(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6217440号
(P6217440)
(45)発行日　平成29年10月25日(2017.10.25)　(24)登録日　平成29年10月6日(2017.10.6)

(51)Int.Cl.　F I
G06F 11/36 (2006.01)　G06F 11/36 184
G06F 11/36 108

請求項の数 4　(全 28 頁)

(21)出願番号　特願2014-28832(P2014-28832)
(22)出願日　平成26年2月18日(2014.2.18)
(65)公開番号　特開2015-153323(P2015-153323A)
(43)公開日　平成27年8月24日(2015.8.24)
審査請求日　平成28年11月2日(2016.11.2)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100146776
弁理士　山口　昭則
(72)発明者　前田　芳晴
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

最終頁に続く

(54)【発明の名称】シンボリック実行プログラム、シンボリック実行方法及びシンボリック実行装置

(57)【特許請求の範囲】
【請求項１】
シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定し、
前記レコード群のうち同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出し、
前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシンボリック処理を実行する、
処理をコンピュータに実行させるためのシンボリック実行プログラム。
【請求項２】
シンボリック実行処理させた結果により得られるパスに基づき、前記検索条件を満たすレコードが特定された場合、特定されたレコードを変更する又は新たなレコードを追加する処理を更に含む、
請求項１に記載のシンボリック実行プログラム。
【請求項３】
シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定し、
前記レコード群のうち同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出し、
前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシ

ンボリック処理を実行する、
処理をコンピュータが実行するシンボリック実行方法。
【請求項４】
シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定するレコード識別設定手段と、
前記レコード群を検索し、同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つを抽出するレコード検索制御手段と、
前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシンボリック処理を実行するシンボリック実行手段と、
を有するシンボリック実行装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、シンボリック実行プログラム、シンボリック実行方法及びシンボリック実行装置に関する。
【背景技術】
【０００２】
プログラムの開発では、作成したプログラムが想定通りに正しく動作するかを確認するため、プログラムのテスト作業（以下、「テスト」という。）が実施される。テスト対象プログラムを網羅的にテストすることが可能なテストケース及びテストデータを作成する作業は手間がかかる。そこで、テスト対象プログラムをシンボリック実行処理してテストケース及びテストデータを生成する技術が提案されている（例えば、非特許文献１参照。）。
【０００３】
シンボリック実行処理は、プログラムに外部から入力する値（引数値）に、「１」や「Ａ」のような確定値（具体値）の替わりにシンボル値（記号値）を設定してプログラムの実行をエミュレート（模擬実行）し、プログラム実行時の種々の情報を収集する技術である。プログラムをシンボリック実行処理により分析した結果、当該プログラムにおいて実行可能な命令文の系列（以下、「パス」または「実行パス」という。）、及び、パスが実行されるためにシンボル値が満たすべき条件（以下、「パス条件」という。）が抽出される。
【０００４】
図１は、シンボリック実行処理の結果の一例を示す。図１には、左側に示されるプログラムの一例をシンボリック実行処理により分析した結果として、右表のパス及びパス条件が抽出された例が示されている。「プログラムの例」で示される矩形ボックスは、該プログラムに対するシンボリック実行処理で条件文を通過したときに満たすべき論理的な制約（パス条件）を表している。シンボリック実行処理では、このようなパス条件を代入文や条件文等を通過する毎に積み上げる処理が行われる。図１のプログラムの例では、「性別が男性か？」の条件文Ｔ１で、「男性である」というパス条件と「男性でない」というパス条件とが積まれ、「性別が女性か？」の条件文Ｔ２で、「女性である」というパス条件と「女性でない」というパス条件とが積み上げられる。
【０００５】
その結果、性別が男性でありかつ性別が女性でないパス条件を持つパス１と、性別が男性でなくかつ性別が女性であるパス条件を持つパス２と、性別が男性でなくかつ性別が女性でないパス条件を持つパス３とが抽出される。性別が男性でありかつ性別が女性であるパス条件は矛盾しており成立しないため、パスは抽出されない。抽出されたパス１～３はテストケースに相当し、パス毎のパス条件を満たす値がテストケース毎のテストデータとなる。このようにして生成された各テストケースのテストデータをテスト対象のプログラムの引数値に入力し、そのプログラムの動作確認を実行することでテスト対象のプログラムを網羅的にテストすることができる。

【０００６】
例えば受発注管理などの業務システムを構成するプログラムでは、マスタデータや処理結果や履歴等を記録するためにデータベース（Ｄａｔａｂａｓｅ、以下、「ＤＢ」ともいう。）を利用する場合がある。テスト対象のプログラムがＤＢを利用する場合、そのプログラムへ入力する値には、引数として設定される入力値と、ＤＢにアクセスした結果としてＤＢ内のレコードから取得されるフィールド値との２種類がある。テスト対象のプログラムの実行パスは、これらの２種類の入力値に依存して変化する。ここで、プログラムの実行パスが変化するとは、入力値に依存して条件分岐命令で選択される分岐が変化することによって、実行される一連の命令文の系列、つまりパスが変化することをいう。このため、ＤＢを利用するプログラムのテストは、プログラムに引数値とＤＢのフィールド値とを設定して実施する必要がある。

【０００７】
そこで、以上のようにＤＢを利用するプログラムのテストにおいて、引数値及びＤＢのフィールド値に対するテストデータの生成にシンボリック実行処理を使用する方法が考えられる。しかしながら、ＤＢのフィールド値に設定できるデータの型は、数値や文字列等の具体値であり、シンボル値を設定することはできない。つまり、ＤＢをそのまま使用したシンボリック実行処理は困難である。

【０００８】
これに対して、ＤＢを利用するプログラムに対するシンボリック実行処理が可能なように、ＤＢをスタブ化する技術が提案されている（例えば、特許文献１参照）。ここで、スタブ化とは、ある処理を代替することを意味する。よって、ＤＢをスタブ化するとは、ＤＢを代替することを意味する。

【０００９】
図２には、ＤＢをスタブ化したＤＢスタブ１００にシンボル値を設定し、ＳＱＬにて記述された検索条件に従いシンボル値に対するシンボリック実行処理を行った結果の一例が示されている。図２のＤＢスタブ１００は、３つのレコードを有し、各レコードの３つのフィールドＦ１，Ｆ２、Ｆ３にシンボル値ｓ１～ｓ９が設定されている。

【００１０】
下記ＳＱＬ(Structured Query Language)検索式により記述したフィールドＦ１に対する検索条件「Ｆ１＝ｃ」に基づき、上記シンボル値が設定されたＤＢスタブ１００を検索するプログラムをシンボリック実行処理する例を挙げて説明する。
ＳＱＬ検索式：SELECT * FROM Table WHERE (F1 = "c")
このときのシンボリック実行処理では、まず、「レコード１のＦ１がｃか？」の条件文で、「ｓ１＝ｃ」というパス条件と「ｓ１≠ｃ」というパス条件とが積まれる。次に、「レコード２のＦ１がｃか？」の条件文で、「ｓ４＝ｃ」というパス条件と「ｓ４≠ｃ」というパス条件とが積み上げられる。次に、「レコード３のＦ１がｃか？」の条件文で、「ｓ７＝ｃ」というパス条件と「ｓ７≠ｃ」というパス条件とが更に積み上げられる。フィールドＦ１の複数のシンボル値がｃになるパス条件は、ＤＢのフィールド値の一意性制約を満たさず成立しない。ここで、パス毎のパス条件を満たす値がテストケース毎のテストデータとなる。この結果、生成されるテストケースは４個であり、図２の右表に示した４つのテストケースのテストデータが生成される。なお、テストデータに示される「""」、「"！"」、「"　"」はすべて「値がｃでない」ことを示す。

【００１１】
具体的に生成されたテストデータは以下である。
テストケース１：検索条件を満たすレコードが１件もヒットしなかった場合
テストケース２：レコード３が１件ヒットした場合であり、テストデータは、ｓ７＝ｃ
テストケース３：レコード２が１件ヒットした場合であり、テストデータは、ｓ４＝ｃ
テストケース４：レコード１が１件ヒットした場合であり、テストデータは、ｓ１＝ｃ

【先行技術文献】
【特許文献】

【００１２】
【特許文献１】特開２０１２－１８６７５号公報
【非特許文献】
【００１３】
【非特許文献１】シンボリック実行：玉井哲雄、福永光一、「記号実行システム」、情報処理、ｐｐ１８－２８、１９８２／０１／１５
【発明の概要】
【発明が解決しようとする課題】
【００１４】
しかしながら、ＤＢスタブの複数のレコードにシンボル値を設定した場合、各レコードのシンボル値は区別されない。例えば、図２のレコード１、２、３のフィールドＦ１に設定されたｓ１、ｓ４、ｓ７のシンボル値は区別されない。よって、テストケース２、３、４は、すべてレコードが１件ヒットし、かつフィールドＦ１にｃの値が設定された場合に該当するので同一のテストケースである。従って、図２では２個のテストケースが重複して抽出されている。このように、上記ＤＢスタブのシンボリック実行処理では、重複したテストケースが抽出されるという課題を有する。
【００１５】
そこで、一側面では、重複のないテストケースを抽出することが可能な、シンボリック実行プログラム、シンボリック実行方法及びシンボリック実行装置を提供することを目的とする。
【課題を解決するための手段】
【００１６】
一つの案では、
シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定し、
前記レコード群のうち同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出し、
前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシンボリック処理を実行する、
処理をコンピュータに実行させるためのシンボリック実行プログラムが提供される。
【発明の効果】
【００１７】
一態様によれば、重複のないテストケースを抽出することができる。
【図面の簡単な説明】
【００１８】
【図１】プログラムをシンボリック実行処理した結果の一例を示す図。
【図２】ＤＢスタブをシンボリック実行処理した結果の一例を示す図。
【図３】ＤＢを利用するプログラムの一例を示した図。
【図４】ＤＢを利用するプログラムをテストする方法の一例を示した図。
【図５】ＤＢをスタブ化する方法（１）～（３）を説明するための図。
【図６】方式（３）によるＤＢの処理部分のスタブ化を説明するための図。
【図７】一実施形態にかかるシンボリック実行装置の機能構成の一例を示した図。
【図８】一実施形態にかかるシンボリック実行処理の一例を示したフローチャート。
【図９】図８のシンボリック実行処理（テストデータ生成処理）を説明するための図。
【図１０】一実施形態にかかるテストデータの利用の一例を説明するための図。
【図１１】一実施形態にかかるＤＢを利用するプログラムとテストドライバの一例を示した図。
【図１２】一実施形態にかかるＪＤＢＣスタブのＡＰＩ（ライブラリ）の一例を示した図。
【図１３】一実施形態にかかるＤＢスタブに設定した値の一例を示した図。

【図１４】一実施形態にかかるレコード識別値の一例を示した図。
【図１５】一実施形態にかかるシンボリック実行処理を行うレコードの検索処理の一例を示したフローチャート。
【図１６】図１４（ａ）のレコード識別値の場合のシンボリック実行結果の一例を示した図。
【図１７】図１４（ｂ）のレコード識別値の場合のシンボリック実行結果の一例を示した図。
【図１８】一実施形態にかかるレコード更新処理の一例を示したフローチャート。
【図１９】一実施形態にかかるレコード更新処理結果の一例を示す図。
【図２０】一実施形態にかかるレコード削除処理の一例を示したフローチャート。
【図２１】一実施形態にかかるレコード削除処理結果の一例を示す図。
【図２２】一実施形態にかかるレコード追加処理の一例を示したフローチャート。
【図２３】一実施形態にかかるレコード追加処理結果の一例を示す図。
【図２４】一実施形態にかかるシンボリック実行装置のハードウェア構成の一例を示す図。

【発明を実施するための形態】

【００１９】

以下、本発明の実施形態について添付の図面を参照しながら説明する。なお、本明細書及び図面において、実質的に同一の機能構成を有する構成要素については、同一の符号を付することにより重複した説明を省く。

【００２０】

（はじめに）

[シンボリック実行処理]

テスト対象プログラムをシンボリック実行処理により分析し、テストケース及びテストデータを効率的に生成する手法がある。シンボリック実行（記号実行）処理は、プログラムに外部から入力する値（引数値）にシンボル値（記号値）を設定してプログラムの実行をエミュレート（模擬実行）し、プログラム実行時の種々の情報を収集する技術である。テスト対象プログラムをシンボリック実行処理により分析した結果、そのプログラムにおいて実行可能なパスと、各パスが実行されるためにシンボル値が満たすべきパス条件とが抽出される。

【００２１】

受発注管理などの業務システムを構成するプログラムでは、マスタデータや処理結果や履歴等を記録するためにＤＢを利用する場合がある。例えば、図３は、ＤＢを利用するＪａｖａ(登録商標)プログラムの一例を示す。業務システムを構成するプログラム１０がＤＢ３０を利用する場合、プログラム１０は、ＪＤＢＣ２０を介してＤＢ３０と接続する。ＪＤＢＣ２０は、ＪａｖａプログラムとＲＤＢ（リレーショナルデータベース）との接続のためのＡＰＩ（Application Programming Interface）である。ＤＢ３０は、ＳＱＬ等のＤＢ操作言語を受け付けて、ＤＢ操作言語に記述された操作内容に従ってＤＢ３０内に格納された情報を返す。

【００２２】

なお、業務システムを構成するプログラム１０は、テスト対象プログラム１１とＤＢアクセサ１２とに分かれた構成であってもよい。ＤＢアクセサ１２は、ＤＢ３０へアクセスする処理を行う専用のプログラムである。テスト対象プログラム１１は、ＤＢ３０を利用するプログラムであり、ＤＢアクセサ１２によりＪＤＢＣ２０を介してＤＢ３０にアクセスする。特に大規模なシステムのプログラムでは、ＤＢへアクセスする処理を共通化するためにＤＢアクセサ１２が作成され、利用されることがある。

【００２３】

図４は、ＤＢ３０を利用するプログラムをテストする方法の一例を示す。まず、ＤＢ３０の各レコードのフィールド値にテストデータ１が設定され（Ｄ１）、テスト対象プログラム１１の引数値にテストデータ２が設定される（Ｄ２）。

【００２４】
次に、テストドライバ４０は、テスト対象プログラム１１を起動する。テスト作業者は、テスト対象プログラムの実行結果を計測し、実行結果の値と期待値とを比較してテストの成否を判定する（Ｄ３）。
【００２５】
ＤＢ３０を利用するプログラム１０のテストにおいても、シンボリック実行処理の手法を用いてテストデータを生成することが考えられる。しかしながら、ＤＢ３０のフィールド値には、数値や文字列などの具体値（確定値）のみを設定することが可能であり、シンボル値を設定することはできない。このため、ＤＢを利用するプログラムをシンボリック実行処理により分析することは困難である。
【００２６】
[ＤＢ処理のスタブ化]
そこで、ＤＢの処理部分をシンボリック実行処理に対応可能なようにスタブ化する必要がある。つまり、シンボリック実行処理に対応していないＤＢ３０及びＪＤＢＣ２０（ＤＢアクセスライブラリ）をスタブ化する必要がある。
【００２７】
図５は、ＤＢの処理部分をスタブ化するための方法（１）～（３）を示す。スタブ化する方法（１）～（３）は、ＤＢの処理部分をどの範囲でスタブ化するかで分類される。図５には、方式（１）～（３）においてそれぞれスタブ化される範囲が示されている。
【００２８】
方式（１）は、ＤＢアクセサ１２の呼び出し時点でＤＢの処理部分をスタブ化する方法である。方式（１）では、ＤＢアクセサ１２のクラスまたはメソッドをスタブに置き換え、ＤＢアクセサ１２の呼び出し処理を代替する。
【００２９】
方式（２）は、ＤＢ操作言語の呼び出し時点でＤＢの処理部分をスタブ化する方法である。ＤＢ操作言語の一例としてはＳＱＬが挙げられる。ただし、ＤＢ操作言語は、ＳＱＬに限られない。ＤＢ操作言語は、テスト対象プログラム１１又はＤＢアクセサ１２内に記述される。方式（２）は、テスト対象プログラム１１又はＤＢアクセサ１２内に記述されたＤＢ操作言語の呼び出し部分の処理を代替する。
【００３０】
方式（３）は、ＤＢ３０の替わりに、シンボリック実行処理に対応可能なようにＤＢ３０をスタブ化したＤＢ（以下、「ＤＢスタブ」ともいう。）及びＤＢアクセスライブラリ（以下、「ＪＤＢＣスタブ」ともいう。）が使用される。テスト対象プログラム１１又はＤＢアクセサ１２は、ＪＤＢＣスタブを介してＤＢスタブにアクセスする。
【００３１】
以上に説明したＤＢ処理をスタブ化する方式のうち、方式（１）及び方式（２）には、次の第１～第３の課題がある。第１には、スタブ化のためにプログラム１０自体を一部修正する必要がある。第２には、ＤＢアクセサ１２毎又はＳＱＬ等のＤＢ操作言語の記述毎にスタブを用意する必要があるため、大規模なプログラムでは用意すべきスタブの個数が多数となる。このため、スタブの箇所の発見とスタブの作成に多大の手間を要し、最終的にテストの工数が増大する。
【００３２】
第３には、シンボリック実行処理による分析によって生成されたテストデータは、ＤＢアクセサ１２等に記述されたＳＱＬの実行の時点の変数に対して生成され、ＤＢのフィールド値に対して生成されるわけではない。このため、ＤＢを使ったテストに、生成したテストデータをそのまま利用することは困難である。
【００３３】
上記３つの課題のため、本実施形態は、方式（１）及び方式（２）を採用せず、方式（３）によりＤＢをスタブ化する方式を採用する。しかしながら、方式（３）においても、以下に説明するようにテストケース及びテストデータが重複して生成されるという課題が

ある。
【００３４】
[テストケース及びテストデータの重複]
以下、方式（３）の課題について、図６及び図２を参照しながら説明する。図６は、方式（３）によるＤＢの処理部分のスタブ化を説明するための図である。図２は、図６に示したＤＢスタブのシンボリック実行処理の結果の一例を示す。
【００３５】
図６に示されるように、方式（３）によるＤＢの処理部分のスタブ化では、ＤＢスタブ１００及びＪＤＢＣスタブ２００の２つのスタブが生成される。
【００３６】
ＤＢスタブ１００は、ＤＢ３０のスタブである。ＤＢスタブ１００は、下記の機能を備える。
・ＤＢスタブ１００のフィールドＦにシンボル値を設定することができる。よって、ＤＢスタブ１００は、シンボリック実行処理に対応できる。
・ＳＱＬ等のＤＢ操作言語を受け付け、ＤＢ操作言語に記述された操作内容に従って、ＤＢスタブ１００内に格納された情報を返す。
【００３７】
ＪＤＢＣスタブ２００は、ＪＤＢＣ２０のスタブである。ＪＤＢＣスタブ２００は、ConnectionクラスのスタブやPrepared Statementクラスのスタブなど、ＪＤＢＣを構成するクラスのスタブから構成され、下記の機能を備える。
・シンボル値を処理できる。
・ＤＢ３０に替わってＤＢスタブ１００にアクセスする。
【００３８】
図６の左に示したプログラムは、ＤＢアクセサ１２と一体化したテスト対象プログラム１１のＤＢアクセス部分をＪａｖａプログラムにより記述した例である。ＪＤＢＣは、Ｊａｖａプログラムの圧縮形式であるＪａｒ形式で提供される。Ｊａｖａプログラムの実行時にクラスパスでＪＤＢＣスタブ２００を指定することによって、ＪＤＢＣ２０に代替してＪＤＢＣスタブ２００を利用できる。ＪＤＢＣスタブ２００は、ＤＢ３０に替わってＤＢスタブ１００にアクセスする。これにより、テスト対象プログラム１１には手を加えずに、ＤＢの処理部分をスタブ化することができる。
【００３９】
なお、図６のＪａｖａプログラムの（ａ）は、ＤＢの接続、Prepared Statement（図６の例ではＳＱＬの検索式）の取得を記述する。Ｊａｖａプログラムの（ｂ）は、Prepared Statementへの入力パラメタの設定（図６の例ではフィールド１にｃを設定）、ＳＱＬに記述された検索条件に従ったシンボリック実行処理を記述する。Ｊａｖａプログラムの（ｃ）は、シンボリック実行処理の結果の返却を記述する。
【００４０】
図２には、ＤＢをスタブ化したＤＢスタブ１００にシンボル値を設定し、ＳＱＬにて記述された検索条件に従いシンボル値に対するシンボリック実行処理を行った結果の一例が示されている。本例でのＳＱＬ検索式（SQL：SELECT * FROM Table WHERE (F1 = "c")）の検索条件は、ＤＢスタブ１００のフィールドＦ１に対して検索条件「Ｆ１＝ｃ」に従い各レコードを検索するものである。
【００４１】
このときのシンボリック実行処理では、まず、「レコード１のＦ１がｃか？」の条件文で、「ｓ１＝ｃ」というパス条件と「ｓ１≠ｃ」というパス条件とが積まれる。次に、「レコード２のＦ１がｃか？」の条件文で、「ｓ４＝ｃ」というパス条件と「ｓ４≠ｃ」というパス条件とが積み上げられる。次に、「レコード３のＦ１がｃか？」の条件文で、「ｓ７＝ｃ」というパス条件と「ｓ７≠ｃ」というパス条件とが更に積み上げられる。フィールドＦ１の複数のシンボル値がｃになるパス条件は、検索条件を満たさず成立しない。ここで、パス毎のパス条件を満たす値がテストケース毎のテストデータとなる。この結果

、生成されるテストケースは４個であり、図２の右表に示した４つのテストケースのテストデータが生成される。なお、テストデータに示される「""」、「"！"」、「"　"」はすべて「値がｃでない」ことを示す。

【００４２】

具体的に生成されたテストデータは以下である。

テストケース１：検索条件を満たすレコードが１件もヒットしなかった場合

テストケース２：レコード３が１件ヒットした場合であり、テストデータは、ｓ７＝ｃ

テストケース３：レコード２が１件ヒットした場合であり、テストデータは、ｓ４＝ｃ

テストケース４：レコード１が１件ヒットした場合であり、テストデータは、ｓ１＝ｃ

ここでＤＢスタブの複数のレコードにシンボル値を設定した場合、各レコードのシンボル値は区別されない。例えば、図２のレコード１、２、３のフィールドＦ１に設定されたｓ１、ｓ４、ｓ７のシンボル値は区別されない。よって、テストケース２、３、４は、すべてレコードが１件ヒットし、かつフィールドＦ１にｃの値が設定された場合に該当するので同一のテストケースである。従って、図２では２個のテストケースが重複して抽出されている。このように、上記ＤＢスタブのシンボリック実行処理では、重複したテストケースが抽出されるという課題を有する。

【００４３】

そこで、以下に示す本実施形態にかかるシンボリック実行装置では、重複のないテストケースを抽出するために、シンボル値が設定されたＤＢスタブ１００の各レコードを区別できるようにする。そして、各レコードの異同を識別してシンボリック実行処理が制御される。これにより、重複のないテストデータを生成することができる。以下、本実施形態にかかるシンボリック実行装置の機能及び動作について詳細に説明する。

【００４４】

[シンボリック実行装置の機能構成]

まず、一実施形態にかかるシンボリック実行装置の機能構成について、図７を参照しながら説明する。図７は、一実施形態にかかるシンボリック実行装置の機能構成の一例を示す。

【００４５】

本実施形態にかかるシンボリック実行装置５０は、ＤＢスタブ１００の各レコードのフィールド値にシンボル値を設定し、検索条件に従いシンボル値に対するシンボリック実行処理を行う。シンボリック実行装置５０は、シンボル値設定手段５１、シンボリック実行手段５２、ＤＢ接続スタブ手段５３、レコード識別設定手段５４、レコード検索制御手段５５、レコード更新制御手段５６、レコード削除制御手段５７及びレコード追加制御手段５８を有する。

【００４６】

ＤＢスタブ１００は、ＤＢ３０をスタブ化してシンボル値を設定可能にしたＤＢ３０の代替である。ＤＢスタブ１００は、シンボリック実行装置５０の内部の記憶領域を使用して生成されてもよく、シンボリック実行装置５０の外部の記憶領域を使用して生成されてもよい。

【００４７】

シンボル値設定手段５１は、ＤＢスタブ１００の各レコードのフィールド値にシンボル値を設定する。

【００４８】

シンボリック実行手段５２は、検索条件に従い、設定されたシンボル値に対するシンボリック実行処理を行う。シンボリック実行処理された結果、ＤＢスタブ１００のシンボル値のパス毎のパス条件が得られる。

【００４９】

ＤＢ接続スタブ手段５３は、シンボル値を処理可能なＪＤＢＣスタブ２００を用いてＤＢ３０に替わってＤＢスタブ１００にアクセスする。

【００５０】

レコード識別設定手段５４は、ＤＢスタブ１００のレコードを区別するためにレコード毎にレコード識別値を設定する。レコード識別設定手段５４は、シンボル値をフィールド値として含むレコード群のうち、フィールド値に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定してもよい。

【００５１】

レコード検索制御手段５５は、検索条件に従いシンボリック実行処理の対象となるレコードを検索する。その際、レコード検索制御手段５５は、レコード群のうち同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出する。

【００５２】

レコード更新制御手段５６は、設定されたレコード識別値に基づきＤＢスタブ１００のレコードの更新処理を制御する。レコード更新制御手段５６は、検索条件を満たすレコードが特定された場合、その特定されたレコードを更新する。

【００５３】

レコード削除制御手段５７は、設定されたレコード識別値に基づきＤＢスタブ１００のレコードの削除処理を制御する。レコード削除制御手段５７は、検索条件を満たすレコードが特定された場合、その特定されたレコードを更新する。

【００５４】

レコード追加制御手段５８は、設定されたレコード識別値に基づきＤＢスタブ１００のレコードの追加処理を制御する。レコード追加制御手段５８は、検索条件を満たすレコードが特定された場合、その特定されたレコードに対して一意性制約に反しないレコードを追加する。一意性制約とはデータを追加や更新する際の制約の一つで、プライマリキーが設定されたフィールドＦの値が一意であること、つまり、プライマリキーが設定されたフィールドに同じデータがないことを要求する。

【００５５】

かかる機能構成を有するシンボリック実行装置５０は、同じレコード識別値が付与された複数のレコードのうちの一つのレコードに対してシンボリック実行処理を行う。この結果、本実施形態にかかるシンボリック実行装置５０によれば、重複のないＤＢのテストケースを生成することができる。

【００５６】

つまり、後述されるように、シンボリック実行処理により抽出されたパス毎のパス条件は、テストケース生成装置６０のテストデータ生成手段６１に入力される。テストデータ生成手段６１は、入力されたパス毎のパス条件に基づきＤＢ３０のフィールド値のテストデータを生成する。これによれば、抽出されたパス及びパス条件は重複していないため、重複のないテストケース及びテストデータを生成することができる。

【００５７】

［シンボリック実行処理（テストデータ生成処理）］

次に、本実施形態にかかるシンボリック実行処理について、図８及び図９を参照しながら説明する。図８は、本実施形態にかかるシンボリック実行処理（テストデータ生成処理）の一例を示したフローチャートである。図９は、図８に示したシンボリック実行処理を説明するための図である。

【００５８】

図８では、まず、テスト対象プログラム１１（又はＤＢアクセサ１２）が、クラスパスでＪＤＢＣスタブ２００を指定し、ＪＤＢＣスタブ２００に接続するように変更される（ステップＳ１０）。これにより、図９の（１）に示されるように、ＤＢアクセサ１２が、ＪＤＢＣスタブ２００と接続可能になる。

【００５９】

次に、テストドライバ４０の制御により、シンボル値設定手段５１は、ＤＢスタブ１００のレコードの各フィールドにシンボル値を設定する（ステップＳ１１）。これにより、図９の（２）に示されるように、ＤＢスタブ１００の各フィールドにシンボル値が設定さ

れる。なお、ＤＢスタブ１００のフィールド値には、シンボル値又は具体値を設定することができる。

【００６０】

次に、シンボリック実行手段５２によりテストドライバ４０が起動され、テスト対象プログラム１１、ＤＢアクセサ１２、ＪＤＢＣスタブ２００及びＤＢスタブ１００が順にアクセスされる。シンボリック実行手段５２は、テスト対象プログラム１１又はＤＢアクセサ１２内のＳＱＬに記述された検索条件に従いシンボル値に対してシンボリック実行処理を行う（ステップＳ１２）。本実施形態では、ＤＢスタブ１００の各フィールドには、テストドライバ４０の制御により指定されたシンボル値が設定されている。よって、検索条件に従い、その設定されたシンボル値に対するシンボリック実行処理が行われる。この結果、図９の（３）に示されたシンボル値に対するシンボリック実行処理が行われ、実行可能なパス及びパス条件が抽出される。

【００６１】

抽出されたパス毎のパス条件は、テストデータ生成手段６１に入力される。テストデータ生成手段６１は、入力されたパス毎のパス条件を解析し、テストケース毎にテストデータを生成し（ステップＳ１３）、本シンボリック実行処理（テストデータ生成処理）を終了する。

【００６２】

［テストデータの利用例］

以上のようにして生成されたテストデータの利用の一例について、図１０を参照しながら説明する。図１０は、一実施形態にかかるテストデータの利用の一例を説明するための図である。

【００６３】

図１０の上段に示される利用手順１では、図８のフローチャートを参照して説明した通り、シンボリック実行処理を用いてテストケース毎のテストデータが生成され、出力される。

【００６４】

図１０の下段に示される利用手順２では、ＤＢ３０の各フィールド値に生成したテストデータが設定される。この状態で、テストドライバ４０によりテスト対象プログラム１１が起動され、テスト対象プログラム１１の動作を確認するテストが実施される。

【００６５】

後述されるように、本実施形態に係るシンボリック実行装置５０では、重複のないテストケース及びテストデータを生成することができる。この結果、上記テスト対象プログラム１１の動作確認のテストを効率的に行うことができる。

【００６６】

［ＤＢを利用するプログラムとテストドライバの例］

次に、一実施形態にかかるＤＢを利用するプログラム（テスト対象プログラム１１の一部）及びテストドライバの一例を、図１１を参照しながら説明する。図１１は、一実施形態にかかるＤＢを利用するプログラム及びテストドライバの一例を示した図である。

【００６７】

図１１に示されるＤＢを利用するプログラムのプログラミング言語はＪａｖａである。このプログラムの例では、プログラムはテストドライバと一体となって、テストドライバから直接にＤＢスタブ１００にアクセス可能である。また、本実施形態においてプログラムが利用する、シンボリック実行処理の対象となるＤＢスタブを呼び出すメソッドは、「ｔｅｓｔＪＤＢＣＳｔｕｂ」である。

【００６８】

図１１のプログラム（テストドライバ）の一例では、下記の（１）～（４）の処理が記述される。

（１）では、ｓ１～ｓ９をフィールドＦに設定するシンボル値として扱うことの宣言が記述される。

（２）では、ｔｅｓｔＪＤＢＣＳｔｕｂのメソッド（ドライバ）から呼び出されるＤＢスタブ１００の各レコードのフィールドＦにシンボル値ｓ１～ｓ９を設定することが記述される。
（３）では、ＤＢを検索する処理が記述される。ＳＱＬの検索式の設定や入力パラメタの設定等の準備後、ＳＱＬの検索条件に従いシンボル値を用いたシンボリック実行処理が行われる。
（４）では、ＤＢ処理結果を利用するための処理が記述される。これにより、シンボル値を用いたシンボリック実行処理による検索結果が返却される。

【００６９】

［ＪＤＢＣスタブ及びＤＢスタブ］

ＪＤＢＣスタブ２００は、ＪＤＢＣ２０を代替し、ＤＢ３０の代わりにＤＢスタブ１００にアクセスする手段を提供する。図１２は、ＪＤＢＣスタブ２００の主要なＡＰＩ（ライブラリ）の例を示す。ＪＤＢＣスタブ２００は、ＪＤＢＣ２０と同一のＡＰＩを提供してもよい。テスト対象プログラム１１（又はＤＢアクセサ１２）は、ＪＤＢＣスタブ２００のＡＰＩを利用してＤＢスタブ１００にアクセスすることができる。

【００７０】

ＤＢスタブ１００は、ＤＢ３０の機能を代替する。図１３に示されるように、ＤＢスタブ１００のフィールドには、例えばＡＢＣや１１のような具体値だけでなく、ｓ１～ｓ１１のようなシンボル値（記号値）を設定できる。ＤＢスタブ１００は、ＪＤＢＣスタブ２００から接続され、ＳＱＬの検索条件に従いシンボル値を用いたシンボリック実行処理が可能である。

【００７１】

具体的には、ＳＱＬなどのＤＢ操作言語と、ＪＤＢＣスタブ２００のＡＰＩの呼出しにより指示された内容に従ってシンボリック実行処理が行われ、ＤＢスタブ１００内のレコードに設定されたシンボル値に対する検索（Ｓｅｌｅｃｔ）、更新（Ｕｐｄａｔｅ）、削除（Ｄｅｌｅｔｅ）及び追加（Ｉｎｓｅｒｔ）などの処理が実行される。

【００７２】

［レコード識別］

次に、ＤＢスタブ１００内の各レコードに付与されたレコード識別値について説明する。レコード識別設定手段５４は、ＤＢスタブ１００内のレコードを区別するためにレコード識別値を設定する。レコード識別設定手段５４は、シンボル値をフィールド値として含むレコード群のうち、フィールド値に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定してもよい。

【００７３】

レコード識別設定手段５４は、ユーザによる指定に応じてレコード識別値を設定してもよい。例えば、ＤＢスタブ１００内の複数のレコードに対して検索条件を満たすレコードのヒット件数を１件にしたい場合、レコード識別設定手段５４は、複数のレコードに１種類のレコード識別値を設定する。ＤＢスタブ１００内の複数のレコードに対して検索条件を満たすレコードのヒット件数をＮ件にしたい場合、レコード識別設定手段５４は、複数のレコードにＮ種類のレコード識別値を設定する。

【００７４】

図１４は、一実施形態にかかるレコード識別値の一例を示す。ＤＢスタブ１００の複数のレコード１０１のフィールドＦ１、Ｆ２、Ｆ３にはシンボル値ｓ１～ｓ９が設定されている。本例では、ＤＢスタブ１００の３つのレコード１０１のそれぞれのフィールド値はすべてシンボル値であり、各レコードを区別することができない。そこで、レコード識別設定手段５４は、これらのレコードを区別できるようにレコード毎にレコード識別値１０２を設定する。

【００７５】

レコード識別設定手段５４は、ＳＱＬの検索条件を満たす複数のレコードが検出された場合にそれらの複数のレコードのうちの一つのレコードに対してシンボリック実行処理が

行われるように、同一の検索条件を満たす複数のレコードに同じレコード識別値を付与する。同じレコード識別値が付与された複数のレコードのうちの一つのレコードに対してシンボリック実行処理が行われた場合、それ以外のレコードは検索対象から除外され、シンボリック実行処理の対象外となる。

【００７６】

例えば、図１４（ａ）は、レコード１～３のレコード識別値１０２に同じ値「Ａ」が付与されている例である。この場合、レコード１～３は、同じレコードであると識別される。よって、図１４（ａ）は、レコード１～３を区別せず、レコード１～３から検索条件を満たすテストケースが１件だけ抽出されるようにしたい場合のレコード識別値の設定例である。この場合、レコード識別設定手段５４は、レコード１～３のレコード識別値１０２に同一の値を設定すればよい。

【００７７】

図１４（ｂ）は、レコード１，２のレコード識別値１０２に同じ値「Ａ」が付与され、レコード３のレコード識別値１０２にレコード１，２と異なる値「Ｂ」が付与されている例である。この場合、レコード１，２は、同じレコードであると識別され、レコード３は、レコード１，２と異なるレコードであると識別される。よって、図１４（ｂ）は、レコード１～３を２種類に区別して、３件のレコードからテストケースが２件抽出されるようにしたい場合のレコード識別値の設定例である。この場合、レコード識別設定手段５４は、レコード識別値１０２に２種類の値を設定すればよい。

【００７８】

このようにして、ＳＱＬ等で記述された検索条件を満たすＤＢスタブの複数のレコードに対して同じレコード識別値を付与することで、検索条件を満たすＤＢスタブの複数のレコードから１件のテストケースを生成することができる。

【００７９】

［レコード検索処理］

レコード検索制御手段５５は、ＤＢスタブ１００のレコード群のうち同一の検索条件を満たし、かつ同一のレコード識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出する。その際、レコード検索制御手段５５は、同一のレコード識別値が設定されたレコードが既にシンボリック実行処理の対象として抽出されたかを示すフラグを用いる。

【００８０】

例えば、本実施形態では、フラグの初期値は「０」であり、既にシンボリック実行処理の対象として抽出されたレコードのレコード識別値に対するフラグの値は、「０」から「１」に変更される。ただし、既にシンボリック実行処理の対象として抽出されたレコードに設定されたレコード識別値か否かを判定できれば、フラグ以外のいかなる手段を用いることもできる。そして、レコード検索制御手段５５は、フラグの値に基づき、既にシンボリック実行処理の対象として抽出されたレコードのレコード識別値と同一のレコード識別値が設定されたレコードについてはシンボリック実行処理を行うレコードの検索対象から除外する。以下、レコード検索制御手段５５が行うレコード検索処理例１、２について詳しく説明する。

【００８１】

［レコード検索処理例１］

まず、本実施形態に係るレコード検索処理例１について、図１５及び図１６を参照しながら説明する。図１５は、本実施形態にかかるシンボリック実行処理を行うレコードの検索処理の一例を示したフローチャートである。図１６は、左表に示されるＤＢスタブ１００のレコード１～３のレコード識別値に「Ａ」が設定された場合の図１５のレコード検索処理の結果の一例を示す。

【００８２】

図１５のレコード検索処理が開始されると、レコード検索制御手段５５は、テスト対象プログラムに記述されたＳＱＬの検索式を解析し、検索条件を取得する（ステップＳ２０

）。以下の処理では、図１６に示されるようにＳＱＬの検索式に基づき、検索条件がフィールドＦ１＝ｃである場合を例に挙げて説明する。

【００８３】

次に、レコード検索制御手段５５は、検索対象のＤＢスタブ１００からレコードｉ（ｉ＝１，２・・・Ｎ）を取得する（ステップＳ２１）。例えば、図１６に示されるＤＢスタブ１００では、レコード１，２，３（レコード数Ｎ＝３）が取得される。

【００８４】

次に、レコード検索制御手段５５は、レコードｉに１を設定し（ステップＳ２２）、レコードｉのレコード識別値が既出かを判定する（ステップＳ２３）。ここで、「レコードｉのレコード識別値が既出か」とは、レコードｉのレコード識別値が、既にシンボリック実行処理の対象として抽出されたレコードに設定されたレコード識別値と同じレコード識別値であるかを意味する。

【００８５】

この時点では、レコード識別値「Ａ」のフラグは「０」に設定されている。よって、レコード検索制御手段５５は、レコード１のレコード識別値「Ａ」は既出でないと判定し、レコード１が検索条件を満たすかを判定する（ステップＳ２４）。

【００８６】

ここでは、レコード検索制御手段５５は、レコード１のフィールドＦ１に設定された値がｃであるかを判定する。レコード１のフィールドＦ１に設定された値がｃでないと判定された場合（ここでは、シンボル値ｓ１がｃでない場合）、ステップＳ２６に進む。一方、レコード１のフィールドＦ１に設定された値がｃであると判定された場合（ここでは、シンボル値ｓ１がｃである場合）、レコード検索制御手段５５は、検索結果を格納するテーブルに検索条件を満たすレコード１を記憶し、レコード１のレコード識別値「Ａ」のフラグに１を設定し、レコード１のレコード識別値を既出とする（ステップＳ２５）。

【００８７】

次に、レコード検索制御手段５５は、レコードｉに１を加え（ステップＳ２６）、レコードｉがレコード数Ｎ（＝３）よりも大きいかを判定する（ステップＳ２７）。この時点では、レコード２は、レコード数３よりも小さいため、ステップＳ２３に戻り、レコード検索制御手段５５は、レコード２のレコード識別値「Ａ」が既出かを判定する。

【００８８】

この時点で、レコード識別値「Ａ」のフラグは「１」に設定されているため、レコード検索制御手段５５は、レコード識別値「Ａ」が既出であると判定し、シンボリック実行手段５２にバックトラックの指示を出す（ステップＳ２８）。この指示に応じて、シンボリック実行手段５２は、レコード２に対するシンボリック実行処理におけるパスのパス条件の追跡を中断し、追跡可能な別のパスが存在するレコードの位置（命令文）まで戻る（バックトラック）。これにより、本実施形態では、既にシンボリック実行処理の対象として抽出されたレコード１と同一のレコード識別値が設定されたレコード２は、検索対象から除外される。

【００８９】

次に、レコード検索制御手段５５は、レコードｉに１を加え（ステップＳ２６）、レコードｉがレコード数Ｎよりも大きいかを判定する（ステップＳ２７）。

【００９０】

この時点では、レコード３は、レコード数３に等しいため、ステップＳ２３に戻り、レコード検索制御手段５５は、レコード３のレコード識別値「Ａ」が既出かを判定する。レコード識別値「Ａ」のフラグは１に設定されているため、レコード検索制御手段５５は、レコード識別値「Ａ」が既出であると判定し、シンボリック実行手段５２にバックトラックの指示を出す（ステップＳ２８）。この指示に応じて、シンボリック実行手段５２は、レコード３に対するシンボリック実行処理におけるパスのパス条件の追跡を中断し、バックトラックする。これにより、本実施形態では、既にシンボリック実行処理の対象として抽出されたレコード１と同一のレコード識別値が設定されたレコード３は、検索対象から

除外される。

【００９１】

次に、レコード検索制御手段５５は、レコードｉに１を加え（ステップＳ２６）、レコードｉがレコード数Ｎよりも大きいかを判定する（ステップＳ２７）。

【００９２】

この時点では、レコード４は、レコード数３よりも大きいため、ステップＳ２９に進む。シンボリック実行手段５２は、検索結果のテーブルに格納されたレコード１からＳＱＬに記述された検索条件に従ってシンボリック実行処理を行い、パス毎のパス条件を抽出して、出力する（ステップＳ２９）。

【００９３】

シンボリック実行手段５２は、検索条件「Ｆ１＝ｃ」に従い、検索結果のテーブルに記憶されたレコード１のフィールドＦ１に設定されたシンボル値ｓ１に対するシンボリック実行処理を行う。この結果、シンボル値ｓ１がｃである場合とｃでない場合の２つのパス条件が抽出される。なお、シンボル値ｓ４及びシンボル値ｓ７はｃである場合は成立せず、ｃでない場合のパス条件が抽出される。

【００９４】

この結果、パス１では、シンボル値ｓ１、ｓ４、ｓ７がすべてｃでないパス条件が抽出され、パス２では、シンボル値ｓ１がｃであり、シンボル値ｓ４、ｓ７がｃでないパス条件が抽出される。このようにして抽出されたパス及びパス条件が出力された後、本レコード検索処理は終了する。

【００９５】

以上、レコード１～３に同一のレコード識別値が設定された場合のレコード検索処理にについて説明した。これによれば、同一のレコード識別値に基づき、レコード１～３はすべて同じレコードであると識別される。そして、本実施形態に係るレコード検索処理では、同じレコード識別値が設定された複数のレコードのうち、検索条件を満たした一つのレコードに対してシンボリック実行処理が行われるようにレコードが検索される。例えば、レコード１が検索条件を満たし、シンボリック実行処理の対象として抽出された場合、同じレコード識別値が設定されたレコード２，３は検索対象から除外される。これにより、同一の検索条件を満たし、かつ同一のレコード識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出できる。以上により、同じレコード識別値が設定された複数のレコードに関して重複したテストケース及びテストデータの生成を防止できる。

【００９６】

なお、本実施形態に係るレコード検索処理例１では、レコード１をシンボリック実行処理させ、レコード１と同じレコード識別値が設定されたレコード２，３は検索対象から除外した。しかしながら、本実施形態に係るシンボリック実行装置５０は、これに限られず、レコード識別値が同じ複数のレコードのうちの任意の一つのレコードに対してシンボリック実行処理が行われるようにしてもよい。例えば、レコード識別値が同じレコード２に対してシンボリック実行処理を行い、レコード１，３は検索対象から除外してもよい。また、レコード識別値が同じレコード３に対してシンボリック実行処理を行い、レコード１，２は検索対象から除外してもよい。

【００９７】

［レコード検索処理例２］

次に、本実施形態に係るレコード検索処理例１について、図１５及び図１７を参照しながら説明する。図１７は、左表に示されるＤＢスタブ１００のレコード１、２のレコード識別値に「Ａ」が設定され、レコード３のレコード識別値に「Ｂ」が設定された場合の図１５のレコード検索処理の結果の一例を示す。

【００９８】

図１５のシンボリック実行処理が開始されると、レコード検索制御手段５５は、ＳＱＬの検索式を解析し、検索条件を取得する（ステップＳ２０）。以下の処理では、図１７に

示されるようにＳＱＬの検索式に基づき、検索条件がフィールドＦ２＝ｂである場合を例に挙げて説明する。

【００９９】

次に、レコード検索制御手段５５は、検索対象のＤＢスタブ１００からレコードｉ（ｉ＝１，２・・・Ｎ）を取得する（ステップＳ２１）。例えば、図１７に示されるＤＢスタブ１００では、レコード１，２，３（レコード数Ｎ＝３）が取得される。

【０１００】

次に、レコード検索制御手段５５は、レコードｉに１を設定し（ステップＳ２２）、レコードｉのレコード識別値が既出かを判定する（ステップＳ２３）。この時点では、レコード識別値「Ａ」のフラグは０に設定されている。よって、レコード検索制御手段５５は、レコード１のレコード識別値Ａは既出でないと判定し、レコード１が検索条件を満たすかを判定する（ステップＳ２４）。

【０１０１】

ここでは、レコード検索制御手段５５は、レコード１のフィールドＦ２に設定された値がｂであるかを判定する。レコード１のフィールドＦ２に設定された値がｂでないと判定された場合（ここでは、シンボル値ｓ２がｂでない場合）、ステップＳ２６に進む。一方、レコード１のフィールドＦ２に設定された値がｂであると判定された場合（ここでは、シンボル値ｓ２がｂである場合）、レコード検索制御手段５５は、検索結果を格納するテーブルにレコード１を記憶し、レコード１のレコード識別値「Ａ」のフラグに１を設定、レコード１のレコード識別値を既出とする（ステップＳ２５）。

【０１０２】

次に、レコード検索制御手段５５は、レコードｉに１を加え（ステップＳ２６）、レコードｉがレコード数Ｎよりも大きいかを判定する（ステップＳ２７）。この時点では、レコード２は、レコード数３よりも小さいため、ステップＳ２３に戻り、レコード検索制御手段５５は、レコード２のレコード識別値「Ａ」が既出かを判定する。

【０１０３】

この時点で、レコード識別値「Ａ」のフラグは「１」に設定されているため、レコード検索制御手段５５は、レコード識別値「Ａ」が既出であると判定し、シンボリック実行手段５２にバックトラックの指示を出す（ステップＳ２８）。この指示に応じて、シンボリック実行手段５２は、レコード２に対するシンボリック実行処理によるパスのパス条件の追跡を中断し、バックトラックする。これにより、レコード２は、検索対象から除外される。

【０１０４】

次に、レコード検索制御手段５５は、レコードｉに１を加え（ステップＳ２６）、レコードｉがレコード数Ｎよりも大きいかを判定する（ステップＳ２７）。

【０１０５】

この時点では、レコード３は、レコード数３に等しいため、ステップＳ２３に戻り、レコード検索制御手段５５は、レコード３のレコード識別値「Ｂ」が既出かを判定する。レコード識別値「Ｂ」のフラグは０に設定されているため、レコード検索制御手段５５は、レコード識別値「Ｂ」は既出でないと判定し、レコード３が検索条件を満たすかを判定する（ステップＳ２４）。

【０１０６】

ここでは、レコード検索制御手段５５は、レコード３のフィールドＦ２に設定された値がｂであるかを判定する。シンボル値ｓ８がｂでないと判定された場合、ステップＳ２６に進む。一方、ステップＳ２４にてシンボル値ｓ８がｂであると判定された場合、レコード検索制御手段５５は、検索結果を格納するテーブルにレコード３を追加し、レコード３のレコード識別値「Ｂ」のフラグに１を設定する（ステップＳ２５）。

【０１０７】

次に、レコード検索制御手段５５は、レコードｉに１を加え（ステップＳ２６）、レコードｉがレコード数Ｎよりも大きいかを判定する（ステップＳ２７）。

【０１０８】
この時点では、レコード４は、レコード数３よりも大きいため、ステップＳ２９に進む。シンボリック実行手段５２は、検索結果のテーブルに格納されたレコード１，３からＳＱＬに記述された検索条件に従ってシンボリック実行処理を行い、パス毎のパス条件を抽出して、出力する（ステップＳ２９）。
【０１０９】
シンボリック実行手段５２は、検索結果のテーブルに記憶されたレコード１に対して検索条件「ｓ２＝ｂ」に応じたシンボリック実行処理を行い、レコード３に対して検索条件「ｓ８＝ｂ」に応じたシンボリック実行処理を行う。この結果、シンボル値ｓ２がｂである場合とｂでない場合、及びシンボル値ｓ８がｂである場合とｂでない場合の４つの組み合わせのパスが抽出される。なお、シンボル値ｓ５はｂでない場合のみ抽出される。
【０１１０】
この結果、パス１では、シンボル値ｓ２、ｓ５、ｓ８のすべてがｂでないパス条件が抽出される。パス２では、シンボル値ｓ２がｂであり、シンボル値ｓ５、ｓ８がｂでないパス条件が抽出される。パス３では、シンボル値ｓ２、ｓ５がｂでなく、シンボル値ｓ８がｂであるパス条件が抽出される。パス４では、シンボル値ｓ２、ｓ８がｂであり、シンボル値ｓ５がｂでないパス条件が抽出される。このようにして抽出されたパス及びパス条件を出力後、本レコード検索処理は終了する。
【０１１１】
以上、二つのレコードに同一のレコード識別値が設定され、残りの一つのレコードには異なるレコード識別値が設定された場合のレコード検索処理について説明した。これによれば、同一のレコード識別値に基づき、レコード１，２は同じレコードであると識別される。これにより、レコード１が検索条件を満たしたら、同じレコード識別値が設定されたレコード２は検索対象から除外される。一方、レコード１と異なるレコード識別値が設定されたレコード３は検索対象から除外されない。このようにして、複数のレコードから最大で２個のレコードが抽出されてシンボリック実行処理によるテストケースが生成される場合、複数のレコードには２種類のレコード識別値が設定されればよい。これにより、複数のレコードからテストケースが２件だけ抽出されるように制御できる。以上により、同じレコード識別値が設定された複数のレコードに関して重複したテストケース及びテストデータの生成を防止できる。
【０１１２】
なお、本実施形態に係るレコード検索処理例１では、レコード１をシンボリック実行処理させ、レコード１と同じレコード識別値が設定されたレコード２は検索対象から除外した。しかしながら、本実施形態に係るシンボリック実行装置５０は、これに限られず、レコード２をシンボリック実行処理させ、レコード１は検索対象から除外してもよい。
【０１１３】
［レコード更新処理］
次に、ＤＢスタブ１００のレコードの更新処理について、図１８及び図１９を参照しながら説明する。図１８は、一実施形態にかかるレコード更新処理の一例を示したフローチャートである。図１９は、一実施形態にかかるレコード更新処理の結果の一例を示す。なお、ＤＢスタブ１００の各レコードには、図１９の左表に示されるシンボル値及びレコード識別値が設定されている。
【０１１４】
レコード更新制御手段５６は、ＳＱＬで記述された検索条件に従いレコード更新処理を制御する。その際、レコード更新制御手段５６は、レコード識別値に基づきレコードの更新処理を制御する。図１８のレコード更新処理のステップＳ２１～Ｓ２９は、図１５にて説明したレコード検索処理のステップＳ２１～Ｓ２９と同一処理であり、レコード検索制御手段５５により実行される。
【０１１５】
図１８のレコード更新処理が開始されると、レコード更新制御手段５６は、ＳＱＬの検

索式を解析し、検索条件を取得する（ステップＳ３０）。以下の処理では、図１９に示されるようにＳＱＬの検索式に基づき、フィールドＦ２＝ｂの検索条件を満たすレコードが特定され、特定されたレコードのフィールドＦ１を「Ｂ」に更新する処理が実行される。この更新処理は、検索条件を満たすレコードが特定された場合、特定されたレコードを変更する処理の一例である。

【０１１６】

次に、レコード更新制御手段５６は、レコード検索制御手段５５にステップＳ２１～Ｓ２９の処理を実行させる。レコード検索制御手段５５は、ステップＳ２１～Ｓ２９の順に各レコードを検索する。その結果、初めにレコード１が抽出され、その際にレコード識別値「Ａ」のフラグに１が設定される。これにより、レコード２，３は検索対象から除外される。

【０１１７】

次に、レコード更新制御手段５６は、抽出されたレコード１をシンボリック実行処理させた結果により得られるパスに基づき、検索条件を満たすレコードを特定する（ステップＳ３２）。

【０１１８】

図１９では、抽出されたレコード１をシンボリック実行処理させた結果により得られるパス１，２のパス条件が示されている。抽出されたレコード１をシンボリック実行処理させた結果、パス１のシンボル値ｓ２、ｓ５、ｓ８がｂでない場合と、パス２のシンボル値ｓ２がｂであり、シンボル値ｓ５、ｓ８がｂでない場合が抽出される。このうち、パス２のパス条件が検索条件Ｆ２＝ｂを満たす。よって、パス２に基づき検索条件を満たすレコード１が特定される。

【０１１９】

次に、レコード更新制御手段５６は、特定されたレコード１に対してＳＱＬに記述されたフィールドＦ１を「Ｂ」に更新する処理を実行し（ステップＳ３４）、本更新処理を終了する。

【０１２０】

この結果、図１９の右下の表に示されるように、レコード１のフィールドＦ１がＢに更新される。以上に説明したレコード更新処理によれば、レコード識別値に基づいてレコードの更新を制御することができる。

【０１２１】

［レコード削除処理］

次に、ＤＢスタブ１００のレコードの削除処理について、図２０及び図２１を参照しながら説明する。図２０は、一実施形態にかかるレコード削除処理の一例を示したフローチャートである。図２１は、一実施形態にかかるレコード削除処理の結果の一例を示す。なお、ＤＢスタブ１００各レコードには、図２１の左表に示されるシンボル値及びレコード識別値が設定されている。

【０１２２】

レコード削除制御手段５７は、ＳＱＬで記述された検索条件に従いレコード削除処理を制御する。その際、レコード削除制御手段５７は、レコード識別値に基づきレコードの削除処理を制御する。図２０のレコード更新処理のステップＳ２１～Ｓ２９は、図１５にて説明したレコード検索処理のステップＳ２１～Ｓ２９と同一処理であり、レコード検索制御手段５５により実行される。

【０１２３】

図２０のレコード削除処理が開始されると、レコード削除制御手段５７は、ＳＱＬの検索式を解析し、検索条件を取得する（ステップＳ４０）。以下の処理では、図２１に示されるようにＳＱＬの検索式に基づき、フィールドＦ２＝ｂの検索条件を満たすレコードが特定され、特定されたレコードを削除する処理が実行される。この削除処理は、検索条件を満たすレコードが特定された場合、特定されたレコードを変更する処理の一例である。

【０１２４】

次に、レコード削除制御手段５７は、レコード検索制御手段５５にステップＳ２１～Ｓ２９の処理を実行させる。レコード検索制御手段５５は、ステップＳ２１～Ｓ２９の順に各レコードを検索する。その結果、初めにレコード１が抽出され、その際にレコード識別値「Ａ」のフラグに１が設定される。これにより、レコード２，３は検索対象から除外される。

【０１２５】

次に、レコード削除制御手段５７は、抽出されたパス及びパス条件に基づきフィールドＦ２＝ｂの検索条件を満たすレコードを特定する。（ステップＳ４２）。

【０１２６】

図２１では、抽出されたレコード１をシンボリック実行処理させた結果により得られるパス１，２のパス条件が示されている。抽出されたレコード１をシンボリック実行処理させた結果、パス１のシンボル値ｓ２、ｓ５、ｓ８がｂでない場合と、パス２のシンボル値ｓ２がｂであり、シンボル値ｓ５、ｓ８がｂでない場合が抽出される。このうち、パス２のパス条件が検索条件Ｆ２＝ｂを満たす。よって、パス２に基づき検索条件を満たすレコード１が特定される。

【０１２７】

次に、レコード削除制御手段５７は、特定されたレコード１に対してＳＱＬに記述されたレコードの削除処理を実行し（ステップＳ４４）、本削除処理を終了する。

【０１２８】

この結果、図２１の右下の表に示されるように、レコード１が削除される。以上に説明したレコード削除処理によれば、レコード識別値に基づいてレコードの削除を制御することができる。

【０１２９】

［レコード追加処理］

次に、ＤＢスタブ１００のレコードの追加処理について、図２２及び図２３を参照しながら説明する。図２２は、一実施形態にかかるレコード追加処理の一例を示したフローチャートである。図２３は、一実施形態にかかるレコード追加処理の結果の一例を示す。なお、ＤＢスタブ１００の各レコードには、図２３の左表に示されるシンボル値及びレコード識別値が設定されている。

【０１３０】

レコード追加制御手段５８は、ＳＱＬで記述された更新値に従いレコード追加処理を制御する。その際、レコード追加制御手段５８は、レコード識別値に基づきレコードの追加処理を制御する。図２２のレコード更新処理のステップＳ２１～Ｓ２９は、図１５にて説明したレコード検索処理のステップＳ２１～Ｓ２９と同様の処理であり、レコード検索制御手段５５により実行される。ただし、本実施形態では、等号の条件に従いレコードを検索する。

【０１３１】

図２２のレコード追加処理が開始されると、レコード追加制御手段５８は、ＳＱＬの検索式を解析し、追加するレコードの更新値を取得する（ステップＳ５０）。以下の処理では、図２３に示されるようにＳＱＬの検索式に基づき、フィールドＦ１、Ｆ２，Ｆ３の更新値がａ、ｂ、ｃのレコードを追加する処理が実行される。この追加処理は、検索条件を満たすレコードが特定された場合、新たなレコードを追加する処理の一例である。本実施形態に係る検索条件については、後述される。

【０１３２】

レコード追加制御手段５８は、プライマリキーが設定されたフィールドＦ１に対して、更新値との一意性制約に関する等号の条件を生成する（ステップＳ５２）。ここで、一意性制約とはデータを追加や更新する際の制約の一つで、プライマリキーが設定されたフィールドＦの値が一意であること、つまり、プライマリキーが設定された各レコードのフィールド値に同一値がないことを要求する。

【０１３３】

本実施形態では、一意性制約に関する等号の条件として、プライマリキーが設定された各レコードのフィールドＦ１の値と、フィールドＦ１に追加する更新値「ａ」との間で等号の条件Ｆ１＝ａが生成される。

【０１３４】

次に、レコード追加制御手段５８は、レコード検索制御手段５５にステップＳ２１～Ｓ２９の処理を実行させる。レコード検索制御手段５５は、ステップＳ２１～Ｓ２９を実行し、等号の条件Ｆ１＝ａを満たすレコードの検索を行う。これにより、レコード検索制御手段５５は、等号の条件Ｆ１＝ａを満たすレコードを抽出する。その結果、初めにレコード１が抽出され、その際にレコード識別値「Ａ」のフラグに１が設定される。これにより、レコード２，３は検索対象から除外される。

【０１３５】

次に、レコード追加制御手段５８は、抽出されたパス及びパス条件に基づき、等号の条件を満たさないレコードを特定する（ステップＳ５４）。つまり、等号の条件を満たすと一意性制約に反することになるため、本実施形態では、等号の条件を満たさない、つまりＦ１≠ａとなるレコードが特定される。また、本実施形態に係る検索条件は、抽出されたレコードのうちプライマリキーが設定されたフィールド値が等号の条件を満たさないことであり、本実施形態では抽出されたレコード１のフィールドＦ１≠ａである。

【０１３６】

図２３では、抽出されたレコード１をシンボリック実行処理させた結果により得られるパス１，２のパス条件が示されている。抽出されたレコード１をシンボリック実行処理させた結果、パス１のシンボル値ｓ１、ｓ４、ｓ７がａでない場合と、パス２のシンボル値ｓ１がａであり、シンボル値ｓ４、ｓ７がａでない場合が抽出される。

【０１３７】

このうち、パス１のパス条件は、検索条件Ｆ１≠ａを満たし、一意性制約に反しない。よって、レコード追加制御手段５８は、ステップＳ５６にて、レコード１は一意性制約に反しないレコードであると判定し、更新値ａ、ｂ、ｃの新たなレコードを追加し（ステップＳ５８）、本追加処理を終了する。これにより、図２３の右下の表に示されるようにレコード４が追加される。

【０１３８】

なお、レコード追加制御手段５８は、ステップＳ５６にて、抽出されたレコードが一意性制約に反するレコードであると判定した場合、新たなレコードを追加せずに（ステップＳ６０）、本追加処理を終了する。

【０１３９】

以上に説明したレコード追加処理によれば、レコード識別値に基づいてレコードの追加を制御することができる。

【０１４０】

なお、レコード更新制御手段５６、レコード削除制御手段５７及びレコード追加制御手段５８は、抽出されたレコードをシンボリック実行処理させた結果により得られるパスに基づき、検索条件を満たすレコードが特定された場合、特定されたレコードを変更する又は新たなレコードを追加するレコード変更制御手段の一例である。

【０１４１】

以上、本実施形態にかかるシンボリック実行装置５０について説明した。これによれば、ＤＢスタブのシンボリック実行処理において、各レコードを識別するレコード識別値に基づき重複のないテストケースを抽出することができる。

【０１４２】

（ハードウェア構成例）

最後に、シンボリック実行装置５０のハードウェア構成例について簡単に説明する。図２４は、本実施形態にかかるシンボリック実行装置５０のハードウェア構成例を示す図である。

【０１４３】

シンボリック実行装置５０は、入力装置１０１、表示装置１０２、外部Ｉ／Ｆ１０３、ＲＡＭ（Random Access Memory）１０４、ＲＯＭ（Read Only Memory）１０５、ＣＰＵ（Central Processing Unit）１０６、通信Ｉ／Ｆ１０７及びＨＤＤ（Hard Disk Drive）１０８を備え、それぞれがバスＢで相互に接続されている。

【０１４４】

入力装置１０１は、キーボードやマウスなどを含み、シンボリック実行装置５０に各操作を入力するのに用いられる。表示装置１０２は、ディスプレイなどを含み、シンボリック実行装置５０を使用してＤＢを利用するプログラムをテストするテスト作業者にテストケース及びテストデータ等のデータを表示する。

【０１４５】

外部Ｉ／Ｆ１０３は、外部装置とのインタフェースである。外部装置には、記録媒体１０３ａなどがある。シンボリック実行装置５０は、外部Ｉ／Ｆ１０３を介して、記録媒体１０３ａの読み取り及び／又は書き込みを行うことができる。記録媒体１０３ａには、ＣＤ（Compact Disk）、及びＤＶＤ（Digital Versatile Disk）、ならびに、ＳＤメモリカード（SD Memory card）やＵＳＢメモリ（Universal Serial Bus memory）等がある。

【０１４６】

ＲＯＭ１０５は、不揮発性の半導体メモリ（記憶装置）であり、各種のプログラムやデータが格納されている。ＲＡＭ１０４は、プログラムやデータを一時保持する揮発性の半導体メモリ（記憶装置）である。

【０１４７】

ＨＤＤ１０８は、プログラムやデータを格納している不揮発性の記憶装置である。格納されるプログラムやデータには、各種機能を提供するアプリケーションソフトウェアなどがある。また、ＨＤＤ１０８は、上記実施形態のシンボリック実行処理やレコードの検索処理等を行うためにＣＰＵ１０６により実行されるＪａｖａプログラムやＡＰＩ（ライブラリ）を格納してもよい。

【０１４８】

ＣＰＵ１０６は、上記記憶装置（例えば「ＨＤＤ」や「ＲＯＭ」など）から、プログラムやデータをＲＡＭ上に読み出し、装置全体の制御や搭載機能を実現する演算装置である。シンボリック実行処理やレコードの検索処理等は、ＨＤＤ１０８等にインストールされたプログラムをＣＰＵ１０６に実行させることにより実現される。

【０１４９】

通信Ｉ／Ｆ１０７は、ネットワークを介して他の機器と接続するためのインタフェースである。

【０１５０】

以上、シンボリック実行プログラム、シンボリック実行方法及びシンボリック実行装置を上記実施形態により説明したが、本発明は上記実施形態に限定されるものではなく、本発明の範囲内で種々の変形及び改良が可能である。

【０１５１】

以上の説明に関し、更に以下の項を開示する。

（付記１）

シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定し、

前記レコード群のうち同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出し、

前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシンボリック処理を実行する、

処理をコンピュータに実行させるためのシンボリック実行プログラム。

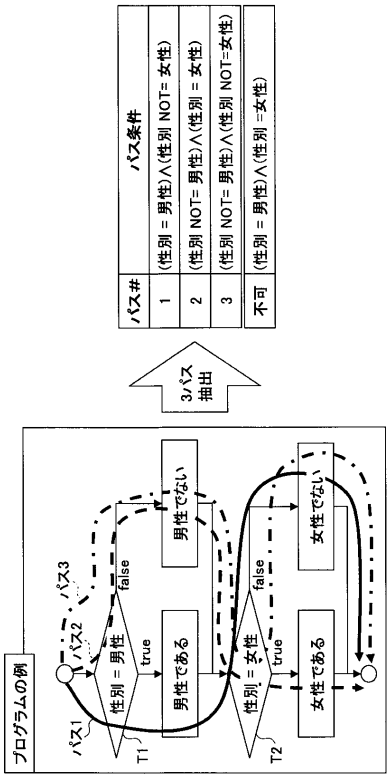
（付記２）

シンボリック実行処理させた結果により得られるパスに基づき、前記検索条件を満たすレコードを特定し、特定されたレコードを変更する、又は新たなレコードを追加する処理

を更に含む、
付記１に記載のシンボリック実行プログラム。
（付記３）
シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定し、
前記レコード群のうち同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つをシンボリック実行処理の対象として抽出し、
前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシンボリック処理を実行する、
処理をコンピュータが実行するシンボリック実行方法。
（付記４）
シンボリック実行処理させた結果により得られるパスに基づき、前記検索条件を満たすレコードが特定された場合、特定されたレコードを変更する又は新たなレコードを追加する処理を更に含む、
付記３に記載のシンボリック実行方法。
（付記５）
シンボル値を項目の値として含むレコード群のうち、前記項目に対する同一の検索条件を満たす複数のレコードに同一の識別値を設定するレコード識別設定手段と、
前記レコード群を検索し、同一の検索条件を満たし、かつ同一の識別値が設定された複数のレコードのうちの一つを抽出するレコード検索制御手段と、
前記抽出されたレコードの前記項目の前記シンボル値に対し、前記検索条件に従ってシンボリック処理を実行するシンボリック実行手段と、
を有するシンボリック実行装置。
（付記６）
シンボリック実行処理させた結果により得られるパスに基づき、前記検索条件を満たすレコードが特定された場合、特定されたレコードを変更する又は新たなレコードを追加するレコード変更制御手段を更に有する、
付記５に記載のシンボリック実行装置。
【符号の説明】
【０１５２】
５０：シンボリック実行装置
５１：シンボル値設定手段
５２：シンボリック実行手段
５３：ＤＢ接続スタブ手段
５４：レコード識別設定手段
５５：レコード検索制御手段
５６：レコード更新制御手段
５７：レコード削除制御手段
５８：レコード追加制御手段
６０：テストデータ生成装置
６１：テストデータ生成手段
１００：ＤＢスタブ
１０１：レコード
１０２：レコード識別値
２００：ＪＤＢＣスタブ
Ｆ１、Ｆ２、Ｆ３：フィールド
ｓ１～ｓ９：シンボル値

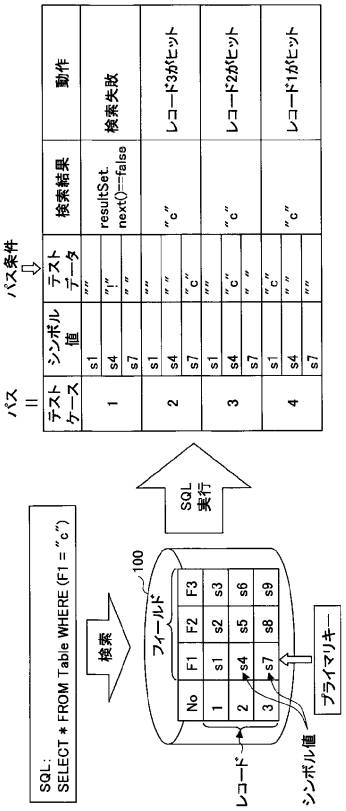
【図１】

プログラムをシンボリック実行処理した結果の一例を示す図



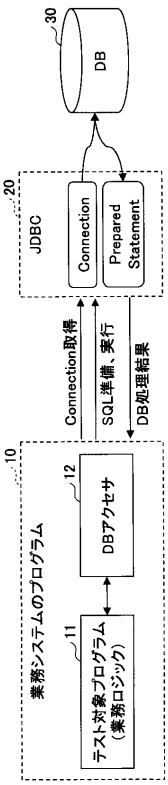
【図２】

DBスタブをシンボリック実行処理した結果の一例を示す図



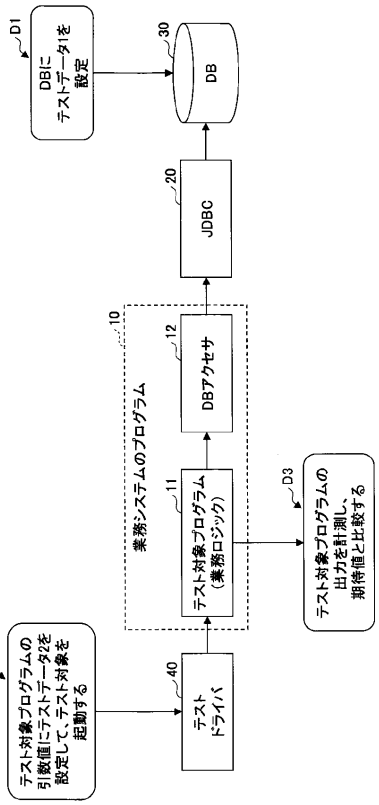
【図３】

DBを利用するプログラムの一例を示した図



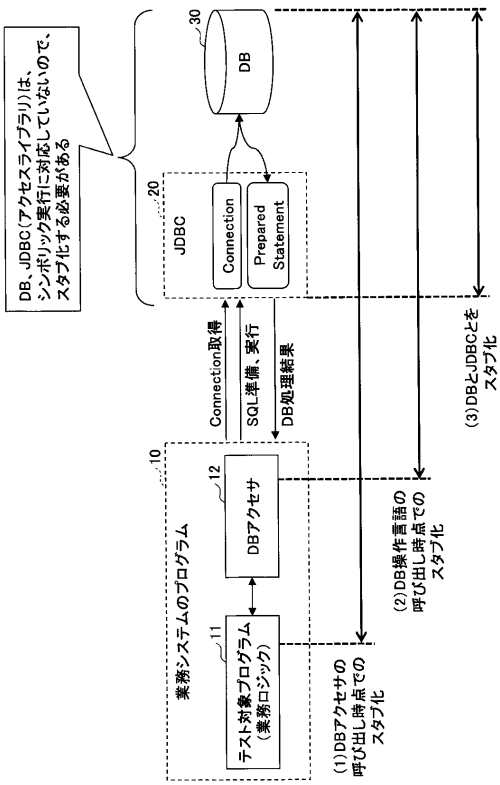
【図４】

DBを利用するプログラムをテストする方法の一例を示した図



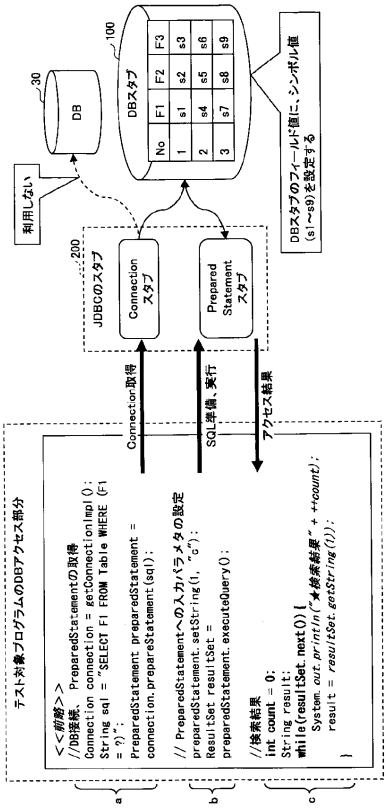
【図５】

DBをスタブ化する方法(1)～(3)を説明するための図



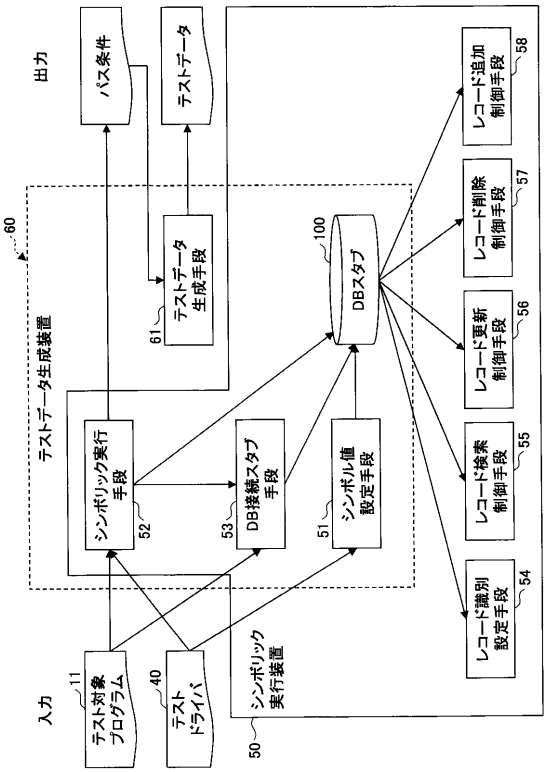
【図６】

方法(3)によるDBのスタブ化を説明するための図



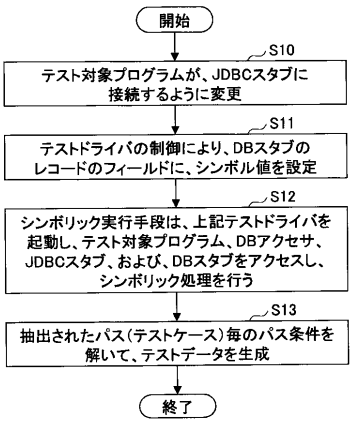
【図７】

一実施形態にかかるシンボリック実行装置の機能構成の一例を示した図



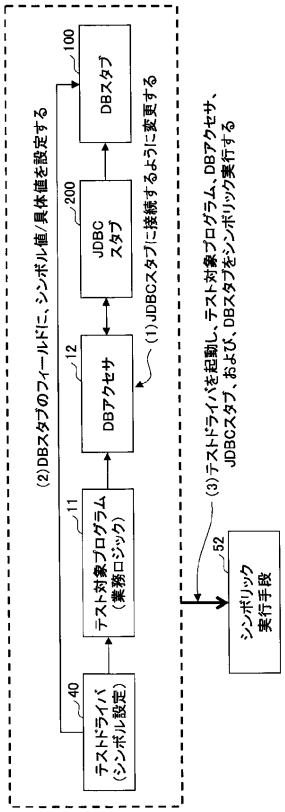
【図８】

一実施形態にかかるシンボリック実行処理の一例を示したフローチャート



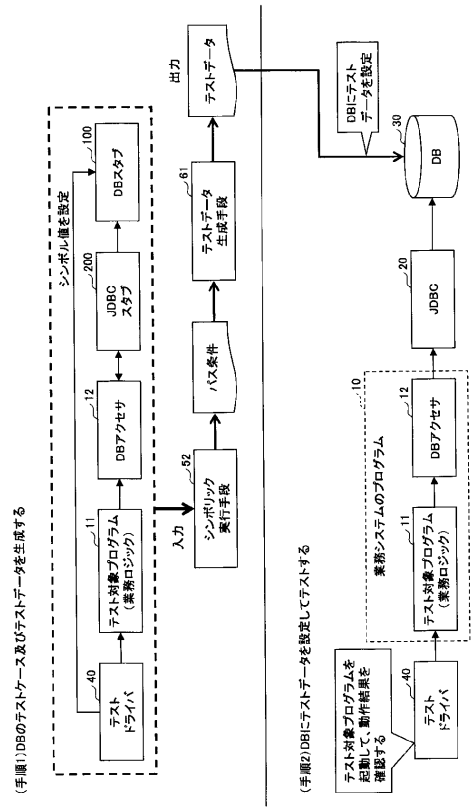
【図９】

図8のシンボリック実行処理(テストデータ生成処理)を説明するための図



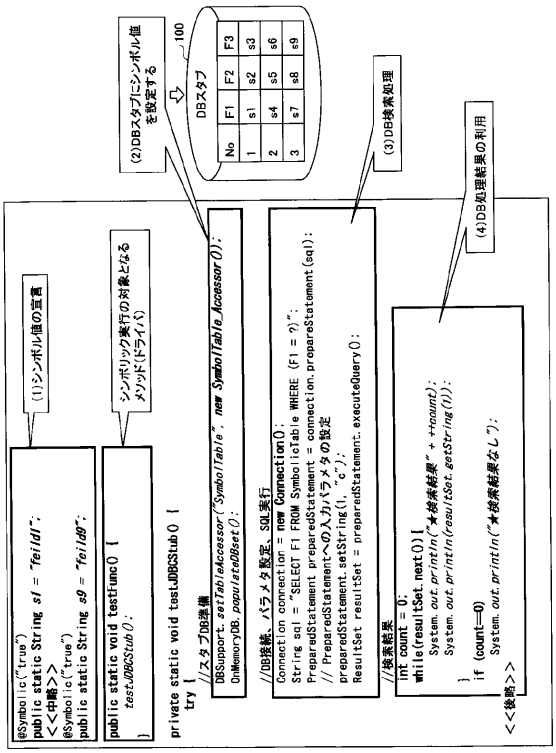
【図１０】

一実施形態にかかるテストデータの利用の一例を説明するための図



【図１１】

一実施形態にかかる
DBを利用するプログラムとテストドライバの一例を示した図



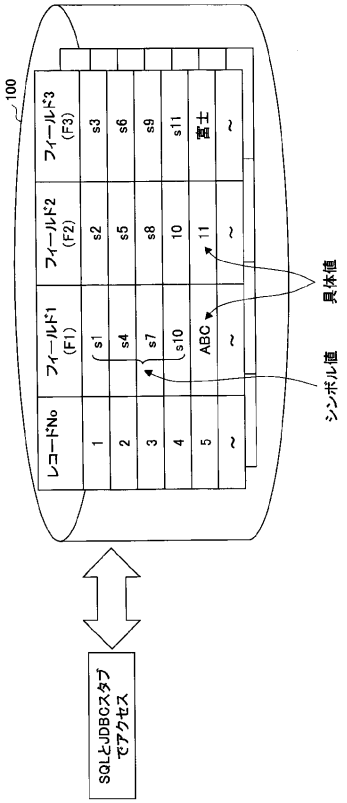
【図１２】

一実施形態にかかる
JDBCスタブのAPI(ライブラリ)の一例を示した図

クラス	メソッド	引数	戻り値
DataSource	getConnection		Connection
Connection	preparedStatement	String	PreparedStatement
	createStatement		Statement
	commit		void
	rollback		void
PreparedStatement	setString	int, String	void
	executeQuary		ResultSet
	executeUpdate		int
Statement	executeQuery	String	ResultSet
	executeUpdate	String	int
ResultSet	next		
	getString	int	String

【図１３】

一実施形態にかかるDBスタブに設定した値の一例を示した図



【図１４】

一実施形態にかかるレコード識別値の一例を示した図

(a)

100

101 レコードNo	F1 フィールド1	F2 フィールド2	F3 フィールド3	102 レコード識別値
1	s1	s2	s3	A
2	s4	s5	s6	A
3	s7	s8	s9	A

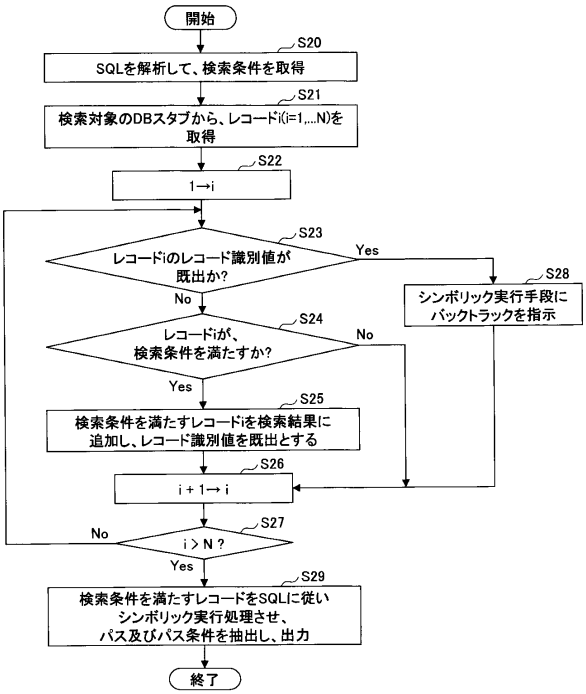
(b)

100

101 レコードNo	F1 フィールド1	F2 フィールド2	F3 フィールド3	102 レコード識別値
1	s1	s2	s3	A
2	s4	s5	s6	A
3	s7	s8	s9	B

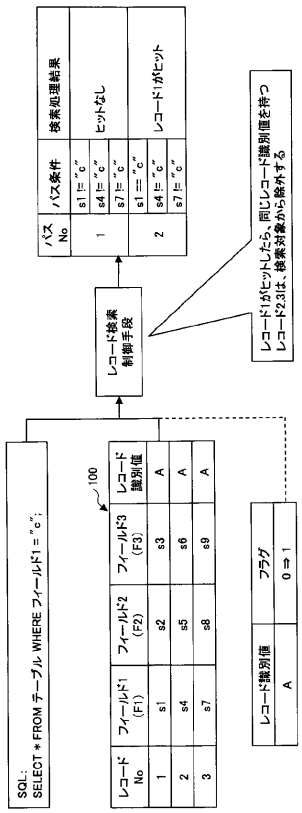
【図１５】

一実施形態にかかる
シンボリック実行処理を行うレコードの検索処理の一例を示したフローチャート



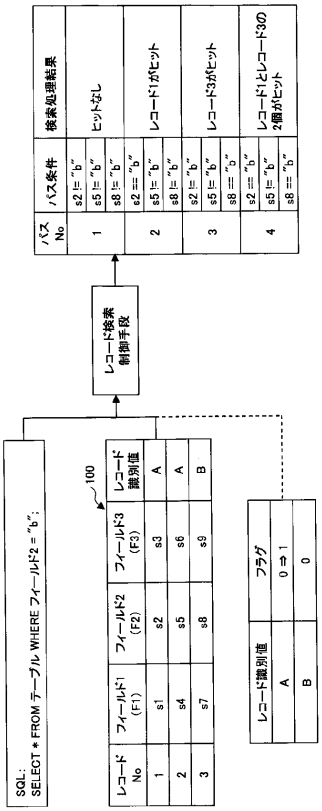
【図１６】

図14(a)のレコード識別値の場合のシンボリック実行結果の一例を示した図



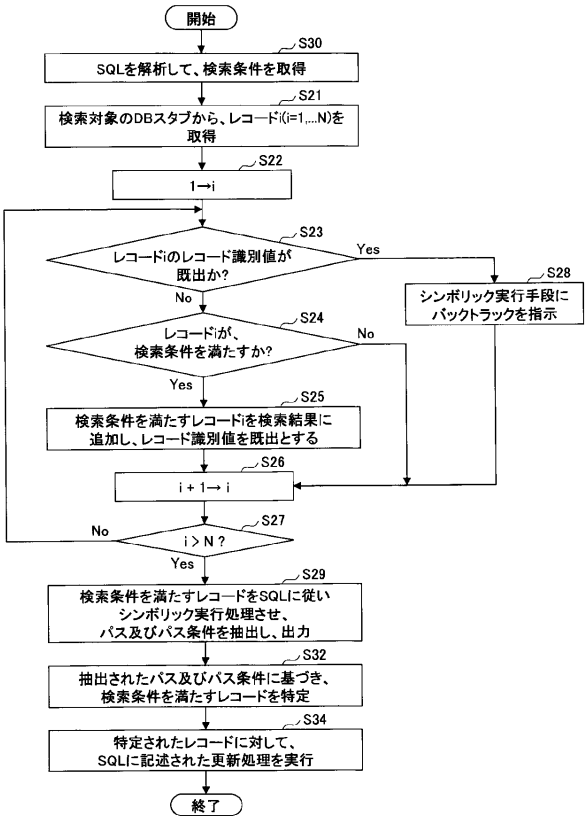
【図１７】

図14(b)のレコード識別値の場合のシンボリック実行結果の一例を示した図



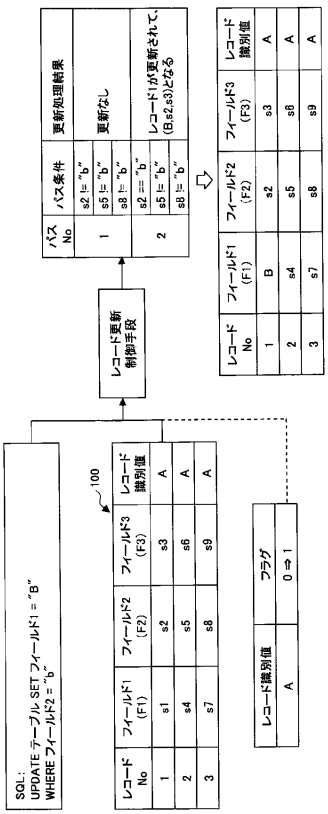
【図１８】

一実施形態にかかるレコード更新処理の一例を示したフローチャート



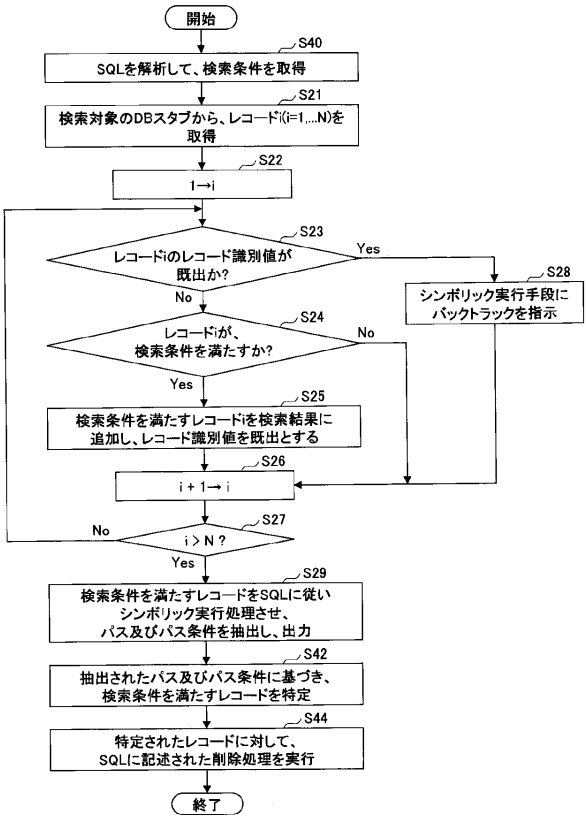
【図１９】

一実施形態にかかるレコード更新処理結果の一例を示す図



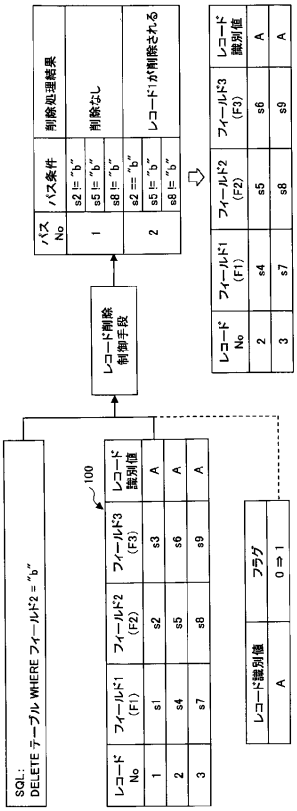
【図２０】

一実施形態にかかるレコード削除処理の一例を示したフローチャート



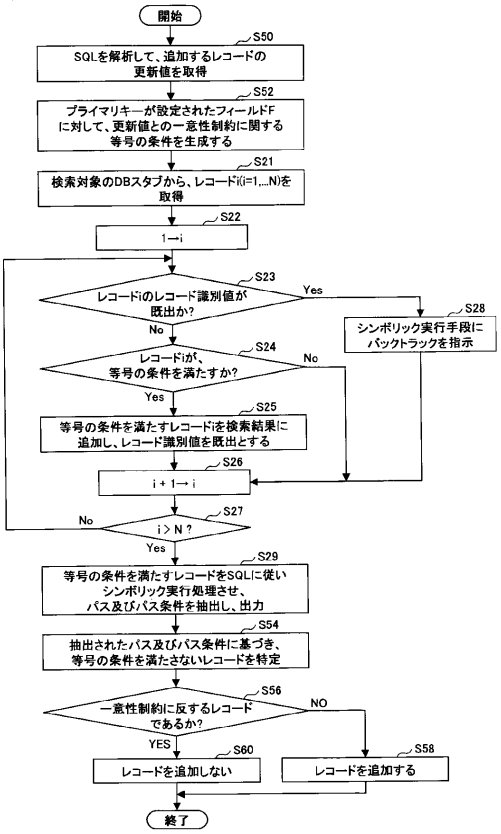
【図２１】

一実施形態にかかるレコード削除処理結果の一例を示す図



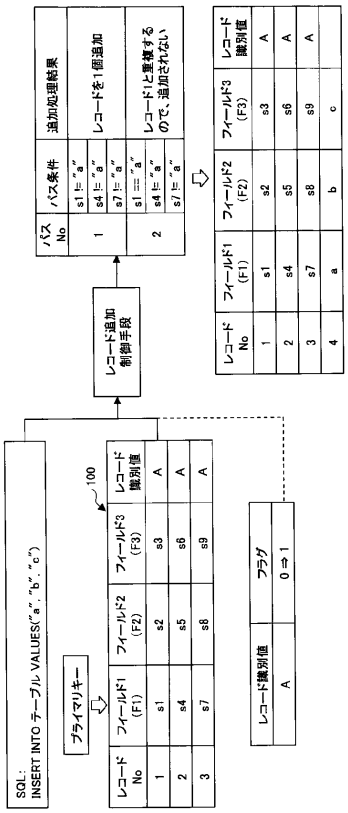
【図２２】

一実施形態にかかるレコード追加処理の一例を示したフローチャート



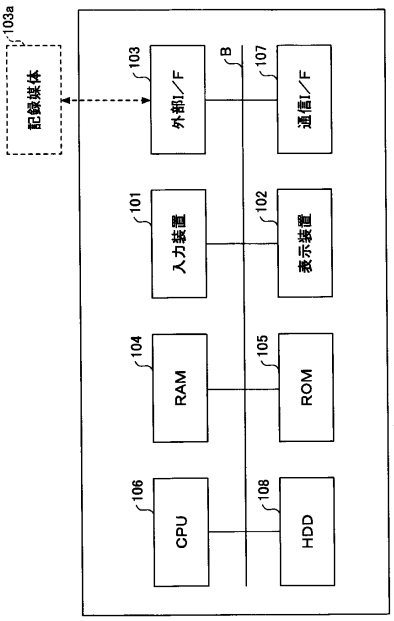
【図２３】

一実施形態にかかるレコード追加処理結果の一例を示す図



【図２４】

一実施形態にかかるシンボリック実行装置のハードウェア構成の一例を示す図



フロントページの続き

(72)発明者　上原　忠弘
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　宗像　一樹
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　徳本　晋
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　藤原　翔一朗
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　片山　朝子
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　モンプラターンチャイ　スッパシット
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

審査官　多賀　実

(56)参考文献　特開２０１２－０１８６７５（ＪＰ，Ａ）
国際公開第２０１２／１０４９０７（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１１／３６