デバイスの集合・連携操作におけるデータベースとクエリの利用

赤星　祐平† 　木俵　豊†† 　田中　克己†

† 京都大学大学院情報学研究科　〒 606–8501 京都市左京区吉田本町
†† 独立行政法人情報通信研究機構　〒 184–8795 東京都小金井市貫井北町 4–2–1
E-mail: †{akahoshi,tanaka}@dl.kuis.kyoto-u.ac.jp, ††kidawara@nict.go.jp

あらまし　ユビキタスコンピューティング環境においては，多種多様なデバイスがネットワークを介して接続することにより，デバイスを協調させて利用するなど，新しい使い方が生まれる．しかし，デバイスごとに違う機能利用の仕組みなどにより，利用のための仕組みの構築やデバイスの操作・管理には困難になりやすいと考えられる．本論文では，ユビキタスコンピューティング環境において多種多様な複数のデバイスを組み合わせて利用する場合を対象として，ラッパーを用意することでそれらの操作・利用を簡単にするための手法を提案する．ラッパーとしてデータベースの仕組みを応用することでデバイスや機能を抽象化し，クエリを用いてデバイスを集合操作したり協調操作するための手法について言及する．

キーワード　ユビキタスコンピューティング

Applying Databases and Queries to Set-oriented Device Operation and Device Collaboration

Yuhei AKAHOSHI†, Yutaka KIDAWARA††, and Katsumi TANAKA†

† Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606–8501 Japan
†† National Institute of Information and Communications Technology
4–2–1 Nukui-Kita-cho, Koganei-Shi, Tokyo, 184–8795 Japan
E-mail: †{akahoshi,tanaka}@dl.kuis.kyoto-u.ac.jp, ††kidawara@nict.go.jp

Abstract　In ubiquitous computing environment, we can use many kinds of devices around the world in new styles such as collaborative device use because of the interconnection of devices through network. However, difference of devices will make it difficult to manage and operate multiple devices. In this paper, we propose a method to make it easy to manage and operate variety of devices in ubiquitous computing environment by introducing wrappers. We introduce wrappers for the devices and abstract devices and functions based on database management method. Using this mechanism, users will operate and manage devices by applying queries.

Key words　Ubiquitous Computing

1. はじめに

ユビキタスコンピューティング環境では，多種多様なデバイスが身の回りのあらゆる場所に存在するようになる．さらに，それらのデバイスやネットワークを介して相互に接続されることで，個々のデバイスが持つ機能やサービスを他のデバイスと共有し，協調して利用するような使い方が生まれる．デバイス間で機能を共有できるようにするために，現在では，個々のデバイスで API や Web サービスといった仕組みが用意されるようになっている．

しかし，実際に多種多様なデバイスを組み合わせて利用する場合には問題が発生する．個々のデバイスごとの API 等が用意されるが，デバイスが異なると API の仕様等が異なってくるため，それらを組み合わせて利用するときには，個々のデバイスの仕様に応じた仕組みを構築する必要がある．そのため，現状ではシステムを構築する場合においては，個別のデバイスにあわせて設計・構築し，利用していることが多い．しかし，デバイスの状態が刻々と変化して利用対象となるデバイスが入れ替わるような場合，現状のような仕組みでは大きな変更を必要とする場合もあり，そのためのコストがかかる可能性がある．また，個々のシステムにあわせて構築を行うため，新たなシステムを構築する場合に既存の仕組みの再利用性は低い．

そこで本論文では，ユビキタスコンピューティング環境において多種多様な複数のデバイスを組み合わせて利用する場合を対象として，それらの操作・利用を簡単にするための手法を提案する．そのため，データベースの仕組みとクエリを応用し，既存の Web サービスや API の個々の違いを吸収するラッパーとなるような仕組みを構築する．そして，クエリを用いて複数のデバイスを集合操作したり連携操作するための仕組みについて言及する．

2. 研究の位置づけ

まず，本論文で提案する手法の背景や位置づけ，関連研究について整理する．

2.1 位置づけ

ユビキタスコンピューティング環境においては，多種多様なデバイスがネットワークを介して相互に接続することにより，これまでにはなかったようなデバイスの利用が可能になると考えられている．特に，個々のデバイスで利用可能な機能に関して，それを API や DLL といったライブラリなどを通してほかのデバイスにも公開・共有できるようにし，これにより状況に応じてたくさんのデバイスを連携させて利用するような使い方がされるようになる．例えば，センサネットワークにおいては，ネットワークを介して多数のセンサデバイスが接続することにより，センサから得られるさまざまなデータを収集し活用することが可能になる．また，ネットワークを介して多くのカメラが接続することにより，周囲の画像を収集しモニタリングするといったことも可能になる．このようなたくさんのデバイスを対象として何らかのシステムを構築する場合を考えると，通常では利用対象となるデバイスをある程度特定し，必要な機能をなどの仕様を考慮したうえで個々に応じた仕組みを構築することになる．

一方で，API や DLL の仕組みに関して考えると，例えば同じような機能を提供してくれるものであっても，それを提供するデバイスのベンダーやモデルの違いによって，利用するプロトコルや機能を使うためのクラス・メソッドの使い方などが違うことがよくある．現在ある Web 検索の API を例として見ると，Yahoo によって提供される検索 API では REST 形式を用いて検索結果を取得することになるが，Google によって提供される検索 API では SOAP が利用され，両者の検索を行うための方法は異なる．デバイスで提供する機能の場合も同様であり，写真を撮る場合に，製造メーカーの違いによって写真撮影のための命令の仕方が違うことが考えられ，このことが多種類のデバイスをまとめて利用する場合，困難さを生じさせることがある．

センサデバイスを集めてモニタリングをするシステムの場合を考えると，定時に全センサからデータを収集するためには，個々のセンサからデータを収集するための方法に応じてシステム中で個別に命令を生成し，それをセンサに対して発行することでデータを収集することになる．このため，システムでは対象となるデバイスを特定し，仕様に応じた命令が発行できるようなものを作ることになる．しかし，このシステムに新たに仕様が異なるが同種のデータを収集可能なセンサを追加して同じ動作をさせるためには，既存のシステムを新たなセンサの仕様に対応できるように変更をすることになり，その変更は場合によってはシステム全体に及び変更にかかるコストが増大する可能性がある．ユビキタスコンピューティングの環境においては，デバイスの多様さや状況に応じた利用デバイスの変更の機会が増えると予想され，これまでのような仕組みでは柔軟な対応がしにくくなると考えられる．

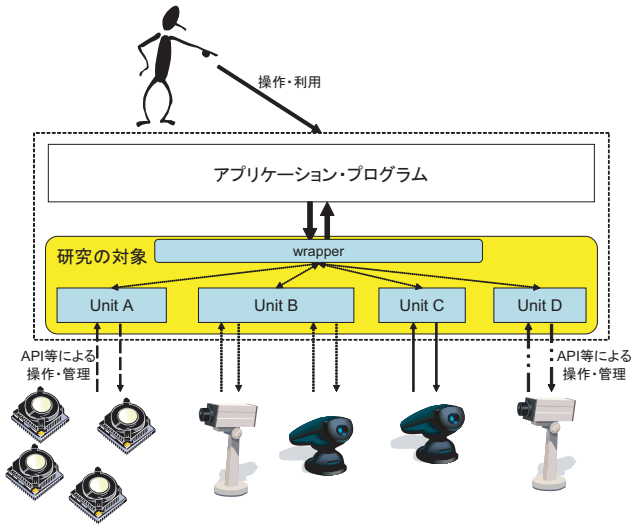


図 1　本研究の概要

そこで本研究では，デバイスの操作や管理にかかわる部分を分離し，かつ，個々のデバイスの仕組みの違いを超えて，デバイスの操作や管理をしやすくするための仕組みの実現を目指す．そのために，本研究では管理対象となるデバイスの仕組みの違いに対応するためのラッパーを用意することを考える．このラッパーを用いることにより，デバイスの仕様が違っていても同じ機能の利用においては同じ方法で利用できるようにすることを目指す．

図 1 で概要を示す．デバイスを管理・操作する場合，通常は個々のデバイスごと，もしくはいくつかまとめたデバイスごと（Unit）に利用のための API 等の仕組みに従って管理・操作をすることになる．しかし，通常では Unit が違うと仕組みが異なってくることから，いくつか Unit を組み合わせてシステムを構築し利用するためには，個々の仕組みに応じたアプリケーションを作る必要がある．そこで，操作対象のデバイスのユニット間の操作の仕組みの違いなどを吸収するための仕組みを用意する．これを用意することにより，デバイスの操作に関する部分と，その上で動くアプリケーション・プログラムが分離され，操作においてアプリケーションプログラムからはデバイスの仕様の違いを意識せずに単一の方法でアクセスできるようになる．本研究では，異なる unit を取りまとめて，アプリケーションプログラムからデバイスの仕組みの違いを考慮せずに操作できるようにするための部分の仕組みの検討を行うものである．

そこで本研究では，このラッパーの構成においてデータベースの手法を応用することを考える．データベースは各種システ

ムにおいてデータの管理部分を分離し集合的に管理操作するための仕組みとして用いられているものであり，データベースの仕組みを利用したラッパーを用意することによる利点としては次のような点が期待される．

- 集合的なデバイス管理・操作
- 再利用性の向上やシステム構築・修正のコスト低下

本来データベースは，テーブルやタプル，クエリを用いることで，データベースに保持された大量の要素を管理，操作することに適した設計になっている．そのため，データベースの仕組みを基礎としてデバイスを抽象化することによって，データベースと同様に，大量のデバイスを対象としても集合的に管理・操作ができるようになると考えられる．これにより，個々のデバイスごとに操作管理をする場合と比べて，操作や管理にかかるコストを削減することが期待される．

また，データベースの導入によってアプリケーションとデータ管理の部分が分離されたことと同様に，デバイス操作や管理にデータベースのような仕組みを使うことで，デバイス利用のためのシステムを作るときに，デバイスの操作や管理をする部分と，それ以外の部分の分離をすることが可能になる．これにより，一度作成したシステムの別のシステムへの利用といった際の再利用性の向上や，デバイスの追加削除などによって生じるシステムの修正にかかるコストを下げることが期待できる．

2.2 関連研究

これまで，デバイスの管理といった点で行われている研究についていくつか言及する．

デバイスの相互接続とそれらの連携利用に関する研究についてここでいくつか触れる．まず，センサーネットワークを対象としてデータベースの手法を用いた管理手法としては，TinyDB [2] [4] が挙げられる．TinyDB では，センサーネットワーク内にある各センサー (mote) で駆動する TinyOS 上に TinyDB のシステムを乗せることで，TinySQL と呼ばれるクエリ言語を用いてセンサーからのデータ収集を実現する仕組みを提案している．TinyDB と本研究の違いは，主に対象とするデバイスの種類であり，我々の研究ではセンサーから PC など多様なデバイスを対象としており，また手法の位置づけも異なっている．寺田らの研究 [3] では，ECA ルールを用いて駆動するユビキタスチップを開発し，ユビキタスチップに接続された機能の動作やチップ間連携において ECA ルールを活用した仕組みについて提案を行っている．ECA ルールを用いたデバイスの管理や操作という点においては類似している点があると考えられる．

家電ネットワーク分野では，uPnP [5] や JINI [6] といったフレームワークにより，ネットワークで接続可能なデバイス間で利用可能な機能の発見し相互のデバイスをつなぐための仕組みが提案されている．また，Task Computing [1] では，機能発見などにセマンティック Web の技術を利用している．これらの研究との違いは，我々の研究では「デバイスがすでに接続されていること」が前提であり，その上でいかに利用可能な機能の検索を行ったり実際に操作するかといったことに着目している点にある．

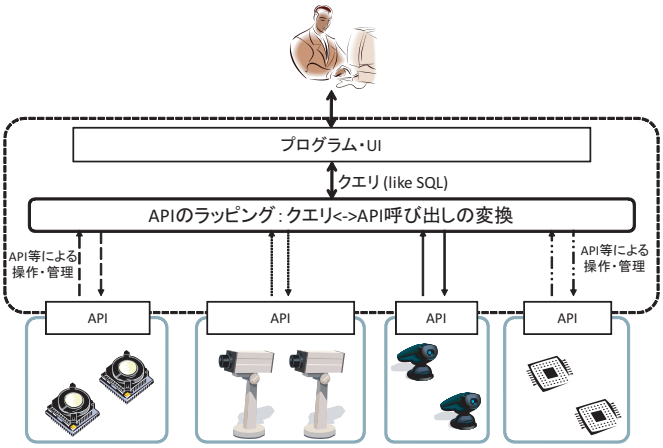


図 2　ラッパーの仕組み

これらいずれの手法について共通に挙げられる違いは，本論文で提案する手法は，これら関連研究によって実現されるデバイスの管理や連携の仕組みのラッパーとして働き，仕組みの違いの吸収やシステム構築・修正等にかかるコストを下げようとするものである．

3. ラッパーの構築

個々のデバイスの違いを吸収するラッパーを実現するためには，まずはデバイスとデバイスにある機能のそれぞれの定義が必要になる．本節では，データベースの仕組みを使ってこれらを定義するための方法について言及する．

3.1 ラッパーの構成について

本研究ではデバイスの操作利用のための仕組みを 3 つの階層に分ける（図 2）．

- ユーザインタフェース・アプリケーションプログラム
- API(デバイス) のラッピングのための階層
- デバイス

まず，ユーザインタフェース・アプリケーションプログラムの階層では，利用可能なデバイスを使ってシステムを構成するためのプログラムや，そのシステムとユーザとのインタフェースを提供する．デバイスの階層は，API などを介して受け取った命令に応じて個々のデバイスの機能を動かし結果を提供する．通常のシステム構築では，ユーザインタフェース・アプリケーションプログラムとデバイスが密接に関わり，プログラム中から直接デバイスの機能を呼び出し利用する．

しかし，本研究ではこの間に API をラッピングするための階層を設けて，これらの分離を試みる．プログラムからデバイスを利用する場合に，まずはデバイスラッパーの仕組みに応じた命令を出し，デバイスラッパーの階層でその命令を解釈し，条件にあったデバイスの機能を利用するための API 等に沿った命令をデバイスに発行する．本研究ではこの API のラッピングの実現のためにデータベースの手法を使うことから，アプリケーションプログラムの階層とラッパーの階層間のやり取りはクエリの形で実現される．個々のデバイスはベンダー等から提供される API などの仕組みがあることから，ラッパーで受け

取ったクエリを解釈し，個々のデバイスへは API などの仕様に応じた命令をデバイスに発行する．

以後，ラッパーの構築のために必要となる，デバイスの表現方法や操作の表現方法について説明する．

3.2 テーブルによるデバイス表現

まずは，デバイスのテーブルを用いた表現について考える．

本研究では，データベースの手法を用いてデバイスの表現を行う．そこでまず，デバイスとデータベースの要素の対応関係を考えることになるが，本手法では，デバイスの管理単位に応じてテーブルを割り当てることで，デバイスの抽象化を実現する．つまり，1 つのデバイスが単体で管理される場合にはその 1 つのデバイスで 1 つのテーブルを構成する．一方でセンサーのように複数のセンサーをまとめて一括管理するような場合には，一括管理されるセンサー群をすべてまとめて 1 つのテーブルで表現を行う．

テーブル設定されるフィールドには，以下のものを含む．またカメラの場合の表現例を図 3 で示す．

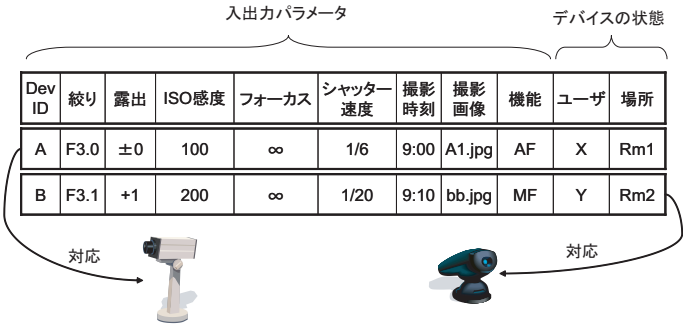
- 適切な入出力パラメータ群
- デバイスのプロパティ情報

まず，入出力パラメータ群については，デバイスの種類ごとに適当なパラメータ群を用意し，それらをフィールドに割り当てる．このときの「入出力パラメータ」は，必ずしも個々のデバイスの機能利用の際に関数・メソッド等の利用の際に実際に使われる入出力パラメータとは一致せず，より抽象的な入出力パラメータである．これは，本研究の趣旨である，個々のデバイスの違いを吸収するためのラッパーを実現するためである．このフィールドとして設定するパラメータは，あらかじめ一般的に機能を利用する際に必要な情報を基にして設定するものである．

たとえばカメラを表現するテーブルを考えると，一般にカメラで画像撮影する場合の入力パラメータとしては「ISO 感度」「露出」「シャッター速度」「焦点距離」「写真の撮影時刻」といったものが挙げられる．一方で出力としては「撮影した写真の画像（ファイル名）」が考えられる．そこで，カメラに対応するテーブルのフィールドには「ISO 感度」「シャッター速度」や「撮影写真の画像名」といったものが含まれることになる．また，テーブルのフラットな構造ではあるタプルの入出力パラメータ群がどのような機能のために利用されたかがわかりにくいため，それを判別しやすくするためのフィールドを入出力パラメータの一種として付加する．

デバイスのプロパティ情報は，デバイスを識別するための ID 情報やの位置情報，現在利用しているユーザ名など，直接デバイスの機能の入出力には関わらないが，デバイスの状態を把握するのに利用される情報である．例えば，個々のデバイスを誰が利用しているかというユーザの情報や，個々のカメラに GPS センサが内蔵されていて緯度経度情報を取得できるときの位置情報などは，撮影の機能の利用においては関係しないが，個々のデバイスの状況を示す属性情報としてフィールドを用意して保持することも考えられる．

このような形でデバイスを表現すると，タプルによって入出力パラメータとデバイスの属性情報がわかることになる．つまり，テーブルに保持されたタプルを参照することでデバイスや機能利用の状況を見ることが可能になる．また，タプルを値を書き換えたり新たにタプルをテーブルに挿入することでデバイスの利用などを表現することが可能になる．



Dev ID	絞り	露出	ISO感度	フォーカス	シャッター速度	撮影時刻	撮影画像	機能	ユーザ	場所
A	F3.0	±0	100	∞	1/6	9:00	A1.jpg	AF	X	Rm1
B	F3.1	+1	200	∞	1/20	9:10	bb.jpg	MF	Y	Rm2

図 3　デバイスのテーブル表現例

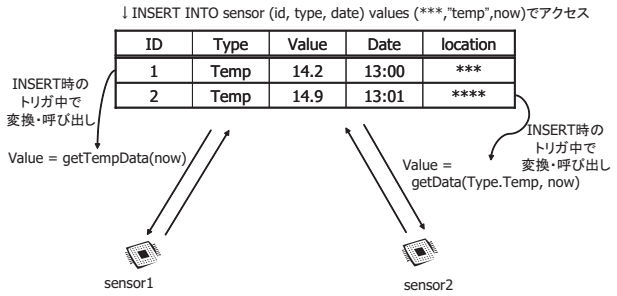


図 4　機能の実現

3.3 機能の表現

入出力パラメータをフィールドとしてデバイスの種類別にテーブルを作ることで，デバイスの抽象化表現ができるが，デバイスで提供される機能については表現ができない．機能の実現方法について本節で述べる．

一般に，デバイスの機能を考えると，次のような流れをたどることになる．

（1） 必要な入力をデバイスに渡す．
（2） 入力に基づいて利用する機能を特定し，実行する．
（3） 機能の実行結果を出力として得る．

そのため，データベーストリガを用いることで個々のデバイスの機能を定義し実現することができる．

個々のデバイスの機能について，適切なパラメータが INSERT 句などを通じて渡された時に機能を実行しその結果を受け取って対応するフィールドに書き込むようなデータベーストリガを定義する．ただし前節で述べたとおり，テーブルの個々のフィールドは，機能利用におけるより一般的な入出力パラメータをフィールドとして持っているため，必ずしもこのすべての情報が個々のデバイスの機能を利用するために必要なパラメータとは限らない．個々の機能を利用するための関数・メソッドの種類やパラメータと，テーブルにあるフィールドとそこに与えられるタプルの値の関係は，データベーストリガで機能を表現する段階にて解決をするようにする．図 4 に例を示す．この例では，温度を取得することができる 2 つのセンサがあるが，

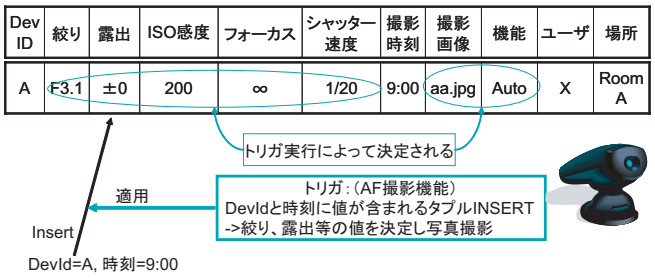


図 5　デバイスの操作

それぞれのセンサでは取得するためのメソッドの構造が異なる．Sensor1 から温度データを取得する場合には getTempData というメソッドを呼び出し，そのときに取得する時刻を引数として渡す必要があり，一方で Sensor2 の場合は getData をいうメソッドを呼び出し，そのときに取得すべきデータタイプと取得時刻を引数として渡す．本手法ではラッパーから個々のデバイスの機能を呼び出す場合にこの違いを解決するため，アプリケーションや UI からラッパーにアクセスするときにはメソッドの違いにかかわらず，sensor テーブルに対する INSERT 句のクエリ発行をする．その後，トリガの中で挿入されたタプルの ID フィールドの値によってメソッドの呼び出し方法を変化させる．ID の値が 1 の場合には Sensor1 からデータを取得するためにメソッドを呼び出し，逆に ID の値が 2 の場合には Sensor2 からデータを取得するためのメソッドを呼び出す．このようにして，トリガによって個々のデバイスの API の違いを処理し，クエリによる機能の利用を実現することが可能になる．

4.　クエリを用いた操作

デバイスと機能をテーブルとデータベーストリガで定義すると，クエリを用いることでデバイスの操作を行うことになる．その方法について説明する．

4.1　基 本 操 作

まず，基本的な操作について考える．

前節で述べたように，テーブルに保持するタプルの値によって，対応するデバイスの状態の参照することができ，また，個々のフィールドはデバイスの持つ機能を利用するためのパラメータとして位置づけられる．このことから，クエリを用いてテーブルに対してタプルの操作を行うとき，

- INSERT/UPDATE によるタプルの挿入・書き換えにより，デバイスの機能の利用
- SELECT によるタプル検索により，デバイスの状態の把握

が，実現されることになる．

INSERT 句を用いた機能利用の場合の流れは以下の通りである．例を図 5 で示す．たとえばカメラで写真を撮る場合，あらかじめ対応するテーブルにトリガの形で機能の定義がされている．カメラで AF 撮影機能を提供する場合，写真を撮るデバイスの ID と撮影時間を含んだタプルがテーブルに挿入されたときに，絞りなどの値を自動的に計算，決定しタプルに反映するトリガが定義される．ユーザが実際にカメラを使って AF 機能

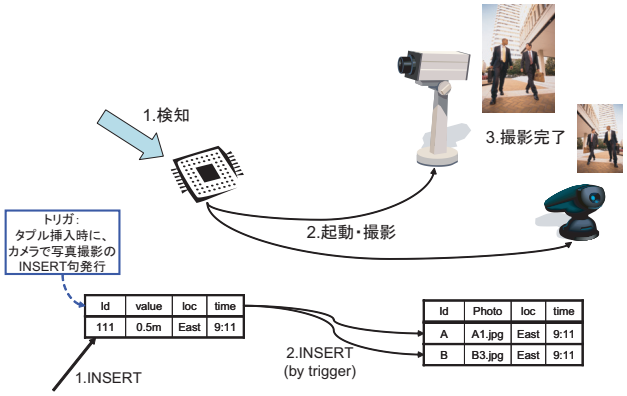


図 6　デバイスの協調操作の例

を利用する状態でシャッターを切る行動をすると，その行動がシステム上でシャッターを切ったカメラの ID と切った時刻の情報を含むタプルのテーブルへの INSERT のクエリとして表現され実行される．INSERT 句のクエリが実行されると，タプルの値をチェックした上で AF 機能に対応するトリガが起動し実際に撮影の動作がカメラで行われる．写真撮影の終了後，トリガの定義に応じてすべてのフィールドの値が決定され書き込みが行われることで，テーブルに挿入されたタプルが今回の動作と対応する表現になる．

そのため，デバイスの状態を参照する場合は，SELECT 句を用いたタプルの検索を行うことで可能である．タプルを参照することにより，機能の利用状態が見ることができるほか，デバイスの位置情報なども参照することが可能である．

4.2　協 調 操 作

ユビキタスコンピューティング環境においては，異なる種類のデバイスを協調して利用することが考えられる．それを本手法で実現する仕組みについて言及する．

デバイスを協調利用する場合には，デバイスの機能をいくつか合わせて利用することになる．つまり，機能を連続的に実行することと同じと見ることが可能である．機能を利用する場合には INSERT 句を使って適当なタプルを挿入することによって実現され，クエリを連続的に条件に応じて発行するにはデータベーストリガを使うことで実現可能であることから，トリガを使った INSERT 句の連続発行によってデバイスの協調操作を実現することができる．

連携させるデバイスの機能について，機能の実行順序を考えた上で，データベーストリガを定義する．このとき，先に利用する機能のあるデバイスに対応するテーブルにトリガを設定し，INSERT 句によって適切なタプルが挿入されたときに，次に利用する機能のあるデバイスに対応するテーブルに INSERT 句を発行し機能を利用するようにする．さらに連続して機能を利用し協調させる場合には，同様にしてデータベーストリガを定義する．これにより，適切な INSERT 句が最初に利用する機能のデバイスに対応するテーブルに対して発行されると，トリガが起動して順次 INSERT 句が発行され連続的に機能が利用されるようになり，デバイスの協調操作が実現する．

図 6 で例を示す．センサーで物体を検知したときに近くのカ

メラで写真を撮る場合を考える．この場合では，センサーで物体検知機能が作動したときにカメラの撮影機能を起動することになるため，センサーに対応するテーブルにトリガを定義する．これにより，センサーで物体を検知すると，対応するテーブルに物体検出機能の利用に対応するタプルが挿入される．タプルが挿入されると前もって定義されたトリガが発動し，カメラに対応するテーブルに写真撮影機能を利用するための INSERT 句のクエリが発行される．これにより適切なカメラに関して写真撮影機能が利用され写真が撮影される．撮影後は撮影機能を利用した記録がタプルの形でテーブルに保持される．

4.3 集合操作

デバイスを連携して使用する場合の一種として，同種の複数デバイスを同時に同様な利用する場合（デバイスの集合操作）が考えられる．このようなデバイスの集合操作の実現方法について考える．

まず集合操作を考えるとき，大きく 2 つのパターンが考えられる．

- 対象デバイスが 1 つのテーブルに含まれる
- 対象デバイスが複数のテーブルに含まれる

集合的に利用するデバイスが元から 1 つのユニットとしてまとめられている場合，デバイスの情報は 1 つのテーブル上にまとめて保持されている．このとき，集合操作をする場合には，デバイスの識別のための値以外はタプルとしてテーブルに挿入すべきフィールドと値は同一になる．そのため，連携の機能の 1 つとしてトリガで INSERT 句によるタプル挿入がされたときに，同時動作対象となるデバイスを決定して同じ動作をさせるように定義することで，実現が可能である．

一方で，対象デバイスがいくつかのユニットに分裂している場合には，複数のテーブルにわたっている可能性がある．このときは，前述の単一テーブルに集約された場合と同様に入出力パラメータは同じになるが，デバイス識別の情報が異なる他に INSERT 句を適用するテーブルが複数になる．そのため，テーブルごとに INSERT 句のクエリを用意し，同時にタプルを挿入することで実現することになる．

4.4 ログの利用

最後に，操作ログに関して簡単に言及する．

提案の手法においては，デバイスの最近の状態を示すタプルの情報によってデバイスの状況を把握し利用することになるが，過去にテーブルに挿入されたタプルの情報は削除せずに保持することによって，デバイスの操作や状態のログを取ることと同じである．ログを保持しておくことにより，過去のデバイスの操作履歴に応じたデバイスの操作などが SELECT 句による履歴検索と INSERT 句によるタプル挿入にによって実現する．

ただし，センサーのログを考える場合，周期的に作動してデータを得るような場合，ログとしてすべてのタプルを保持しておくことはデータ量の多さから難しいことが考えられる．そのため，センサーに対して過去のログを利用する場合には，ストリームデータの処理の要領で，期間を区切ってデータを保持しながら，ログの活用をするといった仕組みが必要である．

5. まとめ

本論文では，ユビキタスコンピューティング環境において多種多様なデバイスを利用する場合を想定して，デバイス間の違いを吸収しながら管理・操作するための仕組みの提案を行った．データベースの仕組みを応用してラッパーを用意することによって，個々のデバイスや機能の利用のために規定された API などの違いを吸収することで，クエリを用いてデバイスの操作や管理を実現する．そのため，デバイスの管理単位ごとにテーブルを用意し，そのデバイスで提供される機能の入出力パラメータとして一般的なものをそのテーブルのフィールドとして設定し，それを受けた個々のデバイスの API などの仕様にあった関数・メソッドの呼び出しは，テーブルにトリガの形で定義をする．これにより，タプルの INSERT や UPDATE によってデバイスの機能の利用や状態変化を表現することが可能になる．また，SELECT 句を用いてタプルを参照することでデバイスの状態や機能利用の状況を把握することが可能であり，タプルを保持し続けることによって過去の状態の参照なども可能になる．

本手法ではデータベースの仕組みを応用することにより，クエリを用いて集合的にデバイスの管理や操作が可能になるため，操作や管理にかかるコストの低下が期待される．また，実際のデバイスの操作・管理の部分とその上で動くアプリケーションプログラムの分離が可能になることから，構築した仕組みの再利用性の向上といった事が期待される．

今後，詳細に手法の検討を行った上で，この手法がシステム構築などにおいてユーザの手間の減少や，既存の仕組みの再利用性の向上などといった点での有効性を検証，評価を行う予定である．

謝　辞

本研究の一部は，文部科学省 21 世紀 COE プログラム「知識社会基盤構築のための情報学拠点形成」（リーダー：田中克己），および，文部科学省科学研究費補助金特定領域研究「情報爆発時代に向けた新しい IT 基盤技術研究」計画研究「情報爆発時代に対応するコンテンツ融合と操作環境融合に関する研究」（研究代表者：田中克己，A-01–00–02，課題番号 18049041），文部科学省科学研究費補助金特定領域研究「情報爆発時代に向けた新しい IT 基盤技術研究」公募研究「ユビキタス情報社会での情報爆発に適した検索・分類・統合化手法」（研究代表者：木俵豊，A01–24）によります．ここに記して謝意を表します．

文　献

[1] R. Masuoka, B. Parsia, Y. Labrou, “Task Computing - The Semantic Web Meets Pervasive Computing”, Proceedings of The 2nd International Semantic Web Conference, pp.866-881, 2003

[2] S. R. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong, “An Acquisitional Query Processing System for Sensor Networks”, ACM Transaction on Database System (TODS) Vol.30, Issue 1, pp.122–173, March 2005

[3] T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, A. Kashitani, S. Nishio, “Ubiquitous Chip: a

Rule-based I/O Control Device for Ubiquitous Computing", Proc. of 2nd Int'l Conf. on Pervasive Computing (Pervasive2004), pp.238–253, 2004
[4] TinyDB http://telegraph.cs.berkeley.edu/tinydb/
[5] uPnP Consortium http://www.upnp.org
[6] JINI http://www.jini.org