスのインスタンスと検索対象のクラスのインスタンスとが何らかのリンクにより関係付けられていなければ検索することはできない。この拡張により、「広さxx m² 以上の台所がある物件」といった検索が可能になる。

(2)-b リンク付けが動的に追加定義・変更・削除・検索

(1)で述べたようなメカニズムにより実現できる。

(2)-c スキーマが動的に追加定義・変更・削除・検索

スキーマの動的な変更（追加定義、変更、削除）は必然的に既存のデータベースの再構築を伴う。その場合に整合性を保ちつつ再構築する必要があるので若干の制約が生じる。これらの詳細については、2. 2. 1. 2 データベースの完全性の項ですでに述べた。

追加定義は新たなクラスが増えるのみであるため、既存のインスタンスには影響を与えないため問題はないが、追加定義か削除かを判定することが困難なため、追加定義の場合もデータベースの再構築を行う。

変更に対しては、特にデータの大きさや型が変更されるため、既存のデータを保持することは困難である。従って、スキーマの動的な変更に対するデータベースの完全性は保証されない。

削除はデータそのものも削除されることを除けば問題がない。

また、スキーマはスキーマファイルにテキストとして記述するため、スキーマの変更はエディタにて行われる。さらに、検索はエディタを開いたうえでのエディタの機能により代替する。

スキーマの記述は2. 2. 3 で述べた文法に従って記述する。スキーマのメンバの型として他のスキーマ型を使用する「定義型の組合せ」は、スキーマのインタプリタ（再構築）を2 パス以上にする必要があるので、今回は行わないこととした。

(2)-d メソッドが動的に追加定義・変更・削除・検索

メソッドはメソッドをハンドリングする「メソッドインタプリタ」が必要となる。スキーマのインタプリタはデータの組み直しのみであったが、こちらはメソッドを解釈実行することが要求されるためプログラムとしてもかなり大きなものとなるので、今回は行わないこととした。

(3) 入出力

(3)-a 図形表示の初期化・表示

図形表示に関しては、属性の表示とは別のコマンドを設けた。当初は、属性表示・図形表示を同一コマンドとし、属性オブジェクトは属性の表示、形状オブジェクトは図形のと自動的に切り換えることも検討されたが、図形オブジェクトも何らかの属性データをもつためこれらの表示のことも考慮して別コマンドとした。

(3)-b DBファイルの初期化・入力・出力

データベースの初期化はコマンドとしては危険なコマンドであるため、主記憶上のデータベースを初期化する機能にとどめた。この後にセーブすることにより本当にデータベ

スが初期化される。

入力はデータベースのオープンにより行うが、管理を簡略化するために同時にオープンできるデータベースはただ1つのみとした。また、クローズコマンドによりセーブされるが、それとは別にインスタンスごとにセーブする機能を設けた。

(3)-c OPファイル

OPファイルからコマンドを読みこんで実行する機能はOPファイルを解釈して実行する処理が必要となるため、今回は実現しない。

(3)-d OPログファイル

OPログファイルは、全てのコマンド(引数を含む)とその実行結果(システムよりテキストベースでリアクションのあったもの)をファイルとして保持する機能とした。さらに実行結果には行頭に#をつけることでコマンド入力と区別し、OPファイル実行が可能となった時のための余地を残した。また、初期化コマンドまたはファイルが一定の大きさとなった時に自動的にログファイルのバックアップにコピーされて初期化される機能を用意した。

2. 3. 4 実現機能の概要

機能要件から必要とされたコマンドの定義とその機能、実現範囲、制限事項について記述する。各コマンドの実現方式の詳細は4. 5で述べる。なお、本プロトタイプを起動するコマンドは含まれていない。

① システム終了機能

〔コマンド名〕

dbexit

〔シンタックス〕

dbexit

〔引数〕

なし

〔意味〕

データベース機能の終了

② データベース初期化

〔コマンド名〕

dbinit

〔シンタックス〕

dbinit

〔引数〕

なし

〔意味〕

主記憶上のデータベースを初期化する。ファイル上の初期化を行うためには本コマンドの実行後、dbcolse を行わなければならない。

③ データベースオープン

〔コマンド名〕

dbopen

〔シンタックス〕

dbopen dbname

〔引数〕

dbname (データベース名称)

〔意味〕

データベースをオープンする

〔備考〕

同時にopenできるデータベースは1つだけである。

④ データベースクローズ

〔コマンド名〕

dbclose

〔シンタックス〕

dbclose dbname

〔引数〕

dbname (データベース名称)

〔意味〕

データベースをクローズする

〔備考〕

openされていないデータベースを指定した場合はエラーとなる

⑤ オブジェクト生成

〔コマンド名〕

objcreate

〔シンタックス〕

objcreate class_name object_name

〔引数〕

class_name (クラス名)

object_name (インスタンス名)

〔戻り値〕

インスタンスのID

〔意味〕

指定クラスのインスタンスを生成する。

インスタンス変数およびインスタンスのリンク付けは別コマンドで設定する。

〔備考〕

インスタンスに対しては、ガーベージコレクションは行わない。つまり、生成したインスタンスは陽に削除しない限り消滅しない。

⑥ オブジェクト格納

〔コマンド名〕

objsave

〔シンタックス〕

objsave class.name object_name

〔引数〕

class_name (クラス名)

object_name (インスタンス名)

〔意味〕

指定のインスタンスをセーブする。インスタンスにリンクづけられた関連インスタンスには影響しない。

⑦ オブジェクト削除

〔コマンド名〕

objdelete

〔シンタックス〕

objdelete class_name object_name

〔引数〕

class_name (クラス名)

object_name (インスタンス名)

〔意味〕

指定インスタンスを削除する。リンクづけられたインスタンスが存在する場合その関係を削除する。リンクづけられたインスタンスからさらに別のインスタンスにリンクしていてもそのインスタンスには影響しない。本コマンドの実行により、表示されているインスタンスにも自動的に再描画がかかる。また、削除できるインスタンスとして正規表現(*, ?)を利用できる。

⑧ オブジェクト検索

〔コマンド名〕

objsel

〔シンタックス〕

objsel class_name class(link) __name mode condition

〔引数〕

class_name (検索条件を設定するクラス名)

class(link) __name (検索対象となるクラスまたはリンクの名称)

mode(リンクモード 自クラスか隣のクラスかリンクか)

condition (検索条件)

〔意味〕

class(link) __nameで指定されたクラスに属するインスタンスまたは指定されたリンクに接続するインスタンスの中で、検索条件を満たすものを検索する。検索条件として指定できるのはclass __nameで指定されたクラスのインスタンス変数に対する範囲指定のみである。

検索条件の指定方法

instance __variable_name relation value

relation: = > < >= <=

value: 数値または文字、文字列

検索された結果はインスタンスの名称一覧として表示される。

(図形表示は別コマンドとする)

⑨ オブジェクト定義

〔コマンド名〕

objset

〔シンタックス〕

objset class__name object__name

〔引数〕

class __name (クラス名)

object__name (インスタンス名)

〔意味〕

指定インスタンスに対するインスタンス変数の値をセットする。

〔備考〕

具体的なインスタンスに対する値の設定はウィンドウを用いて行う。

⑩ オブジェクト属性表示

〔コマンド名〕

objdisp

〔シンタックス〕

objdisp class__name object__name

〔引数〕

class __name (クラス名)

object__name (インスタンス名)

〔意味〕

指定インスタンスの属性を表示する。

〔備考〕

指定インスタンスが図形データであった場合でも、属性データが表示される。

⑪ 形状表示

〔コマンド名〕

drawdisp

〔シンタックス〕

drawdisp class __name object__name

〔引数〕

class __name (クラス名)
object __name (インスタンス名)

〔意味〕

指定インスタンスの図形データを表示する。

〔備考〕

指定インスタンスに子供としてリンクづけられた全ての図形データが表示される。指定インスタンスに図形データインスタンスがリンクされていない場合は何も表示されない。

⑫ リンク生成

〔コマンド名〕

link

〔シンタックス〕

link class1 __name object1 __name class2 __name object2 __name logical __name

〔引数〕

class1 __name (クラス名)
object1 __name (インスタンス1名称)
class2 __name (クラス名)
object2 __name (インスタンス2名称)
logical __name (論理リンク名)

〔意味〕

インスタンス1をインスタンス2にリンク付けする。

リンク付けはインスタンス1、インスタンス2の双方から行われる。

本コマンドの実行により表示中のインスタンスにも自動的に再描画がかかる。

⑬ リンク削除

〔コマンド名〕

linkdel

〔シンタックス〕

linkdel link __name class1 __name object __name

〔引数〕

link __name(リンク名)
class __name (クラス名)
object __name (インスタンス名称)

〔意味〕

インスタンスをリンク名で指定されるリンクから削除する。

〔備考〕

リンク名、クラス名、インスタンス名には正規表現を使用できる。

⑭ リンク表示

〔コマンド名〕

linkdisp

〔シンタックス〕

linkdisp class __name object __name

〔引数〕

class __name (クラス名)
object __name (インスタンス 1 名称)

〔意味〕

指定インスタンスのリンク状態を表示する。

⑮ スキーマ更新

〔コマンド名〕

schupdate

〔シンタックス〕

schupdate scheme__file mode

〔引数〕

scheme __file (スキーマファイル名)
mode (インスタンスの再構築の有無)

〔意味〕

指定されたスキーマで新しいスキーマを生成し、modeにより全インスタンスの

再構築を行う。

〔備考〕

スキーマファイルが変更されていなくてもスキーマのupdate並びにインスタンスの再構築が行われる。本コマンドの実行により表示されているインスタンスにも自動的に描画がかかる。

⑩ 操作履歴初期化

〔コマンド名〕

oploginit

〔シンタックス〕

oploginit

〔引数〕

なし

〔意味〕

OPログファイルをバックアップにコピーし、初期化する。

〔備考〕

OPログファイルは正とバックアップが存在し、初期化又はファイルサイズが限界値に達した場合に正からバックアップにコピーされる。

本コマンドによって、それまでのバックアップファイルは無条件に廃棄される。

⑪ 操作履歴表示

〔コマンド名〕

oplogdisp

〔シンタックス〕

oplogdisp

〔引数〕

なし

〔意味〕

OPログファイルを表示する。

2. 4 ユーザインタフェース機能の概要

2. 4. 1 機能要件

ユーザインタフェース機能の要件は使いやすさ、操作性の観点から以下のように設定された。

- ・ データ操作言語の利用を画面から容易にマウス等を用いて直接行える。
- ・ 建築レイアウト図の表示や属性情報の表示・確認が行える。
- ・ 漢字の入出力が行える。

2. 4. 2 実現機能の概要

ユーザインタフェース機能の実現にあたっては、ウィンドウマネージャとして「Sun View」を用いた。

画面上は、大別すると、

- ・ コマンド選択ウィンドウ
- ・ パラメータ入力ウィンドウ
- ・ 図形表示ウィンドウ
- ・ 属性表示ウィンドウ
- ・ Shellウィンドウ

から構成される。

図2-4に画面の表示例を示す。

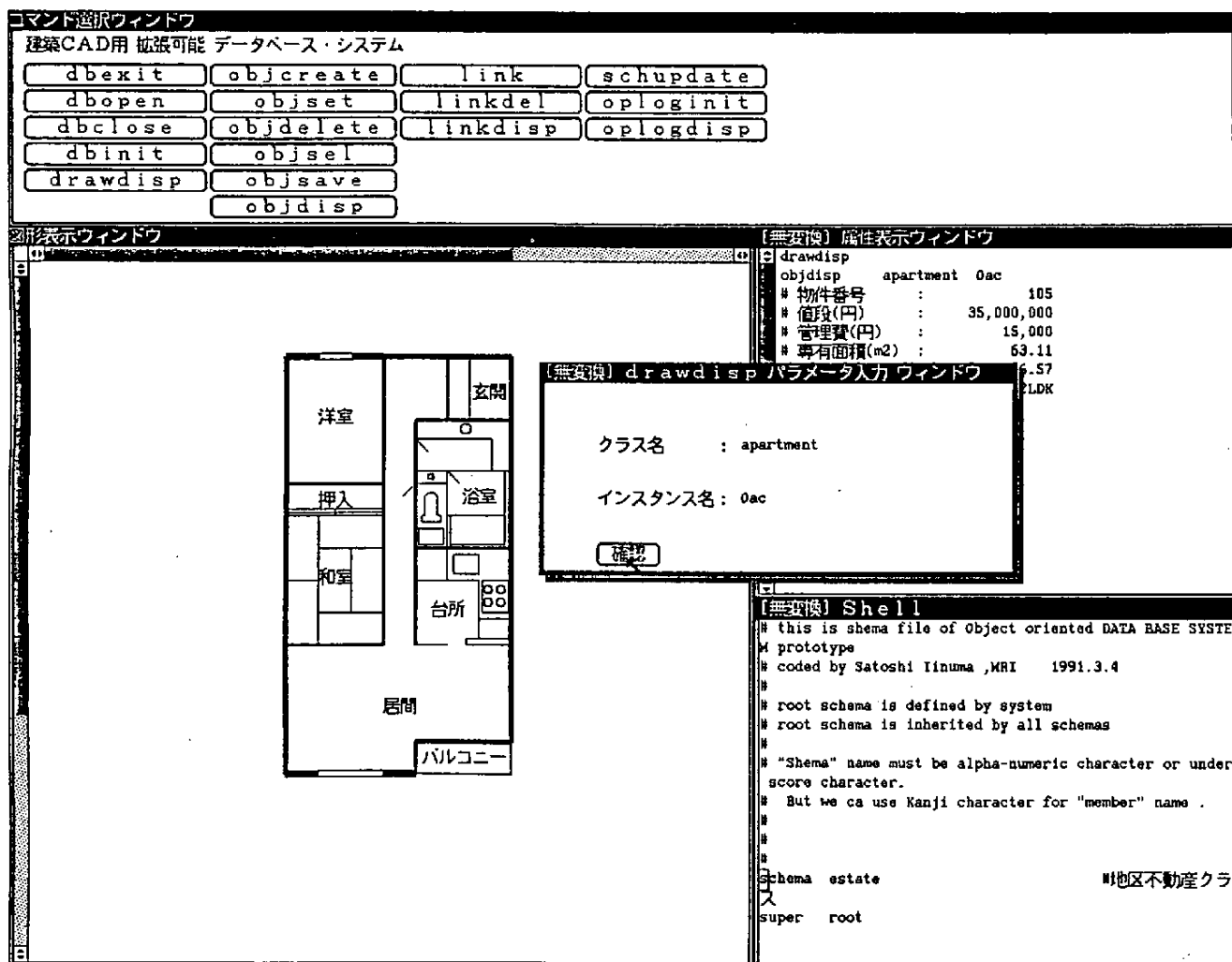


図 2-4 画面の表示例

3. データ構成

3. 1 建築CAD用のデータ構成

建築CAD用のデータベースに対する要件として、図面に対する処理・管理を中心に部品を表した図面と全体図との関係や、部品図同士の関係、図面と部品との関係などをうまく表現できることが挙げられている。この図形データをうまく扱いたいという要件に対して、従来のRDB（関係型データベース）を用いた方式だけでは、表現構造に多様性があり、またそのデータに対する取扱に関する情報も併せて取り扱う必要のある図形データを管理するには不十分であった。

本委託開発作業のプロトタイプ試作で作成した建築CAD用拡張可能データベースのモデル化に際しては、

- ・ 属性データと形状データ（表示）を独立して扱える。
- ・ 両者を表現するオブジェクトを関係づけるリンクを用いて図形データを扱う手続きを表現しやすくする。

等を前提とした設計を行った。

本委託開発における拡張可能データベースシステムの論理構成を図3-1に示す。

建築CAD用拡張可能データベースの論理構成は、マンションを対象とした、2次元の建築レイアウト図であり、以下のクラスから成り立つ。

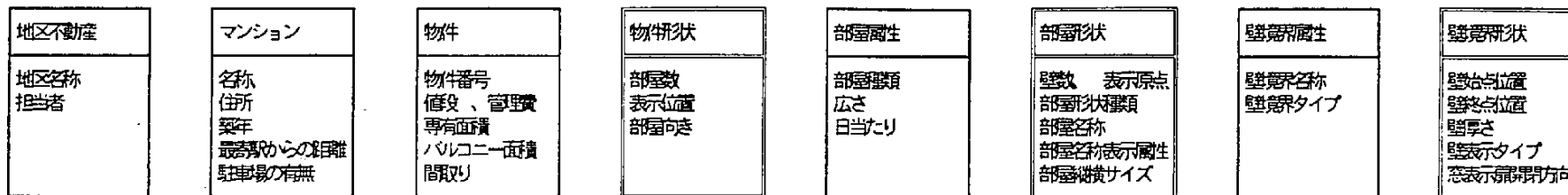
- ① 地区不動産クラス
- ② マンションクラス
- ③ 物件クラス
- ④ 物件形状クラス
- ⑤ 部屋属性クラス
- ⑥ 部屋形状クラス
- ⑦ 壁境界属性クラス
- ⑧ 壁境界形状クラス

このうち、地区不動産クラス、マンションクラス、物件クラス、部屋属性クラス、壁境界属性クラスが属性データを格納するためのクラスである。

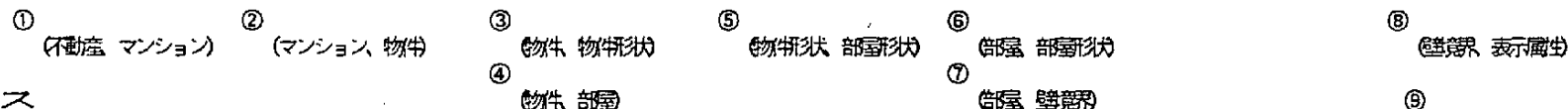
また、物件形状クラス、部屋形状クラス、壁境界形状クラスが形状データを格納するためのクラスである。

以降、3. 2に属性データの構成内容、3. 3に形状データの構成内容を示す。

クラス



リンク



インスタンス

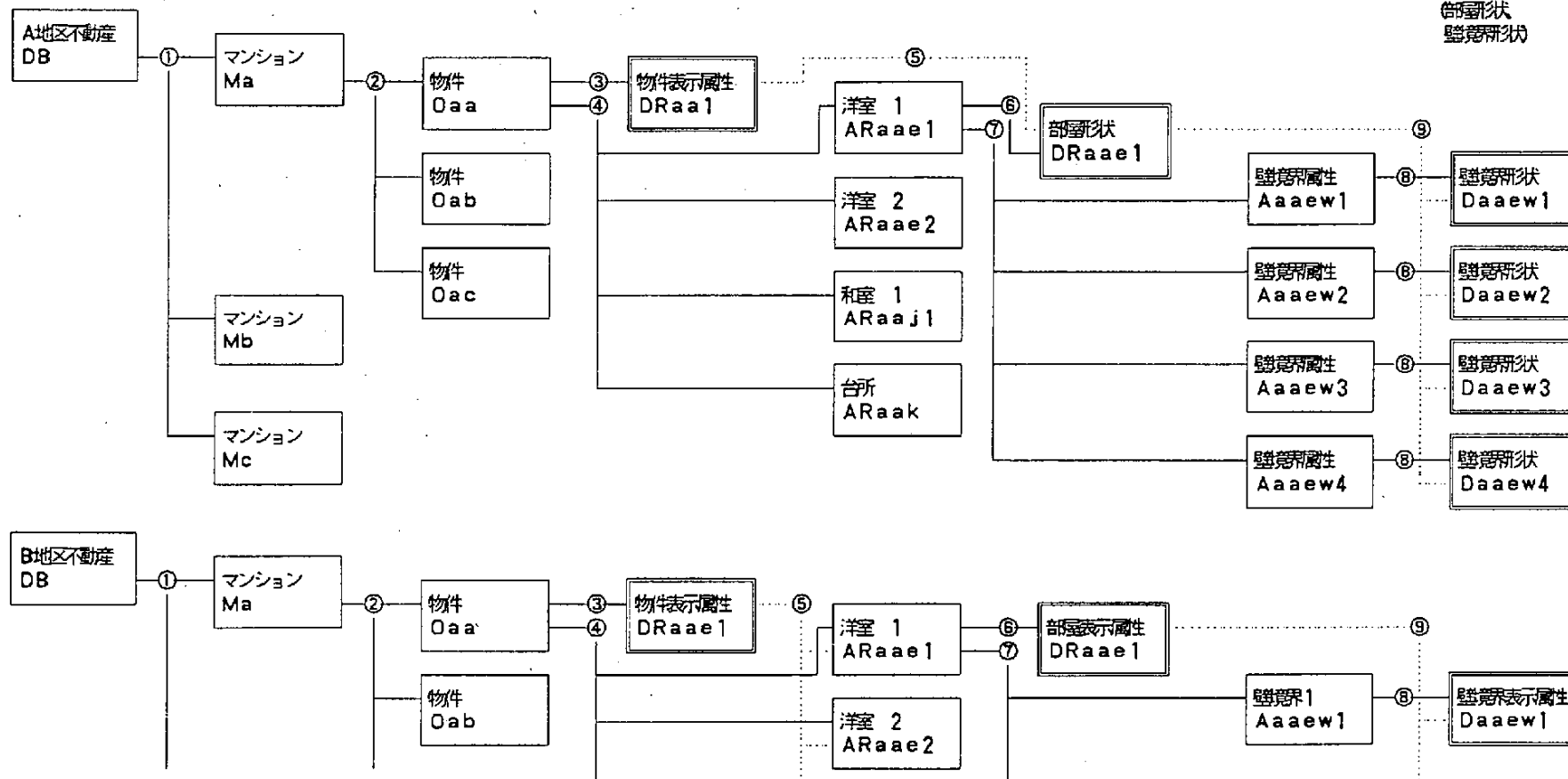


図3-1 データベースの論理構成

3. 2 属性データの構成

属性データを構成するクラスは、地区不動産クラス、マンションクラス、物件クラス、部屋属性クラス、壁境界属性クラスの5つのクラスである。属性データは、日本語を含む文字列情報と、実数、整数情報が格納される。格納内容を表3-1に示す。

表 3-1 属性データの格納内容 (その1)

クラス名称	インスタンス変数	型	サイズ (バイト)	意 味
① 地区不動産	地区名称	文字列	24	不動産対象地域名称
	担当者	文字列	24	データの管理者
② マンション	名称	文字列	24	例 マンション大船西
	住所	文字列	100	例 鎌倉市大船2-5
	築年	整数	4×2	西暦、築月
	最寄駅からの距離	実数	4	駅迄の距離 (m)
	駐車場の有無	文字列	2	例 有、無
③ 物件	物件番号	文字列	12	例 203号
	値段	実数	8	単位 円 金額表示
	管理費	実数	8	単位 円 金額表示
	専有面積	実数	4	単位 m ²
	バルコニー面積	実数	4	単位 m ²

表 3—1 属性データの格納内容 (その2)

クラス名称	インスタンス変数	型	サイズ (バイト)	意 味
③ 物件 (つづき)	間取り	文字列	8	2LDK、3LDK等
④ 部屋属性	部屋種類	文字列	8	例 和室、洋室、台所
	広さ	実数	4	単位 m ²
	日当たり	文字列	12	例 best good worst
⑤ 壁境界属性	壁境界名称	文字列	24	壁につけられた名称
	壁境界タイプ	文字列	8	例 中厚、窓、ふすま

3. 3 形状データの構成

形状データを構成するクラスは、物件形状クラス、部屋形状クラス、壁境界形状クラスの3つのクラスである。

形状データには、日本語を含む文字列情報と、実数、整数情報に加えて、レイアウト図を表示するための位置・形状情報（直線、円、円弧、文字列）が格納される。格納内容を表3-2に示す。

表 3-2 形状データの格納内容 (その1)

クラス名称	インスタンス変数	型	サイズ (バイト)	意 味
① 物件形状	部屋数	整数	4	部屋の数
	表示位置	整数	8 × 2	画面上の座標
	部屋向き	文字列	4	例 N E S W NE ES SW
② 部屋形状	壁数	整数	4	壁の数
	表示原点	整数	8 × 2	物件中心からの座標
	部屋形状種類	文字列	8	部屋表示モード A B
	部屋名称	文字列	2 4	例 和室、リビング
	部屋名称表示属性	整数	8 × 2	名称の表示開始位置
	部屋縦横サイズ	整数	8 × 2	ピクセル単位
③ 壁境界形状	壁始点位置	整数	8	ピクセル単位
	壁終点位置	整数	8	ピクセル単位

表 3-2 形状データの格納内容 (その2)

クラス名称	インスタンス変数	型	サイズ (バイト)	意 味
③ 壁境界形状 (つづき)	壁厚さ	整数	8	壁の厚さ
	壁表示タイプ	文字列	12	壁境界タイプと同じ
	窓扉開閉方向	文字列	12	窓の開閉方向 (内外)

更に、属性情報、形状情報が格納されているインスタンス間を関係づけるためのリンクとして、以下の9種類がある。

- ① (不動産、マンション)
- ② (マンション、物件)
- ③ (物件 物件形状)
- ④ (物件、部屋)
- ⑤ (物件形状、部屋形状)
- ⑥ (部屋、部屋形状)
- ⑦ (部屋、壁境界)
- ⑧ (壁境界、表示属性)
- ⑨ (部屋形状、壁境界形状)

4. プロトタイプシステムの概要

4. 1 システムの概要

本委託開発における拡張可能データベースシステムの全体構成を図4-1に示す。

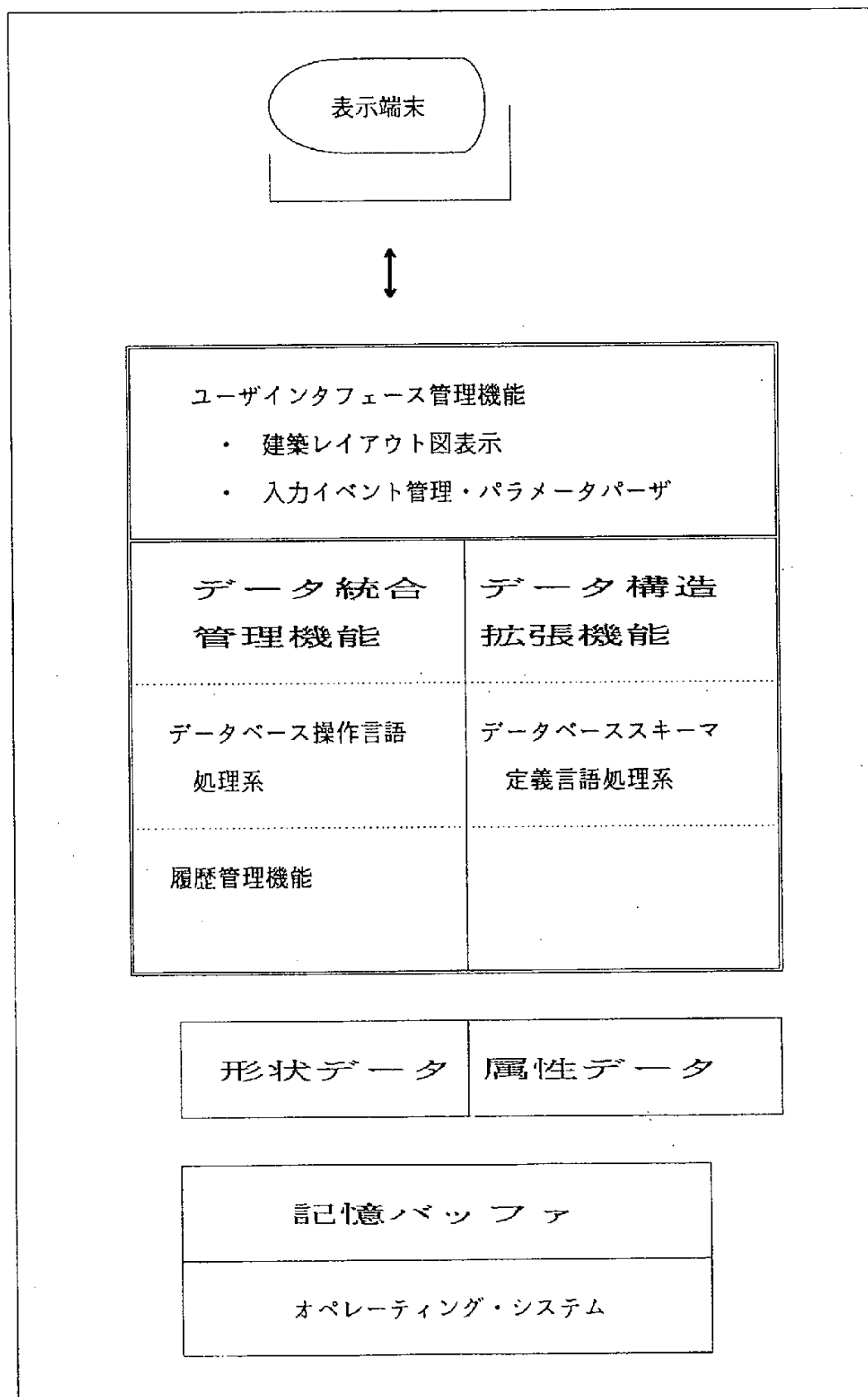


図4-1 拡張可能データベースのシステム構成

4. 2 開発環境とハードウェア構成

詳細設計に基づき、ワークステーション Sparc Station-1 上 (OS は、UNIX 系の Sun-OS 4. 0 3 + JLE 1. 0 3) で、各モジュールの製作を行った。

4. 3 ソフトウェア構成

利用 OS 機能と記述言語について述べる。

① 利用 OS 機能

グラフィック (ウィンドウシステム及びグラフィックライブラリ) には、

「SunView」

を用いている。以下に、主な使用ライブラリを示す。

suntool		sun 上のツールライブラリ
mle		日本語ライブラリ
sunwindow		ウィンドウ管理用ライブラリ
pixrect		描画ライブラリ
m		数学ライブラリ

② 記述言語

クラス定義、継承等を持つ開発言語による実現も想定し、C++ や g++ によるインプリメントが検討されたが、現在使用されているワークステーションのほぼ全てで C 言語が利用可能なことから、移植性を重視し、今回は C 言語による方式を採用した。

4. 4 操作イメージ

本システムを用いた作業操作の概要を図4—2に示す。

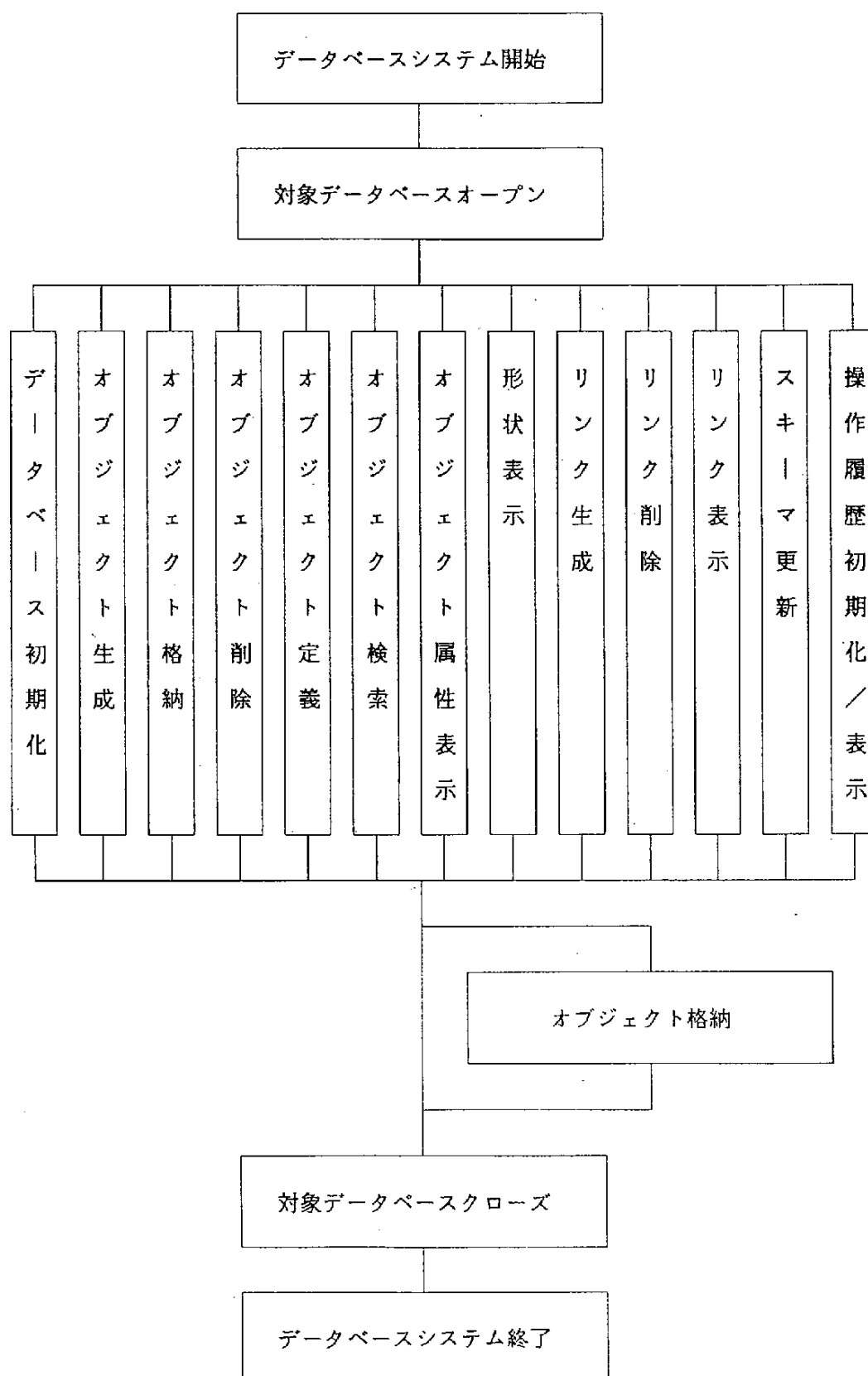


図 4—2 作業操作の流れ

4. 5 実現方式

プロトタイプシステムにおける実現方法に関して、以下の4点について説明する。

- (1) ファイル構成
- (2) データ構成
- (3) 全体制御
- (4) 操作言語の実現方式

4. 5. 1 ファイル構成

プロトタイプシステムで使用するファイルについて、そのフォーマットおよび格納項目について説明する。

データベースを構成するデータバッファは、ファイルとして記憶される。

各データバッファそれぞれに対して1つのファイルを用いるため、1つのデータベースは5つのファイルによって構成されることになる。

この5つのファイルにはそれぞれ異なった拡張子を与え、またファイルの先頭の8バイトに、そのファイルがどのデータバッファを記録したものであるかを識別するためのキーワードを設定している。

データバッファとファイル名等との対応関係は以下の様にする。

<データバッファ>	<ファイル名>	<識別キーワード>
(1) クラス管理テーブル	(DB名). c m t	CLSMTBL ¥0
(2) インスタンス管理テーブル	(DB名). o m t	OBJMTBL ¥0
(3) リンク管理テーブル	(DB名). l m t	LNKMTBL ¥0
(4) クラスインスタンス雛型データ	(DB名). c m c	CLSMOCK ¥0
(5) インスタンスデータ	(DB名). o b r	OBJDATA ¥0

これらのファイルの記憶形式を以下に示す。

(1) クラス管理テーブル

先頭: 00h	char[8] ファイル識別用キーワード	CLSMTBL ¥0
08h	long	クラスの総数

0Ch	クラス管理データ 0
+00h long	クラスID
+04h char[15]	クラス名 (14文字)
+13h long	クラスインスタンス型データのサイズ
	クラス管理データ 1
	:
	:
	:
	クラス管理データ n

(2) インスタンス管理テーブル

先頭: 00h	char[8] ファイル識別用キーワード OBJMTBL ¥0
08h	long インスタンスの総数
0Ch	インスタンス管理データ 0
+00h long	インスタンスID
+04h char[15]	インスタンス名 (14文字)
+13h long	クラスID
	インスタンス管理データ 1
	:
	:
	:
	インスタンス管理データ n

(3) リンク管理テーブル

先頭: 00h	char[8] ファイル識別用キーワード LNKMTBL ¥0
08h	long リンクの総数

0Ch	リンク管理データ 0
+00h long	リンク I D
+04h char[15]	リンク名 (1 4 文字)
+13h long	親インスタンス I D
	リンク管理データ 1
	:
	:
	:
	リンク管理データ n

(4) クラスインスタンス雛型データ

先頭: 00h	char[8] ファイル識別用キーワード CLSMOCK ¥0
08h	long クラスの総数 クラス管理テーブルでの値と等しい
0Ch	クラスインスタンス雛型データ 0
	クラスインスタンス雛型データのメモリマップと等しい
	クラスインスタンス雛型データ 1
	:
	:
	:
	クラスインスタンス雛型データ n

(5) インスタンスデータ

先頭: 00h	char[8] ファイル識別用キーワード OBJDATA ¥0
08h	long インスタンスの総数 インスタンス管理テーブル値と等しい
0Ch	インスタンスデータ 0
+00h long	クラス I D
+04h	メンバ変数領域

	変数値 0
	変数値 1
	:
	:
	:
	変数値 n
	インスタンスデータ 1
	:
	:
	:
	インスタンスデータ n

4. 5. 2 データ構成

プロトタイプシステムを実現するにあたって使用しているデータベース管理用の内部データ、および各インスタンスのメモリ上の展開方法ならびにデータ項目について説明する。

本データベース上においてクラス概念を表現するために、5種類のデータバッファを使用した。

- (1) クラス管理テーブル
- (2) インスタンス管理テーブル
- (3) リンク管理テーブル
- (4) クラスインスタンス雛型データ
- (5) インスタンスデータ

データベース上における各インスタンスに対して、インスタンスデータは1対1に対応し、インスタンスにおけるメンバ変数、およびリンク情報などはすべてこの領域に記憶されることになる。

このインスタンスデータを生成するために、クラスインスタンス雛型データというものを定義する。クラスインスタンス雛型テーブルとは、各クラスに対するインスタ

ンスデータの初期データを示したものであり、新たなインスタンスデータの生成は、これをコピーすることによって行なうことが可能となる。

クラス管理テーブルとは、クラスインスタンス難型データを管理するためのものであり、各クラスの名称、難型データのサイズなどの情報が、記録されている。各クラス難型データには、クラスIDナンバーが与えられており、内部ではこのIDナンバーによってクラス難型データを管理している。このクラスIDナンバーによりこのテーブル上の情報を取得し、クラス難型データ領域へアクセスすることになる。

同様にインスタンス管理テーブルは、インスタンスデータを管理するためのものであり、各インスタンスの名称、実際のインスタンスデータへのポインタなどの情報が記録されている。クラス管理テーブルと同様に、各インスタンスに割り振られているインスタンスIDナンバーによりこのテーブル上の情報を取得し、インスタンスデータ領域へアクセスを行なうことになる。

リンク管理テーブルとはインスタンス間のリンク情報を管理するものであり、これにはリンク名称、リンク関係における親インスタンスのインスタンスIDなどの情報が記録されている。このリンクデータにもIDナンバーが割り振られており、インスタンスデータ領域に設定されているリンク情報もIDナンバーによるものである。

次にこれらのメモリマップを示す。

(1) クラス管理テーブル

先頭: long クラス数	
クラス管理データ (クラスID: 0)	
char[15]	クラス名 (14文字)
long	クラスインスタンス難型データのサイズ
char*	クラスインスタンス難型データへのポインタ
short	データ存在フラグ
(このフラグが立っていない場合は、このIDの難型データは使用されることがないか、またはすでに削除済みのデータであることを示す。以下同様)	

クラス管理データ (クラスID: 1)

:

:

:

クラス管理データ (クラスID: n)

(2) インスタンス管理テーブル

先頭: long インスタンス数

インスタンス管理データ (インスタンスID: 0)

char[15] インスタンス名 (14文字)

long クラスID

char* インスタンスデータへのポインタ

short データ存在フラグ

インスタンス管理データ (インスタンスID: 1)

:

:

:

インスタンス管理データ (インスタンスID: n)

(3) リンク管理テーブル

先頭: long リンク数

リンク管理データ (リンクID: 0)

char[15] リンク名 (14文字)

long 親インスタンスID

short データ存在フラグ

リンク管理データ (リンクID: 1)

:
:
:
リンク管理データ (リンクID : n)

(4) クラスインスタンス雛型データ

先頭:	long	クラスID
	long	メンバ変数の数
メンバ変数管理領域		
変数属性値	0	
	long	変数ID
	char[15]	変数名 (14文字)
	long	変数領域への先頭からのオフセット値
	long	変数領域のサイズ
	short	変数の型
変数属性値	1	
	:	
	:	
	:	
変数属性値	n	
メンバ変数領域		
変数初期値	0	
変数初期値	1	
	:	
	:	
	:	
変数初期値	n	

(5) インスタンスデータ

先頭: long クラス I D
long メンバ変数の数

メンバ変数管理領域

ここまではクラスインスタンス雛型データと同じ

メンバ変数領域

変数値 0

変数値 1

:

:

:

変数値 n

(注) インスタンス I Dおよびクラス名、インスタンス名、およびリンク情報はメンバ変数の一種 (ルートインスタンスクラス変数) として、メンバ変数領域内に保持されている。

ルートインスタンス変数

long インスタンス I D

char[15] インスタンス名

char[15] クラス名

system システム管理領域 (リンク情報など)

4. 5. 3 全体制御

プロトタイプシステムにおけるプログラムの制御方式について、イベントハンドリングを中心として説明する。

イベントハンドリングは、図4-3に示すように以下の3つの処理から成り立つ。

① イベント制御

SunviewのWindow_main_loopを用いて制御を行う。

② イベント検出

Window_main_loopによって検出されたイベントの種類に応じて、ユーザインタフェース関数への制御を移している。

③ イベント処理

ユーザインタフェース関数は、ユーザ入力を取り込み、制御情報としてコマンド処理関数へ引きわたしている。

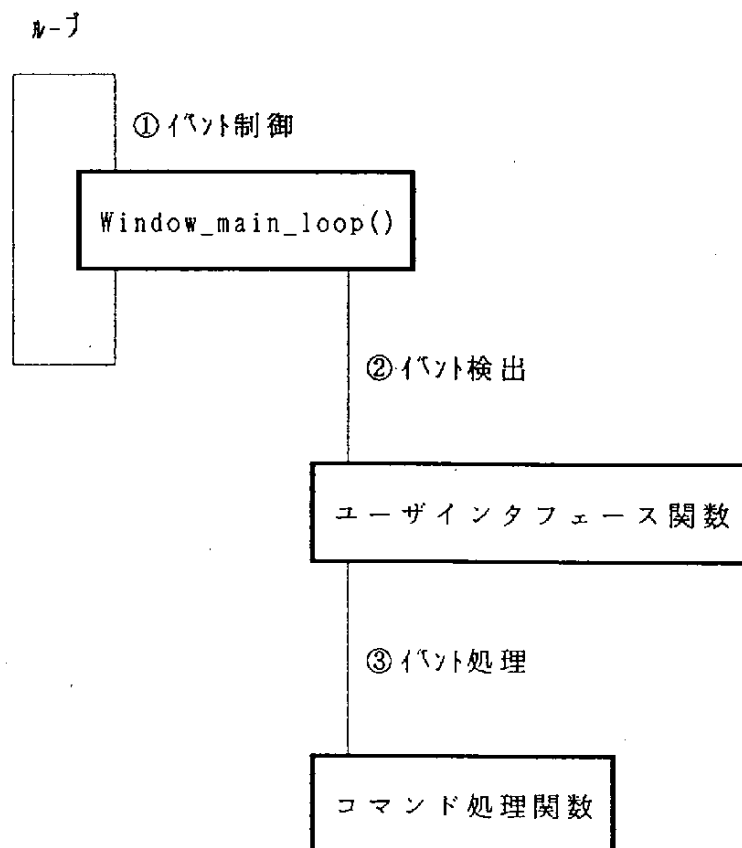


図4-3 イベント処理の概要

4. 5. 4 操作言語の実現方式

各モデル操作コマンドの実現方式について説明する。

前に述べたデータバッファに対する処理の内、主なものの実現方法を述べる。

ここでは、以下の処理を取り上げる。

- (1) `schupdate`
- (2) `objcreate`
- (3) `objset`
- (4) `link`
- (5) `objsel`

(1) `schupdate`

この処理は指定されたスキーマファイルを解釈し文法が合っているかを調べ、さらにフラグの値によっては、解釈したスキーマファイルより新しいクラスの生成を行なうものである。ここで用いられるスキーマファイルとは、各クラスを定義するクラス定義文が記述されたファイルのことである。

このスキーマ定義文によって、各クラスにおけるクラス名、上位クラス名、クラスのメンバ変数、およびその初期値についての設定を行なうことができる。

新しいクラスの生成処理としてシステムは、スキーマファイルに記述されているクラス定義文を解釈し、クラスインスタンス雛型データを作成することになる。

スキーマ定義文の解釈は次のようにして行なう。

まずスキーマファイルからクラス定義文を一行分読み込む。

次に定義文の文頭からコメント記号である '`#`' を探していき、もし存在した場合は '`#`' を `NULL` に置き換える。この処理によって '`#`' 以降の文は無視されコメントとなるのである。

そして次に定義文中の先頭の単語を取り出し、予約語である `schema`, `super`, `member` と比較する。

このときいずれとも一致しない場合はエラーとする。

一つのクラスデータの定義には、クラス名を定義するための `schema` 文が1行、上位クラスを定義するための `super` 文が1行、そしてクラスのメンバ変数

の初期値を定義するための member 文がメンバ変数の数だけ必要となる。

これらは schema, super, member の順に記述されなければならない。

これより、1つのクラス定義文は schema 文からファイルの終わりまたは次に出現する schema 文の直前までとなる。

クラスデータの再構築フラグが立っていない場合、ここまでの処理が正常に行なわれたかどうかを表示する。またこの再構築フラグが立っている場合、さらに以下のような処理を行なう。

こうして取得した各スキーマ定義データからクラスインスタンス雛型データを生成するために、次の様な処理を行なう。

1. メンバ変数定義文より本クラスにおいて生成されるメンバ変数のデータ領域として必要なメモリ量を計算する。
2. 上位クラス定義文とクラス管理テーブルより上位クラスが持っているメンバ変数の定義に必要なデータ領域の大きさを取得し、1のメモリ量と合計することによって、クラスインスタンス雛型データ領域のメモリ量を求める。
3. クラスインスタンス雛型データに必要なメモリ領域を確保する。
4. クラス管理テーブルの値を以下のようにして設定する。
 - (a) クラス管理データのデータ存在フラグを順次参照し、データが存在しない空のクラス管理データブロックを探し、クラスIDを定める。
 - (b) クラス管理データをクラス名定義文などより設定する。
5. クラスインスタンス雛型データを以下のように設定する。
 - (a) クラスID、メンバ変数の数を設定する。
 - (b) 上位クラスのメンバ変数のデータを、データ領域のメモリ上に複写し、対応するメンバ変数管理領域値を設定する。
 - (c) 本クラスのメンバ変数定義文より、メンバ変数の初期値をデータ領域上に書き込み、対応するメンバ変数管理領域値を設定する。

以上に示した処理は上位クラスから順に処理する必要がある。

これは、クラスの順位テーブルを上位クラス名の定義文より作成し、そのテーブル順にクラス生成処理を行なうことで実現することができる。ただし、“スキーマファイルはトップダウン式に記述する”という制約を与えれば、こうしたクラスの順位評価処理は不要となる。

(2) objcreate

この処理は指定したクラスのインスタンスを指定した名前で、新たに生成するものである。

ただし、このインスタンス名は同一クラス内において唯一のものでなければならない。

ここでインスタンスの生成処理としてシステムは、前述の `schupdate` において生成したクラスインスタンス雛型データをコピーし、その内部にデータを設定することになる。

1. クラス管理テーブル中から、生成すべきインスタンスのクラス名を探す。
2. クラス管理テーブルより、インスタンス雛型データの大きさを取得し、その大きさのインスタンスデータ領域を確保する。
3. インスタンス雛型データをインスタンスデータ領域にコピーする。
4. インスタンス名、クラス名、インスタンスIDをメンバ変数の一種としてインスタンスデータ領域に設定する。

インスタンス雛型データは、各メンバ変数の初期値が設定された状態のインスタンスデータと同じ形となっているため、これをコピーすることによって、自動的に初期化されたインスタンスデータを生成することができるのである。

以上の処理によって新しいインスタンスの生成を行なうことができる。

(3) objset

この処理は指定したインスタンス内部のメンバ変数の値を変更するものである。

この処理は、今までのインスタンス内部のメンバ変数の値を取得する部分と、変更値をメンバ変数に設定する部分の、2つに分けることができる。

実際の処理は次の様になる。

- 1) `objset1` を呼び出す。
(メンバ変数値を取得)
- 2) メンバ変数値の変更処理を行なう。

3) o b j s e t 2 を呼び出す。

(メンバ変数値を変更)

前半部の処理は次の様になる。

1. インスタンス管理テーブルより指定されたインスタンス名を探し、インスタンスデータへのポインタを取得する。
2. メンバ変数の数を取得し、この数だけ以下の処理を繰り返しメンバ変数値を取得する。
 - (a) インスタンスデータの先頭のポインタに、各変数領域へのオフセット値を加え変数領域へのポインタを得る。
 - (b) 変数領域より変数値を得る。
 - (c) 変数の型と変数領域のサイズに注意して、変数値を文字による表記に変換する。

(例) (long)100 --> (char[]) "100"
 - (d) 文字表記した変数の先頭のポインタを返す。

こうして得た文字表記された変数を変更するイベント処理を行ない、その結果を用いて次の様にして後半部の処理を行なう。

1. メンバ変数の数だけ以下の処理を繰り返す。
 - (a) 変数の文字表記を、変数の型に注意して変数値に変換する。

(例) (char*) "200" --> (long) 200

このとき変換不能な文字 (例えば long における数字以外の文字) があった場合、入力誤りとしエラーコードを返す。
 - (b) 変数領域へ変数値を格納する。

これによりインスタンスのメンバ変数の変更を行なうことが可能となる。

(4) l i n k

この処理は指定したインスタンス間にリンクを張るためのものである。

実際には、インスタンスのメンバ変数におけるルートオブジェクト変数のシステム管理領域値を操作することによって、リンク処理を行うことになる。

システム管理領域は次の様な構造体になっている。

system 構造体

```
{  
    long    親へのリンクの数  
    親へのリンクのデータ [m個]  
    {  
        long    リンク I D  
        long    親のインスタンス I D  
        long    兄のインスタンス I D  
        long    弟のインスタンス I D  
        short   リンクデータ削除フラグ  
    }  
  
    long    子へのリンクの数  
    子へのリンクのデータ [n 個]  
    {  
        long    リンク I D  
        long    リンク中の子供の数  
        long    長男のインスタンス I D  
        long    末っ子のインスタンス I D  
        short   リンクデータ削除フラグ  
    }  
  
    short   データ変更フラグ  
    short   サイズ変更フラグ  
}
```

これを説明するために、リンク形式について述べる。

本データベースにおけるインスタンスのリンク形式を部分的に取り出すと次のようになっている。

+---- 子インスタンス 1

|

|

親インスタンス -----+----- 子インスタンス 2

|
|

+----- 子インスタンス 3

ここで子インスタンスは、親インスタンスのIDの他に、それぞれにおける兄インスタンスと、弟インスタンスのIDを持つことによって、リスト形式を成している。

そして親インスタンスは、子インスタンスリストの長子インスタンスと末子インスタンスのIDを持つことによって、リンク処理を行なうことができるのである。

具体的なリンク処理は次の様なものとなる。

1. 指定された親インスタンス名と子インスタンス名からインスタンス管理テーブルより、インスタンスIDを取得する。
2. 指定されたリンク名がすでに存在するかどうかをリンク管理テーブルより調べ、次の様に処理を行なう。
 - (a) 同名のリンクが存在しないならば、新規にリンクを登録し、リンクデータを設定する。
 - (b) 同名のリンクが存在するならばその親インスタンスIDが一致するかどうかを調べ、親インスタンスが異なるのに同じリンク名を付けようとしている場合はエラーコードを返す。
3. 各システム管理領域の設定を行なう。このとき、子インスタンスは末子インスタンスとしてリンク付けされる。

こうしてリンク処理を行なうことができる。

(5) o b j s e l

この処理は、指定した条件を満たすインスタンスを検索するものである。

ここでの条件とは、指定されたインスタンスクラスにおけるメンバ変数値によるものであるため、条件の指定としてメンバ変数の所属するインスタンスクラスと、実際の

条件式が必要となる。

(例) クラス名 : room

条件式 : 広さ > 15

検索結果出力法として次の様な3つのモードが存在する。

- a) 条件に一致したインスタンスの名称を出力する。
- b) 出力インスタンスのクラス名を指定することによって、条件に一致したインスタンスにリンク付けされているインスタンスの名称を出力する。
- c) 出力インスタンスへのリンク名を指定することによって、条件に一致したインスタンスにリンク付けされているインスタンスの名称を出力する。

具体的には次の様な処理を行なうことになる。

1. インスタンスクラス名よりクラスIDをクラス管理テーブルより取得する。
2. インスタンス管理テーブルより、1. で求めたクラスのインスタンスデータを取得し、以下の様な処理を行なう。
 - (a) 条件式のメンバ変数値を取得する。
 - (b) 条件と照らし合わせ、条件に一致するかどうか調べる。
 - (c) 検索出力モードによって、それぞれの形式によって出力を行なう。

以上の処理によってインスタンスの検索を行なうことになる。

5. 実行方法

5. 1 実行方法の概略

本システムの機能構成の概略を、図5-1に示す。

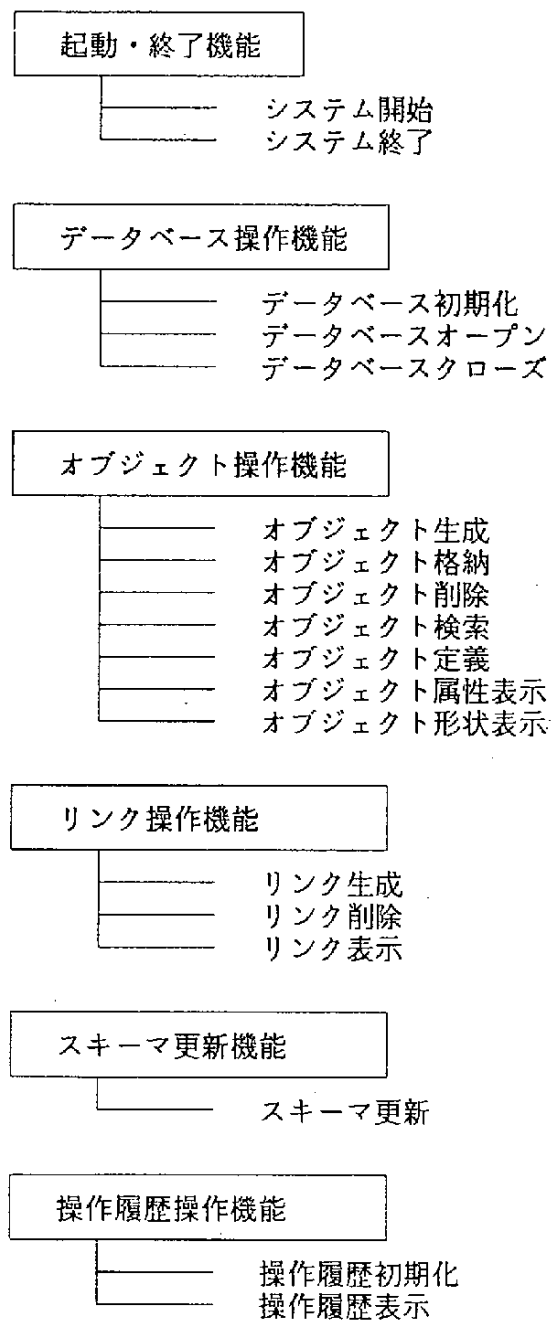


図5-1 拡張可能データベースの機能構成

以下、5. 2～5. 17に各機能の操作方法と実行事例を示す。

5. 2 システム開始機能

コ マ ン ド 名 称		d b s t a r t			
機 能 概 要	拡張可能データベースシステム機能の開始を行う。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	な し			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		な し			
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ d b s t a r t ()			
	引数名	I/O	型	サイズ	備 考
	な し				
備 考	0 : 正常終了 - 1 : ログファイルがオープンできない				

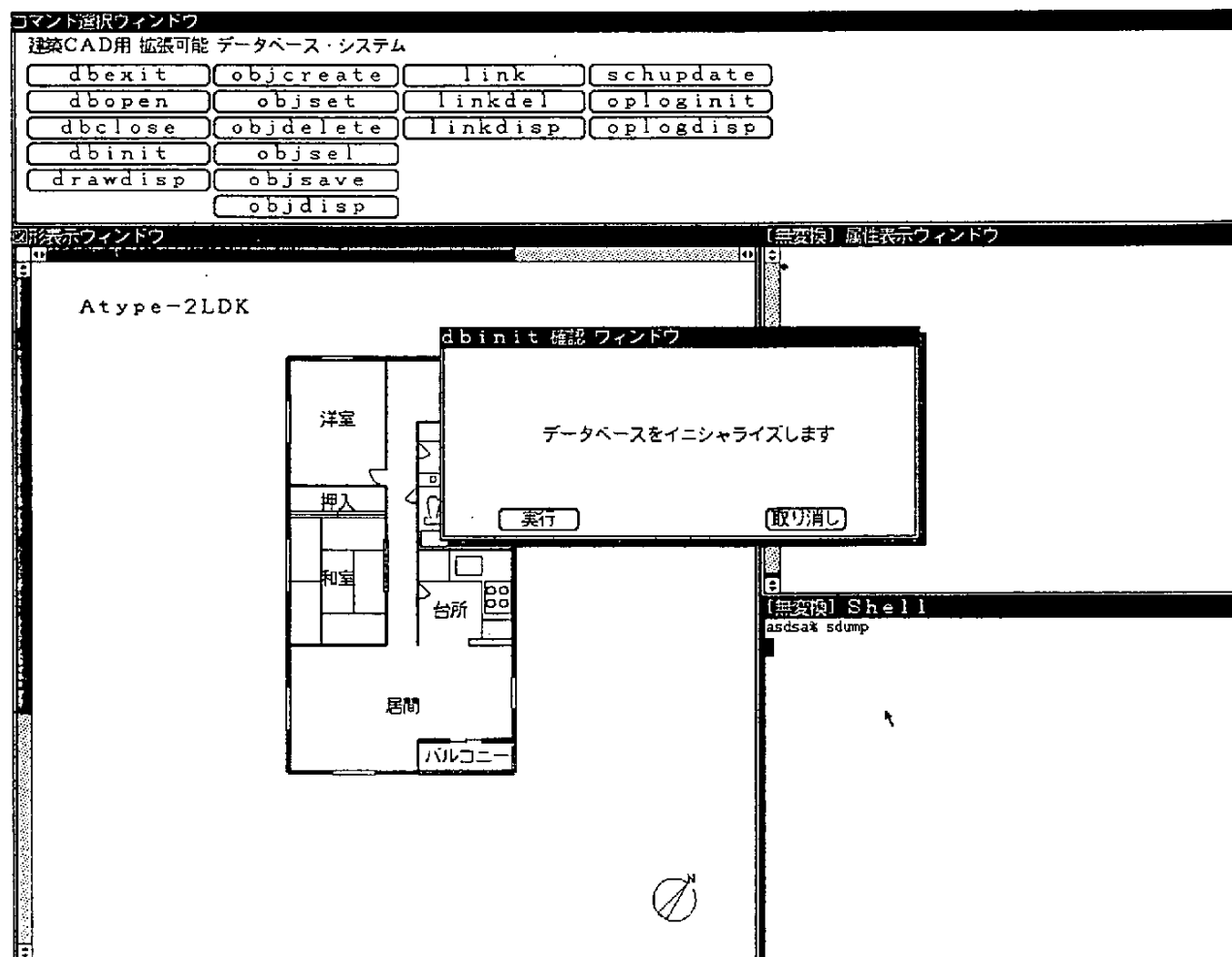
5. 3 システム終了機能

コマンド名称		dbexit			
機能概要	拡張可能データベースシステム機能の終了を行う。				
操作インタフェース	ボタン	dbexit ボタンをクリック			
	キーボード	内 容	サイズ	備 考	
		な し			
言語インタフェース	定義	int co_dbexit()			
	引数名	I/O	型	サイズ	備 考
	な し				
備考	0 : 正常終了 - 1 0 : データベース機能が開始していない				

5. 4 データベース初期化機能

コ マ ン ド 名 称		d b i n i t			
機 能 概 要	データベースを初期化する。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	dbinit ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		な し			
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ d b i n i t ()			
	引数名	1/0	型	サイズ	備 考
	な し				
備 考	0 : 正常終了 - 1 0 : 初期化失敗				

実行例 (画面内容)



動作説明 (dbinit-1)

(図形表示ウィンドウには0acの物件が表示されている)

・ dbinitボタンクリック

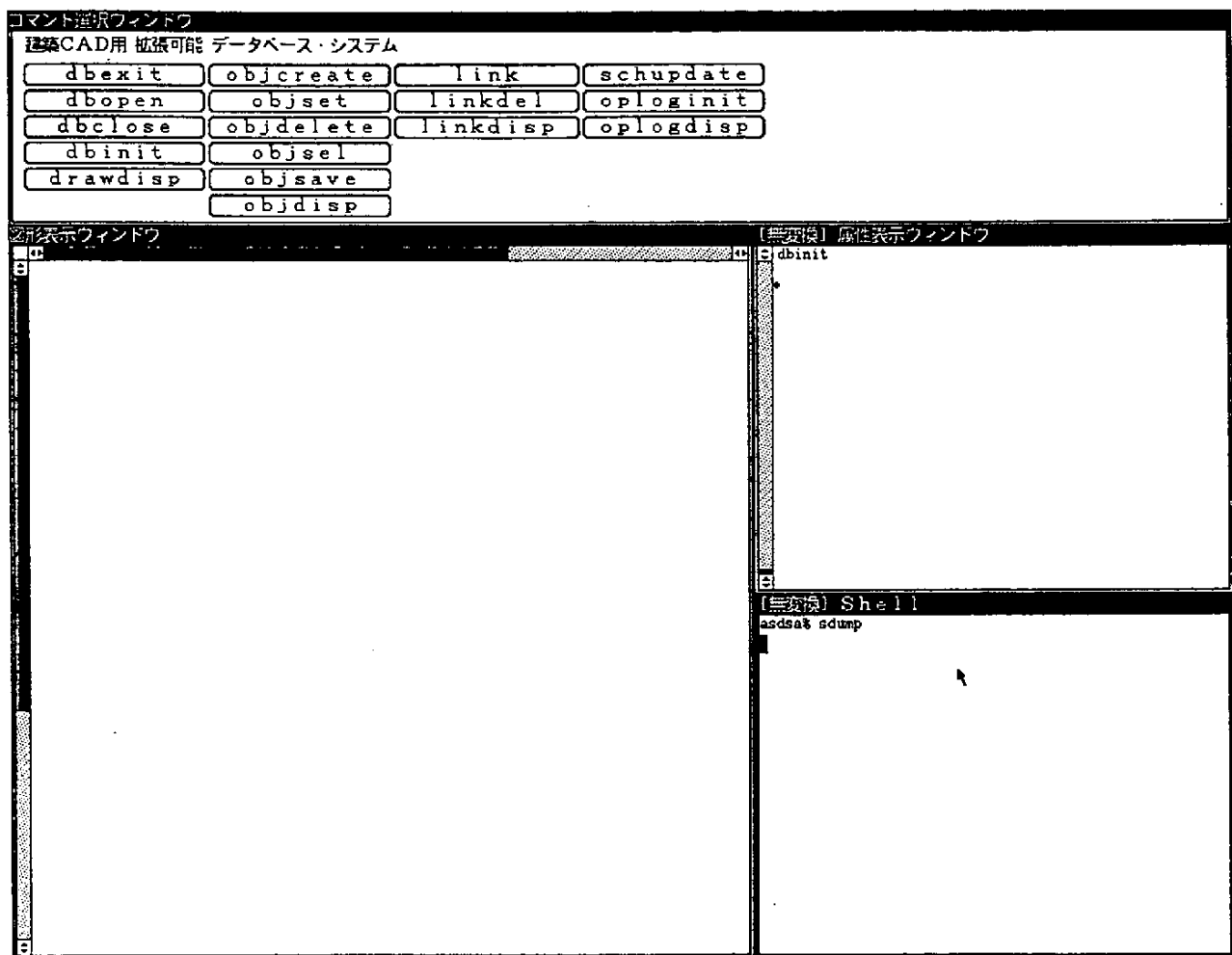
→ 確認ウィンドウが表示される

“データベースをイニシャライズします”

確 認

取 り 消 し

実行例 (画面内容)



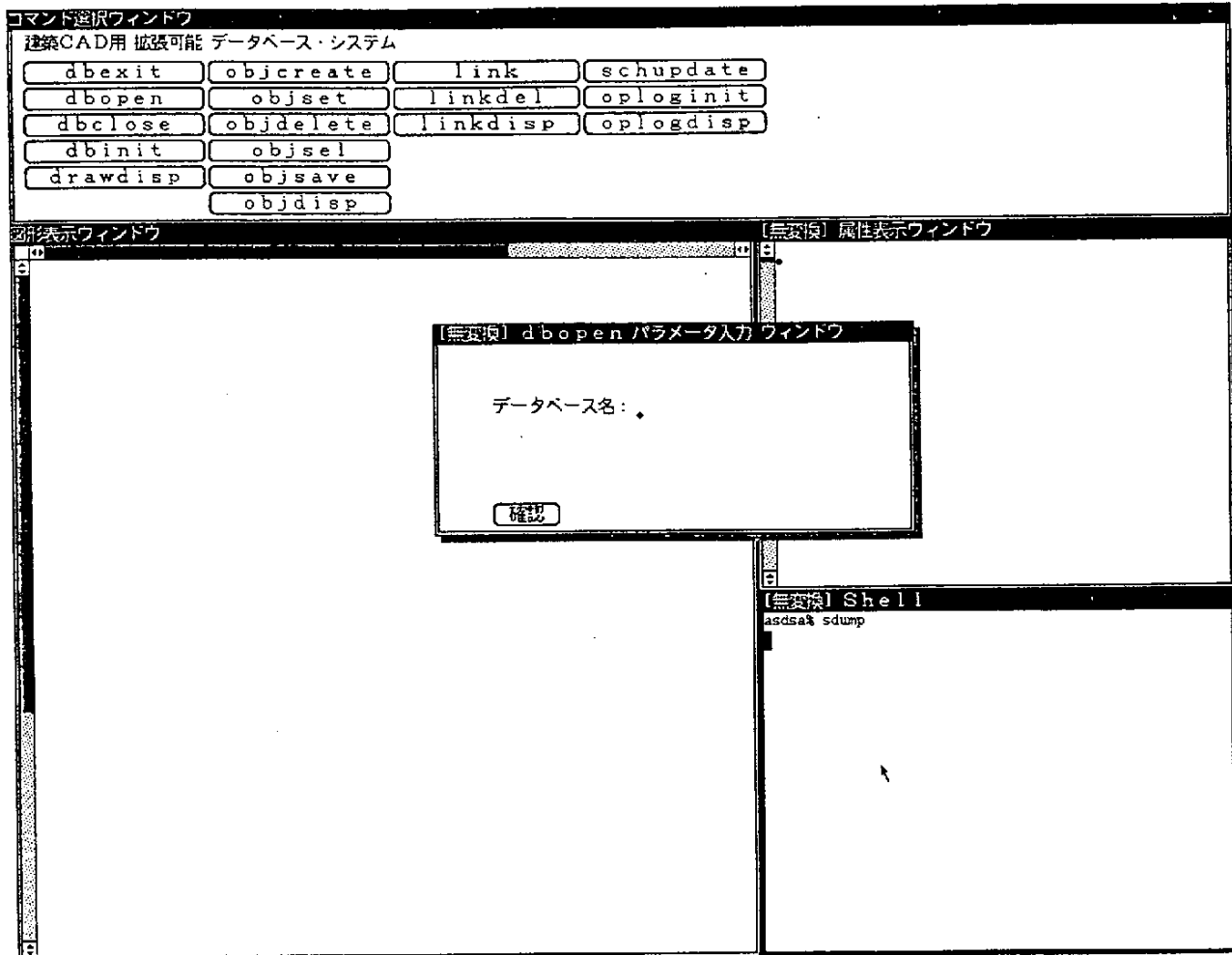
動作説明 (dbinit-2)

- 確認ウィンドウの「確認」をクリックする
- 属性表示ウィンドウにエコー表示
“dbinit”
- 図形表示ウィンドウがクリアされる

5. 5 データベースオープン機能

コ マ ン ド 名 称		d b o p e n			
機 能 概 要	指定されたデータベースをオープンする。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	dbopen ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		データベース名	2 4	漢字で最大12文字	
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ d b o p e n (d b n a m e)			
	引数名	1/0	型	サイズ	備 考
	dbname	1	char*	2 4	データベース名
備 考	0: 正常終了 -1: すでに別のDBがオープンされている -2: すでに同じDBがオープンされている -3: 指定DB名が存在しない -4: DBの形式がおかしい -10: -100:メモリアロケート失敗				

実行例 (画面内容)



動作説明 (dbopen-1)

(図形表示ウィンドウ、属性表示ウィンドウには何も表示されていない)

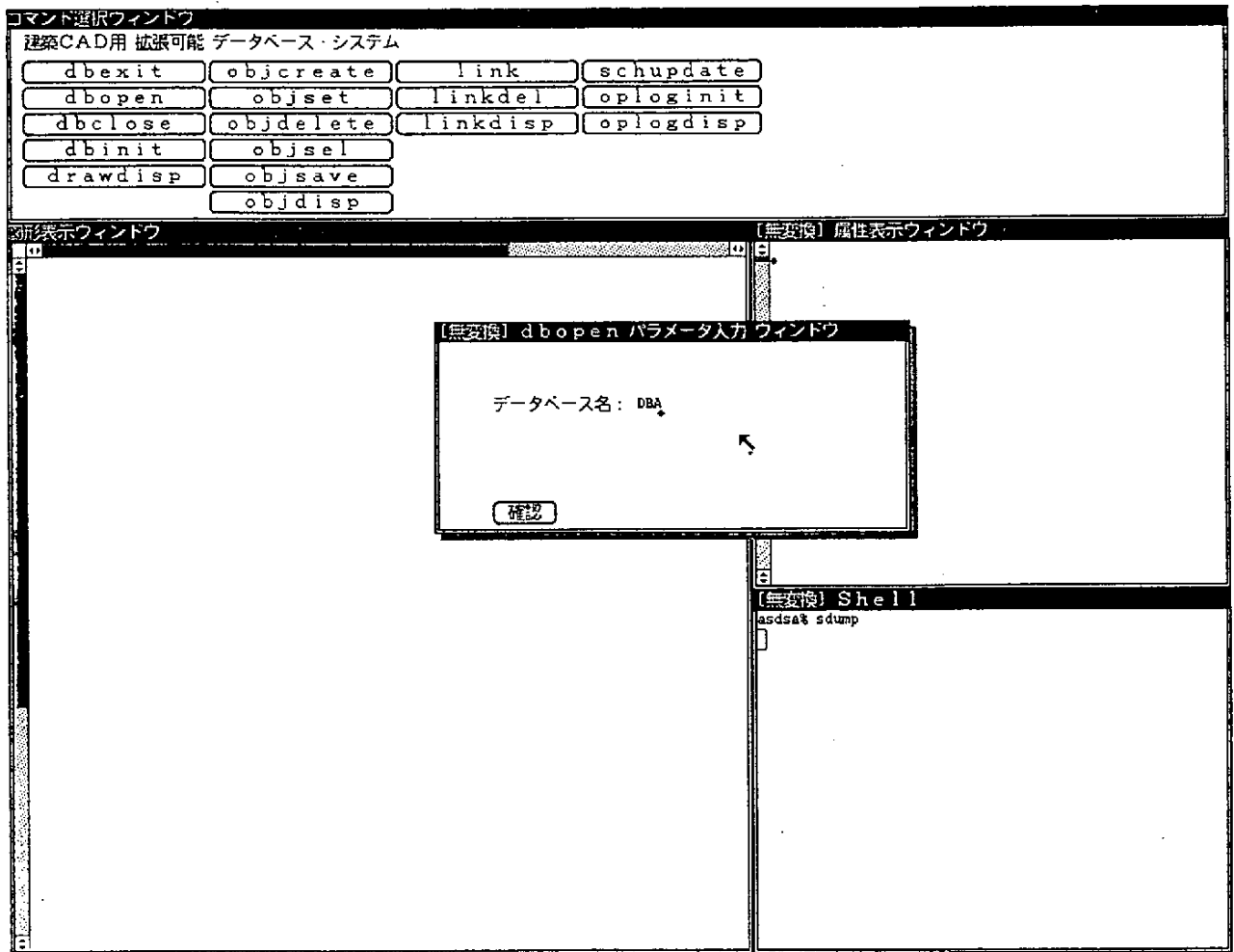
・ dbopen ボタンクリック

→ パラメータ入力ウィンドウ表示

(内容) データベース名:

確 認

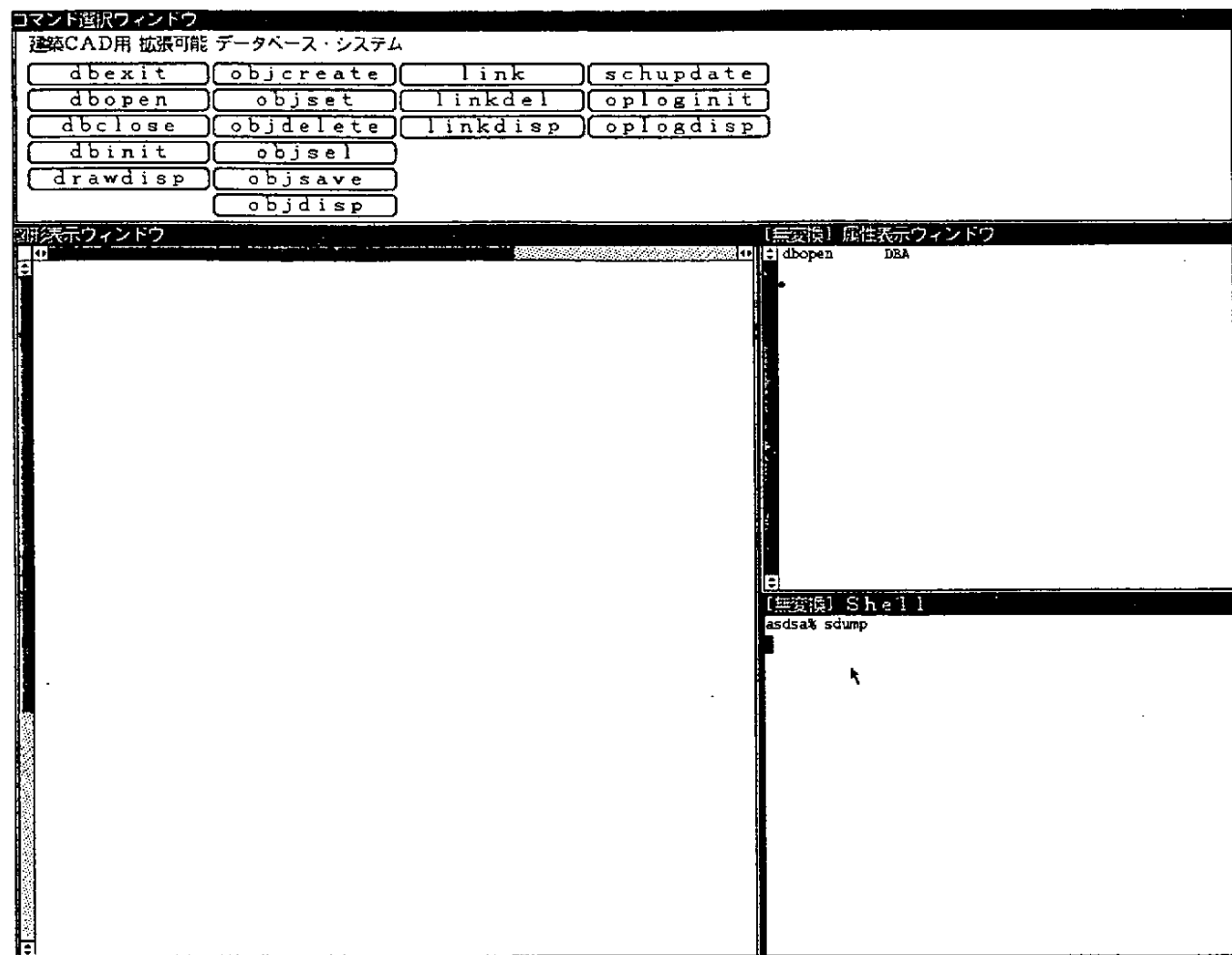
実行例 (画面内容)



動作説明 (dbopen-2)

- データベース名称 "DBA"を入力して「確認」をクリック
→パラメータ入力ウィンドウは消える

実行例 (画面内容)



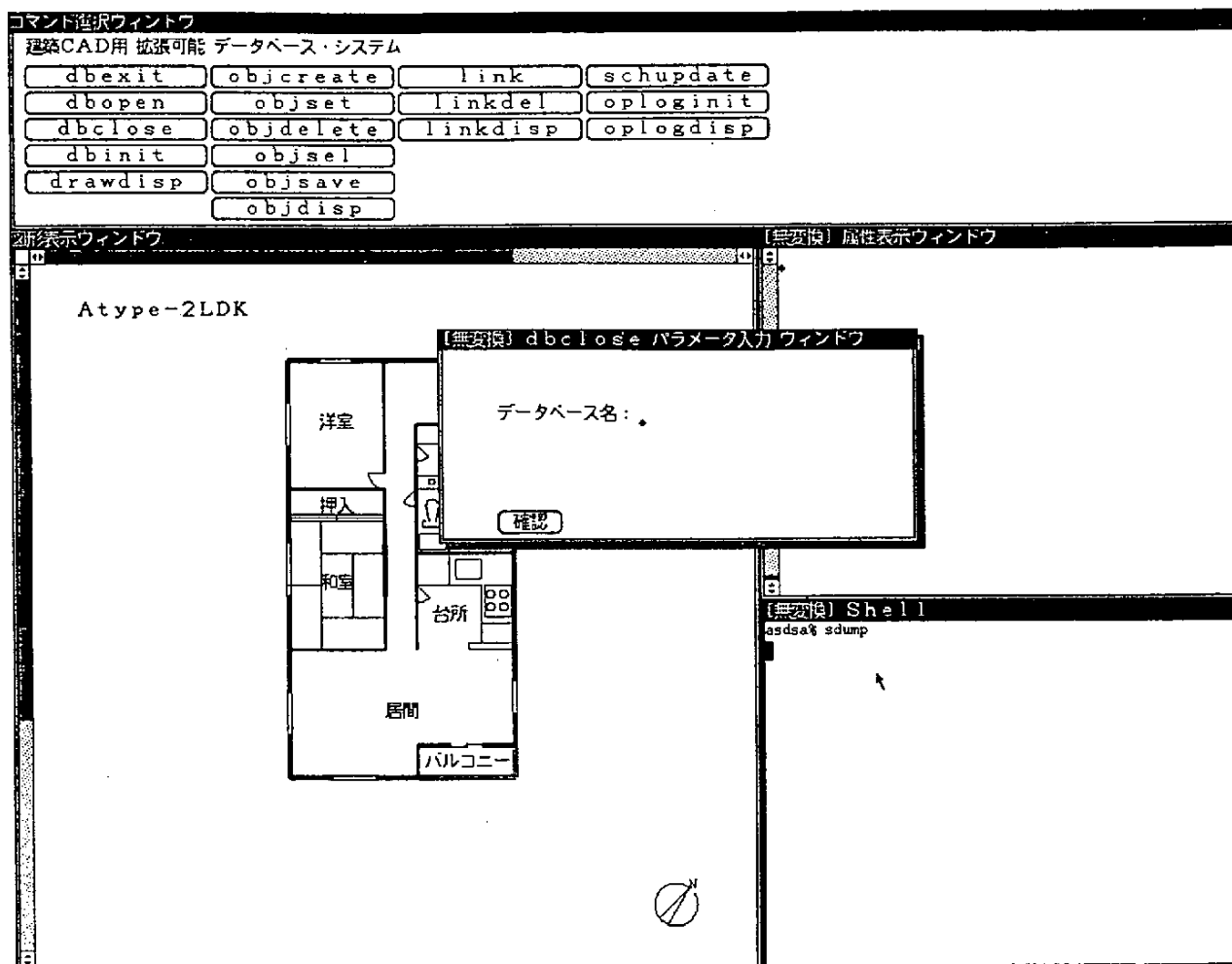
動作説明 (dbopen-3)

→属性表示ウィンドウにコマンド実行のエコー表示
 "dbopen DBA"

5. 6 データベースクローズ機能

コ マ ン ド 名 称		d b c l o s e			
機 能 概 要	指定されたデータベースをクローズする。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	d b c l o s e ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		データベース名	2 4	漢字で最大12文字	
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ d b c l o s e (d b n a m e)			
	引数名	I/O	型	サイズ	備 考
	d b n a m e	I	char*	2 4	データベース名
備 考	0 : 正常終了 - 5 : 指定DBはオープンされていない - 1 0 :				

実行例 (画面内容)



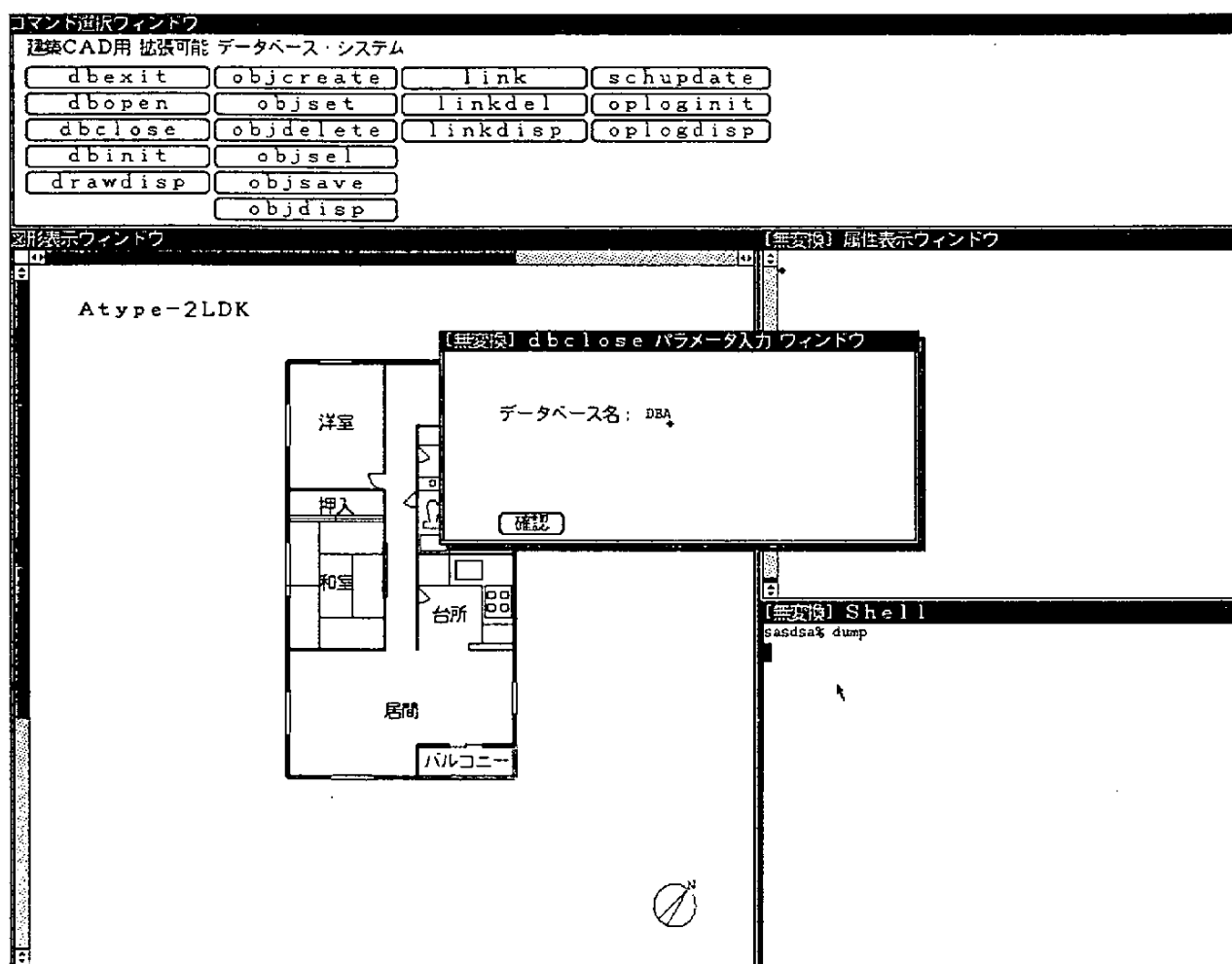
動作説明 (dbclose-1)

(図形表示ウィンドウには0acの物件が表示されている)

- ・ dbclose ボタンクリック
→パラメータ入力ウィンドウ表示
(内容) データベース名:

確 認

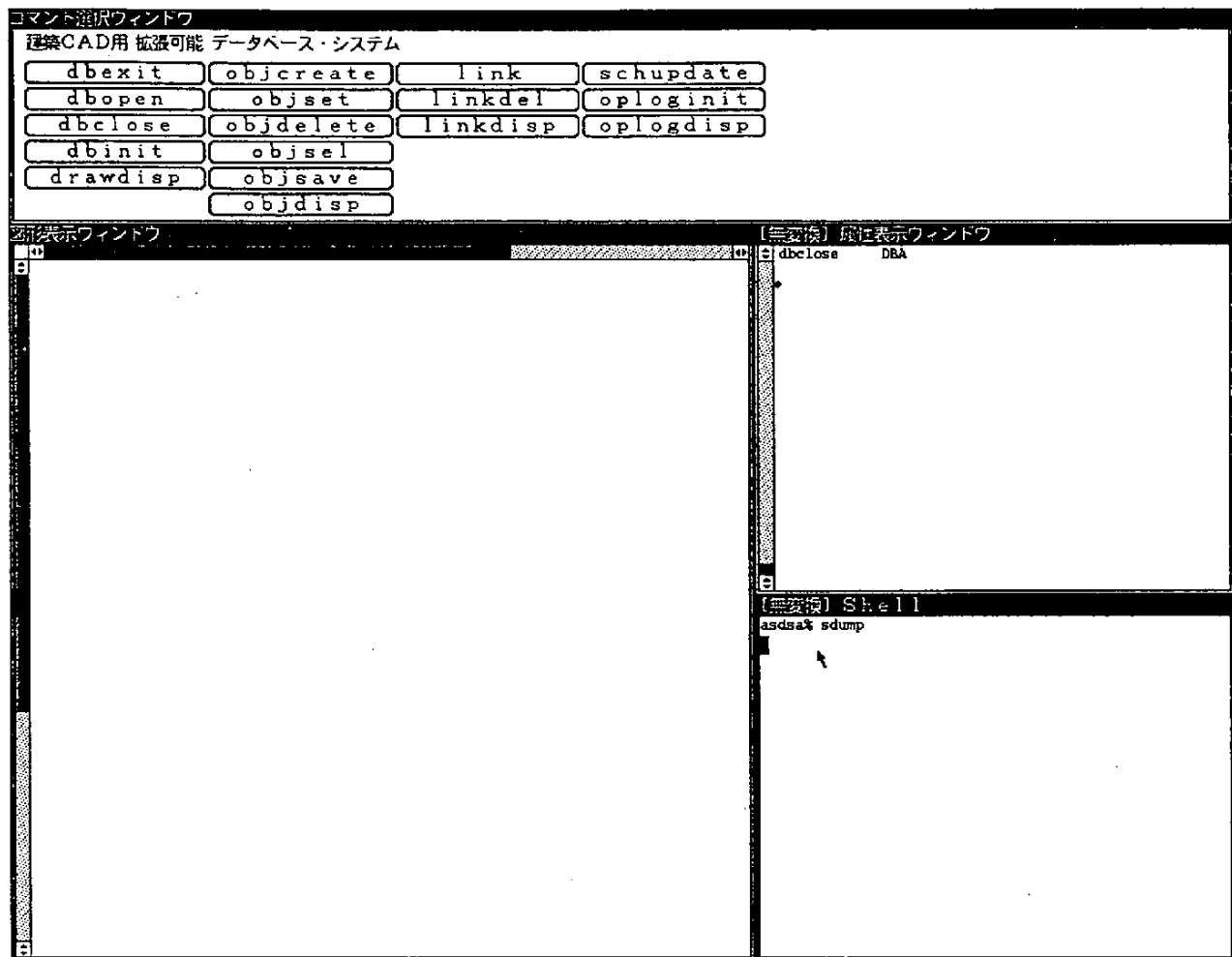
実行例 (画面内容)



動作説明 (dbclose-2)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック
 データベース名: DBA
 →パラメータ入力ウィンドウは消える

実行例 (画面内容)



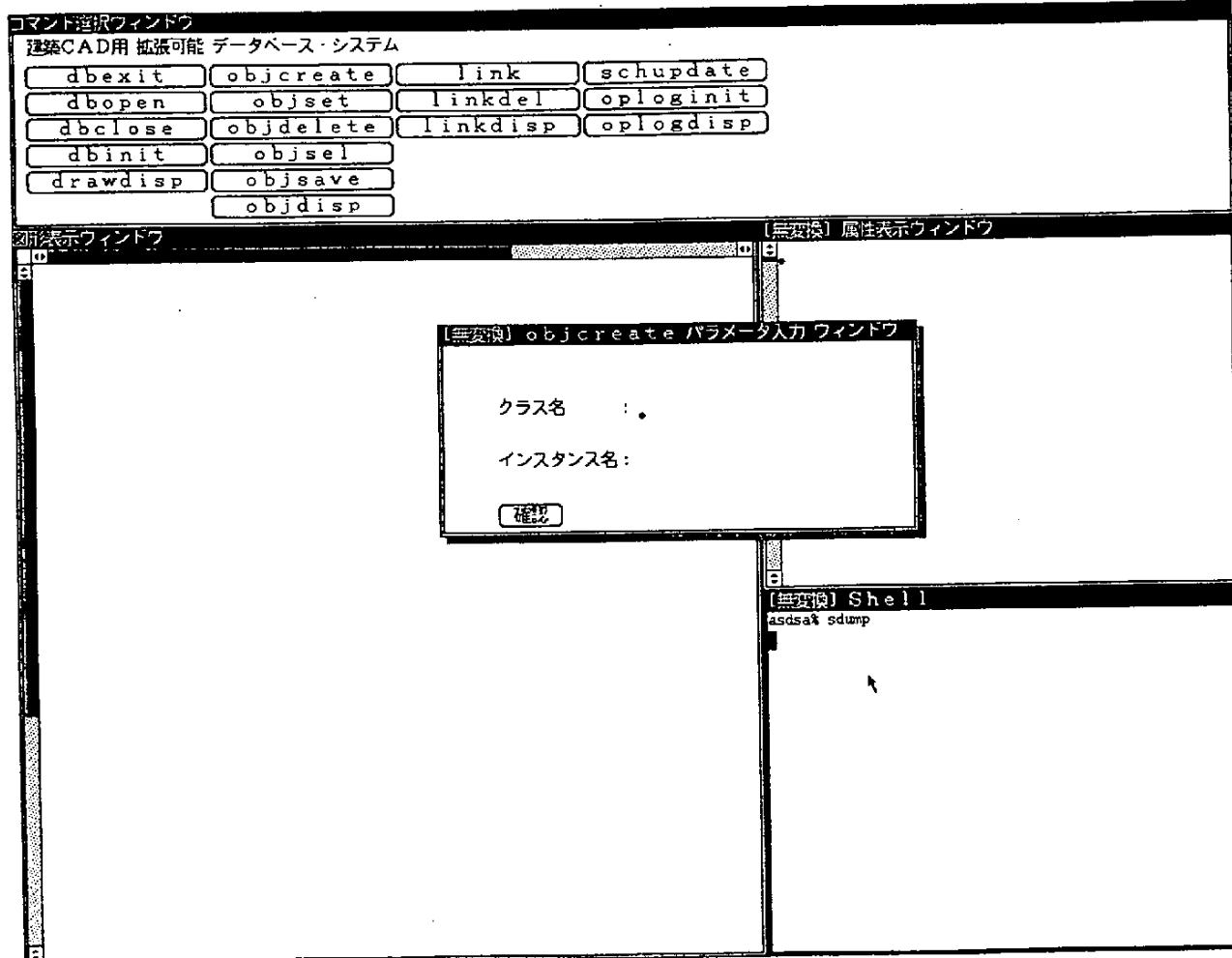
動作説明 (dbclose-3)

- 属性表示ウィンドウにエコー表示
"dbclose DBA"
- 図形表示ウィンドウはクリアされる

5. 7 オブジェクト生成機能

コ マ ン ド 名 称		o b j c r e a t e			
機 能 概 要	指定されたクラスのオブジェクトを指定された名称で生成する。				
操 作 インタフェース	ボ タ ン	o b j c r e a t e ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		クラス名 インスタンス名	2 8 2 8	漢字で最大14文字 漢字で最大14文字	
言 語 インタフェース	定義	int co_objcreate(c_name, o_name, o_id)			
	引数名	I/O	型	サイズ	備 考
	c _ n a m e o _ n a m e o _ i d	I I O	char* char* long*	2 8 2 8 4	クラス名 インスタンス名 インスタンス I.D
備 考	0: 正常終了 -10: -11: クラス名が定義されていない -12: インスタンス名が長すぎる -13: すでに同じ名が存在する -14: インスタンスの最大個数に達している -15: Rootクラスの変数が設定できない (クラス定義が不正) -100: メモリアロケート失敗 生成と同時にRootクラスの変数 { //Object_ID //Object_name } の値が設定される				

実行例 (画面内容)



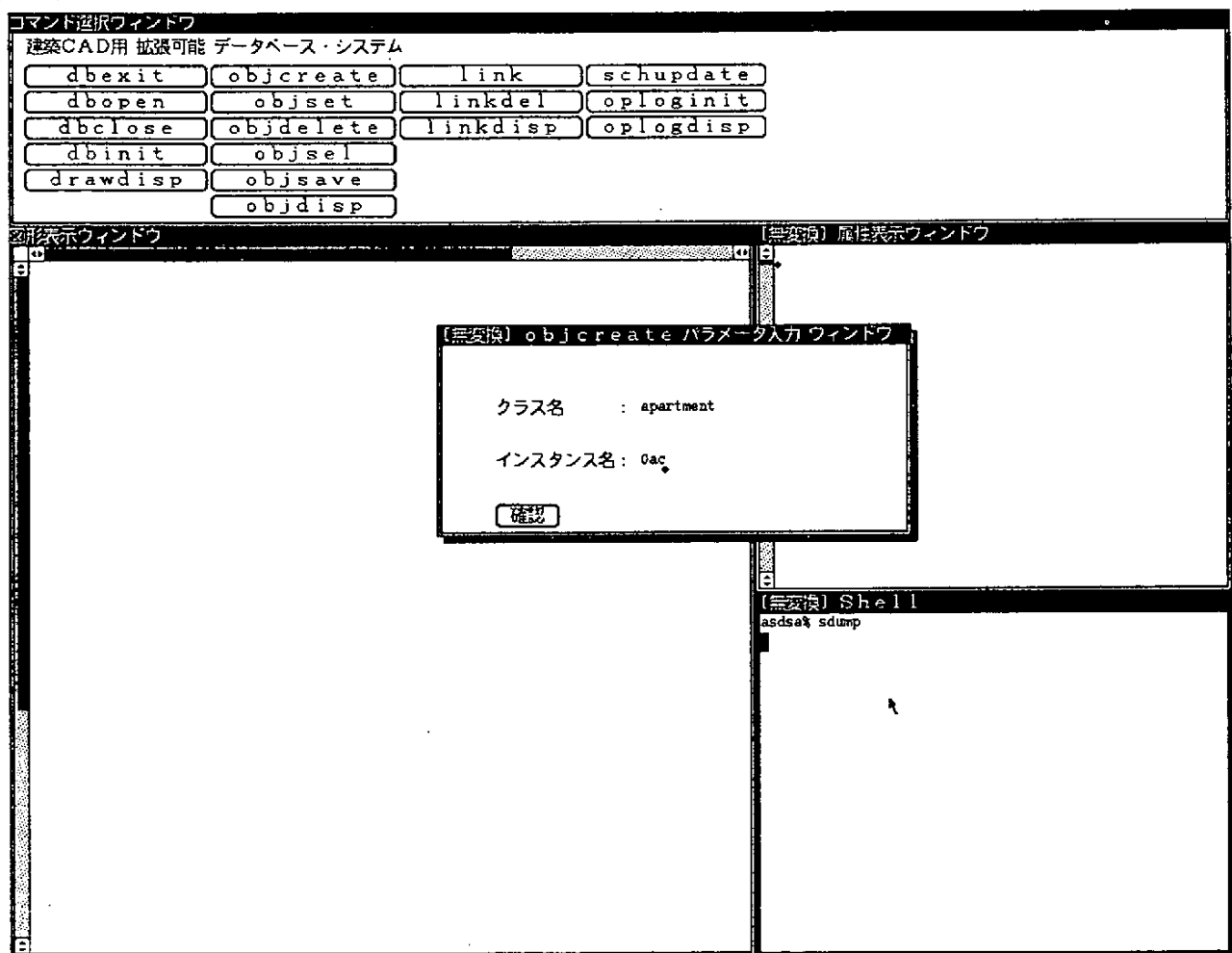
動作説明 (objcreate-1)

(図形表示ウィンドウには何も表示されていない)

- ・ objcreate ボタンをクリック
→ パラメータ入力ウィンドウ表示
(内容) クラス名 :
インスタンス名 :

確認

実行例 (画面内容)

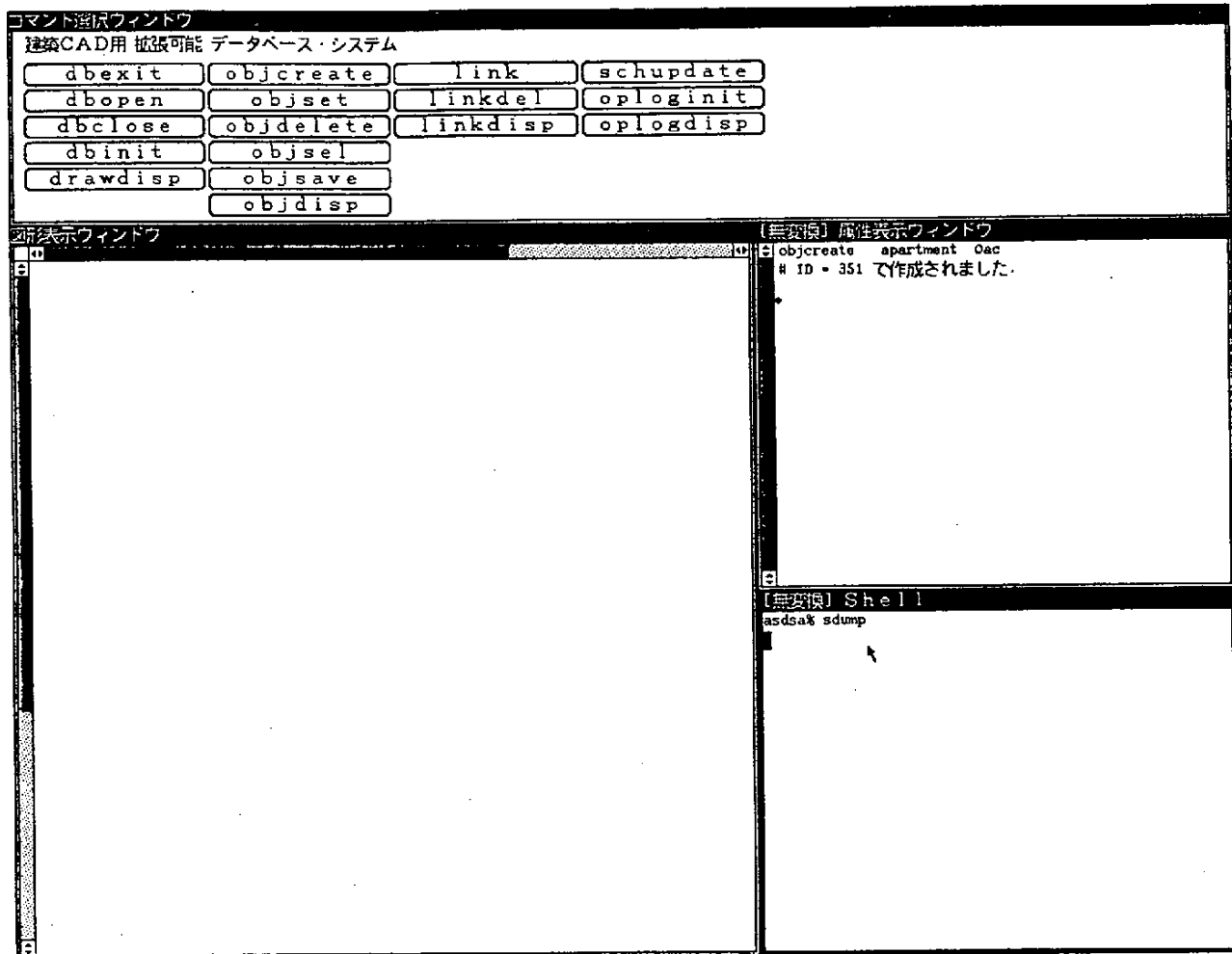


動作説明 (objcreate-2)

・クラス名称"apartment", インスタンス名称"0ac"を入力して「確認」をクリック

→パラメータ入力ウィンドウは消える

実行例 (画面内容)



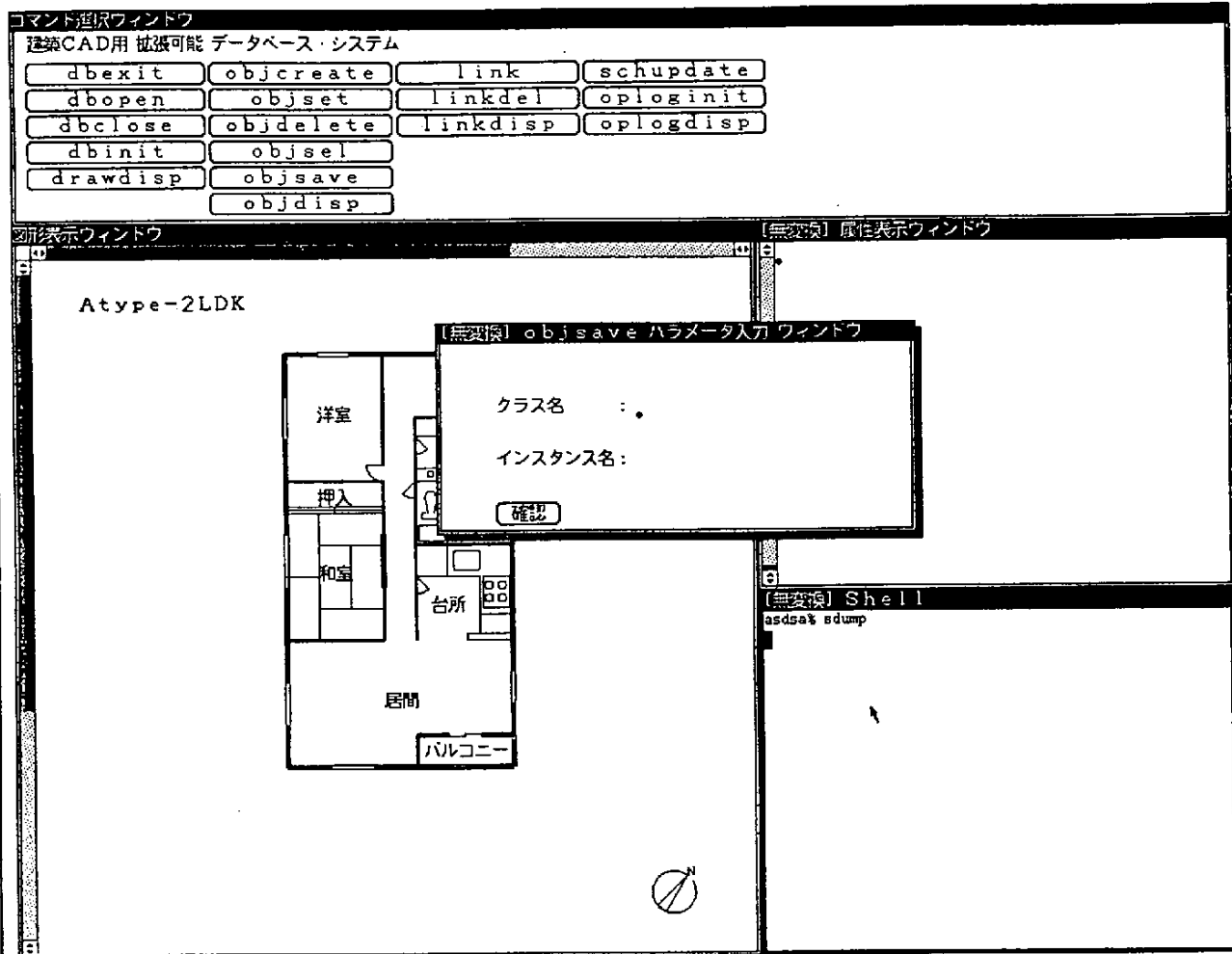
動作説明 (objcreate-3)

- 属性表示ウィンドウにエコー表示
"objcreate apartment 0ac"
- 属性表示ウィンドウに表示
"ID で作成されました"

5. 8 オブジェクト格納機能

コ マ ン ド 名 称		o b j s a v e			
機 能 概 要	指定のオブジェクトをオープン中のデータベースへ格納する。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	o b j s a v e ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		クラス名	8 0		
		インスタンス名	8 0		
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ o b j s a v e (c _ n a m e , o _ n a m e)			
	引数名	1/0	型	サイズ	備 考
	c _ n a m e	1	char*	8 0	クラス名
	o _ n a m e	1	char*	8 0	インスタンス名
備 考	0 : 正常終了				
	- 1 0 : 失敗				

実行例 (画面内容)



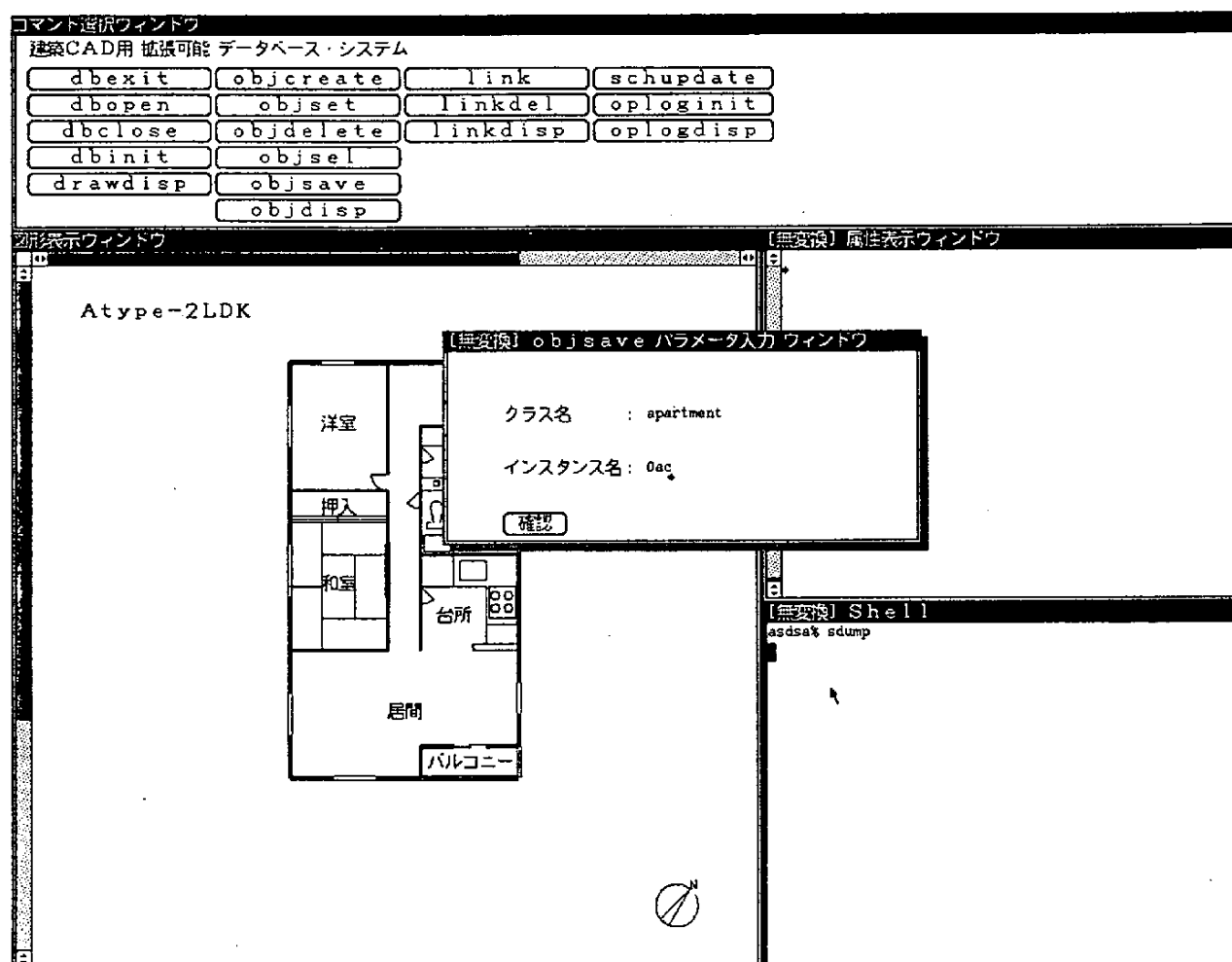
動作説明 (objsave-1)

(図形表示ウィンドウは0acの物件が表示されている)

- ・ Objsave ボタンクリック
- パラメータ入力ウィンドウ表示
- (内容) クラス名 :
- インスタンス名 :

確 認

実行例 (画面内容)



動作説明 (objsave-2)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

クラス名 : apartment

インスタンス名 : 0ac

→パラメータ入力ウィンドウは消える

実行例 (画面内容)

コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit	objcreate	link	schupdate
dbopen	objset	linkdel	oploginit
dbclose	objdelete	linkdisp	oplogdisp
dbinit	objsel		
drawdisp	objsave		
	objdisp		

図形表示ウィンドウ

Atype-2LDK

属性表示ウィンドウ

objsave apartment 0ac

Shell

asdsa% sdump

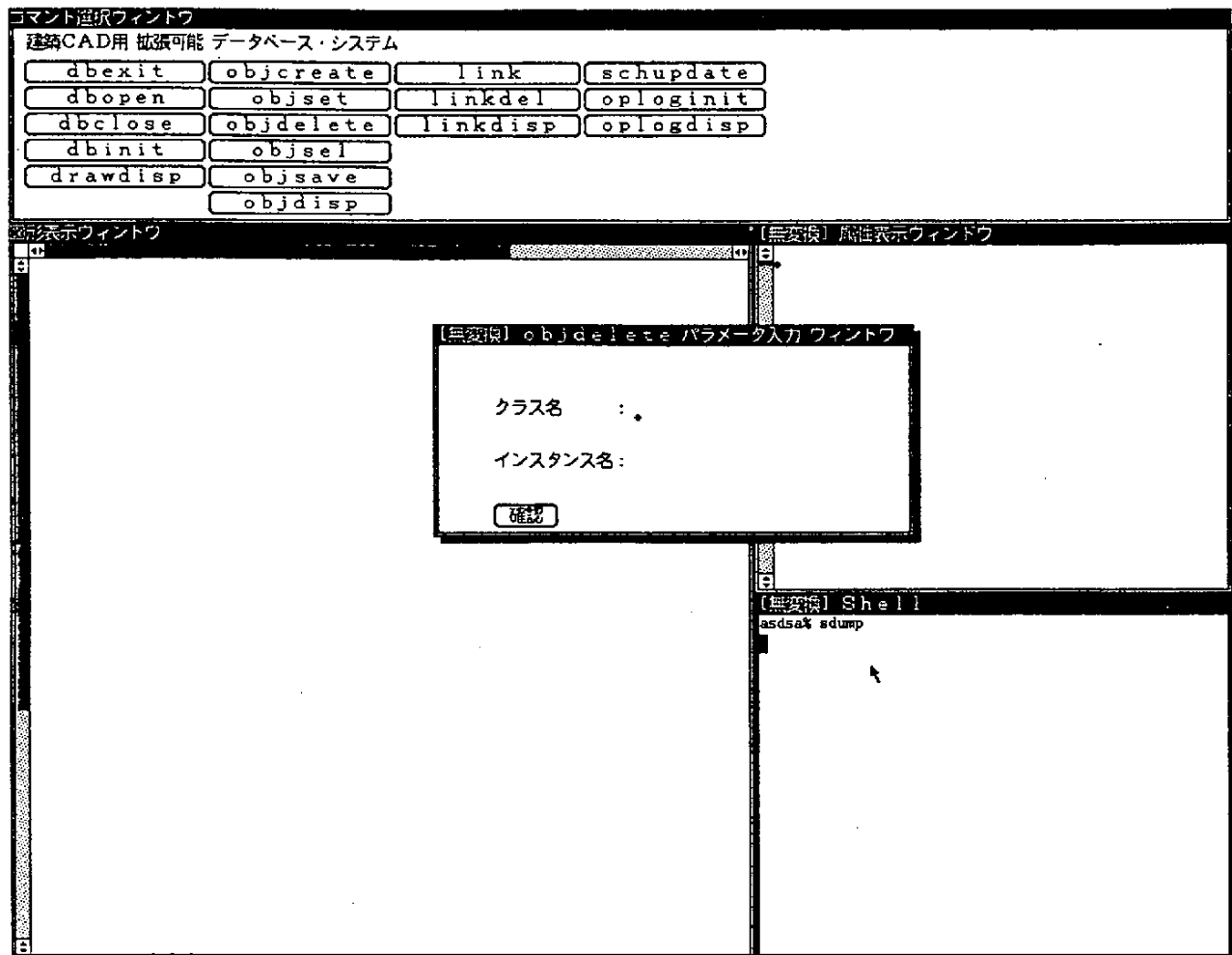
動作説明 (objsave-3)

→属性表示ウィンドウにエコー表示
"objsave apartment 0ac"

5. 9 オブジェクト削除機能

コ マ ン ド 名 称		o b j d e l e t e			
機 能 概 要	正規表現で指定されたオブジェクトを削除する。 (関連するリンクも削除する。)				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	o b j d e l e t e ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		クラス名	8 0	正規表現も可能	
		インスタンス名	8 0	正規表現も可能	
言 語 イ ン タ フ ェ ー ス	定義	long co_objdelete(c_name, o_name)			
	引数名	I/O	型	サイズ	備 考
	c _ n a m e	I	char*	8 0	クラス名 → 正規表現
	o _ n a m e	I	char*	8 0	インスタンス名 → 正規表現
備 考	正 または 0 : 削除されたオブジェクト数 - 1 : クラス名の正規表現がおかしい (正規表現で使えるのは - 2 : インスタンス名の正規表現がおかしい ? と * のみ) - 10 : インスタンス・データがおかしい				

実行例 (画面内容)



動作説明 (objdelete-1)

(図形表示ウィンドウには何も表示されていない状態)

・ Objdelete ボタンクリック

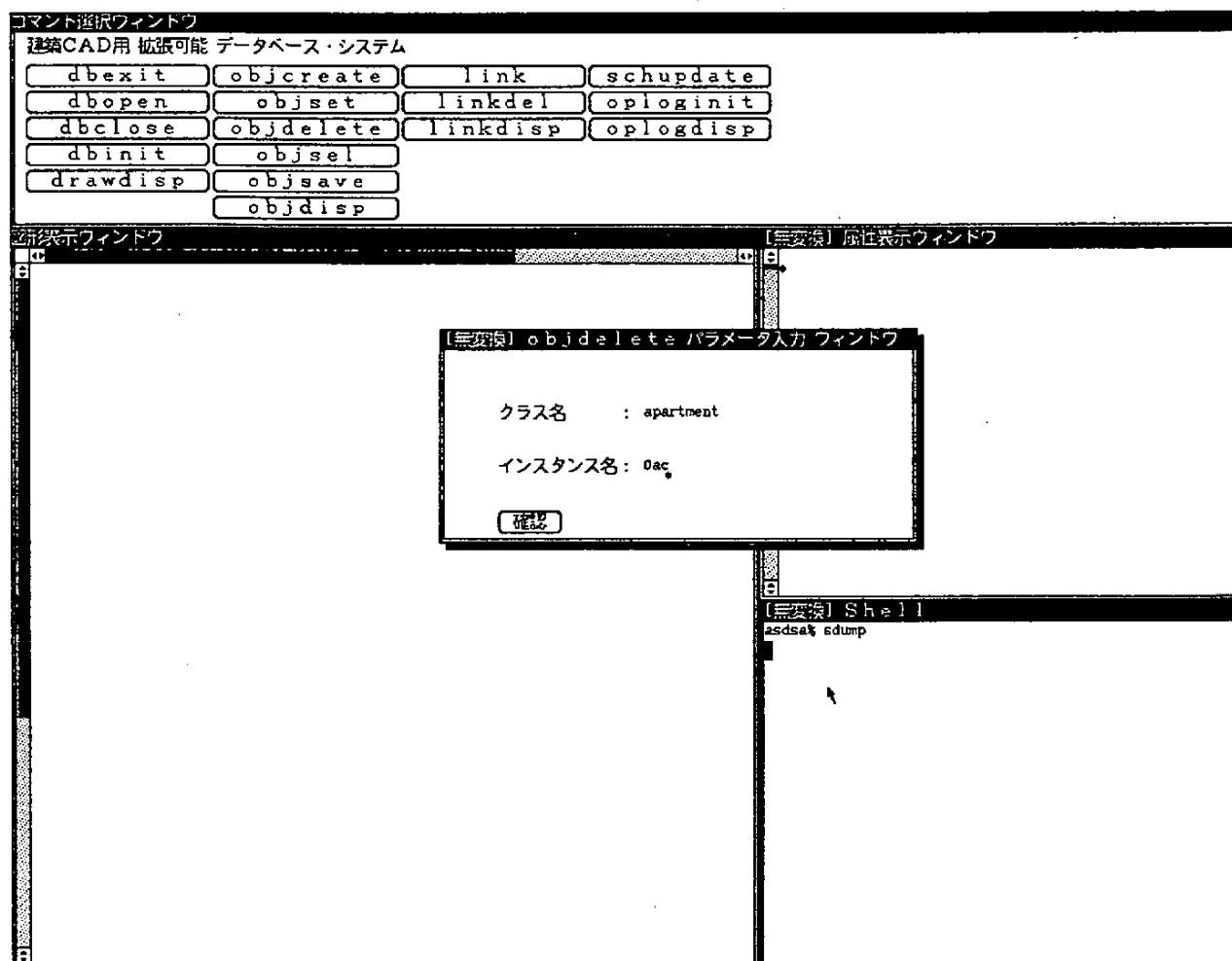
→パラメータ入力ウィンドウ表示

(内容) クラス名:

インスタンス名:

確認

実行例 (画面内容)



動作説明 (objdelete-2)

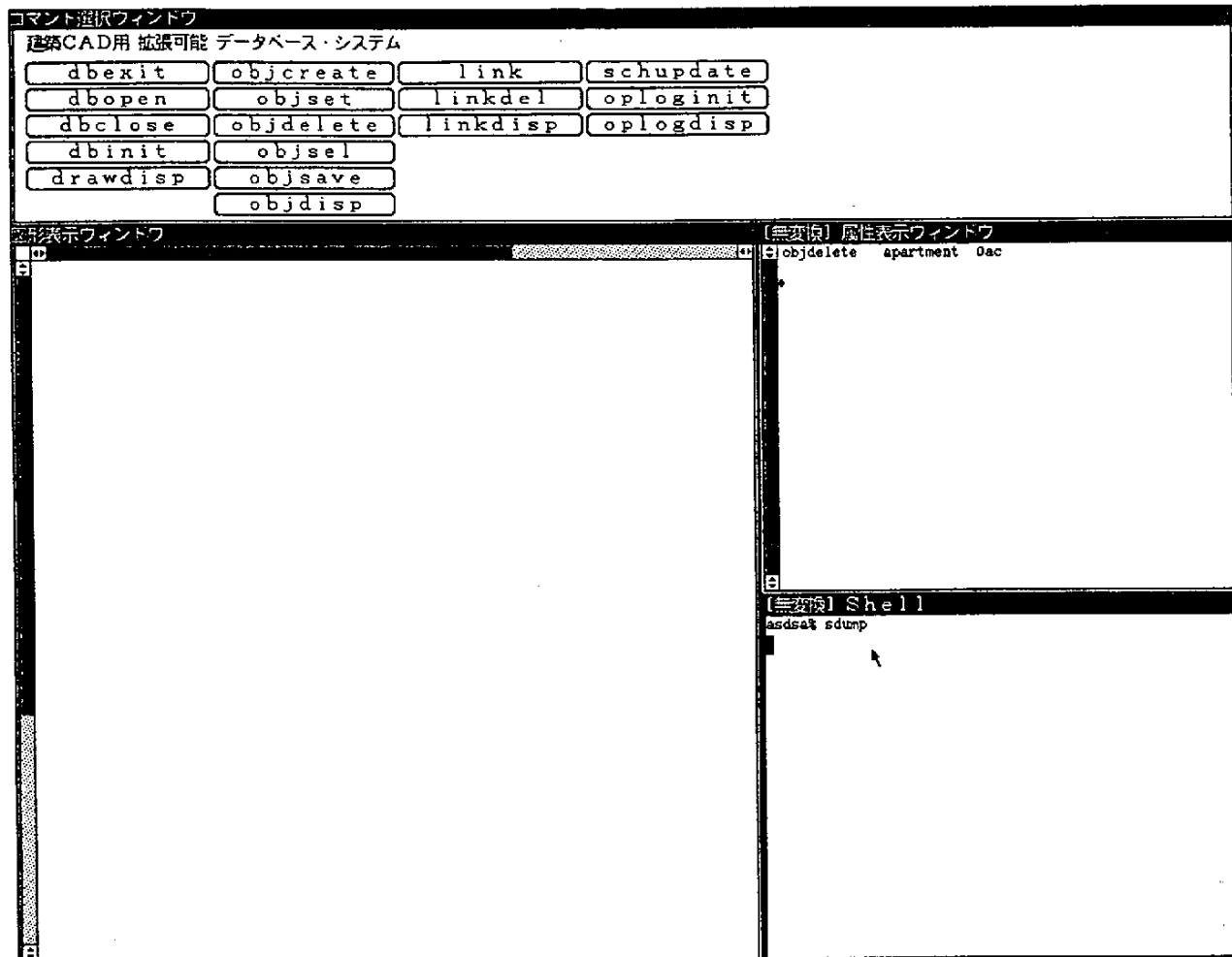
パラメータ入力ウィンドウに以下の値を入力後、「確認」をクリック

クラス名 : apartment

インスタンス名 : 0ac

→パラメータ入力ウィンドウは消える

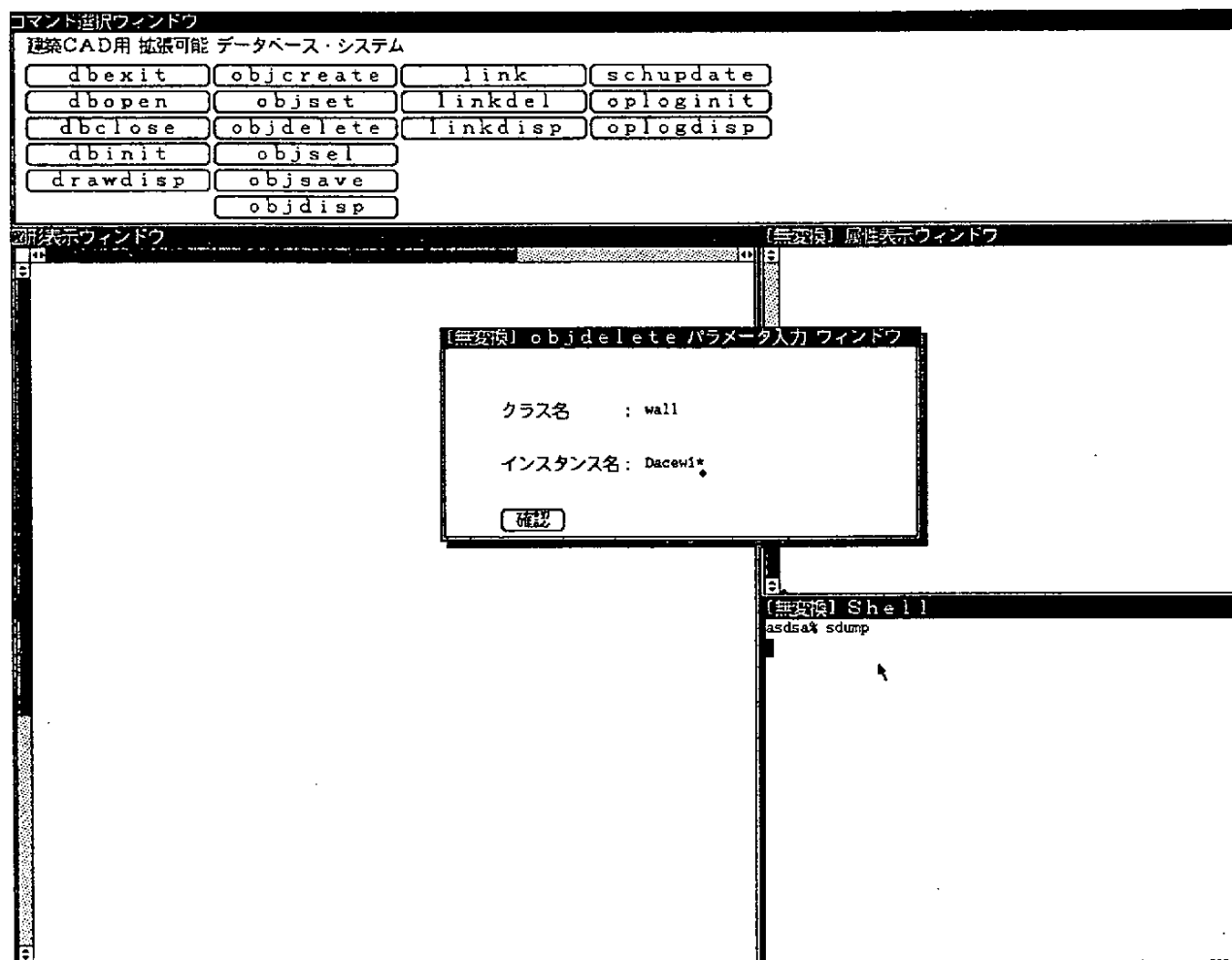
実行例 (画面内容)



動作説明 (objdelete-3)

→ 属性表示ウィンドウにエコー表示
"objdelete apartment 0ac"

実行例 (画面内容)



動作説明 (objdelete-4)

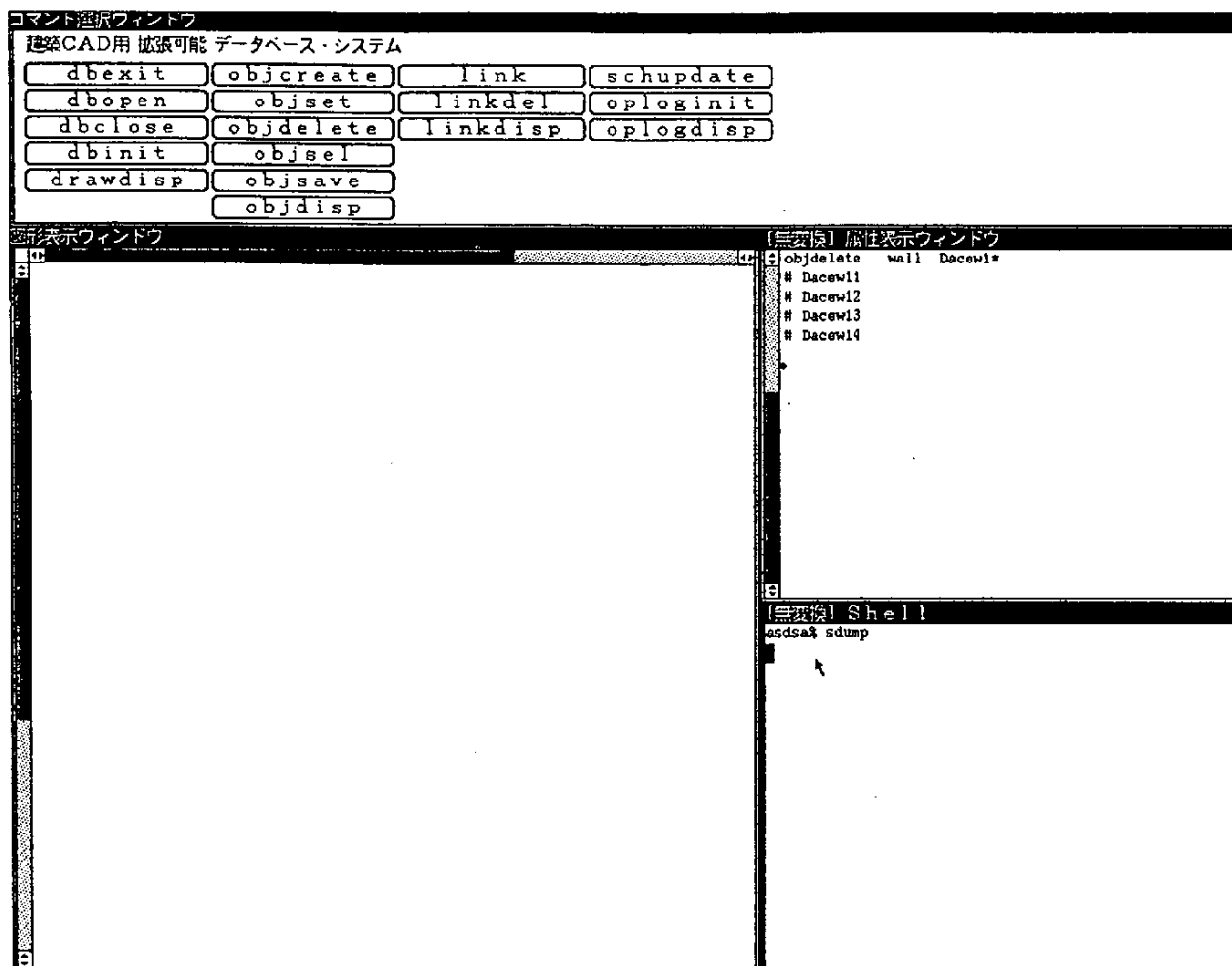
パラメータ入力ウィンドウに以下の値を入力後、「確認」をクリック

リンク名称 : wall

クラス名 : Dacew1*

→パラメータ入力ウィンドウは消える

実行例 (画面内容)



動作説明 (objdelete-5)

→属性表示ウィンドウにエコー表示

"objdelete wall Dacew1"

→属性表示ウィンドウに削除されたオブジェクト表示

Dacew11

Dacew12

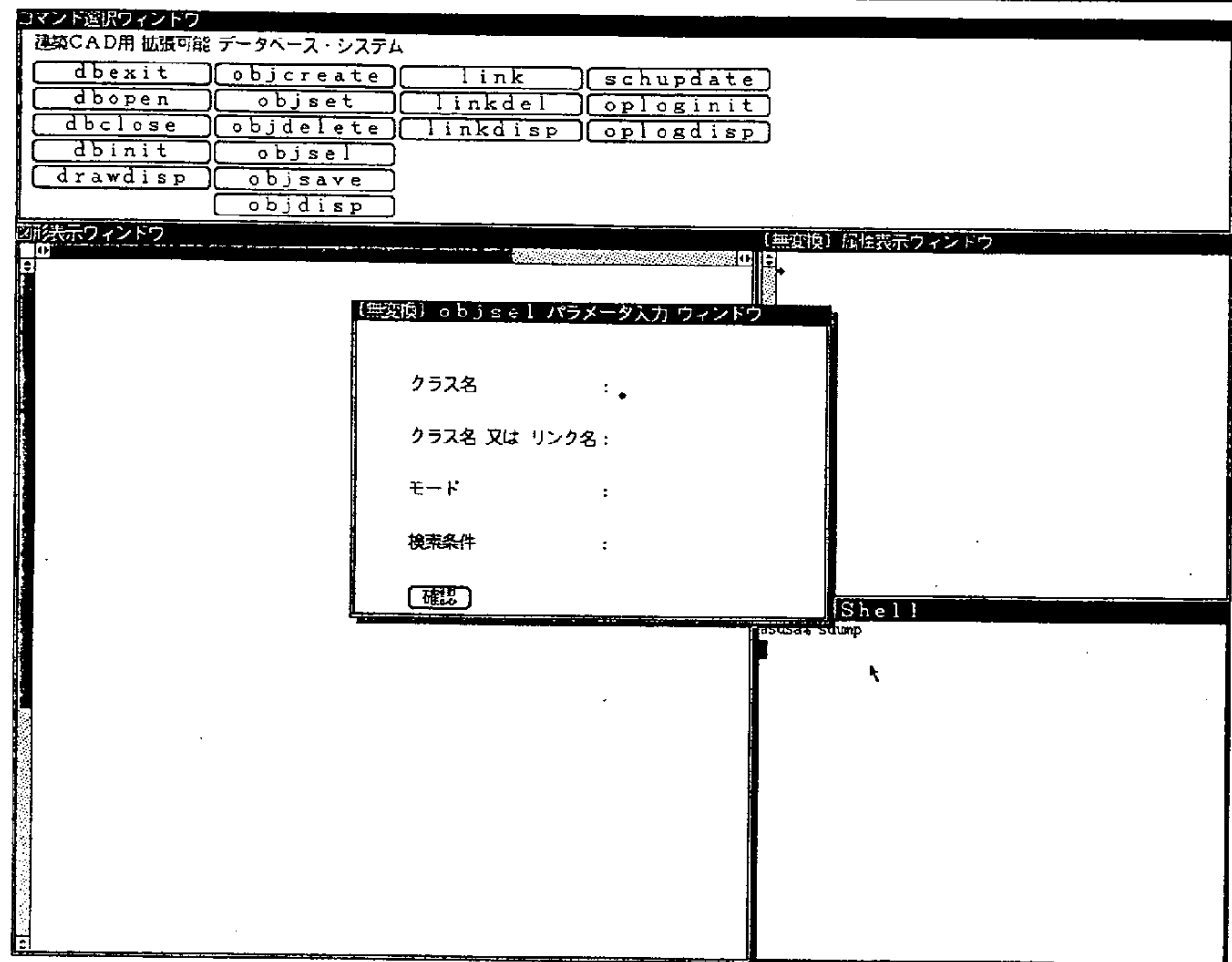
Dacew13

Dacew14

5. 1 0 オブジェクト検索機能

コ マ ン ド 名 称		o b j s e l			
機 能 概 要	指定クラスのオブジェクトの中で、検索条件を満たすインスタンスの				
	・オブジェクトID ・オブジェクト名 ・対象変数値 を表示する。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	o b j s e l ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		クラス 名	8 0	(0:自分のクラス 1:隣のクラス 2:リンク)	
		クラス 名 または リンク名	8 0		
		モード	2		
検索条件	8 0				
言 語 イ ン タ フ ェ ー ス	定義	long co_objsel(c_name, s_name, mode, cond)			
	引数名	1/0	型	サイズ	備 考
	c _ n a m e	I	char*	8 0	比較される数値の属するクラス クラス名 または リンク名 0 : 自分のクラス 1 : 隣のクラス 2 : リンク
	s _ n a m e	I	char*	8 0	
	m o d e	I	short	2	
c o n d	I	char*	8 0		
備 考	0 または正 : 検索結果の数 - 1 0 : エラー				

実行例 (画面内容)



動作説明 (objsel-1)

(図形表示ウィンドウには何も表示されていない)

・objselボタンをクリック

→パラメータ入力 ウィンドウ表示

(内容)

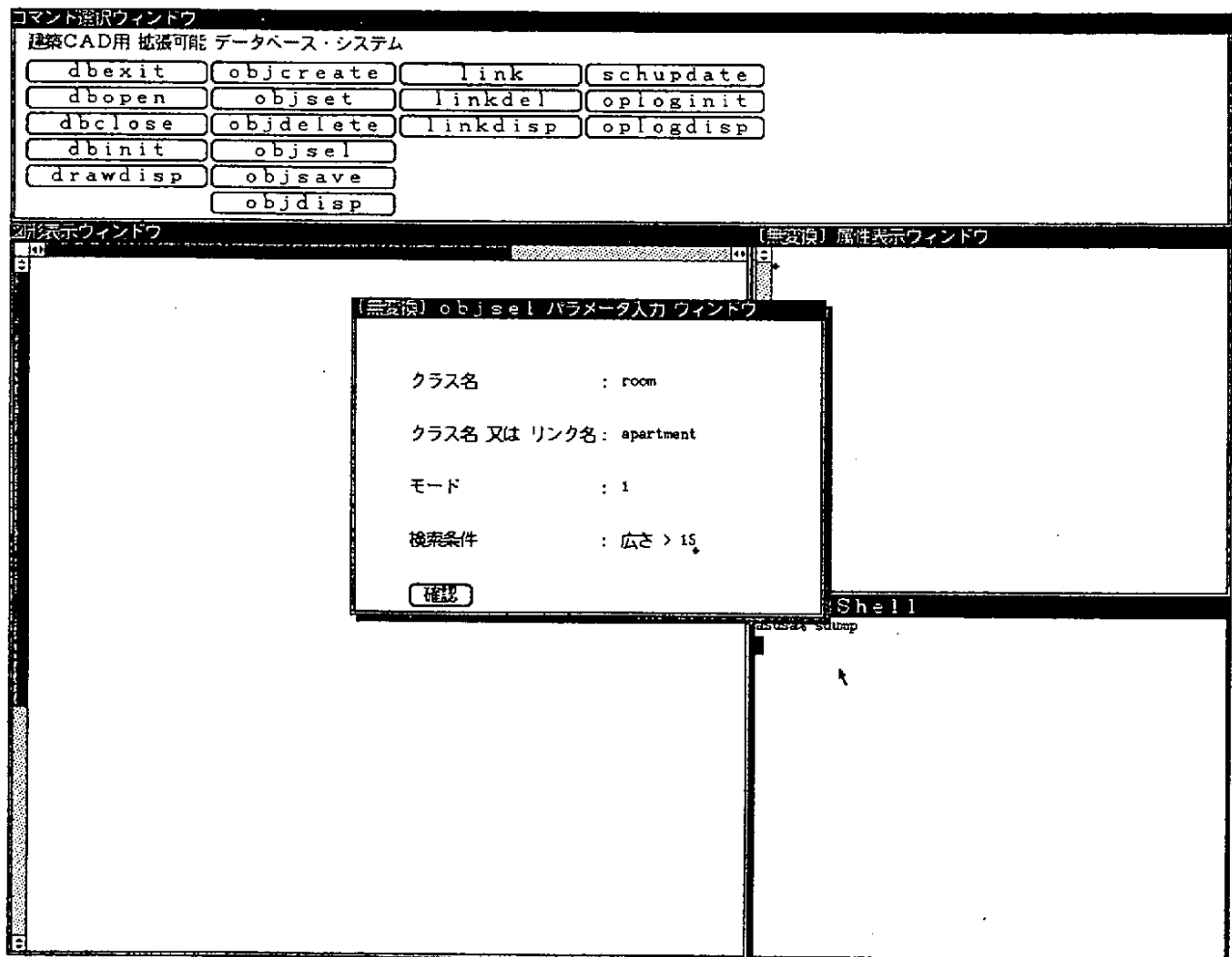
クラス名:

クラス名又はリンク名:

モード:

検索条件:

実行例 (画面内容)



動作説明 (objsel-2)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

クラス名 : room

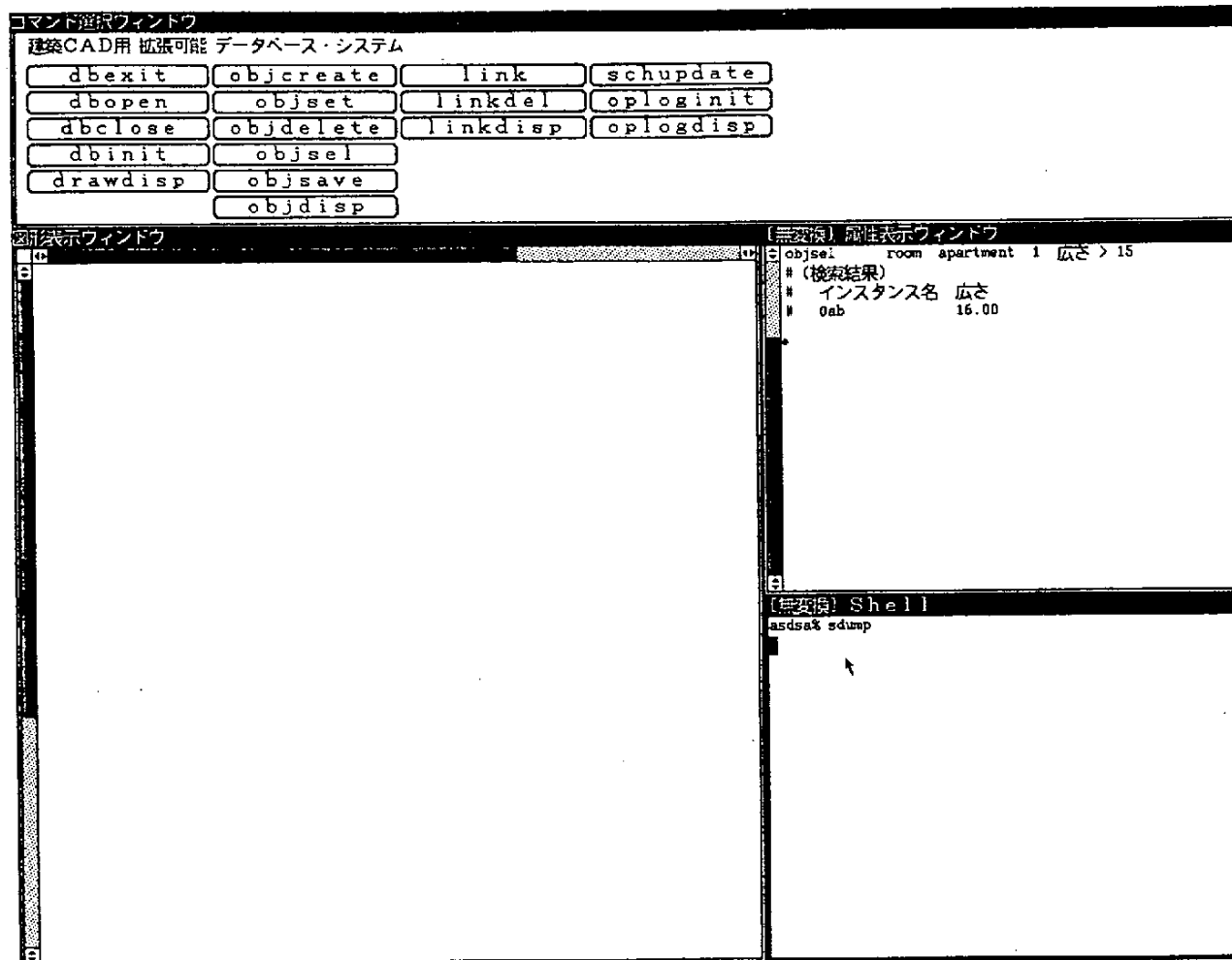
クラス名又はリンク名 : apartment

モード : 1

検索条件 : 広さ > 15

→パラメータ入力ウィンドウが消える

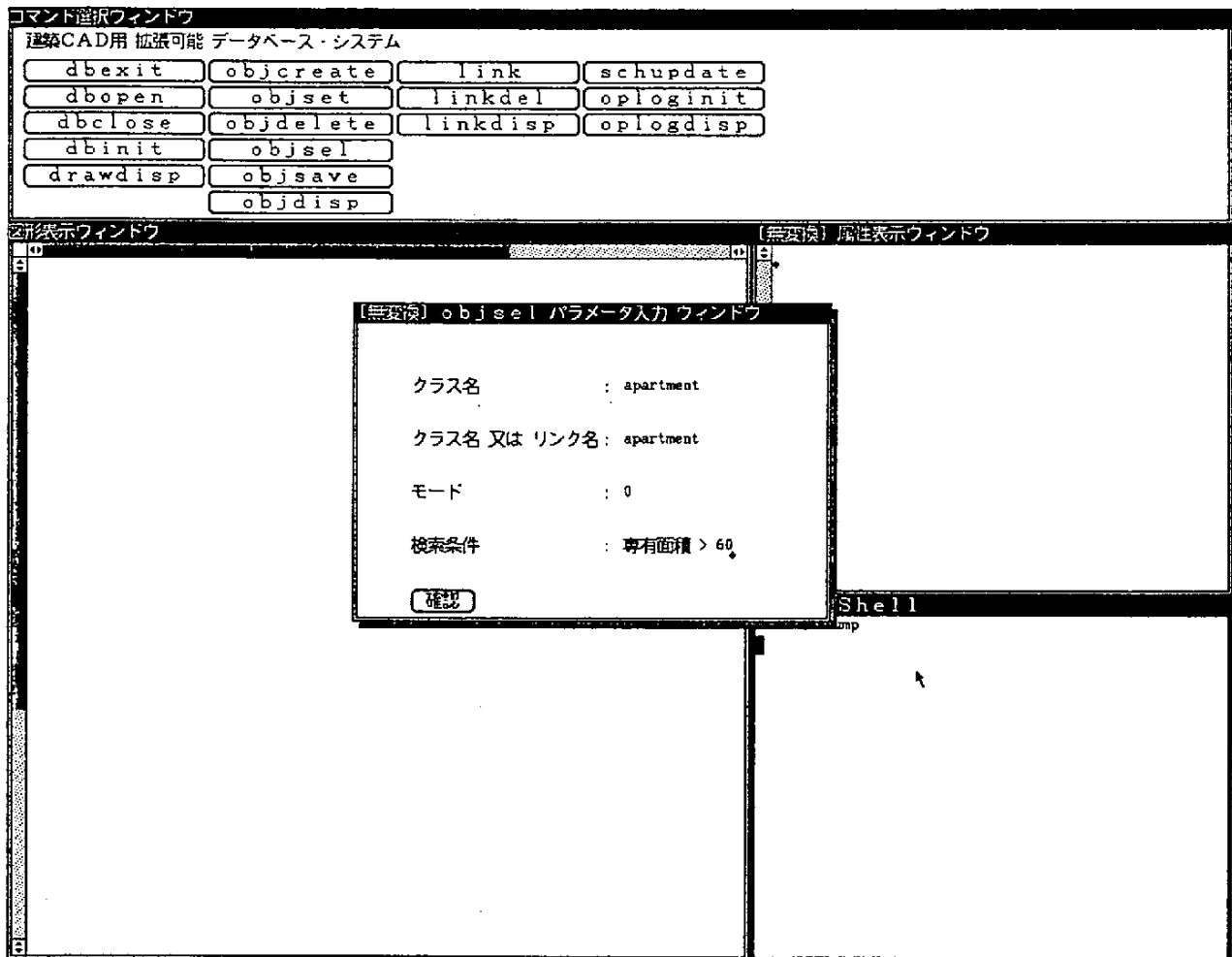
実行例 (画面内容)



動作説明 (objsel-3)

- 属性表示ウィンドウにエコー表示
"objsel room apartment 1 広さ > 15"
- 検索結果が属性表示ウィンドウに表示される。
検索結果
インスタンス名 0ab 広さ 16

実行例 (画面内容)



動作説明 (objsel-4)

パラメータ入力 ウィンドウに以下の値を入力し、「確認」をクリック

クラス名 : apartment

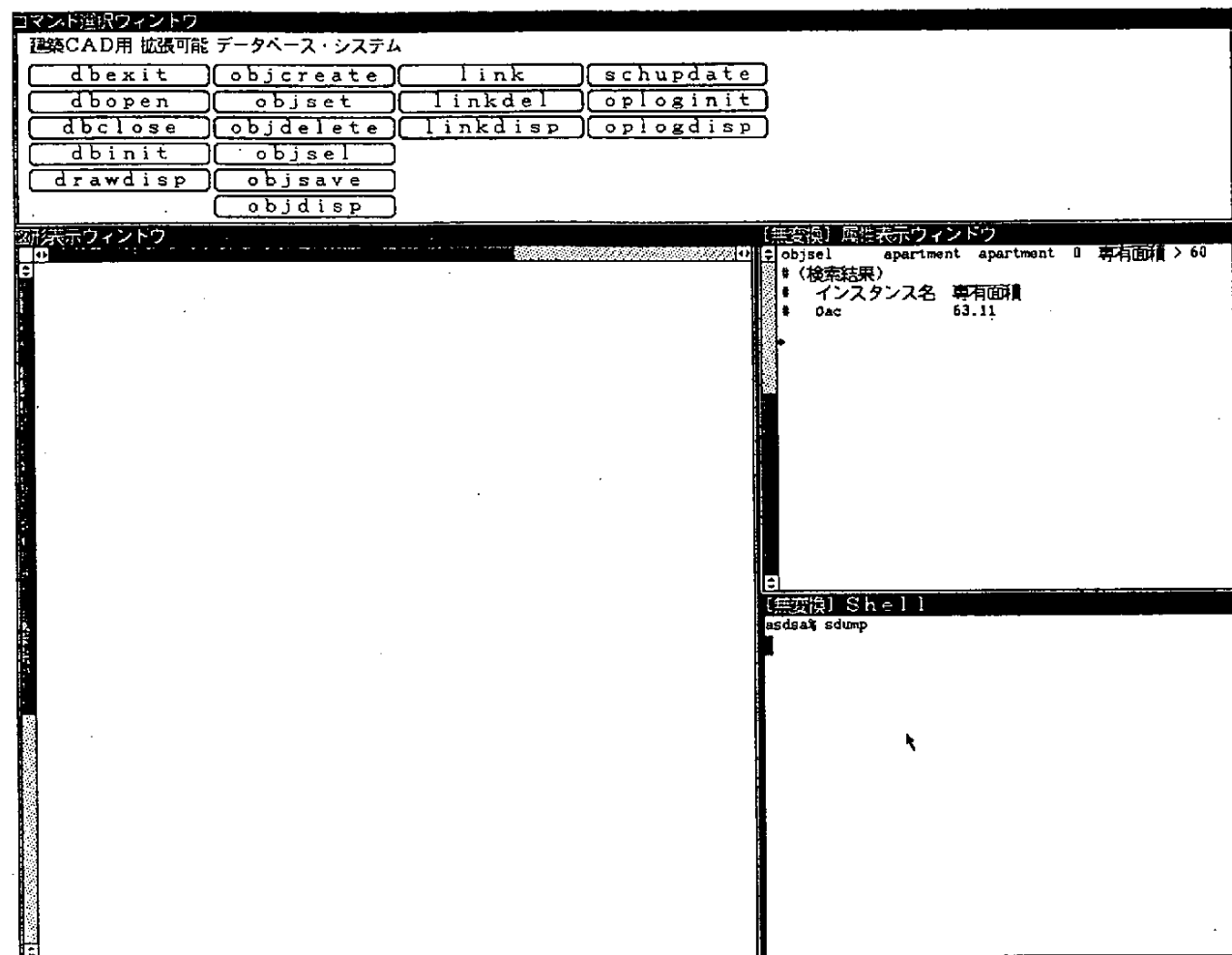
クラス名又はリンク名 : apartment

モード : 0

検索条件 : 専有面積 > 60

→パラメータ入力ウィンドウが消える

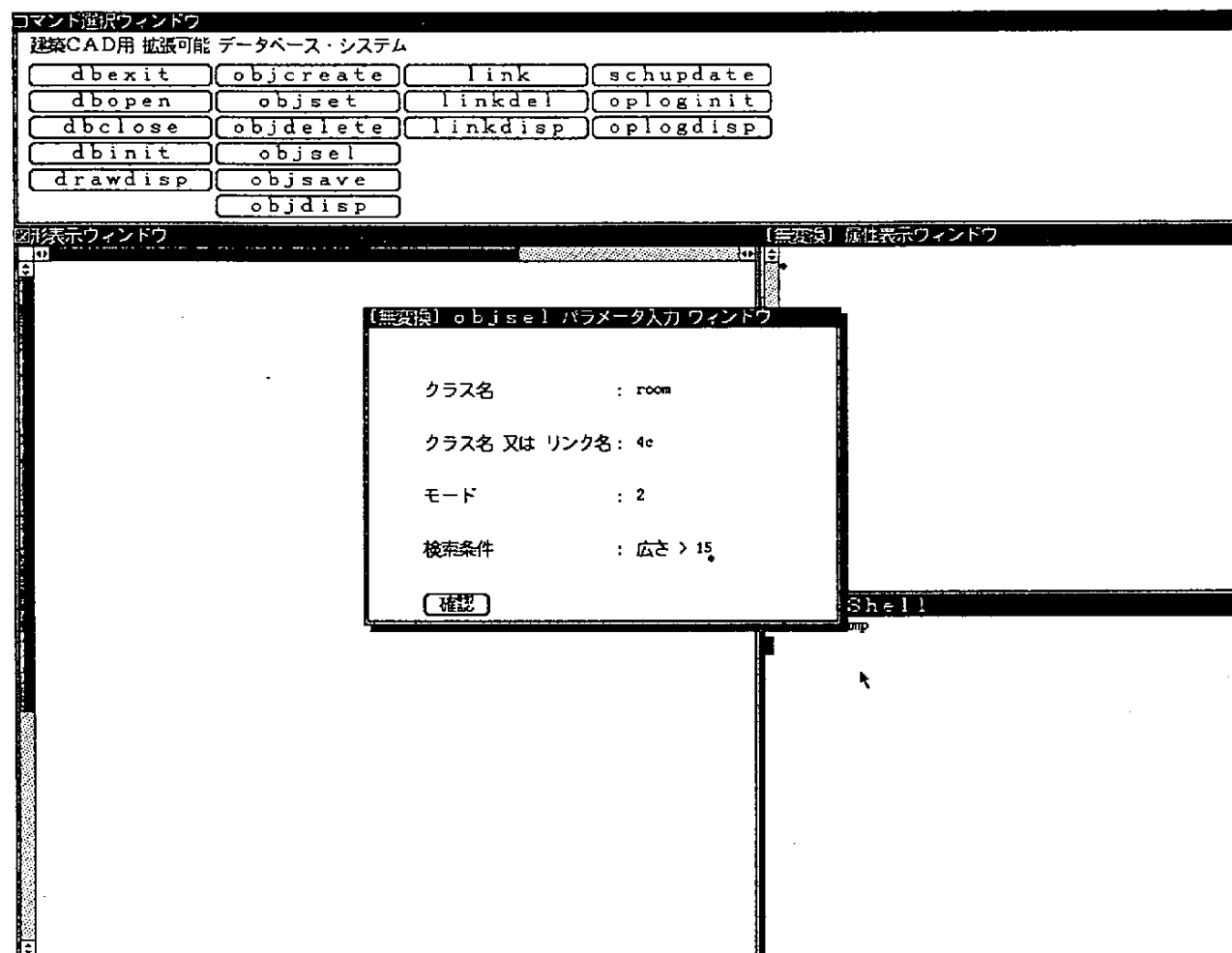
実行例 (画面内容)



動作説明 (objsel-5)

- 属性表示ウィンドウにエコー表示
 "objsel apartment apartment 0 専有面積 > 60"
- 検索結果が属性表示ウィンドウに表示される。
 検索結果
 インスタンス名 0ac 専有面積 63.11

実行例 (画面内容)



動作説明 (objsel-6)

パラメータ入力 ウィンドウに以下の値を入力し、「確認」をクリック

クラス名 : room

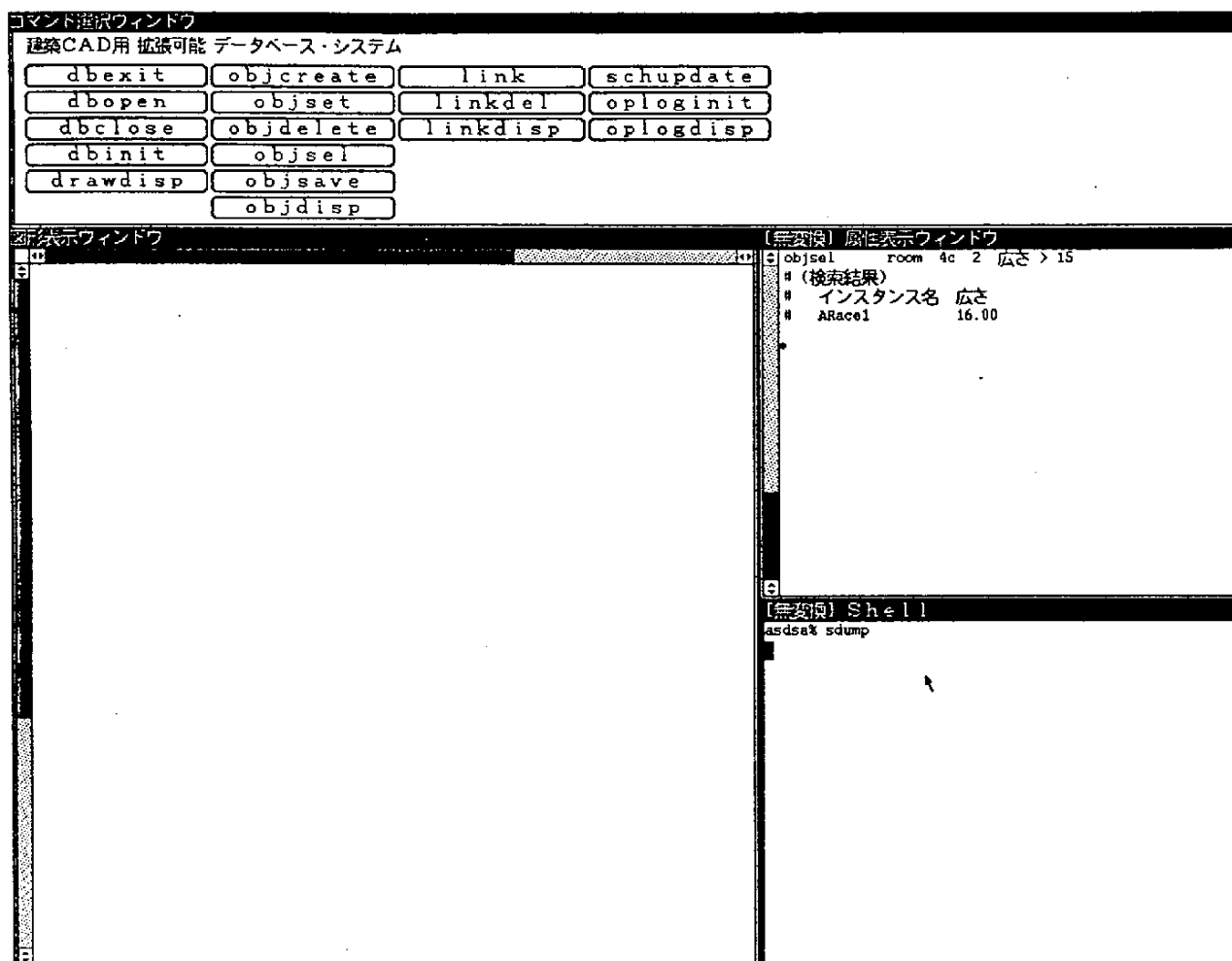
クラス名又はリンク名 : 4c

モード : 2

検索条件 : 広さ > 15

→パラメータ入力ウィンドウが消える

実行例 (画面内容)



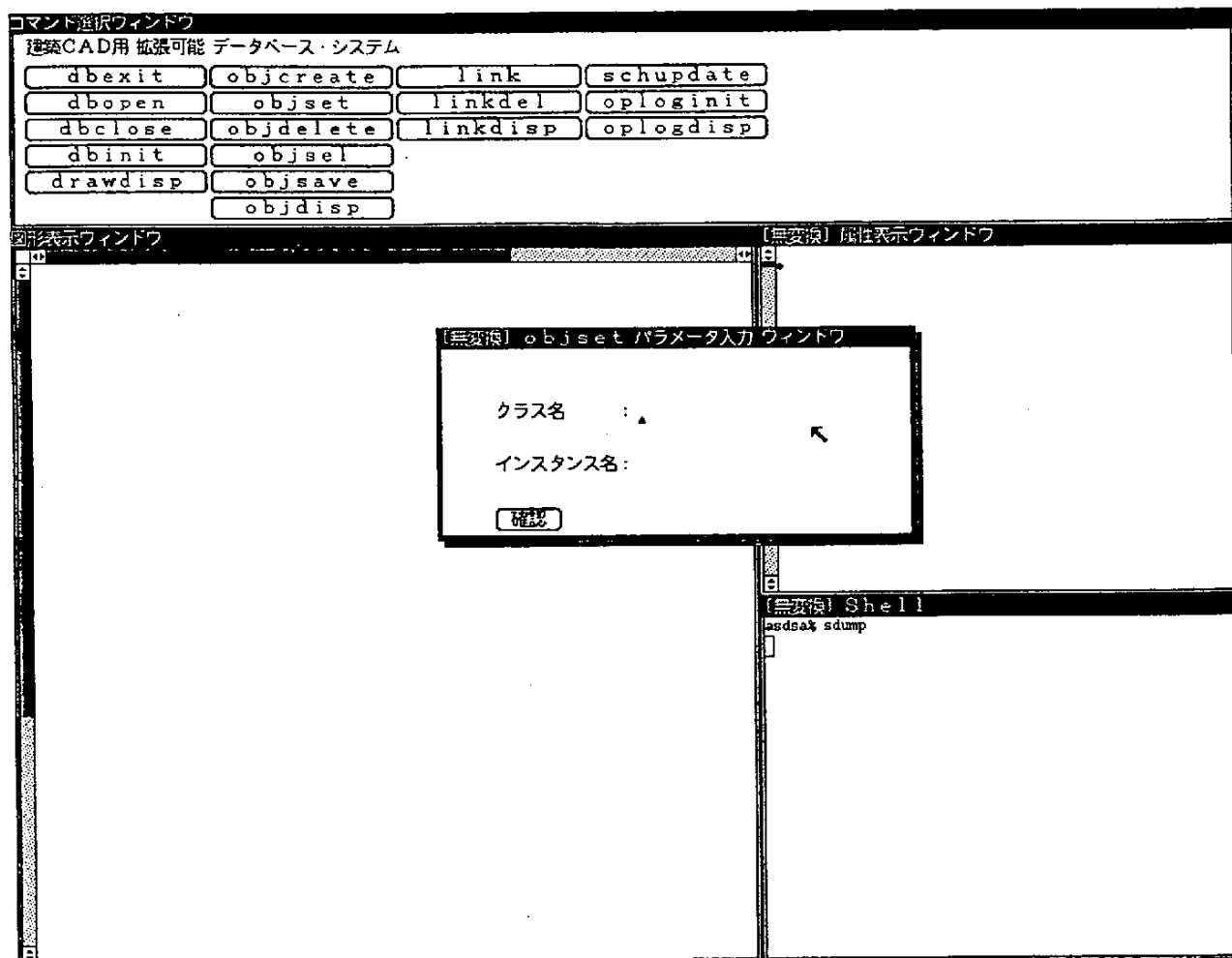
動作説明 (objsel-7)

- 属性表示ウィンドウにエコー表示
"objsel room 4c 2 広さ > 15"
- 検索結果が属性表示ウィンドウに表示される。
検索結果
インスタンス名 ARace1 広さ 16

5. 1 1 オブジェクトデータ定義機能（その1）

コ マ ン ド 名 称		o b j s e t			
機 能 概 要	指定したオブジェクトのデータを定義する。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	o b j s e t			
	キ ー ボ ー ド	内 容	サ イ ズ	備 考	
		クラス名	8 0		
		インスタンス名	8 0		
言 語 イ ン タ フ ェ ー ス	定 義	int co_objset1(c_name, o_name, v_str)			
	引 数 名	1/0	型	サ イ ズ	備 考
	c _ n a m e	I	char*	8 0	クラス名
	o _ n a m e	I	char*	8 0	インスタンス名
	v _ s t r	O	VAR_10*	4	変数データ構造体へのポインター
備 考	0 : 正常終了 - 1 : クラス名がおかしい - 2 : インスタンス名がおかしい - 1 0 : インスタンスデータがおかしい				

実行例 (画面内容)



動作説明 (objset-1)

(図形表示ウィンドウに何も表示されていない)

・ Objsetボタンクリック

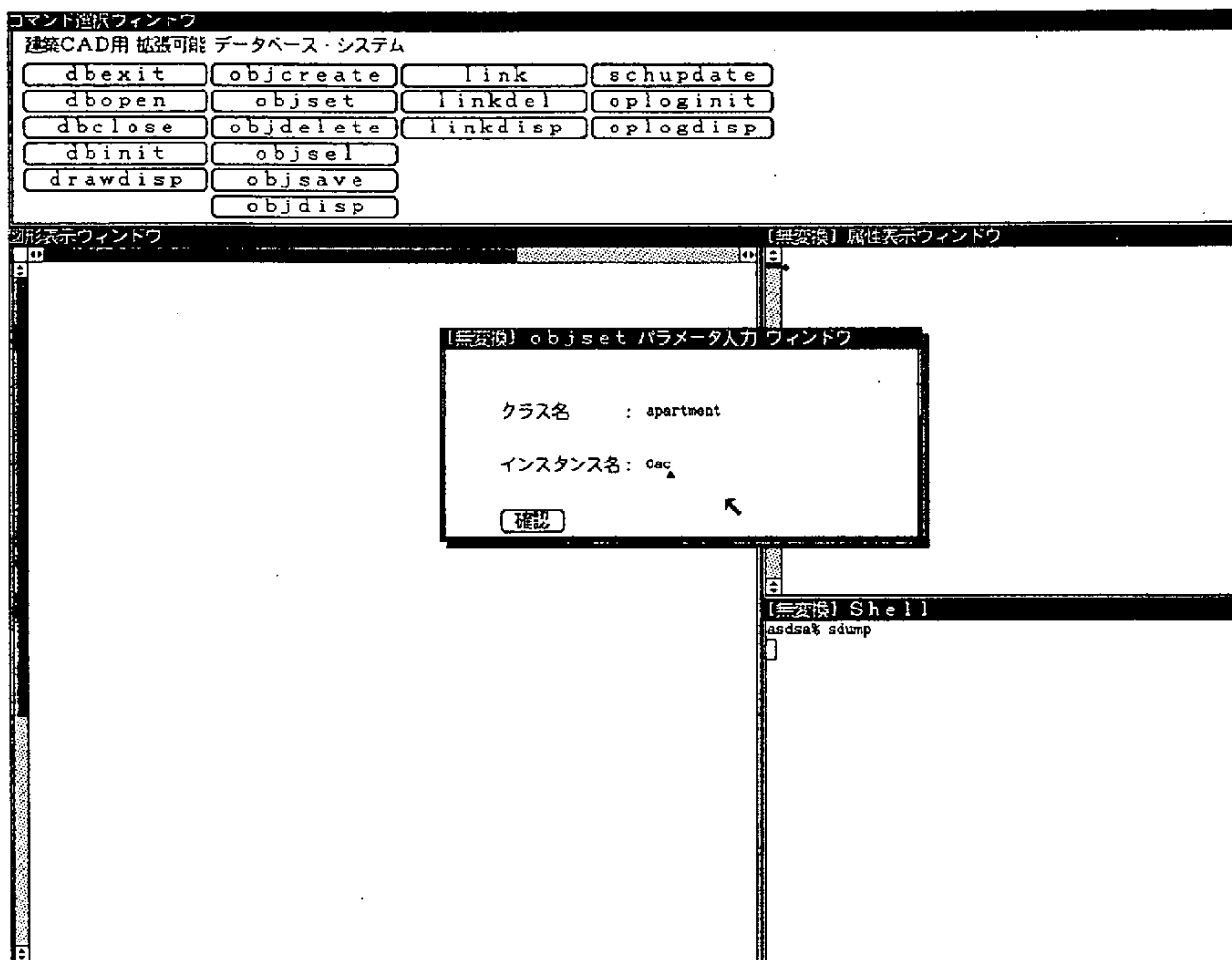
→パラメータ入力ウィンドウ表示

(内容) クラス名:

インスタンス名:

確認

実行例 (画面内容)



動作説明 (objset-2)

クラス名称 "apartment"、インスタンス名称 "0ac" を入力後、「確認」をクリック
→パラメータ入力ウィンドウは消える

確 認

5. 1 1 オブジェクトデータ定義機能 (その2)

コマンド名称		o b j s e t			
機能概要	指定したオブジェクトのデータを定義する。				
操作インタフェース	ボタン				
	キーボード	内 容	サイズ	備 考	
言語インタフェース	定義	int co_objset2(v_str)			
	引数名	I/O	型	サイズ	備 考
	v _ s t r	I	VAR_IO*	4	変数データ構造体へのポインター
備考	0 : 正常終了 - 1 : 登録ができない - 1 0 : インスタンスデータがおかしい				

実行例 (画面内容)

コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit	objcreate	link	schupdate
dbopen	objset	linkdel	oplogininit
dbclose	objdelete	linkdisp	oplogdisp
dbinit	objsel		
drawdisp	objsave		
	objdisp		

属性表示ウィンドウ

objset apartment 0ac

(無変換) objset インスタンス変数 設定 ウィンドウ

物件番号 : *

値段 (円) :

管理費 (円) :

専有面積 (m²) :

バルコニー面積 :

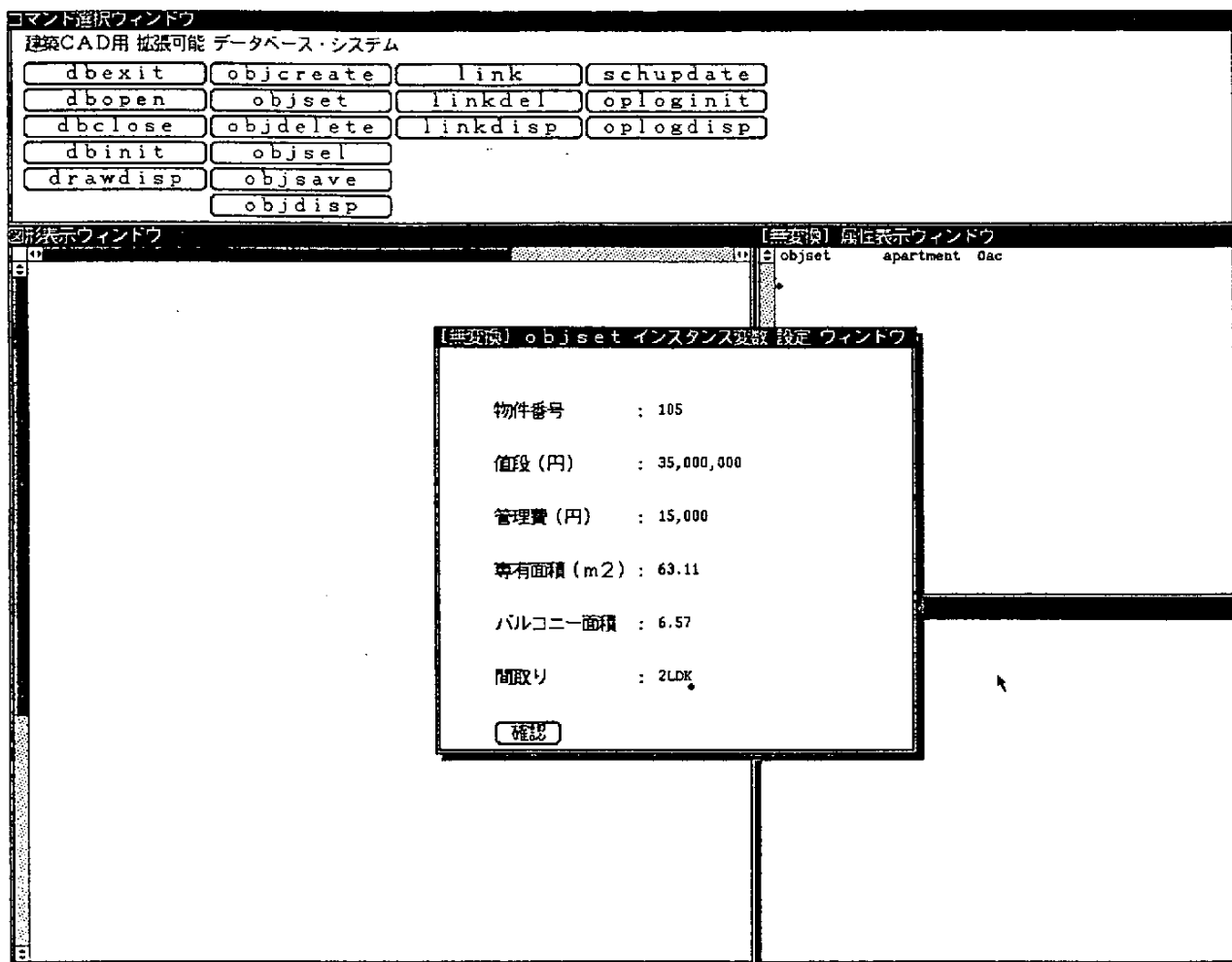
間取り :

確認

動作説明 (objset-3)

- 属性表示ウィンドウにエコー表示
"objset apartment 0ac"
 - インスタンス変数設定ウィンドウ表示
- 内容
- 物件番号 :
- 値段 (円) :
- 管理費 (円) :
- 専有面積 (m²) :
- バルコニー面積 :
- 間取り :

実行例 (画面内容)



動作説明 (objset-4)

インスタンス変数を下記のように設定して「確認」をクリック

物件番号 : 105
 値段 (円) : 35,000,000
 管理費 (円) : 15,000
 専有面積 (m²) : 63.11
 バルコニー面積 : 6.57
 間取り : 2LDK

→ インスタンス変数設定ウィンドウは消える

→ 属性表示ウィンドウに上記内容が表示される

(変数名 : 左ツメ 値 : 右ツメ)

実行例 (画面内容)

コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit

objcreate

link

schupdate

dbopen

objset

linkdel

oploginit

dbclose

objdelete

linkdisp

oplogdisp

dbinit

objsel

drawdisp

objsave

objdisp

表示ウィンドウ

属性表示ウィンドウ

objset

apartment

0ac

物件番号

:

105

値段(円)

:

35,000,000

管理費(円)

:

15,000

専有面積(m2)

:

63.11

バルコニー面積

:

6.57

間取り

:

2LDK

Shell

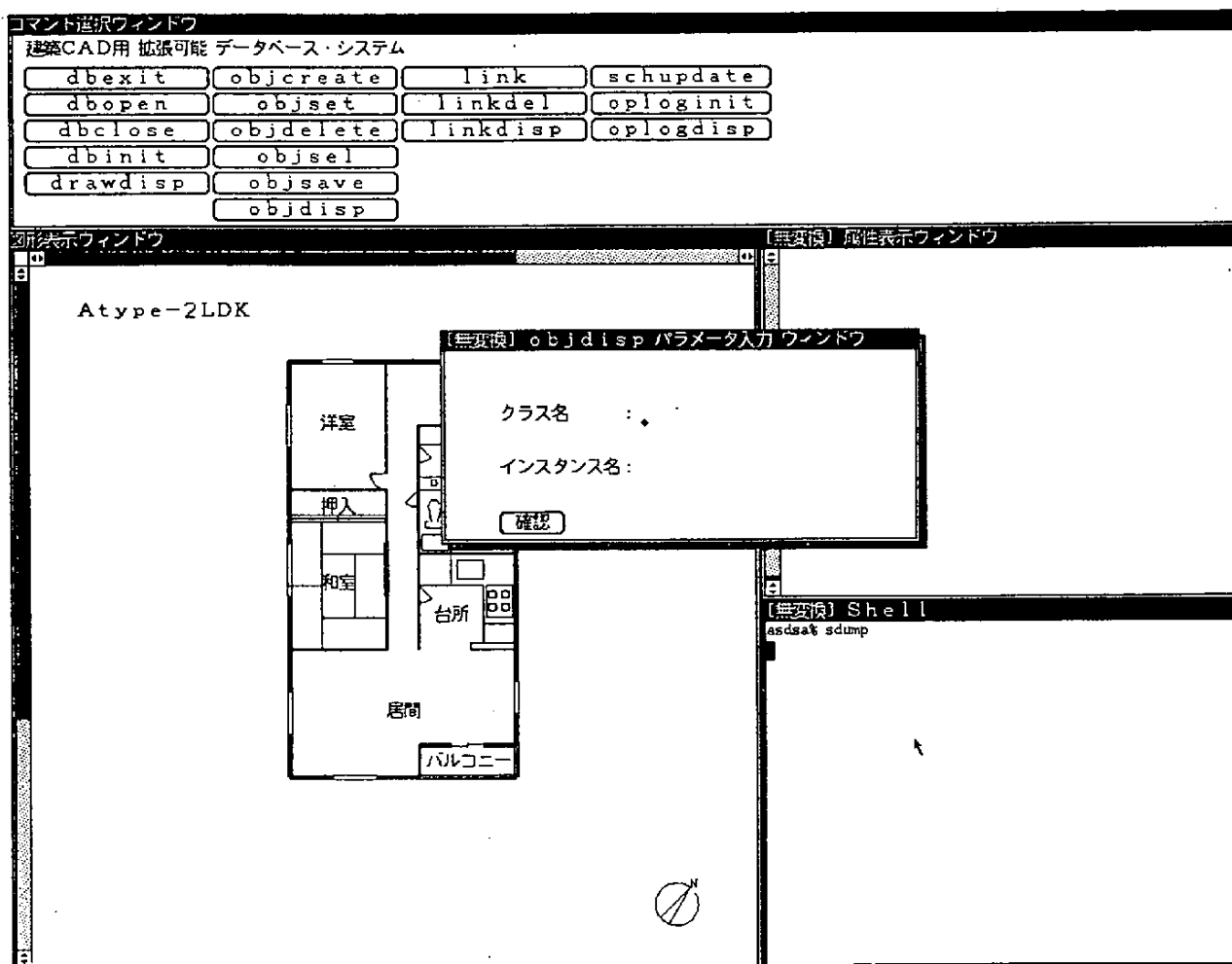
asdsa% sdump

動作説明 (objset-5)

5. 1 2 オブジェクト属性表示機能

コマンド名称		o b j d i s p			
機能概要	指定されたオブジェクトの各変数値を表示する。				
操作インタフェース	ボタン	o b j d i s p ボタンをクリック			
	キーボード	内 容	サイズ	備 考	
		クラス名	8 0		
		インスタンス名	8 0		
言語インタフェース	定義	i n t c o _ o b j d i s p (c _ n a m e , o _ n a m e)			
	引数名	I/O	型	サイズ	備 考
	c _ n a m e	I	char*	8 0	クラス名
	o _ n a m e	I	char*	8 0	インスタンス名
備考	0 : 正常終了 - 1 : クラス名がおかしい - 2 : インスタンス名がおかしい - 1 0 : インスタンス・データがおかしい SYSTEM_AREA の値は表示されない				

実行例 (画面内容)



動作説明 (objdisp-1)

(図形表示ウィンドウには0acの物件が表示されている)

objdispボタンクリック

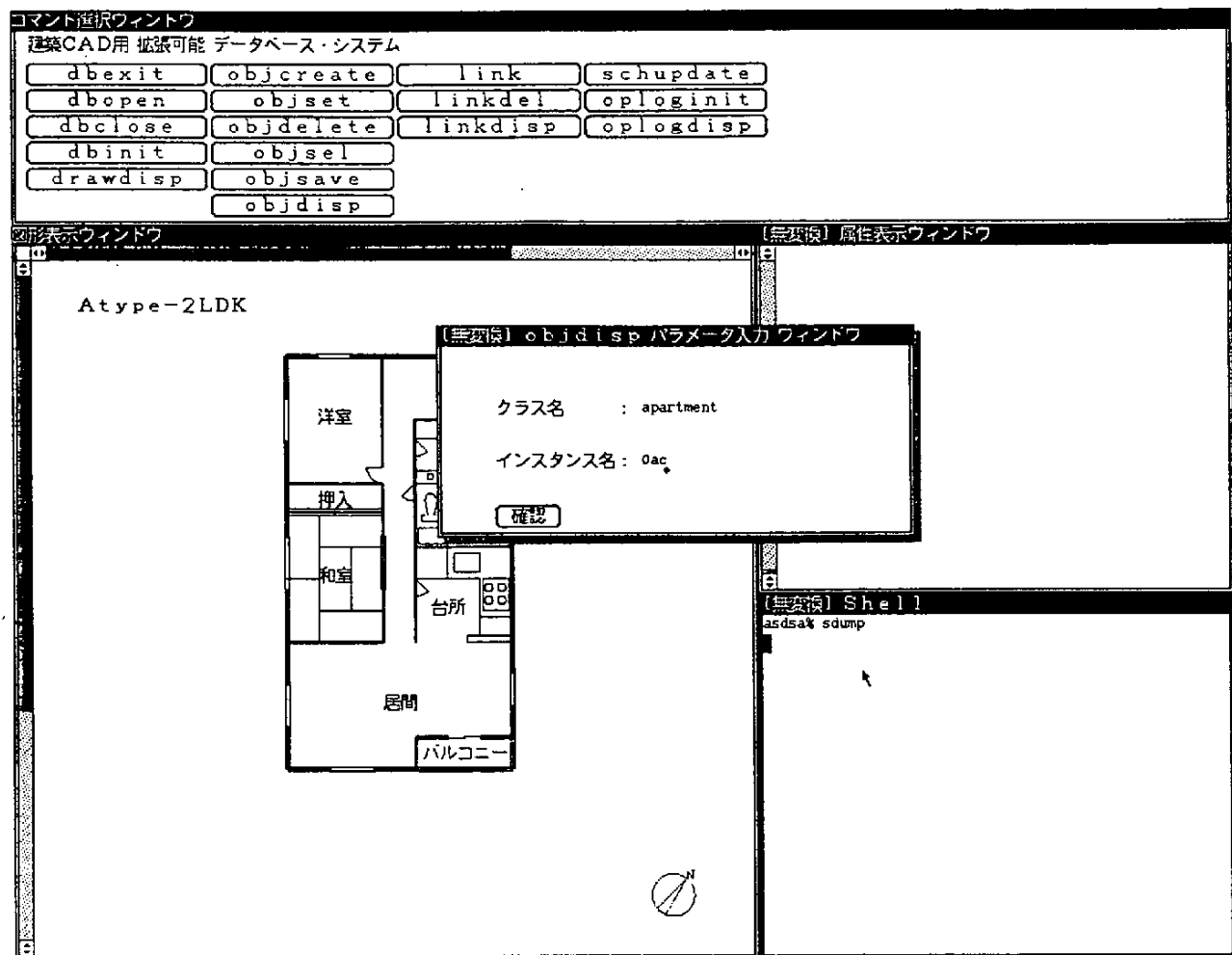
→パラメータ入力ウィンドウ表示

(内容) クラス名:

インスタンス名:

確 認

実行例 (画面内容)



動作説明 (objdisp-2)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

クラス名称 : apartment

インスタンス名称 : 0ac

→パラメータ入力ウィンドウは消える

実行例 (画面内容)

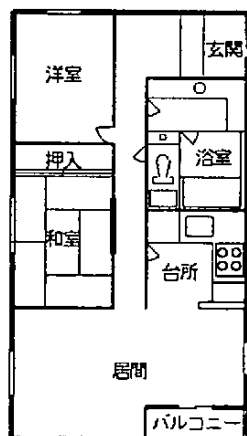
コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit	objcreate	link	schupdate
dbopen	objset	linkdel	oplogininit
dbclose	objdelete	linkdisp	oplogdisp
dbinit	objsel		
drawdisp	objsave		
	objdisp		

表示ウィンドウ

Atype-2LDK



【無変換】属性表示ウィンドウ

```
objdisp apartment 0ac
# 物件番号      : 105
# 値段(円)      : 35,000,000
# 管理費(円)    : 15,000
# 専有面積(m2)  : 63.11
# バルコニー面積 : 6.57
# 間取り        : 2LDK
```

【無変換】Shell

```
asdse% sdump
```

動作説明 (objdisp-3)

→属性表示ウィンドウにエコー表示

“objdisp apartment 0ac”

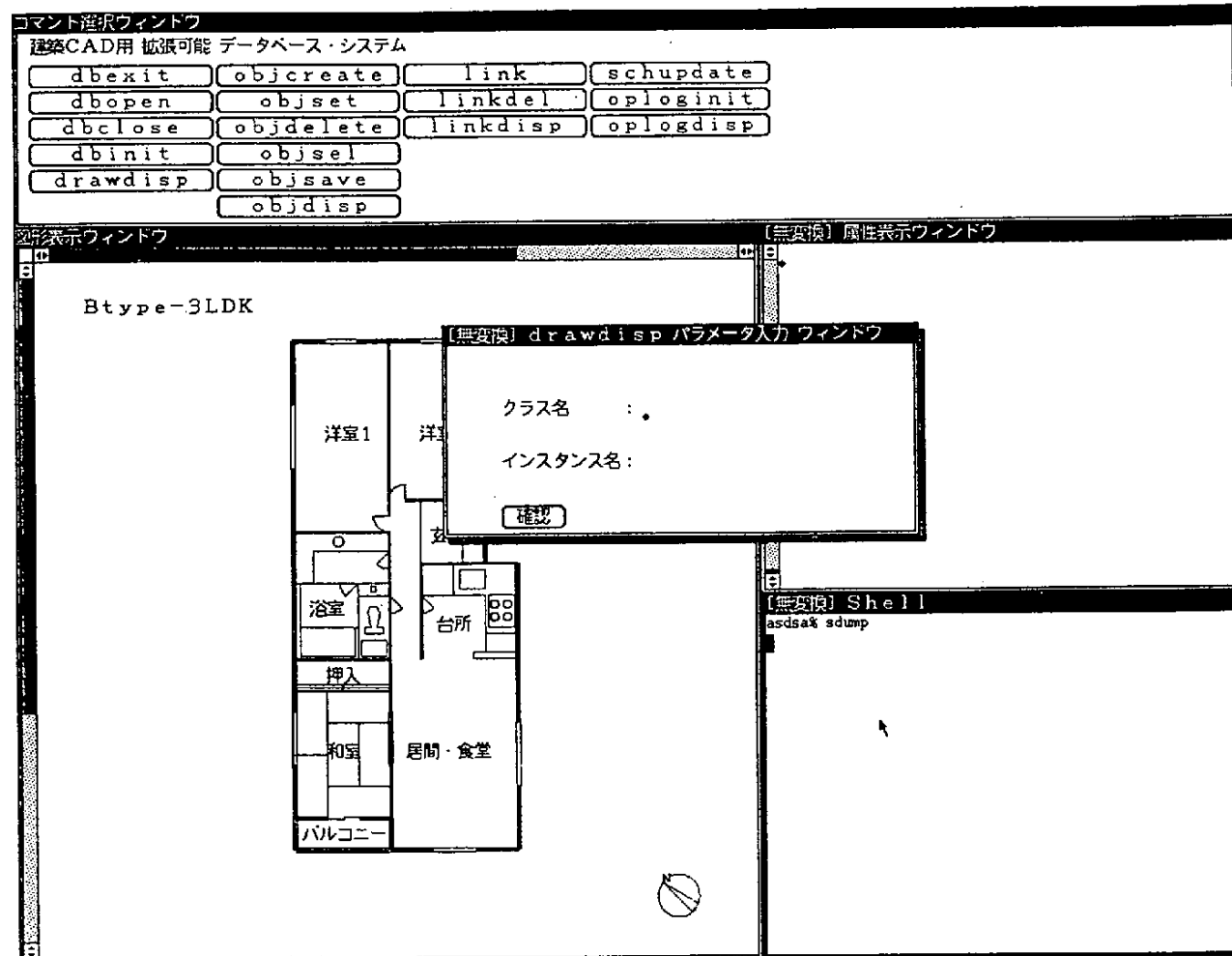
→属性表示ウィンドウに属性を表示

物件番号	105
値段	35,000,000
管理費	15,000
専有面積	63.11
バルコニー面積	6.57
間取り	2LDK

5. 1 3 形状表示機能

コマンド名称		drawdisp			
機能概要	指定された形状を表示する。				
操作インタフェース	ボタン	drawdisp ボタンをクリック			
	キーボード	内 容	サイズ	備 考	
		クラス名	80		
		インスタンス名	80		
言語インタフェース	定義	int co_drawdisp(c_name, o_name)			
	引数名	I/O	型	サイズ	備 考
	c_name	I	char*	80	クラス名
	o_name	I	char*	80	インスタンス名
備考	0: 正常終了 -1: クラス名がおかしい -2: インスタンス名がおかしい -3: 作画できるインスタンスではない -10: インスタンスデータがおかしい				

実行例 (画面内容)



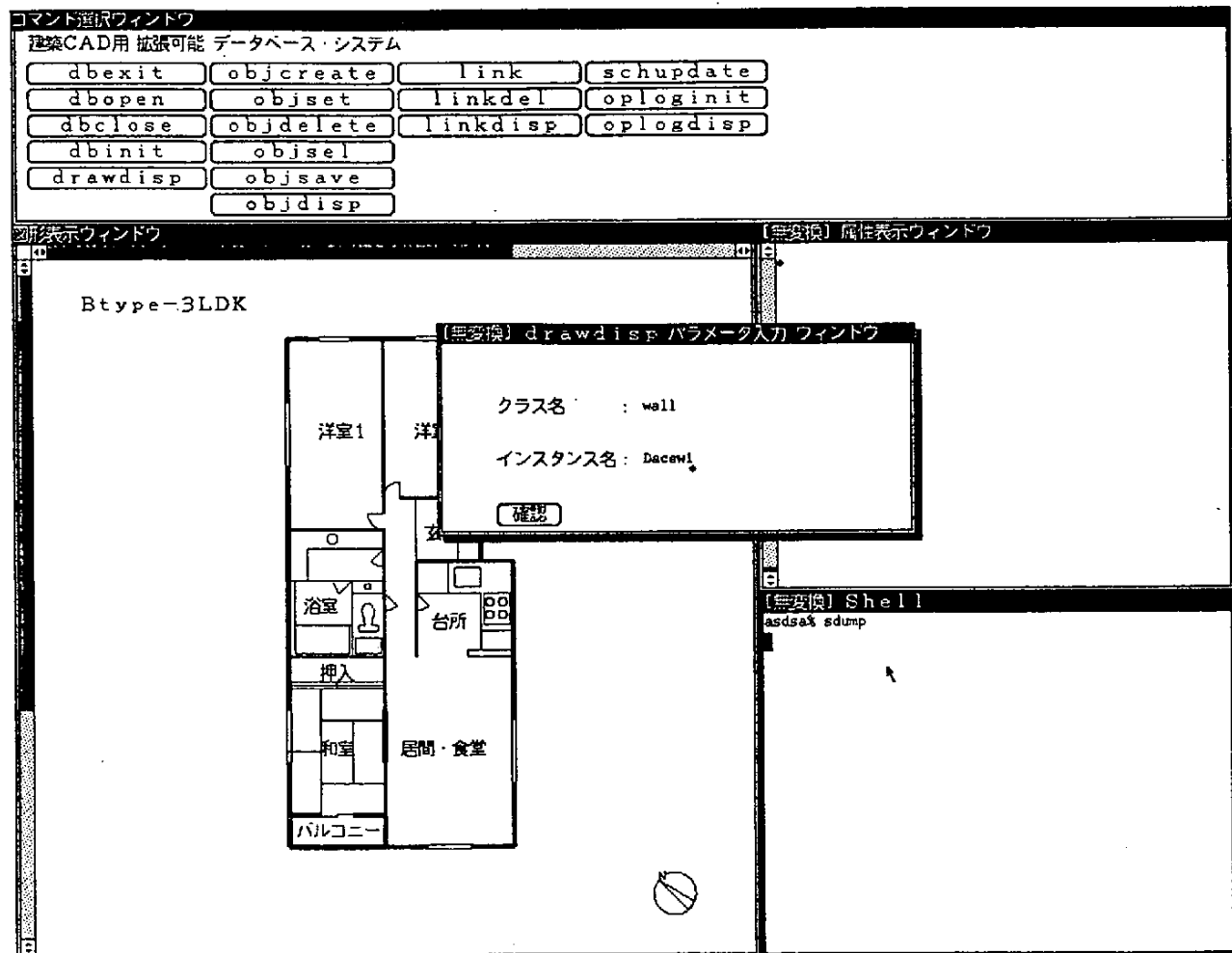
動作説明 (drawdisp-1)

(図形表示ウィンドウには0abの物件が表示されている)

- ・ drawdisp ボタンクリック
 - パラメータ入力ウィンドウ表示
 - (内容) クラス名 :
 - インスタンス名 :

確 認

実行例 (画面内容)



動作説明 (drawdisp-2)

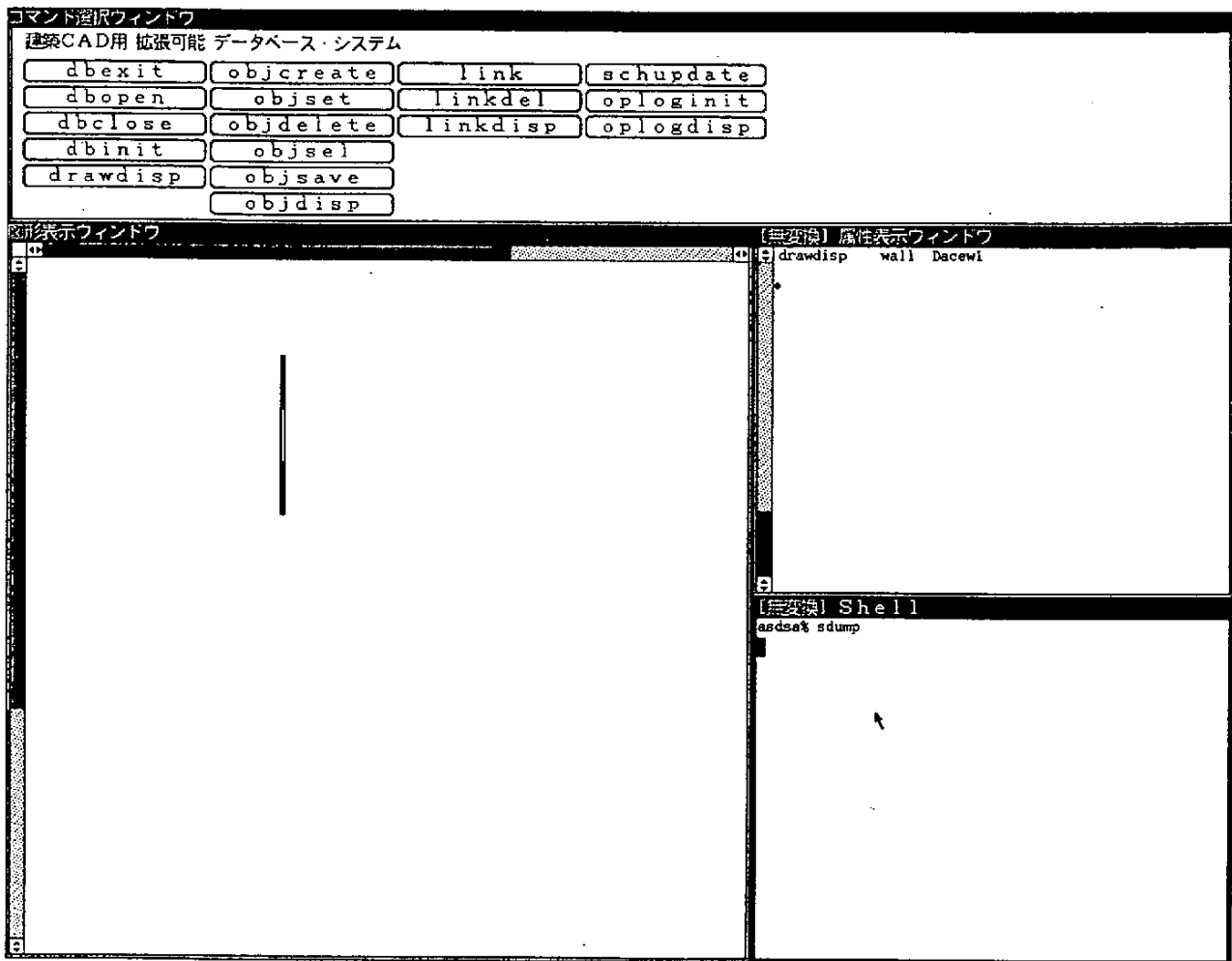
パラメータ入力ウィンドウに以下の値を入力後、「確認」をクリック

クラス名 : wall

インスタンス名 : Dacewl

→パラメータ入力ウィンドウは消える

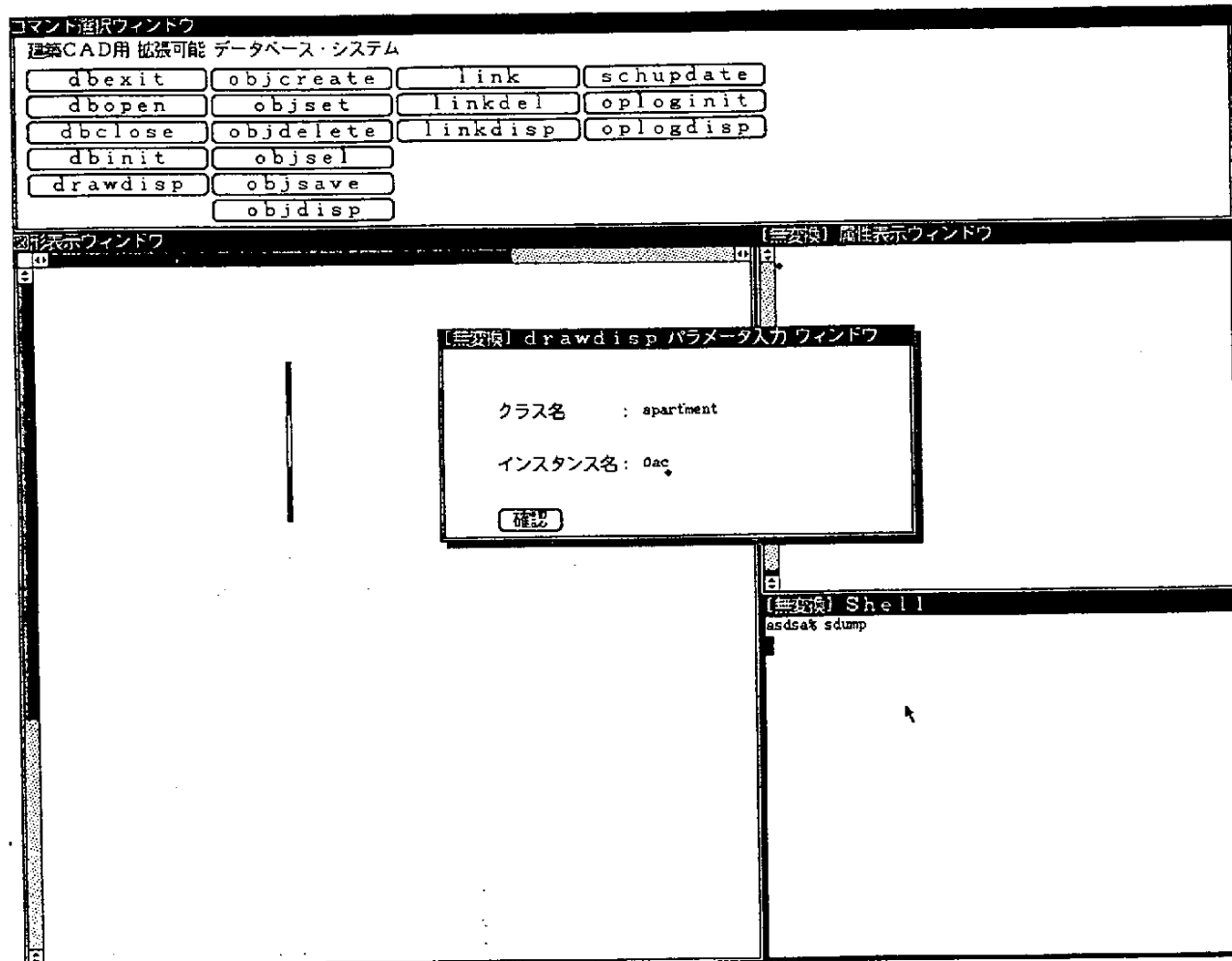
実行例 (画面内容)



動作説明 (drawdisp-3)

- 属性表示ウィンドウにエコー表示
"drawdisp wall Dacew1"
- 図形表示ウィンドウに壁Dacew1が表示される

実行例 (画面内容)



動作説明 (drawdisp-4)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

クラス名 : apartment

インスタンス名 : 0ac

→パラメータ入力ウィンドウは消える

実行例 (画面内容)

コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit	objcreate	link	schupdate
dbopen	objset	linkdel	oploginit
dbclose	objdelete	linkdisp	oplogdisp
dbinit	objsel		
drawdisp	objsave		
	objdisp		

図形表示ウィンドウ

Atype-2LDK

属性表示ウィンドウ

drawdisp apartment 0ac

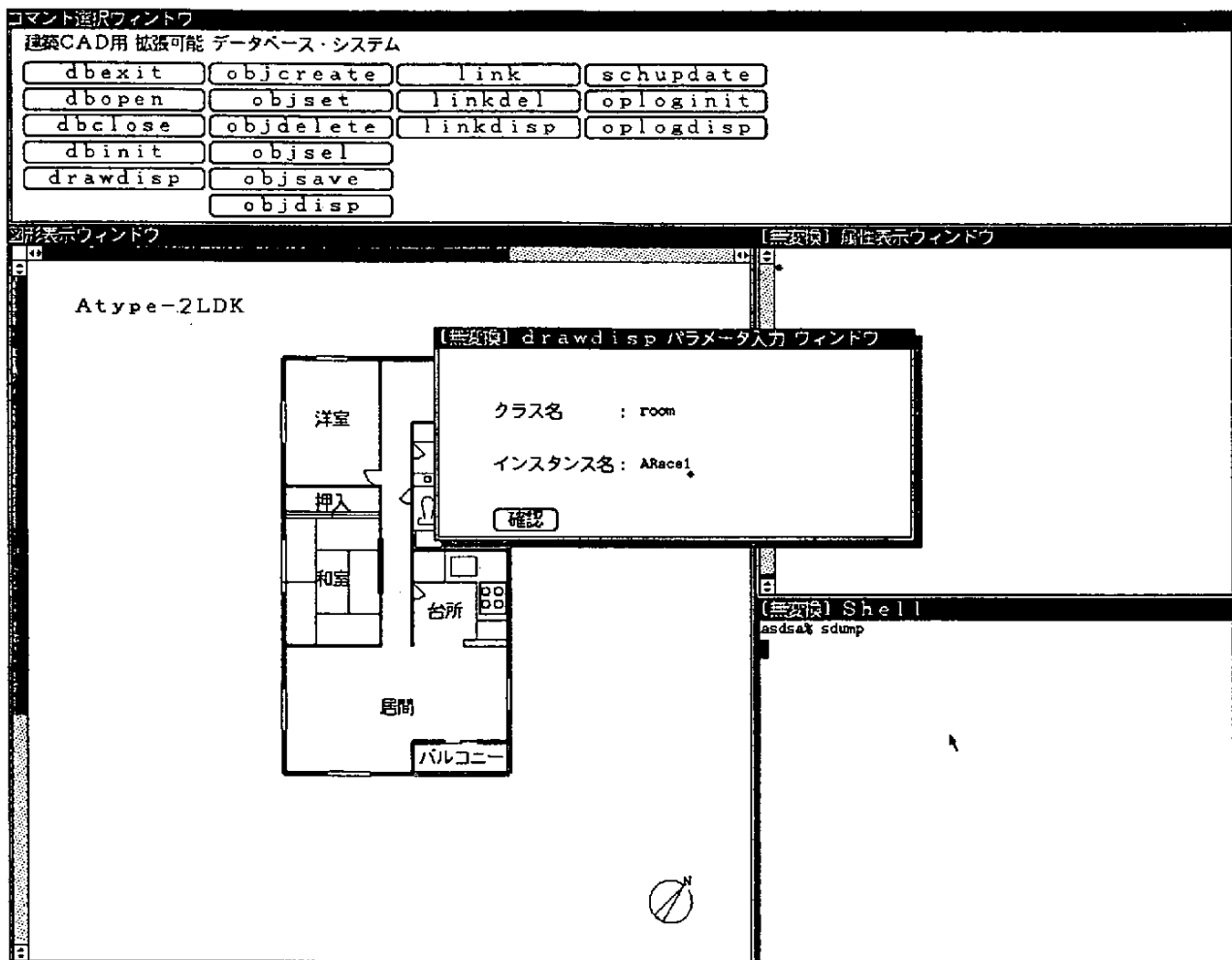
Shell

asdsa% sdump

動作説明 (drawdisp-5)

- 属性表示ウィンドウにエコー表示
"drawdisp apartment 0ac"
- 図形表示ウィンドウに物件0acが表示される

実行例 (画面内容)



動作説明 (drawdisp-6)

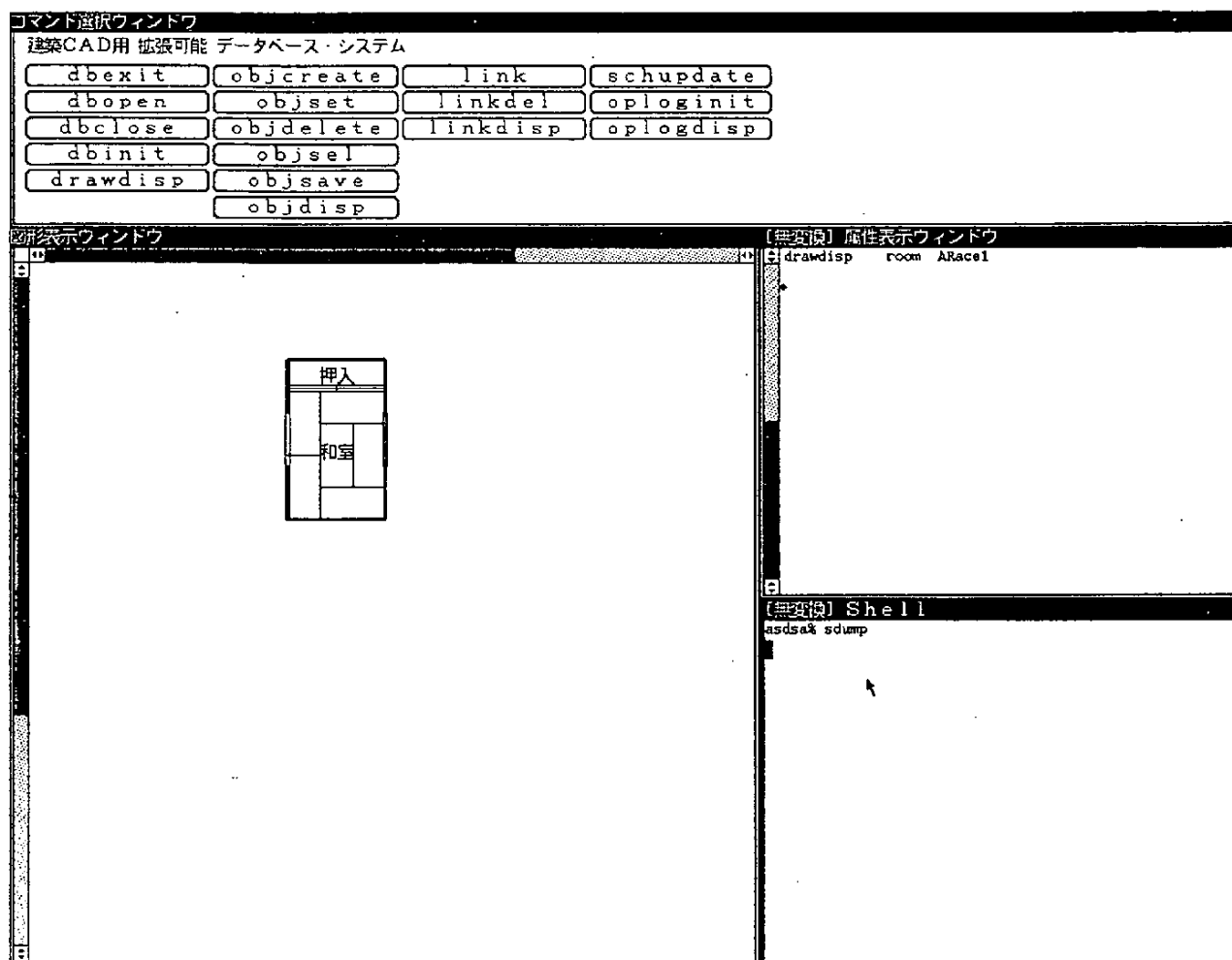
パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

クラス名 : room.

インスタンス名 : ARace1

→パラメータ入力ウィンドウは消える

実行例 (画面内容)



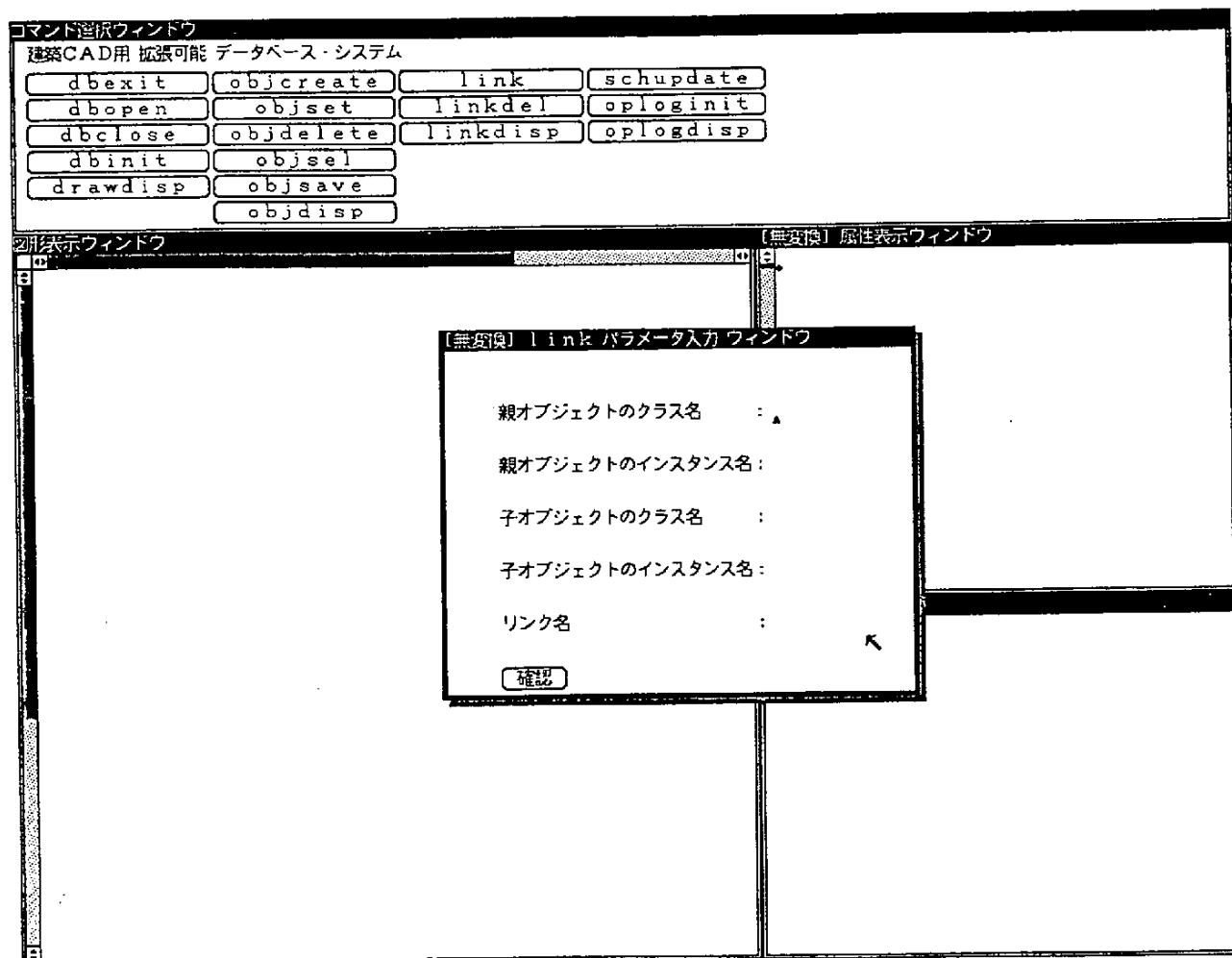
動作説明 (drawdisp-7)

- 属性表示ウィンドウにエコー表示
"drawdisp room ARace1"
- 図形表示ウィンドウに部屋ARace1が表示される

5. 1 4 リンク生成機能

コ マ ン ド 名 称		l i n k			
機 能 概 要	2つのオブジェクトのリンク付けを行う。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	l i n k ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		親オブジェクトのクラス名	8 0		
		親オブジェクトのインスタンス 名	8 0		
		子オブジェクトのクラス名	8 0		
		子オブジェクトのインスタンス 名	8 0		
リンク名	8 0				
言 語 イ ン タ フ ェ ー ス	定義	int co_link(pc_name, po_name, cc_name, co_name, l_name)			
	引数名	1/0	型	サイズ	備 考
	pc_name	1	char*	8 0	親オブジェクトのクラス名
	po_name	1	char*	8 0	親オブジェクトのインスタンス 名
	cc_name	1	char*	8 0	子オブジェクトのクラス名
	co_name	1	char*	8 0	子オブジェクトのインスタンス 名
	l_name	1	char*	8 0	リンク名
備 考	1: 新リンクの生成（正常終了） -3: 子オブジェクト・クラス名がおかしい 0: 旧リンクへの 接続（正常終了） -4: 子オブジェクトのインスタンス 名がおかしい -1: 親オブジェクト・クラス名がおかしい -5: リンク 名がおかしい -2: 親オブジェクトのインスタンス 名がおかしい -10:				

実行例 (画面内容)



動作説明 (link-1)

(図形表示ウィンドウには何も表示されていない)

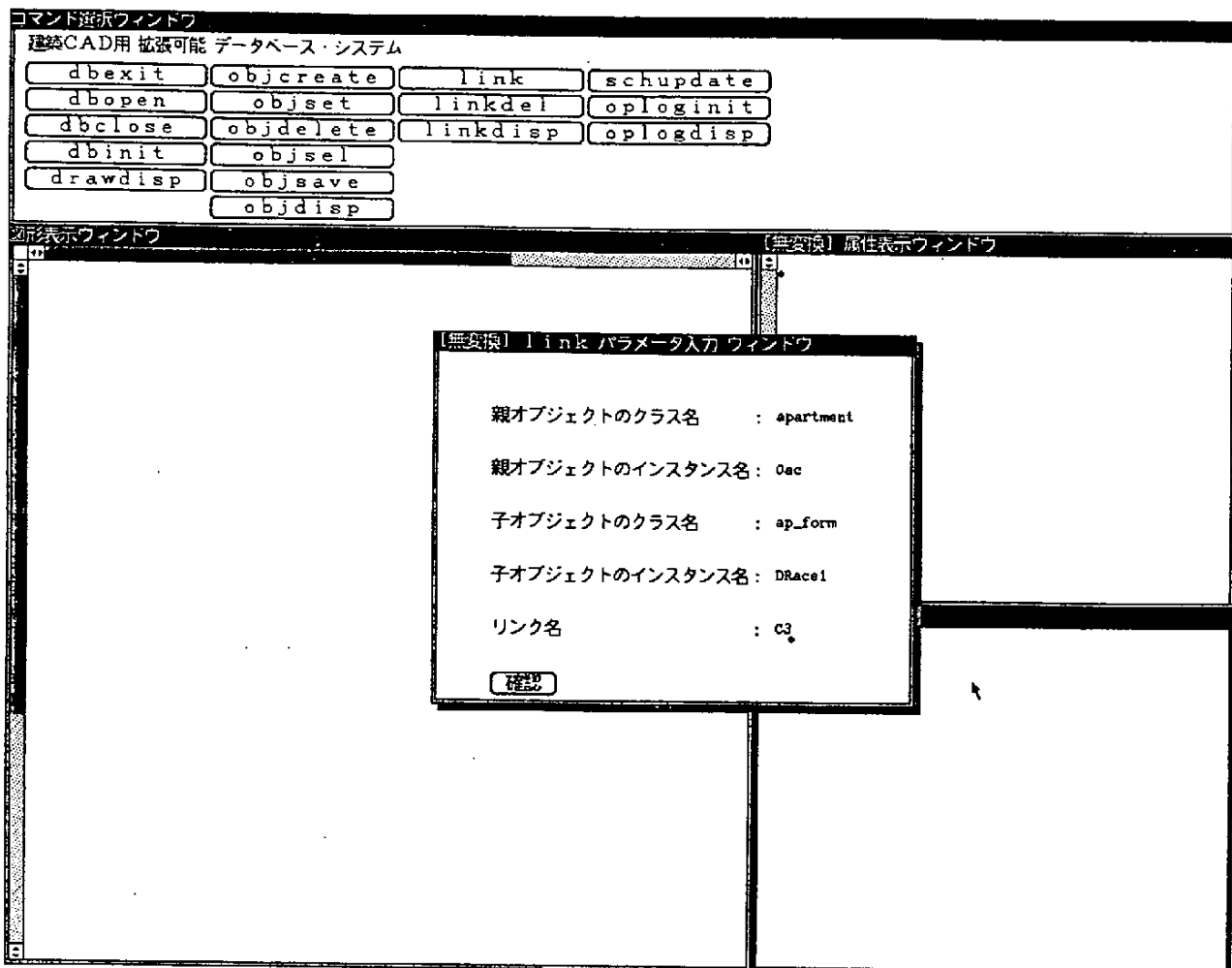
・linkボタンクリック

→パラメータ入力ウィンドウ表示

内容	親オブジェクトのクラス名 :
	親オブジェクトのインスタンス名 :
	子オブジェクトのクラス名 :
	子オブジェクトのインスタンス名 :
	リンク名 :

確 認

実行例 (画面内容)



動作説明 (link-2)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

親オブジェクトのクラス名 : apartment

親オブジェクトのインスタンス名 : 0ac

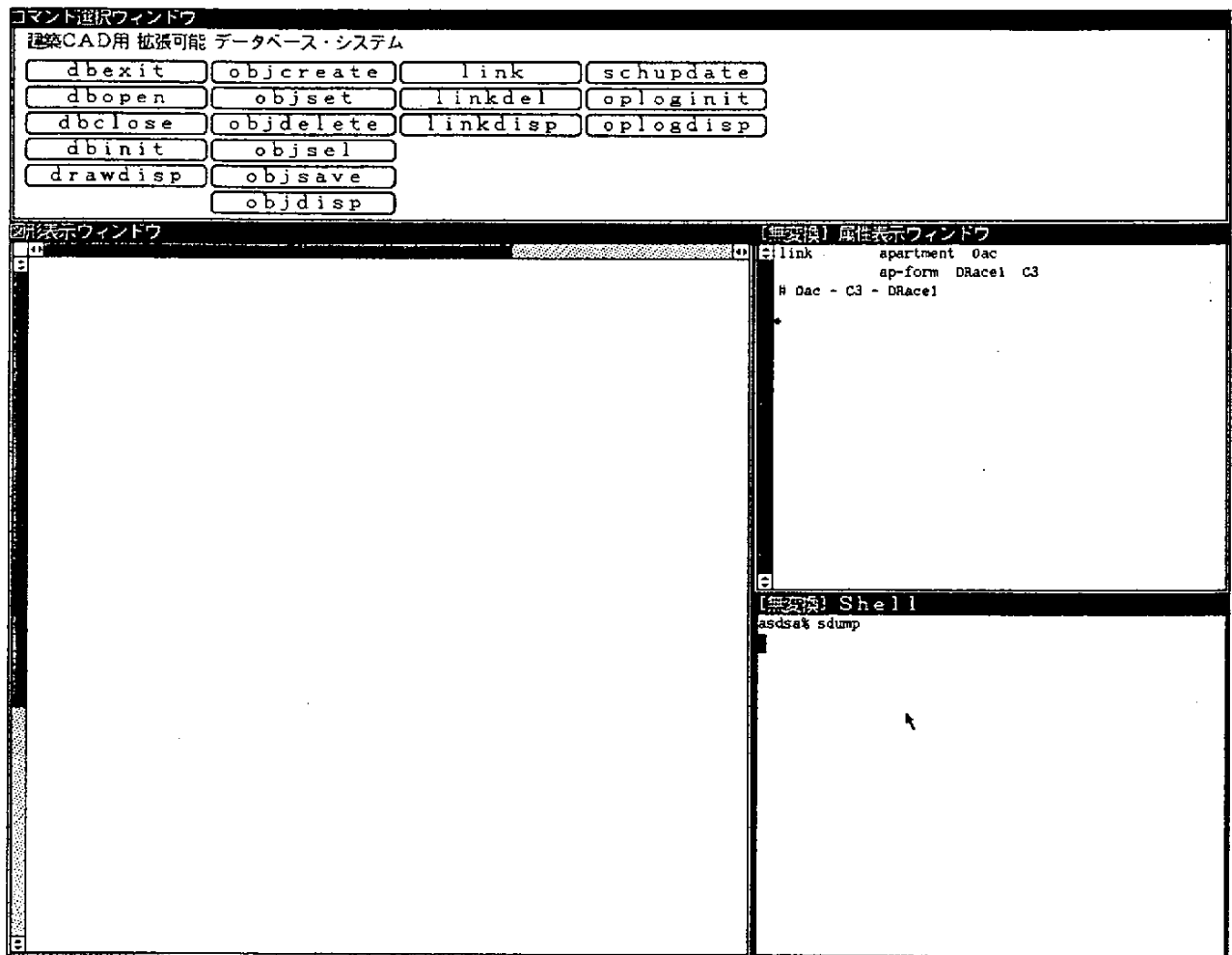
子オブジェクトのクラス名 : ap-form

子オブジェクトのインスタンス名 : DRace1

リンク名 : C3

→パラメータ入力ウィンドウは消える

実行例 (画面内容)



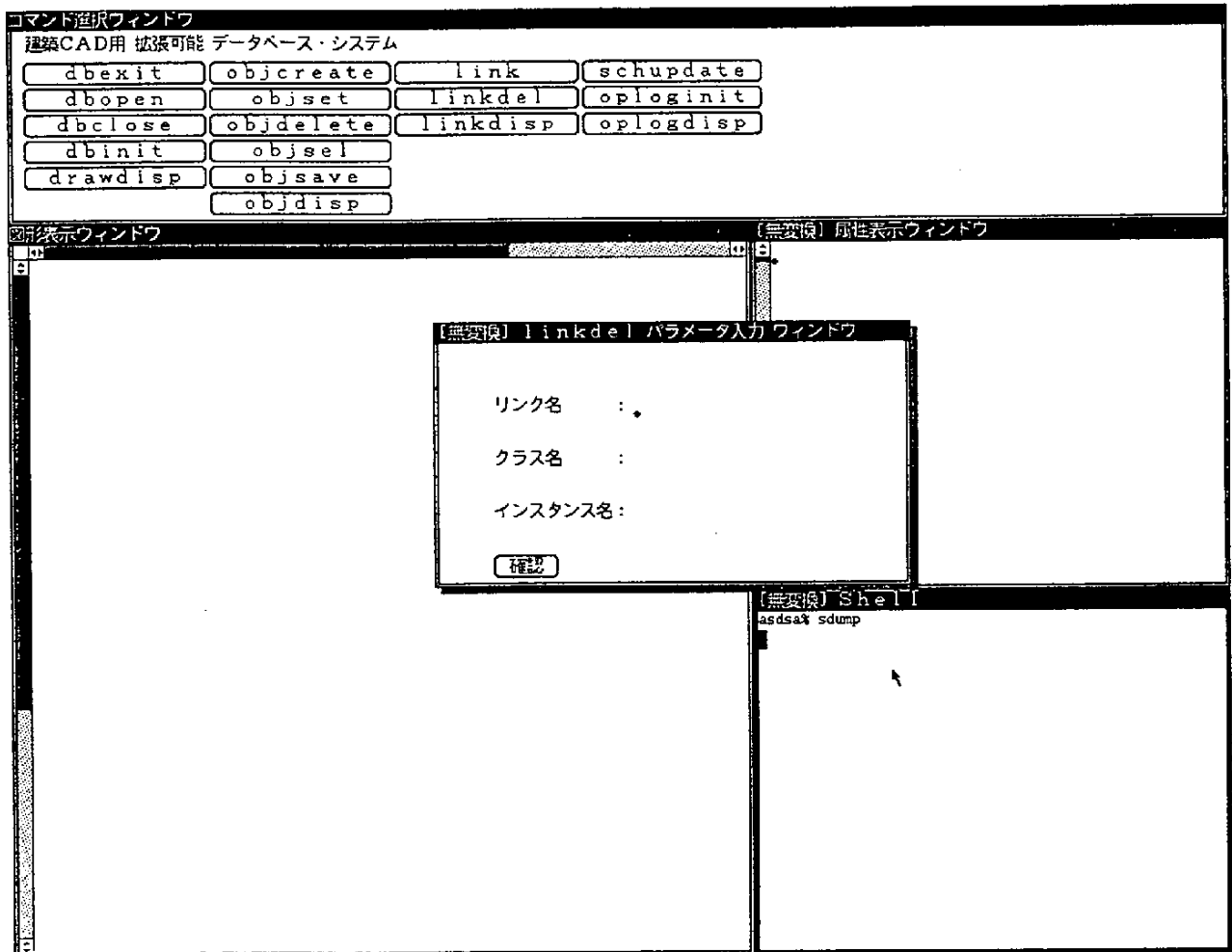
動作説明 (link-3)

- 属性表示ウィンドウにエコー表示
 - ・"link apartment 0ac ap-form DRace1 C3"
- 属性表示ウィンドウにリンク内容表示
 - 0ac-C3-DRace1

5. 1 5 リンク削除機能

コマンド名称		linkdel			
機能概要	正規表現で指定されたリンクから、正規表現で指定された、子オブジェクトをはずす。				
操作インタフェース	ボタン	linkdel ボタンをクリック			
	キーボード	内 容	サイズ	備 考	
		リンク名 クラス名 インスタンス名	80 80 80	正規表現も可能 正規表現も可能 正規表現も可能	
言語インタフェース	定義	int co_linkdel (l_name, c_name, o_name)			
	引数名	I/O	型	サイズ	備 考
	l_name	I	char*	80	リンク名 (正規表現)
	c_name	I	char*	80	クラス名 (正規表現)
	o_name	I	char*	80	インスタンス名 (正規表現)
備考	正 規 表 現 で 指 定 さ れ た リンク から、正 規 表 現 で 指 定 さ れ た、子 オブ ジェ ク ト を はず す。 正 規 表 現 で 使 用 で き る の は * と ? のみ				

実行例 (画面内容)



動作説明 (linkdel-1)

(図形表示ウィンドウには何も表示されていない)

・ linkdel ボタンクリック

→パラメータ入力ウィンドウ表示

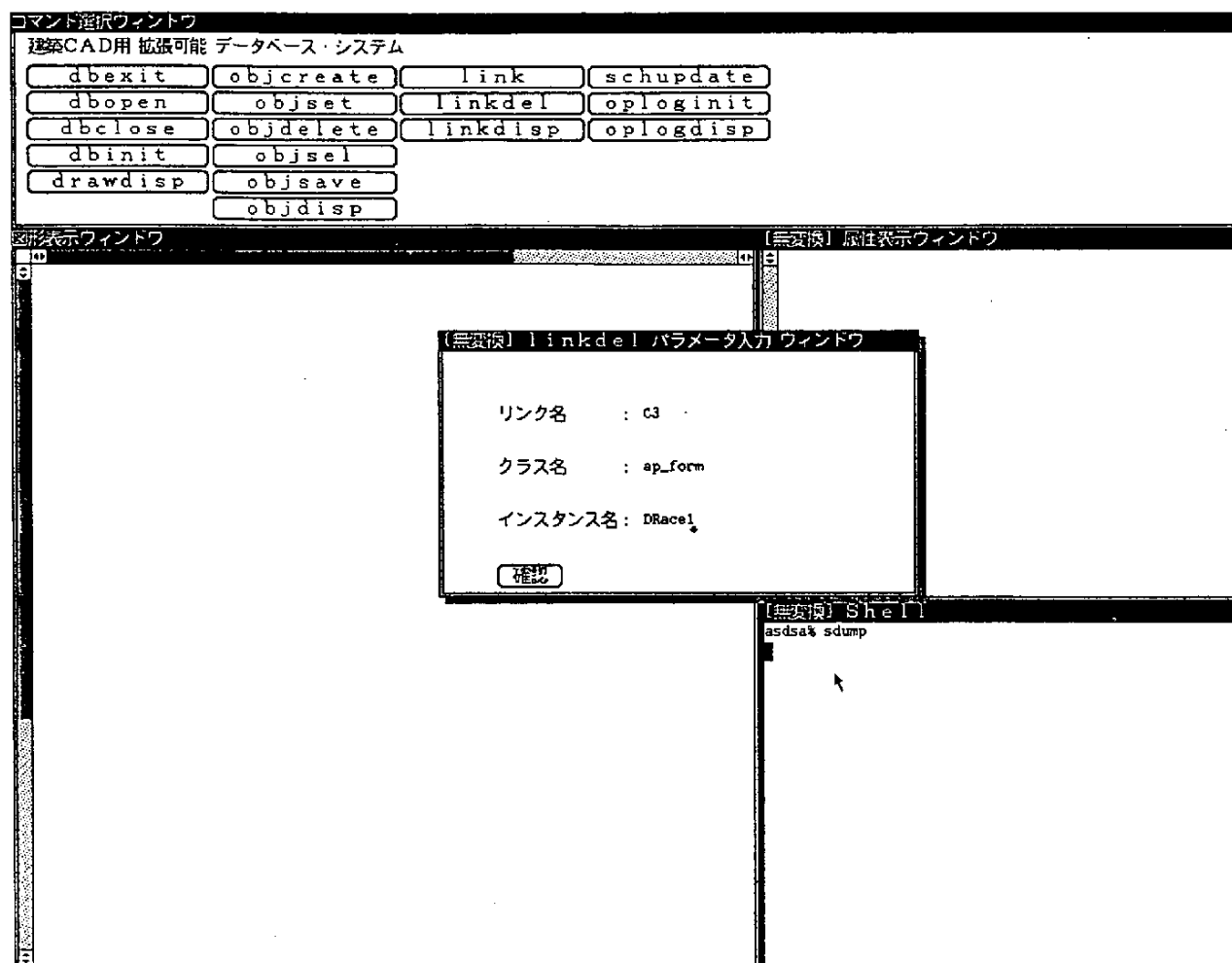
(内容) リンク名 :

 クラス名 :

 インスタンス名 :

確 認

実行例 (画面内容)



動作説明 (linkdel-2)

パラメータ入力ウィンドウに以下の値を入力し、「確認」をクリック

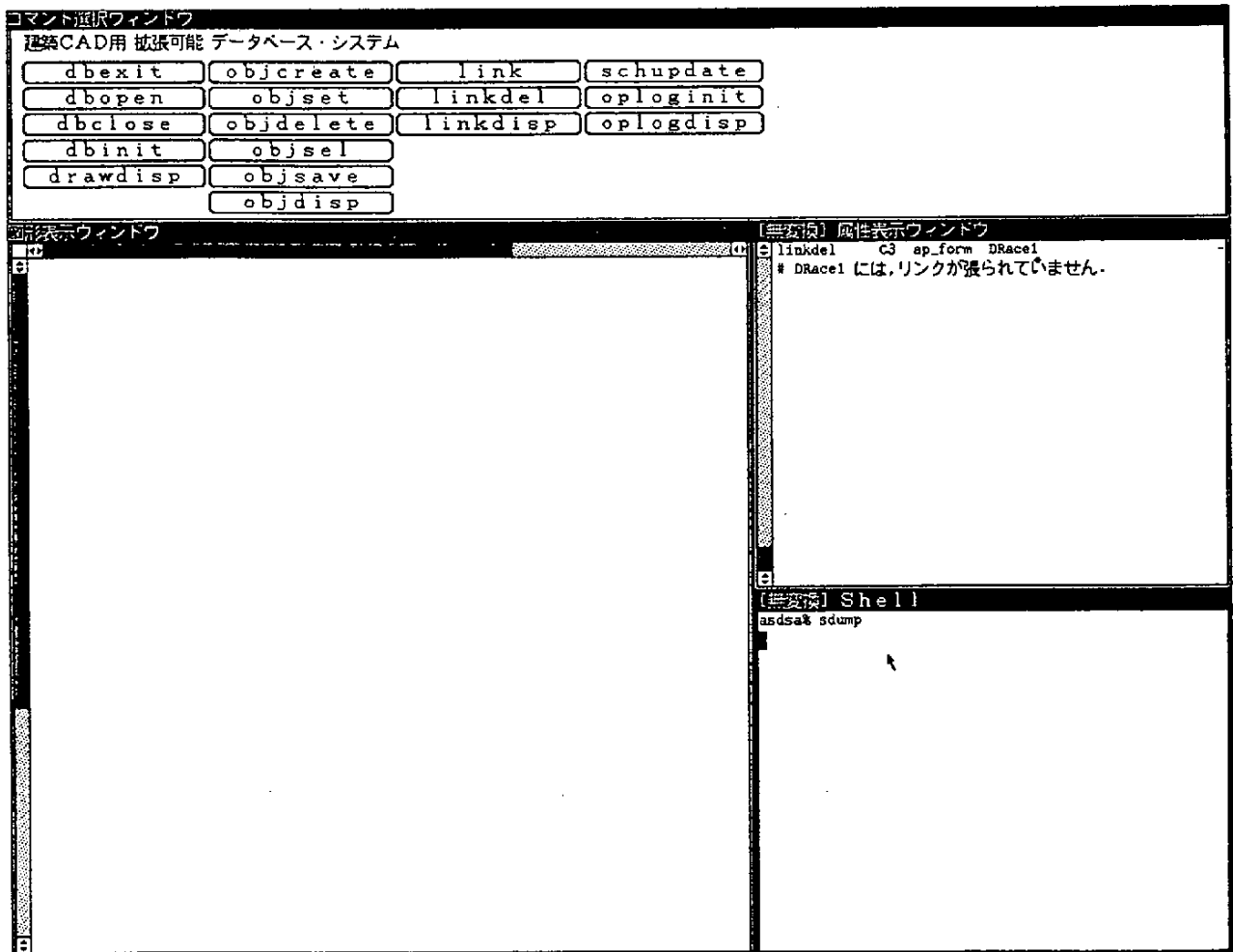
リンク名 : C3

クラス名 : ap_form

インスタンス名 : DRace1

→パラメータ入力ウィンドウは消える

実行例 (画面内容)



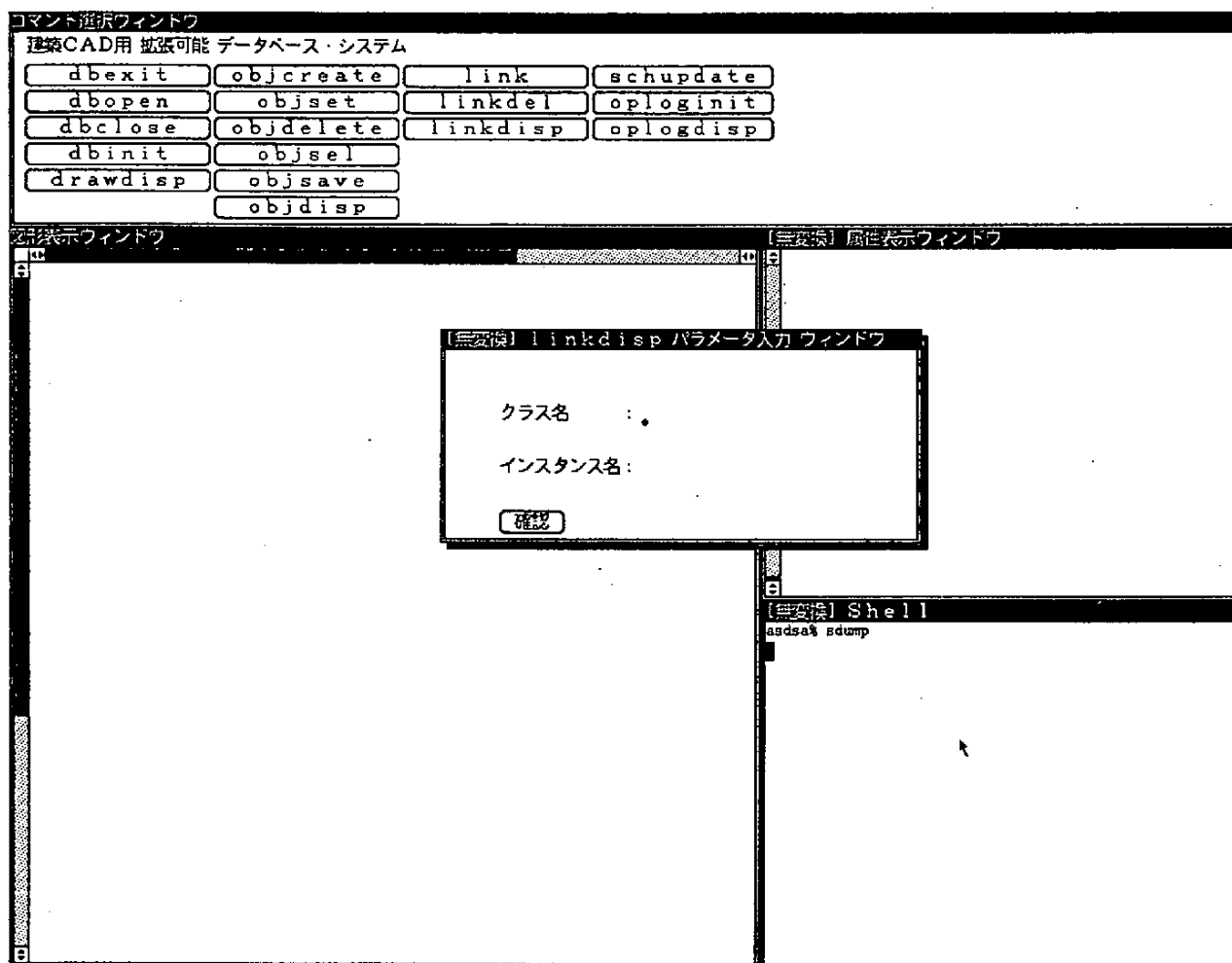
動作説明 (linkdel-3)

- 属性表示ウィンドウにエコー表示
"linkdel C3 ap-form DRace1"
- 属性表示ウィンドウにリンク内容表示
"DRace1 にはリンクが張られていません"

5. 1 6 リンク表示機能

コ マ ン ド 名 称		l i n k d i s p					
機 能 概 要	指定されたオブジェクトのリンク状態を表示する。						
操 作 イ ン タ フ ェ ー ス	ボ タ ン	l i n k d i s p					ボタンをクリック
	キ ー ボ ー ド	内 容		サイズ	備 考		
		クラス名		80			
		インスタンス名		80			
言 語 イ ン タ フ ェ ー ス	定 義	i n t c o _ l i n k d i s p (c _ n a m e , o _ n a m e)					
	引数名		I/O	型	サイズ	備 考	
	c _ n a m e		I	char*	80	クラス名	
	o _ n a m e		I	char*	80	インスタンス名	
備 考	0:正常終了 -10: -1:クラス 名がおかしい -15:インスタンス・データがおかしい -2:インスタンス名がおかしい						

実行例 (画面内容)



動作説明 (linkdisp-1)

(図形表示ウィンドウには何も表示されていない)

- ・ linkdispボタンクリック
 - パラメータ入力ウィンドウ表示
- (内容) クラス名 :
 インスタンス名 :

確 認

実行例 (画面内容)



動作説明 (linkdisp-2)

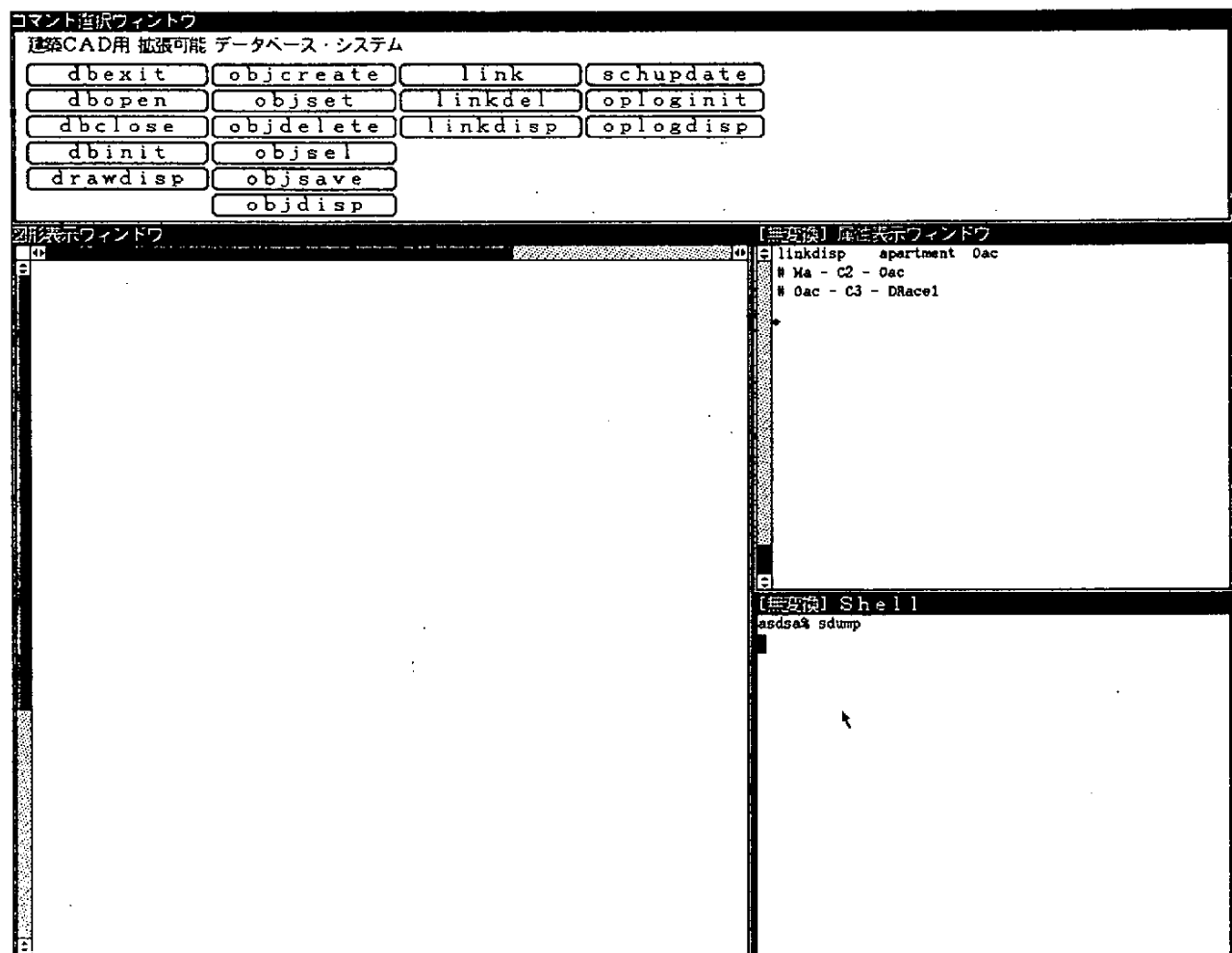
パラメータ入力ウィンドウに以下の値を入力後、「確認」をクリック

クラス名 : apartment

インスタンス名 : 0ac

→パラメータ入力ウィンドウは消える

実行例 (画面内容)



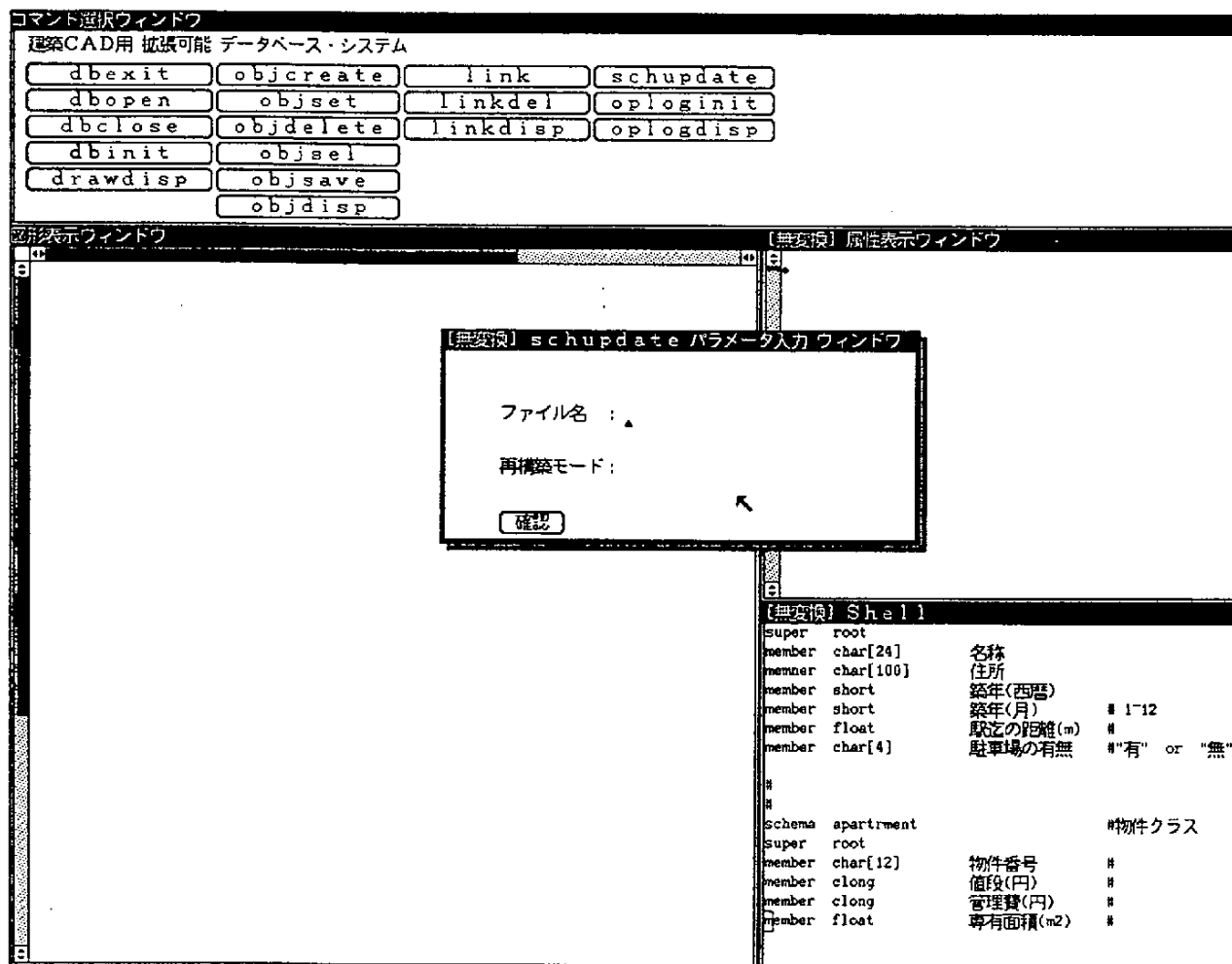
動作説明 (linkdisp-3)

- 属性表示ウィンドウにエコー表示
- 属性表示ウィンドウにリンク内容表示
 - Ma-C2-0ac
 - 0ac-C3-DRace1

5. 17 スキーマ更新機能

コ マ ン ド 名 称		s c h u p d a t e			
機 能 概 要	指定されたスキーマファイルより新しいスキーマを生成し modeによって、全インスタンスの再構築を行う。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	s c h u p d a t e ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		ファイル名 再構築モード	8 0 2	インスタンスの再構築モード 0 : しない 1 : する	
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ s c h u p d a t e (f i l e n a m e , m o d e)			
	引数名	I/O	型	サイズ	備 考
	f i l e n a m e	I	char*	8 0	ファイル名
	m o d e	I	short*	2	インスタンスの再構築モード 0 : しない 1 : する
備 考	0 : 正常終了 - 1 0 : スキーマ・ファイルにミスがある - 1 1 : スキーマが重複する				

実行例 (画面内容)



動作説明 (schupdate-1)

(図形表示ウィンドウには何も表示されていない)
 (shellはスキーマファイルをエディタで開いた状態)

- ・schupdate ボタンクリック
 - パラメータ入力ウィンドウ表示
- (内容) ファイル名:
 再構築モード:

確 認

実行例 (画面内容)

コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit	objcreate	link	schupdate
dbopen	objset	linkdel	oplogininit
dbclose	objdelete	linkdisp	oplogdisp
dbinit	objsel		
drawdisp	objsave		
	objdisp		

表示ウィンドウ

【無変換】 schupdate パラメータ入力ウィンドウ

ファイル名 : schema

再構築モード : 1

確認

【無変換】 Shell

super	root		
member	char[24]	名称	
member	char[100]	住所	
member	short	築年(西暦)	
member	short	築年(月)	# 1~12
member	float	駅迄の距離(m)	#
member	char[4]	駐車場の有無	# "有" or "無"
#			
#			
schema	apartment		#物件クラス
super	root		
member	char[12]	物件番号	#
member	clong	値段(円)	#
member	clong	管理費(円)	#
member	float	専有面積(m2)	#

動作説明 (schupdate-2)

ファイル名 : schema
再構築モード : 1

実行例 (画面内容)

コマンド選択ウィンドウ

建築CAD用 拡張可能 データベース・システム

dbexit	objcreate	link	schupdate
dbopen	objset	linkdel	oploginit
dbclose	objdelete	linkdisp	oplogdisp
dbinit	objsel		
drawdisp	objsave		
	objdisp		

表示ウィンドウ

【無交換】属性表示ウィンドウ

schupdate schema 0

データベースを再構築中です...

終了しました.

【無交換】Shell

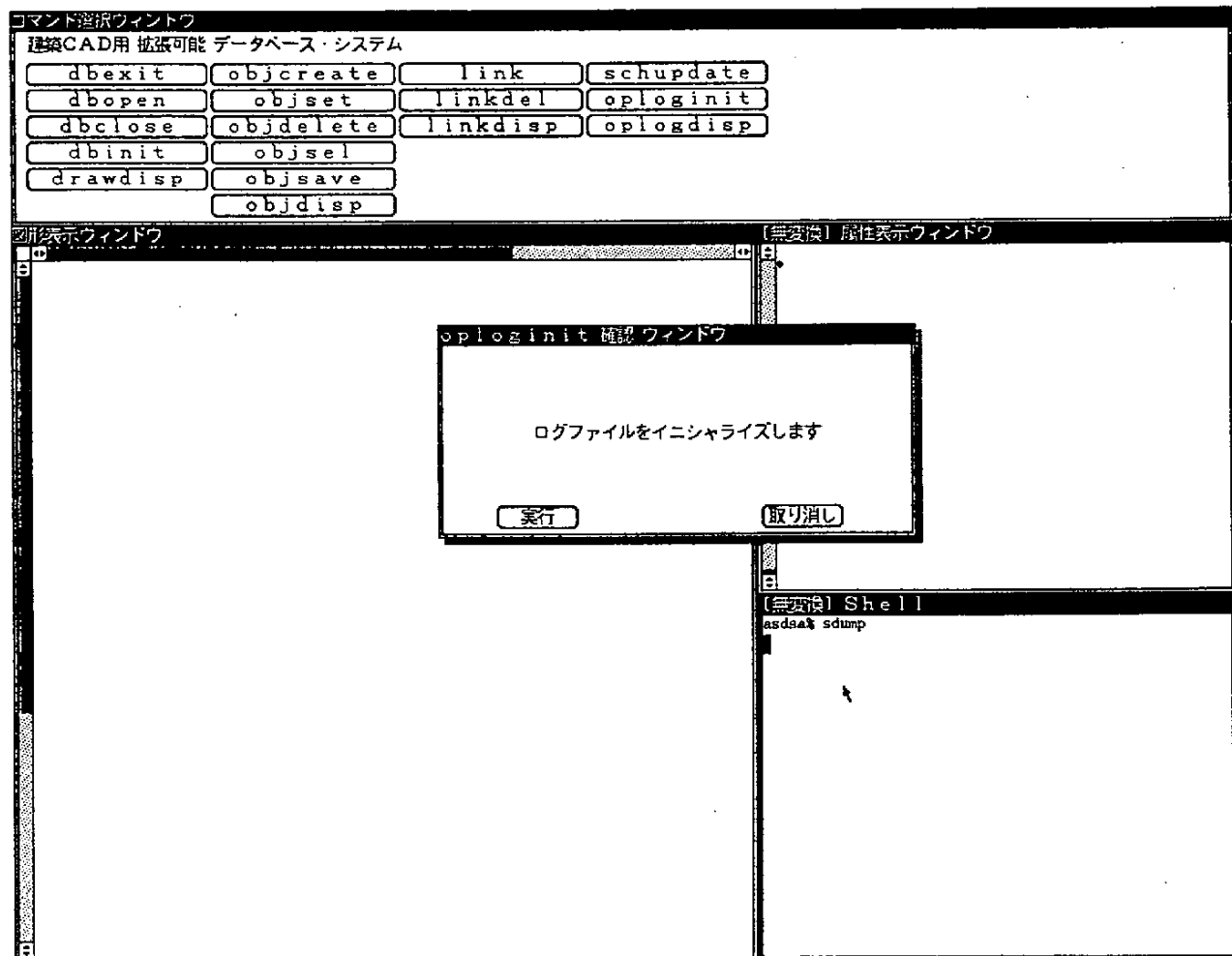
super	root		
member	char[24]	名称	
member	char[100]	住所	
member	short	築年(西暦)	
member	short	築年(月)	# 1~12
member	float	駅迄の距離(m)	#
member	char[4]	駐車場の有無	# "有" or "無"
#			
#			
schema	apartment		#物件クラス
super	root		
member	char[12]	物件番号	#
member	clong	値段(円)	#
member	clong	管理費(円)	#
member	float	専有面積(m2)	#

動作説明 (schupdate-3)

5. 1 8 操作履歴初期化機能

コ マ ン ド 名 称		o p l o g i n i t			
機 能 概 要	ログファイルを初期化する。 (. b a k に元のログの名前を変更する)				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	o p l o g i n i t ボタンをクリック			
	キ ー ボ ー ド	内 容	サイズ	備 考	
		な し			
言 語 イ ン タ フ ェ ー ス	定義	i n t c o _ o p l o g i n i t ()			
	引数名	I/O	型	サイズ	備 考
	な し				
備 考	0 : 正常終了 - 1 0 : 失敗				

実行例 (画面内容)



動作説明 (oploginit-1)

(図形表示ウィンドウには何も表示されていない)

・ oploginit ボタンクリック

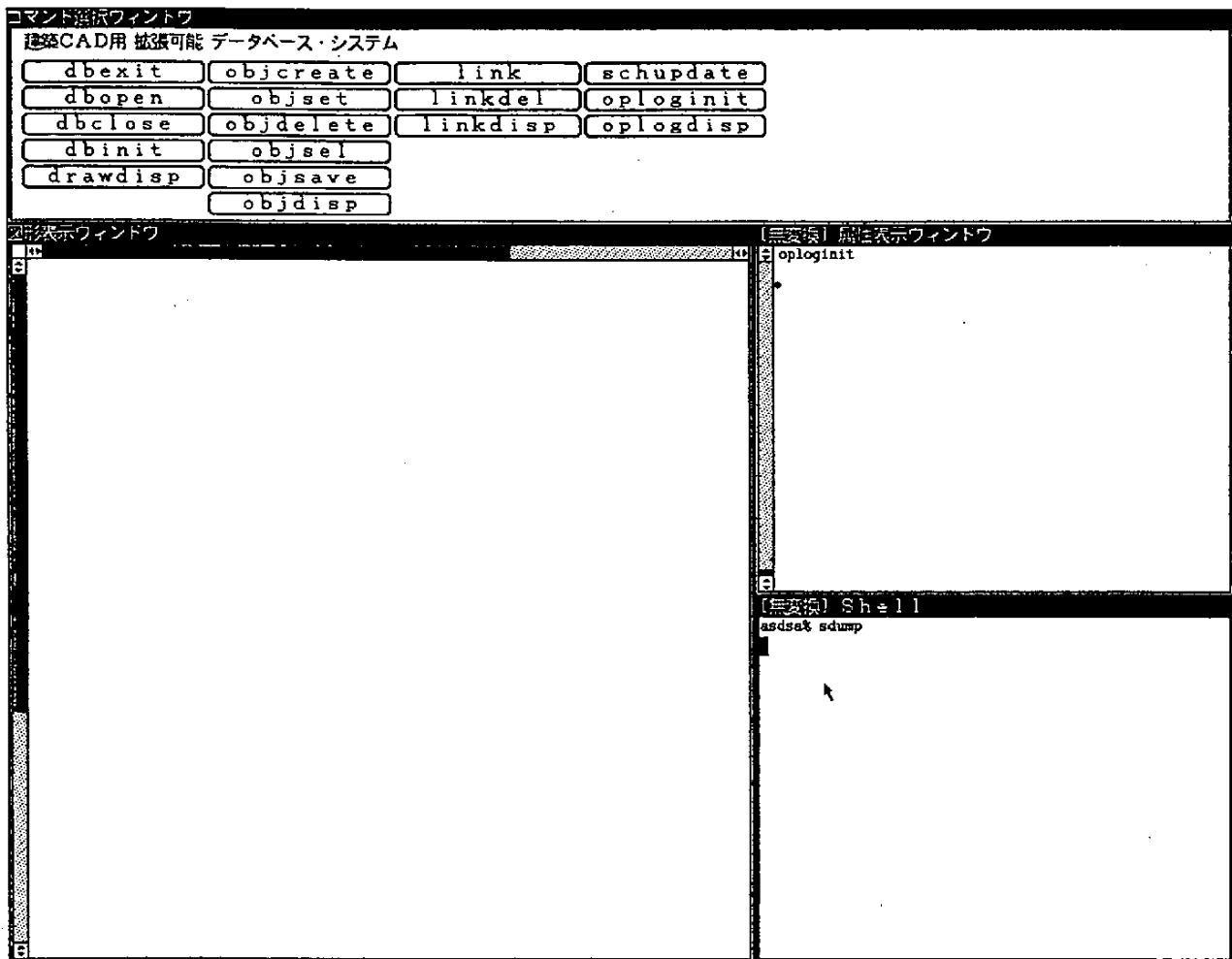
→ 確認ウィンドウ表示

(内容) ログファイルをイニシャライズします

確 認

取 り 消 し

実行例 (画面内容)



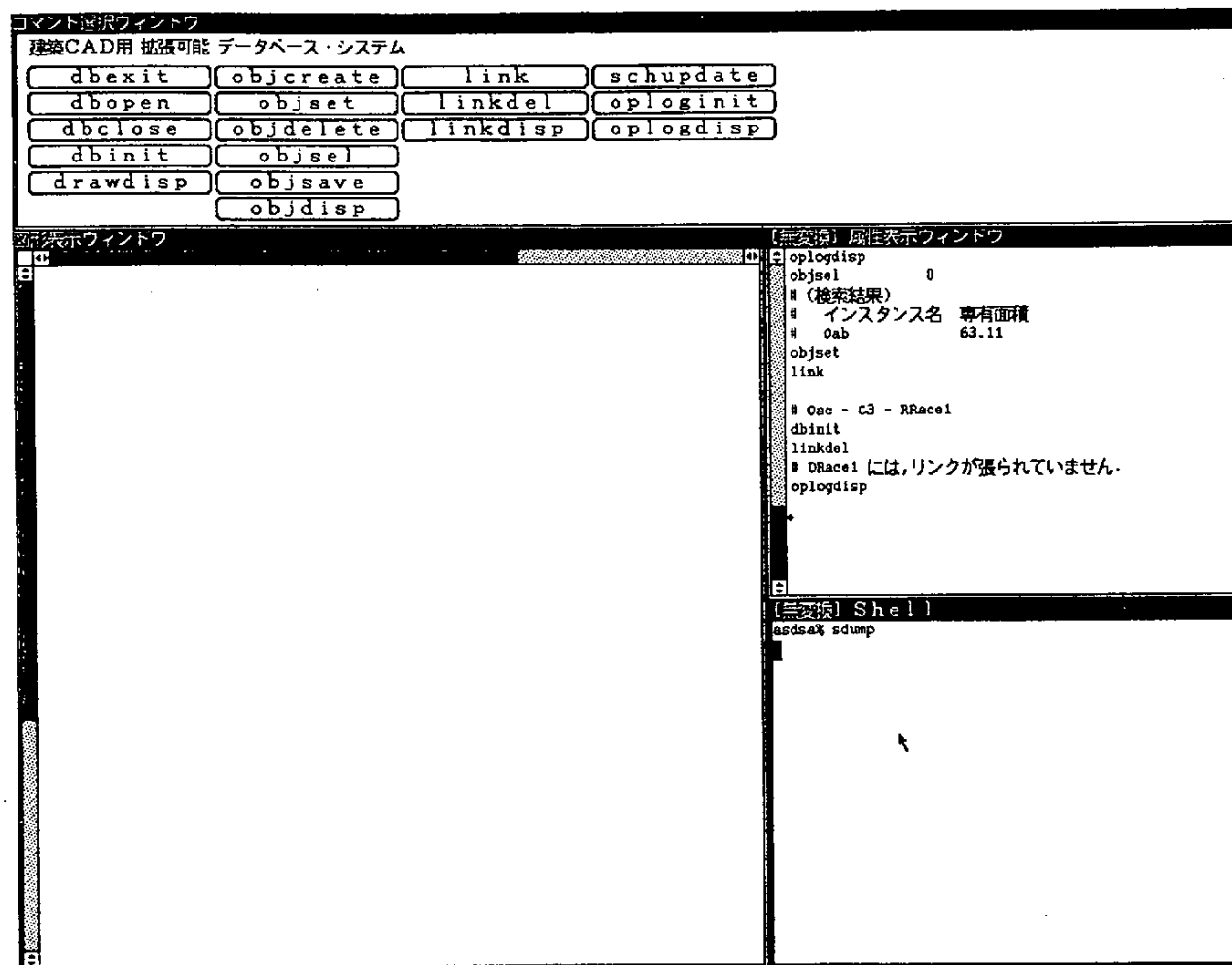
動作説明 (oploginit-2)

- 確認ウィンドウの「確認」ボタンクリック
 - 確認ウィンドウは消える
 - 属性表示ウィンドウにエコー表示
"oploginit"

5. 1 9 操作履歴表示機能

コ マ ン ド 名 称		o p l o g d i s p			
機 能 概 要	ログファイルの内容を表示する。				
操 作 イ ン タ フ ェ ー ス	ボ タ ン	o p l o g d i s p ボタンをクリック			
	キ ー ボ ー ド	内 容	サイ ズ	備 考	
		な し			
言 語 イ ン タ フ ェ ー ス	定 義	i n t c o _ o p l o g d i s p ()			
	引 数 名	I/O	型	サイ ズ	備 考
	な し				
備 考	0 : 正常終了 - 1 0 : 失敗				

実行例 (画面内容)



動作説明 (oplogdisp)

(図形表示ウィンドウには何も表示されていない)

- ・ oplogdisp ボタンクリック
 - 属性表示ウィンドウにエコー
"oplogdisp"
 - 属性表示ウィンドウにログの内容表示

6. 今後の課題と展望

本委託作業において目標とした各実現機能毎に現状と今後の課題を示す。

(1) 形状データと属性データの統合管理機能

- ・ モデリング能力の拡充
- ・ 多彩な検索機能の追加

(2) 階層的データベース構造の拡張機能

- ・ スキーマ言語記述における「入れ子表現」の実現

(3) 建築レイアウト図の表示機能

- ・ 操作性の向上
- ・ グラフィカルなスキーマ編集操作の実現

その他、今後の共通的な課題としては、以下のような項目が想定される。

(1) 実用的なDBMSが具備する機能の追加。

- ・ 処理速度の向上
- ・ メモリー効率の向上
- ・ バージョン管理機能の実現
- ・ スキーマ、リンク単位のセキュリティ管理機能の実現
- ・ リンクが容易にたどれるようなブラウザの実現

(2) データベース周辺要素技術の進展への対応

- ・ マルチメディア化に対応可能なモデリング能力の拡充
- ・ 分散化環境への適合
- ・ 建築分野以外への拡張適用

本委託開発においては、建築CADを利用分野として想定したが、今回実現した機能群は、建築CAD特有のものではなく、広く、エンジニアリングあるいはオフィス・オートメーション用のデータベースの基盤技術となるものである。

また、近年話題になっている拡張可能データベースやオブジェクト指向データベースの開発基盤として利用が期待される。

—— 禁無断転載 ——

平成 3 年 3 月発行

発 行 財団法人 データベース振興センター
東京都港区浜松町二丁目4番1号
世界貿易センタービル7階
TEL 03-3459-8581

委託先 三菱電機株式会社
東京都千代田区丸の内2丁目2番3号
TEL 03-3218-2111

印刷所 菱電印刷株式会社
神奈川県鎌倉市常盤58番地

