筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

スケールアウト型データベース製品に対応したデータ移行支援ツールの開発
－Transform 機能の設計開発－

HU BEIQI

修士（工学）

（コンピュータサイエンス専攻）

指導教員　田中二郎

２０１５年　３月

概要

本プロジェクトは、筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻、高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムにおける研究開発プロジェクトとして、日本電気株式会社(以下、NEC)が提案した「スケールアウト型データベース製品に対応したデータ移行支援ツールの開発」というテーマで、「InfoFrame Relational Store(以下、IRS)」を用いて、他データベース製品へのデータ連携ツールの企画立案、設計開発および評価を学生 4 名で実施した。我々は NEC の要求に基づき、ETL ツールの特長を参考にして、IRS から他 DB やデータウェアハウスへデータを移行するツール(以下、NET ツール)を提案した。

本プロジェクトで開発する NET ツールは通常の ETL ツールと同様に、IRS からデータを抽出する Extract 機能、データ変換を行う Transform 機能、データを PostgreSQL に挿入する Load 機能および GUI 表示機能から構成されている。

本プロジェクトでは、筆者は IRS サーバの管理と Transform 機能の設計開発を担当した。

IRS サーバ管理担当として、開発準備段階で、筆者は IRS の特長、構成およびインストール方法について調査し、開発環境とする IRS のインストールを行った。また、設計開発段階で、IRS の運用・保守を担い、障害が発生する場合、IRS の復旧を行った。

Transform 機能の設計開発担当として、筆者は NET ツールの Transform 機能を設計・開発・テストした。Transform 機能は主にデータ変換、データマッピング、および Transform プラグイン化機能を含んでいる。設計開発段階で、筆者は「Element」という方法を考え、データ変換機能を実現した。また、Graphical Editing Framework(以下、GEF)を利用することでデータマッピングを実現し、Eclipse プラグインの拡張ポイントを使って Transform 機能をプラグインした。

そして、納品段階で、筆者は Transform プラグイン作成マニュアル、一部のユーザマニュアルおよびデータ変換ルールの作成を担当した。

本報告書は、本プロジェクトの開発背景からシステムの概要、プロジェクトの経過、そして筆者が担当する部分などについてまとめたものである。

目次

図目次

表目次

第1章　はじめに

本章では、本プロジェクトの背景、目的および提案を述べる。

1.1　開発背景

ビッグデータ時代を迎え、大量のデータを高速に処理するデータベースの需要が急激に高まっている。しかし、従来から普及しているリレーショナルデータベース(以下 RDB)では一貫性や操作性重視のため、サーバの台数による性能のスケーラビリティが低いという問題がある [1]。分散型キーバリューストア(以下 KVS)は伸縮性と可用性に優れているが、SQL などのような柔軟な操作を一部犠牲にした [1] [2]。

InfoFrame Relational Store(以下 IRS) [3]は、日本電気株式会社 (以下 NEC)により開発された RDB の汎用性 [4]と KVS のスケールアウト特性 [5]を融合したビッグデータ時代のスケールアウト型データベース製品である。IRS のアーキテクチャは一般的な RDB と異なるため、すべての RDB を IRS に置き換えられない。IRS の高いスケールアウト性能とトランザクションを活用するため、データを IRS に格納し、必要に応じて他 DB やデータウェアハウス(以下 DWH)に移行して、移行先で分析を行う。

そのための既存手法の一つとして、さまざまな Extract-Transform-Load ツール(以下 ETL ツール)がある。ETL ツールとは、分析用のデータをさまざまなデータソースから抽出し、適切な形式に変換して、他 DB や DWH に格納するツールである [6]。ETL ツールを利用して、データ移行処理のコストや時間を削減することができる [7]。

しかし、IRS に対応する ETL ツールは存在しない。大量のデータを IRS から他 DB や DWH に移行する場合、IRS を利用するデータ分析者に対して求められる知識が多く、大きな負担になる。

1.2　プロジェクトの目的と提案

ETL ツール調査を元に、データを IRS から分析ツールへ移行するツールを開発することで、IRS を用いた分析環境を使用するユーザを支援することを本プロジェクトの目的とする。

顧客の要求は本製品に蓄積されたデータを他のデータベース製品へ転送するツールの設計開発である。また、GUI により双方のデータベース間の転送元・転送先の対応関係を定義する機能が要求された。

既存の ETL ツールの特長を参考にして、我々は IRS に対応した ETL ツールを開発する。GUI で移行元、移行先のデータベース間の対応関係を定義しデータの移行を行う。また、開発するツールの特長として、IRS の HadoopAPI を利用することで高速にデータの抽出が可能にし、データの加工をプラグイン化することで追加開発も容易にする [8]。

IRS を用いて分析環境を構築している企業のデータ分析者を想定ユーザとする。

1.3 報告書の構成

本報告書は 8 章で構成されている。各章の主要内容を表 1.1 に示す。

章番号	主要内容
第 1 章	本研究開発プロジェクトの開発背景、目的、提案を紹介する。
第 2 章	開発に必要な知識(関連サービス、開発プラットフォーム、開発ツールの同類製品など)をまとめる。
第 3 章	開発する NET ツールの開発環境と構成について説明する。
第 4 章	本研究開発プロジェクトの関係者構成と役割、およびスケジュールについて説明する。
第 5 章	筆者が担当した IRS のインストールと運用について説明する。
第 6 章	筆者が担当した Transform 機能の設計開発について説明する。
第 7 章	本研究開発プロジェクト第 1 イテレーションと第 2 イテレーション終了後の振り返りを述べる。
第 8 章	本研究開発プロジェクトの結論として、開発を通じて筆者が学んだことをまとめる。

第2章　　前提知識

本章では、本研究開発プロジェクトに必要な前提知識(IRS、Eclipse、ETL に関する知識を含む)について述べる。

2.1　　InfoFrame Relational Store (IRS)

InfoFrame Relational Store(以下、IRS)は、NEC によって開発され、SQL インターフェース、トランザクション処理、基幹業務に適用できる高信頼性を備えた、RDB の汎用性と KVS のスケールアウト特性を併せ持つデータベースソフトウェアである。

2.1.1　　IRS の特長

IRS の特長として、データアクセス処理、トランザクション処理、ストレージ三層の構造があるので、それぞれ必要に応じてスケールアウトができる [9]。

IRS の三層構造を図 2.1 に示す。

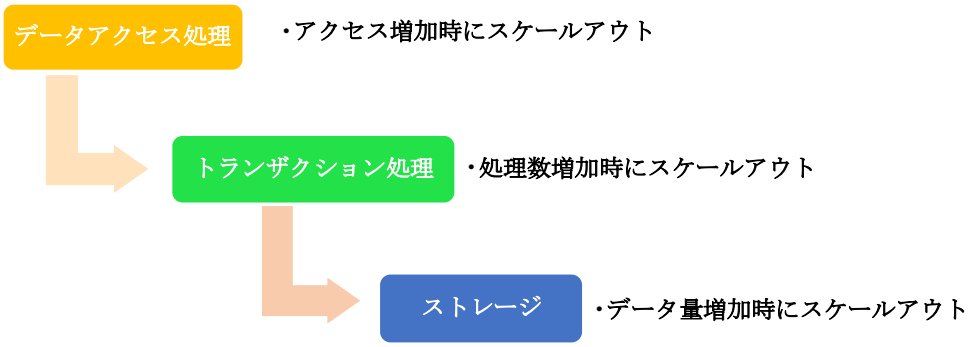


図 2.1 IRS の三層構造

IRS の Hadoop 連携機能を利用することで、大量のデータに高速でアクセスすることも可能になる。IRS は、高いスループットが求められるトランザクション業務、とにかくデータを蓄積する業務、システムを停止できない業務が得意だと考えられる [10]。

2.1.2　　IRS の構成

IRS は、RSUserDB、RSMasterDB、構成管理の 3 つのシステムから構成される。RSUserDB システムは Partiqle サーバ、トランザクションサーバ, ストレージサーバから構成され、RDB のユーザデータ領域のようにユーザデータを保持する。RSMasterDB は Partiqle サーバとトランザクションサーバから構成され、RDB のシステムカタログのように RSUserDB の構成

情報などを保持する。構成管理システムは構成や運用の管理を行う [11]。

IRS の構成については図 2.2 に示す。

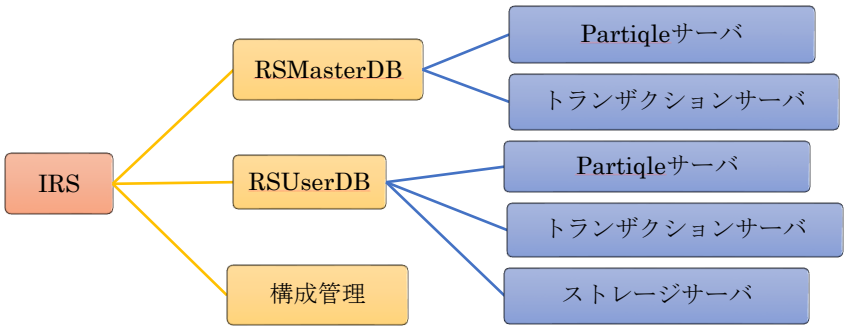


図 2.2 IRS の構成概要

Partiqle サーバはアプリケーションが発行する SQL を解析し、KVS アクセスへ変換する。トランザクションサーバはトランザクションデータを格納し、コミット時に排他制御を行う。ストレージサーバは実データを格納し、データ量に応じてサーバを追加できる。

2.1.3 データベース構成

IRS のデータはすべて Key-Value 形式で格納される。マッピングデザインを変更することで、テーブル形式のデータをどのような Key-Value 形式で格納するか指定できる。マッピングデザインには、フラット、クラスタ、レンジクラスタという 3 種類の手法がある。 [12]

2.2 Eclipse

Eclipse は統合開発環境 (IDE) の一つ。高機能ながらオープンソースであり、Java をはじめとするいくつかの言語に対応する [13]。

2.2.1 プラグイン開発

Eclipse はプラグインアーキテクチャを採用している。Eclipse には PDE(Plugin Development Enviroment)というツールが標準で組み込まれており、自分でプラグインを開発することができる [13]。

プラグインを作成したら、拡張ポイントを利用することで、他の開発者に機能を追加する機会を提供することができる。

2.2.2 GEF について

Graphical Editing Framework(以下 GEF)は Eclipse に搭載されている、すでに存在するアプリケーションモデルからリッチなグラフィカルエディターを作成するためのフレームワ

ークである。GEF の提供する API を利用し、特定のドメイン用に拡張することで開発者は比較的簡単にリッチなエディターを作成することができる [14]。

GEF はグラフィクスの描画を行う Draw2D(org.eclipse.draw2d)と、ユーザインタラクションや Eclipse ワークベンチとの統合を行う GEF(org.eclipse.gef)で構成されている。Draw2D は SWTCanvas を拡張して単純な図形描画のみを行う GraphicsContext をラップし、GraphicsContext にフィギュアとレイアウト、コネクションの概念を追加したものである。GEF は MVC(Model-View-Controller)アーキテクチャを採用したエディターを作成するためのフレームワークである。ビューの部分で Draw2D を利用し、グラフィクスを描画する [15]。

2.3 ETL について

ETL(Extract-Transform-Load)とは、外部の情報源からデータを抽出して、抽出したデータをビジネスでの必要に応じて変換・加工し、最終的ターゲットに変換・加工済みのデータをロードする工程を指す。

Extract 工程は情報源となるシステムからデータを抽出することである。抽出においては、次の変換・加工工程に適したフォーマットに変換する。

Transfrom 工程では、情報源から抽出したデータに一連の規則や関数を適用し、ターゲットにロードできるデータにする。Transofrom の方式はデータのマッピング、合併、計算、修正などを含んでいる。

Load 工程は、データを最終ターゲットにロードする。ロード工程では、データロード時の変更について履歴と監査証跡を保持する場合もある [16]。

既存 ETL 製品としては、Talend などがある。しかし、IRS に対応する ETL ツールは存在していない。

第3章　　開発システム

本章では、本研究開発プロジェクトで開発する NET ツールの開発環境と構成について説明する。

3.1　　開発システムの概要

本プロジェクトで開発する NET ツールは通常の ETL ツールと同様に、IRS からデータを抽出する Extract 機能、データ変換を行う Transform 機能、データを PostgreSQL に挿入する Load 機能および GUI 表示機能から構成されている。

全 ETL 工程で NET ツールの位置づけを図 3.1 に示す。

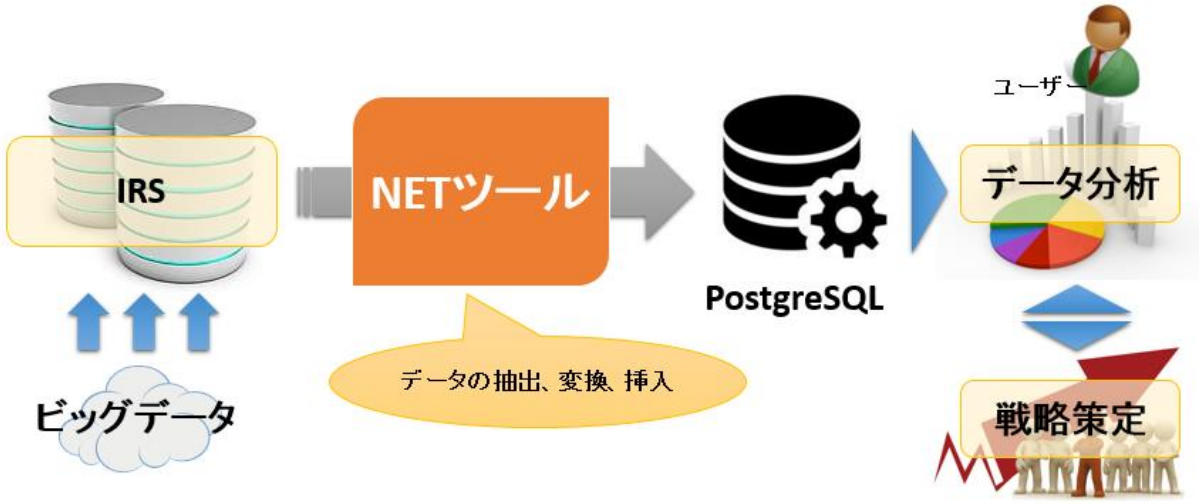


図 3.1 NET ツールの位置づけ

本章では、NET ツールの開発環境とシステム構成を説明する。

3.2　　開発環境

NET ツールの開発環境は IRS の実験環境と開発 PC から構成されている。

IRS の実験環境は高度 IT コース学生に貸与されているデスクトップ PC 四台から構成されている。NEC から提供された InfoFrame Relational Store 3.1 をインストールした。IRS インストレーションガイドによると、スペックの最小台数は 8 台であるが、限定的な利用のため、NEC の提案通り、4 ノードの形式を採用した。各ノードの位置づけを図 3.2 に示す。

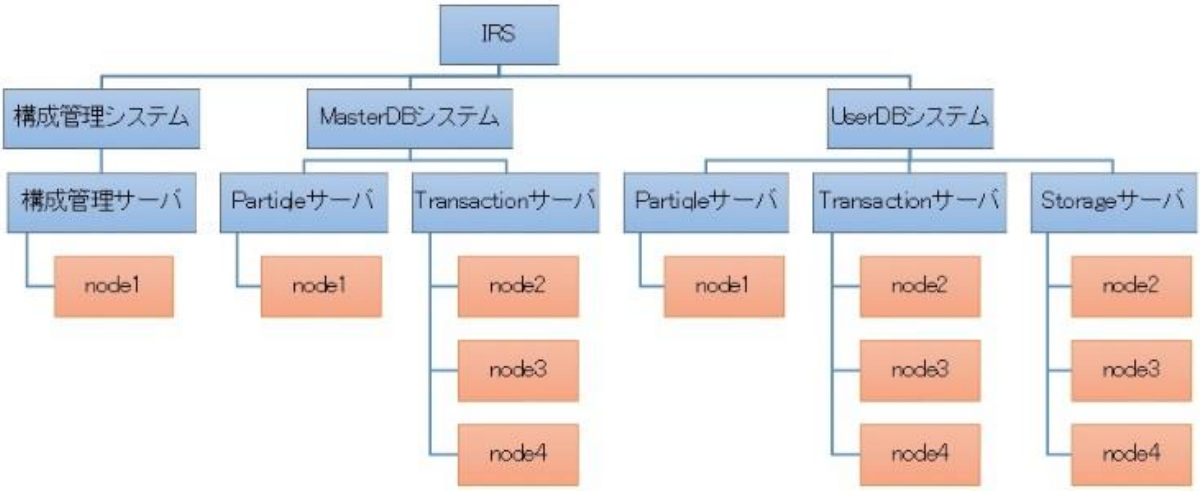


図 3.2 各ノードの位置づけ

開発 PC としては、高度 IT コース学生に貸与されているノート PC を使っている。開発作業は CentOS を搭載する仮想マシン(以下 VM)で行われ、学生居室 LAN を通じて IRS を利用するというような形式になっている。NET ツールは Eclipse プラグインとして設計・開発される。

VM についての情報を表 1 に示す。

表 3.1 VM の情報

VM バージョン	
VMware Workstation 10.0.3	
ハードウェア環境	
メモリ	2 GB
ハードディスク	40 GB
ネットワーク	ホストの IP アドレスを共有して使用
ソフトウェア環境	
OS	CentOS 6.5
データベース	PostgreSQL 8.4.20
Eclipse	Eclipse 4.3.2

本研究開発プロジェクトでは、Java を開発言語として使った。

3.3 開発システムの構成

本研究開発プロジェクトで開発する NET ツールは Extract 機能、Transform 機能、Load 機能および GUI 表示機能から構成されている。

NET ツールの構成を図 3.3 に示す。

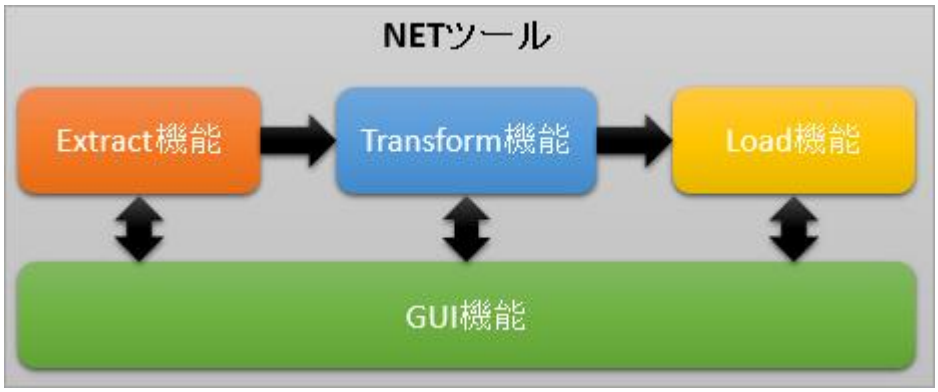


図 3.3 NET ツールの構成

3.3.1 Extract 機能

Extract 機能とは、データの抽出を行う機能である。IRS に接続し、IRS からユーザに指定された条件にしたがって、データの抽出を行っている。IRS の HadoopAPI を利用することで、高速なデータ抽出が可能になる。IRS のテーブル情報の表示機能も含まれている。

Extract 機能は NET ツールの開発で IRS に最も緊密に関連している機能である。

3.3.2 Transform 機能

Transform 機能とは、データの加工を行う機能である。Extract 機能から取得したデータを指定された規則にしたがって簡単な四則演算やデータ変換など、一般的な ETL ツールでサポートされている多様な変換機能を行っている [17]。

また、顧客の要求に応じて、Transform 機能をプラグイン化することで、変換方式が自由に指定できるようにする。

NET ツール開発の第 1 イテレーションでは、IRS から抽出したデータを PostgreSQL のデータ型に対応させるためのデータ型変換機能を実現した。第 2 イテレーションでは、Transform 機能のプラグイン化を実現して、Transform 機能プラグインの作成方法を顧客に提供した。

3.3.3 Load 機能

Load 機能は、データの挿入を行う機能である。PostgreSQL に接続して、Transform 機能から取得したデータを PostgreSQL に挿入する。PostgreSQL テーブルの新規作成機能とテーブル情報表示機能も含まれている。

3.3.4 GUI 機能

データ移動作業の作成、IRS と PostgreSQL への接続、Extract 条件の編集、Transform に

おけるデータのマッピングなど、すべての機能は GUI で使用されている。NET ツールを Eclipse プラグインとして開発するため、Eclipse の GUI を活用した。

3.4 開発システムの想定利用シナリオ

NET ツールの想定利用シナリオは以下に示す。

ある組織では、“Project Based Learning” (以下、PBL)という課題解決研修を行っている。PBL では、チームで顧客に対してヒアリングし、システムを設計、実装、テストを行う。PBL の開発環境として、Apache HTTP Server(以下、Apache)を用いている。そこで、Apache のアクセスログを蓄積し、分析することで各プロジェクトの運営状況を明確に把握できると考えられる。

このような環境において、我々が開発する NET ツールは IRS から分析ツールに移行する際に GUI によるインターフェースを提供し、SQL や Hadoop に造詣が深くないユーザでもデータ分析を可能にする。

想定利用シナリオの概要を図 3.4 に示す。

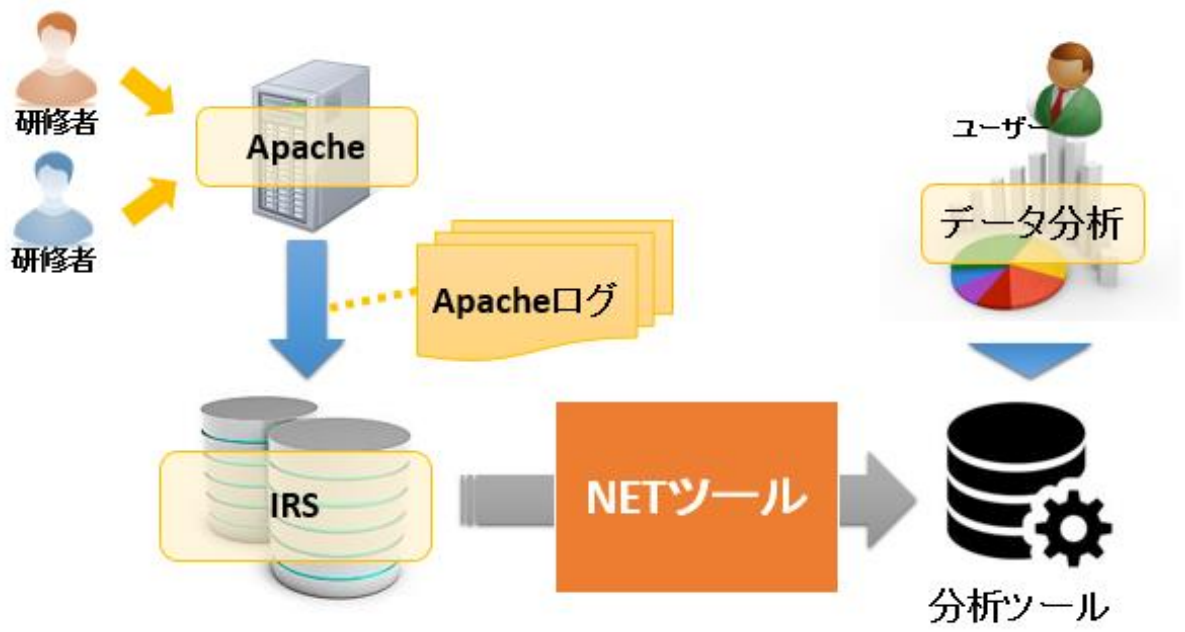


図 3.4 想定利用シナリオの概要図

第4章　開発体制

本節では、本研究開発プロジェクトの関係者構成と役割、およびスケジュールについて説明する。

4.1　関係者構成と役割

本研究開発プロジェクトの関係者は NEC から 1 人と筑波大学の先生 1 人、および学生 4 人、全部で 6 人となる。関係者構成を表 4.1 に示す。

表 4.1 プロジェクトの関係者

NEC	顧客	システムソフトウェア事業部
筑波大学	担当者	天笠 俊之 先生
	開発者	斎藤 優太
		成澤 翔平
		HU BEIQI
		唐 可

NET ツール開発の役割分担を表 4.2 に示す。

表 4.2 役割分担

斎藤 優太	チームリーダー、要件定義、Load 部分の設計・開発・テスト、NET ツールの実行画面、性能調査、納品報告書・セットアップマニュアル作成
成澤 翔平	画面定義書、TransformWindow 以外の GUI 設計・開発・テスト、一部ユーザマニュアルの作成、総合テスト(第 2 イテレーション)
HU BEIQI	IRS サーバ管理、Transform 機能と TransformWindow の設計・開発・テスト、総合テスト(第 1 イテレーション)、一部ユーザマニュアルの作成、Transform 機能のプラグイン化、Transform プラグイン作成マニュアル
唐 可	Extract 部分の設計・開発・テスト，Hadoop 環境の構築・障害対処

4.2　スケジュールと実績

本研究開発プロジェクトの予定としては，11 月末までに 2 回のウォータフォール型の開発を行う。第 1 回目のウォータフォール型の開発を第 1 イテレーション、第 2 回目のウォータフォール型の開発を第 2 イテレーションと呼ぶ。第 1 イテレーションの前に開発準備期間を設けた。第 2 イテレーション終了後、報告準備期間を設けた。

各段階の予定を表 4 に示す。

表 4.3 各段階の予定

段階	概要
開発準備期間	全体の要件を決め、開発構想書を作成し、開発環境を構築する
第 1 イテレーション	開発構想書の開発概要に基づき要件を明確化し、第 1 イテレーションで実装する機能と担当を決めた後、設計開発を行う
第 2 イテレーション	第 1 イテレーションの状況によって詳細を検討し，追加開発を行う
報告準備期間	特定課題研究報告書の執筆と特定課題研究報告会での発表用デモンストレーションの作成を行う

開発実績として、第 1 イテレーションでは、見積もりや技術不足のため，実装が予定通り完了しなかった。このため、第 1 イテレーションで一部機能を落とし、2 週間ぐらい延期した。

第 1 イテレーション終了時、振り返りを行い、延期の原因をまとめて、コアタイム設置などの対策を考え、対策を第 2 イテレーションに適用した。また、中間報告の準備をするための中間報告準備期間が追加された。

各段階の予定期間と実績を図 4.1 に示す。

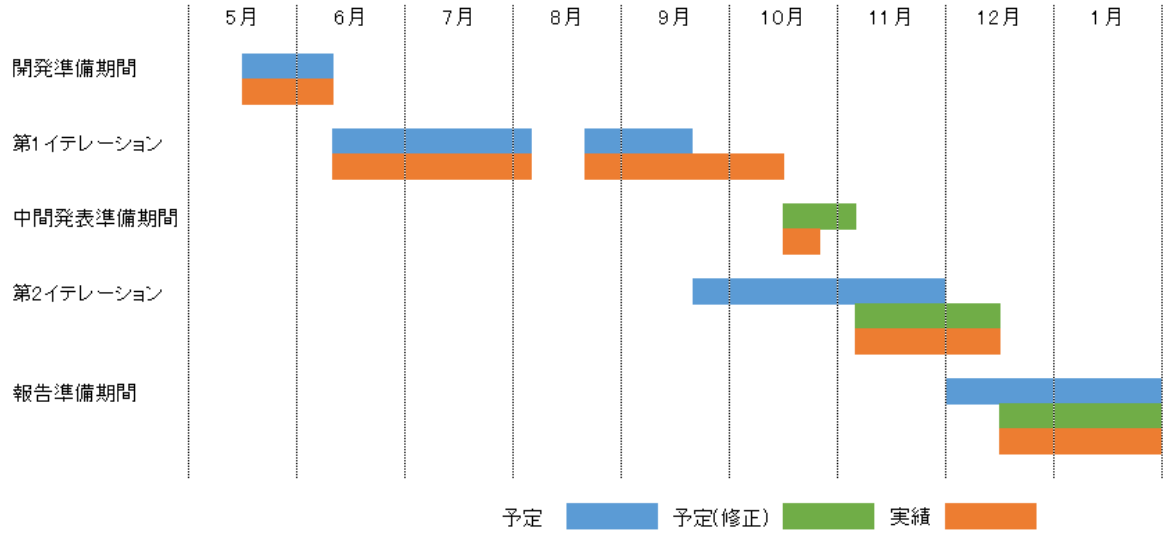


図 4.1 各段階の予定期間と実績

第 1 イテレーションについて、各フェーズの予定と実績を図 4.2 に示す。

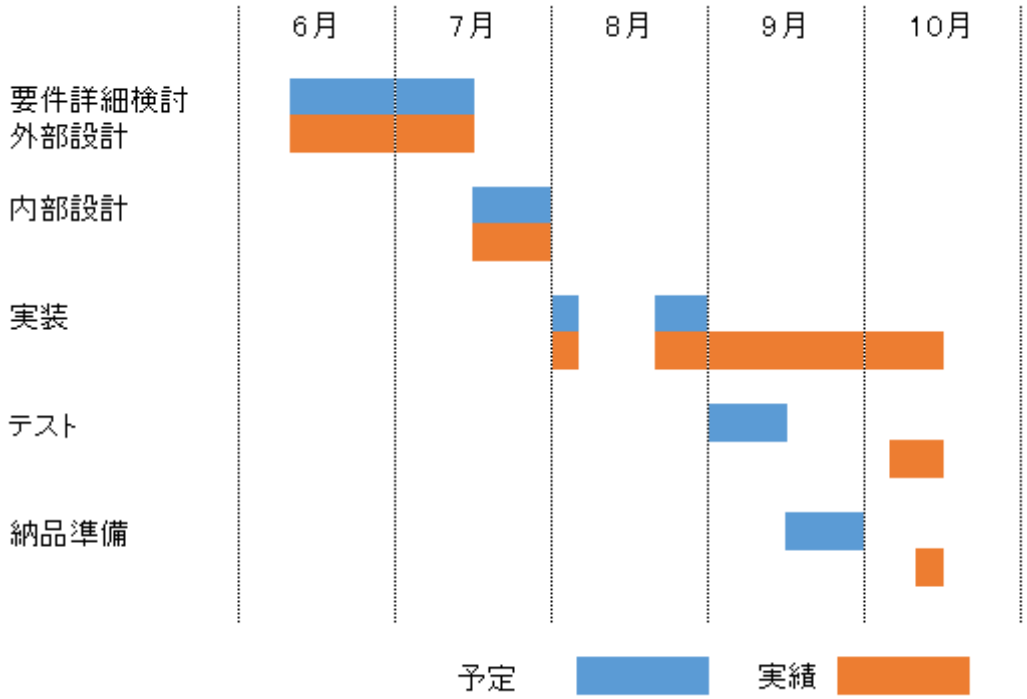


図 4.2 第 1 イテレーションのスケジュール詳細

筆者が担当した Transform 機能の設計開発の実績は図 4.3 に示す。

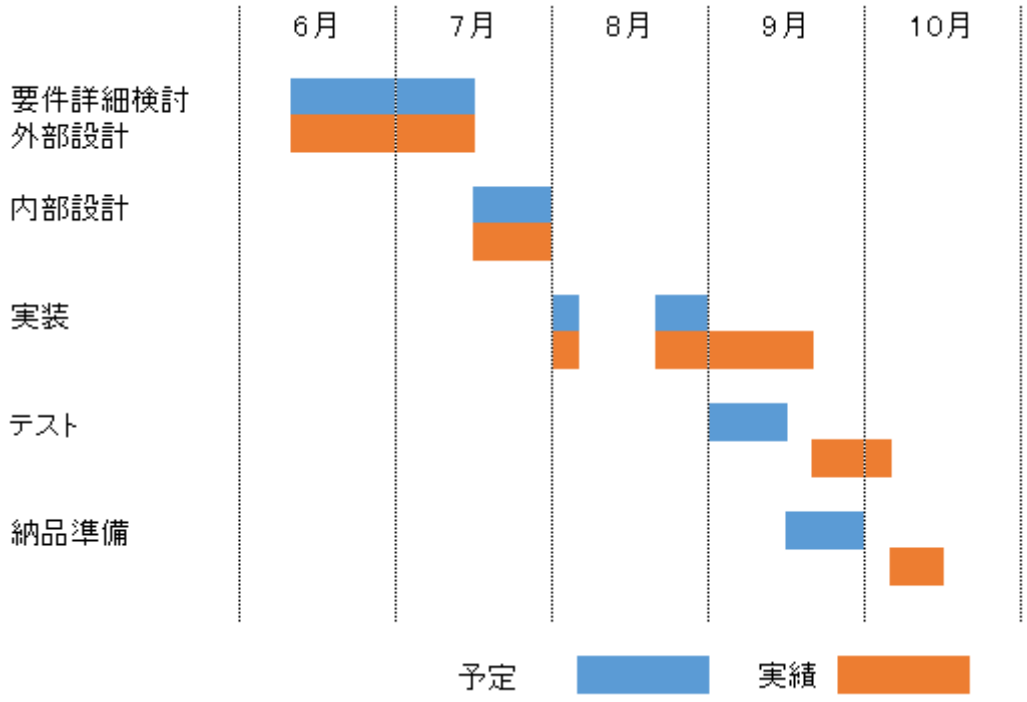


図 4.3 Transform 機能の設計開発実績

第 2 イテレーションでは、第 1 イテレーションで残されたバグをフィックスし、顧客の意見に応じて修正・追加開発を行った。第 1 イテレーションの経験を踏まえて、第 2 イテレーションで予定通り顧客に納品した。第 2 イテレーションで、筆者が担当した Transform プラグイン化機能の設計開発の実績は全体の実績と同じであった。

第 2 イテレーションについて、各フェーズの予定と実績を図 4.4 に示す。

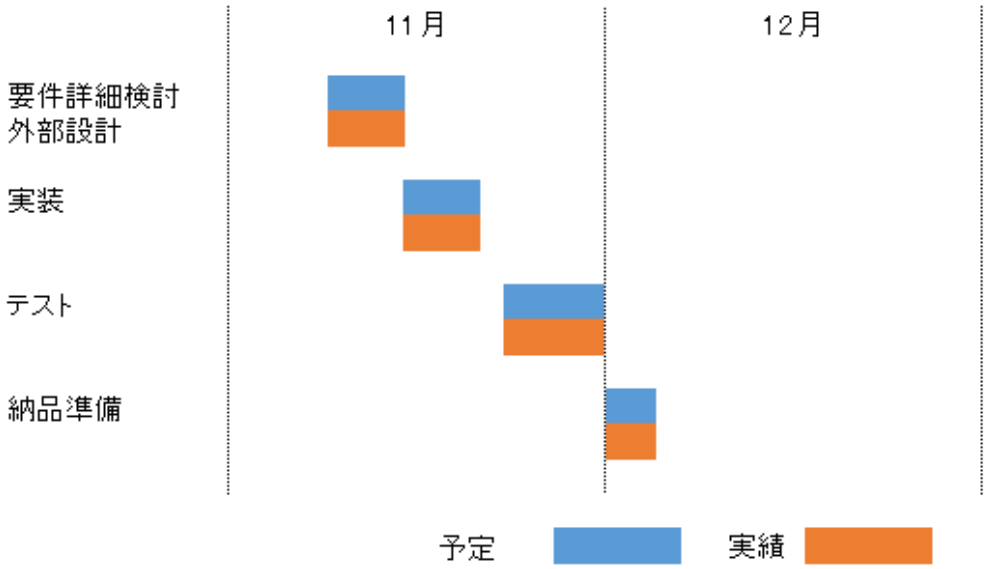


図 4.4 第 2 イテレーションのスケジュール詳細

第 2 イテレーションでは、チーム内のコミュニケーションを促進するために、コアタイムを導入した。コアタイムの時間を表 4.4 に示す。

表 4.4 コアタイムの時間設定

	月	火	水	木	金
午前	10~12 時	10~12 時	10~12 時	10~12 時	10~12 時
午後	13~15 時	—	13~15 時	—	13~15 時

第5章　NET ツール開発環境の構築

NET ツールの開発環境は IRS サーバとチームメンバー各自の開発 PC から構成されている。筆者は IRS サーバのインストールと運用・保守を担当した。本章では、IRS サーバのインストール、運用について説明する。

5.1　IRS インストール前の準備

NEC から提供されたインストレーションガイドによると、すべてのサーバは Red Hat Enterprise Linux 6 で動作するが、限定的な利用のため、CentOS を使用することを提案して、NEC の合意を取った。本研究開発プロジェクトで、高度 IT コース学生に貸与されているデスクトップ PC 四台を集め、CentOS 6.5 をインストールした。

NEC からいただいたインストレーションガイドに記載されている IRS 各サーバの動作環境を表 5.1 に示す [18]。

表 5.1 IRS インストールに必要な動作環境

	Partiqle サーバ	トランザクションサーバ	ストレージサーバ	構成管理サーバ
OS	Red Hat Enterprise Linux 6 (x64) 以上 (RHEL 6.4 推奨)	Red Hat Enterprise Linux 6 (x64) 以上 (RHEL 6.4 推奨)	Red Hat Enterprise Linux 6 (x64) 以上 (RHEL 6.4 推奨)	Red Hat Enterprise Linux 6 (x64) 以上 (RHEL 6.4 推奨)
必要メモリ	16GB 以上	16GB 以上	16GB 以上	4GB 以上
必要ディスク	100GB 以上	100GB 以上	100GB 以上	100GB 以上
シェル	Bash	Bash	Bash	Bash
Java	Java SE 7 (update 45 以降)	Java SE 7 (update 45 以降)	Java SE 7 (update 45 以降)	Java SE 7 (update 45 以降)
AP サーバ	—	—	—	WebOTX V9.1
最小台数	1 台	3 台	3 台	1 台

本研究開発プロジェクトにおいて IRS インストールの実際環境を表 5.2 に示す。

表 5.2 IRS インストールの実際環境

ホスト		node1	node2	node3	node4
役割	1	構成管理サーバ	Transaction サーバ（MasterDB）	Transaction サーバ（MasterDB）	Transaction サーバ（MasterDB）
	2	Partiqle サーバ(MasterDB)	Transaction サーバ（UserDB）	Transaction サーバ（UserDB）	Transaction サーバ（UserDB）
	3	Partiqle サーバ（UserDB）	Storage サーバ（UserDB）	Storage サーバ（UserDB）	Storage サーバ（UserDB）
IP		192.168.11.44	192.168.11.36	192.168.11.43	192.168.11.128
OS		CentOS 6.5	CentOS 6.5	CentOS 6.5	CentOS 6.5
メモリ		15.5 GB	15.5 GB	15.5 GB	15.5 GB
ディスク		200 GB	200 GB	200 GB	200 GB
シェル		Bash	Bash	Bash	Bash
Java		Java SE 7 (update 45)	Java SE 7 (update 45)	Java SE 7 (update 45)	Java SE 7 (update 45)
WebOTX		WebOTX V9.1	—	—	—

本研究開発プロジェクトにおいて、IRS に対する性能要求がないので、開発コストを節約するため、NEC からの提案通り 4 ノードのインストール形式を採用した。各 PC は複数のサーバの役割を担当している。

また、CentOS に貸与された PC のネットワーク設備に対応するドライバーがないため、ドライバーのインストールを行った。

5.2 IRS のインストール

IRS のインストールには、事前準備(OS の基本設定)、パッケージのインストール、カーネルパラメータの設定、RSMasterDB の構築、構成管理の構築、RSUserDB の構築、IRS の起動などの段階がある。

IRS インストレーションガイドによると、RSUserDB の構築と IRS の起動の方式にはコマンドと管理コンソール 2 種類が存在している。本プロジェクトでは、コマンドでの方式を採用した。

IRS インストールの流れを図 5.1 に示す。

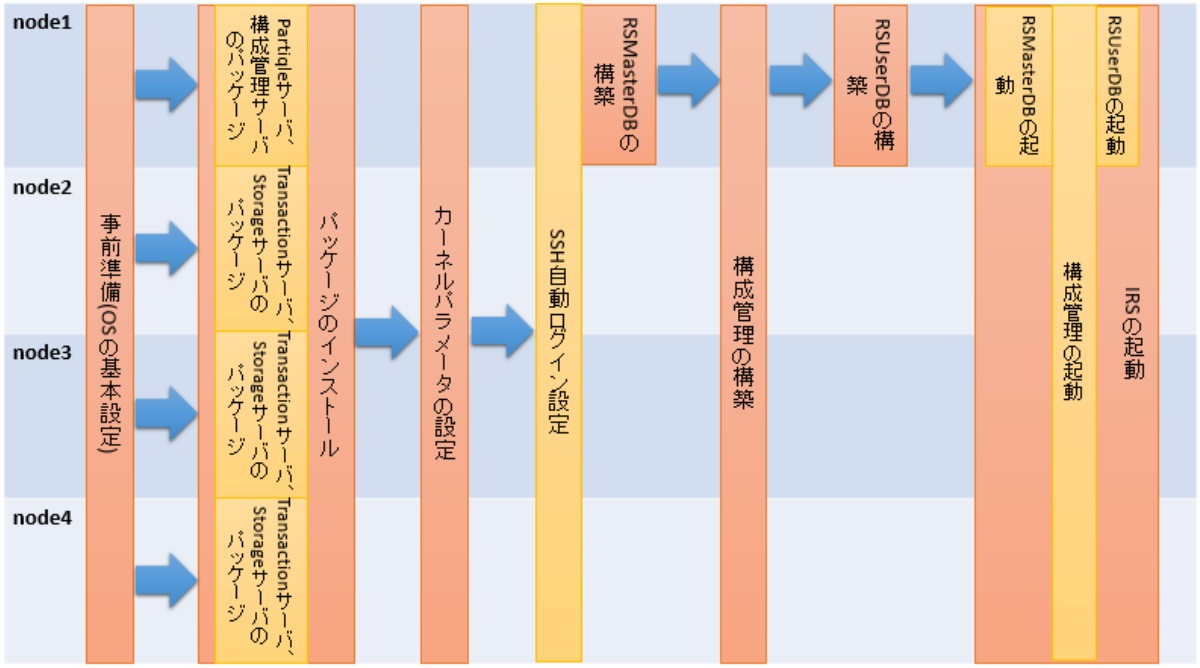


図 5.1 IRS インストールの流れ

5.3 IRS の運用

第 1 イテレーションと第 2 イテレーションの設計開発フェーズで、ネットワーク環境の不安定などの原因で IRS に障害が発生した。

障害が発生し、トラブルシューティングを行ったが、不慣れな操作で非常に時間がかかるため、NEC と相談した結果、プロジェクトの効率を考えたうえで、IRS の再インストールを決めた。その際に、筆者が IRS サーバの管理担当として、IRS の再インストールを行った。

また、本研究開発プロジェクトには IRS の利用が不可欠なので、障害が発生する時早めに対応する必要がある。

第6章　　Transform 機能の設計開発

本研究開発プロジェクトで、筆者は NET ツール Transfrom 機能の設計開発を担当した。本章では、Transform 機能の設計開発について説明する。

6.1　　データ変換

6.1.1　　データ変換機能の概要

Transform のデータ変換は簡単な四則計算、データの合併、単語の置換、データ型変換などを含む。機能のイメージは図 6.1 に示す。

図 6.1 Transform 機能のイメージ図

想定として、Transform 作業の入力データから出力データまで、ユーザはデータ変換の規則を関数(ファンクション)として定義する。また、前項から出力する結果とユーザから入力された情報も四則計算の引数とする可能性がある。

Transform 過程の例を図 6.2 に示す。

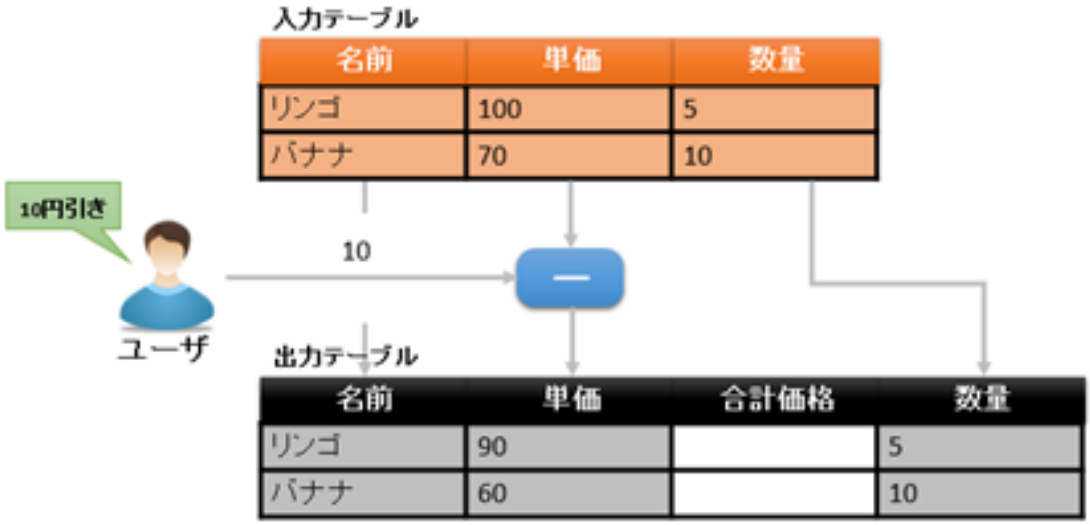


図 6.2 Transform 過程の例(a)

図 6.2 では、減算の引数は入力テーブルから取得した単価とユーザに定義された割引金額である。また、何も変換せずに出力テーブルに移動する場合(例えば、入力テーブルの数量から出力テーブルの数量)もある。

さらに減算した上で、別の変換を追加することもできる。乗算が追加された場合を図 6.3 に示す。

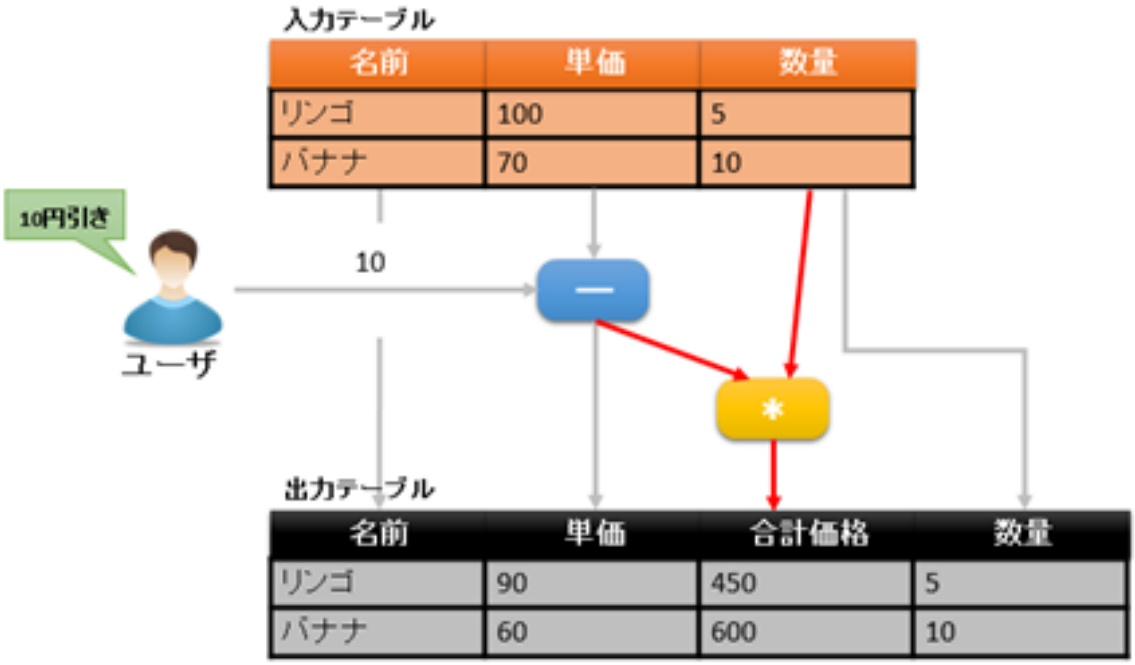


図 6.3 Transform 過程の例(b)

図 6.3 のように、算出した減算の結果を乗算の引数とすることも可能である。

NET ツールの Transform 機能では、変換関数を作成し、変換の流れを自由に指定することができる。

この Transform 機能は 1 レコードのデータを対象とする。Extract 機能から IRS のテーブル情報と抽出されたデータ、Load 機能から PostgreSQL のテーブル情報、GUI 機能からユーザに定義された変換方式を取得することは Transform 機能の前提条件である。Transform 機能完了後、変換したデータを Load 機能に送る。

利用するテーブル情報を NET ツールで共通の Table クラスで管理しており、データを共通の DataSet クラスで管理している。Table クラスは基本的にデータベース名、テーブル名、カラム数、および各カラムの名前、データ型、サイズを持っている。DataSet クラスは基本的に 1 レコードのデータを格納するデータリストとカラム数を持っている。

Transform 機能に関する入出力情報を図 6.4 に示す。

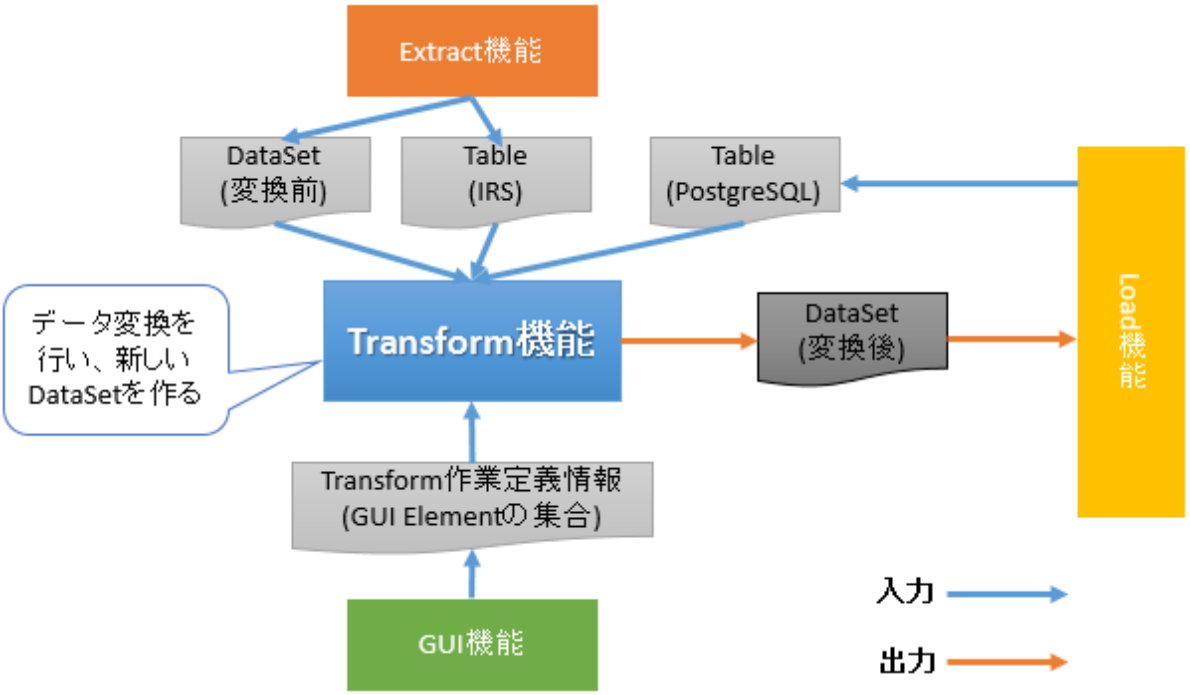


図 6.4 Transform 機能に関する入出力情報

6.1.2 Transform 機能用「Element」の定義

Transform 機能のこの柔軟性を実現するために、入力カラム、出力カラム、四則演算ファンクションなどの要素を本 Transform 機能で定義した。本 Transform 機能で、これらの要素を「Element」と呼ぶ。「Element」の定義を図 6.5 に示す。

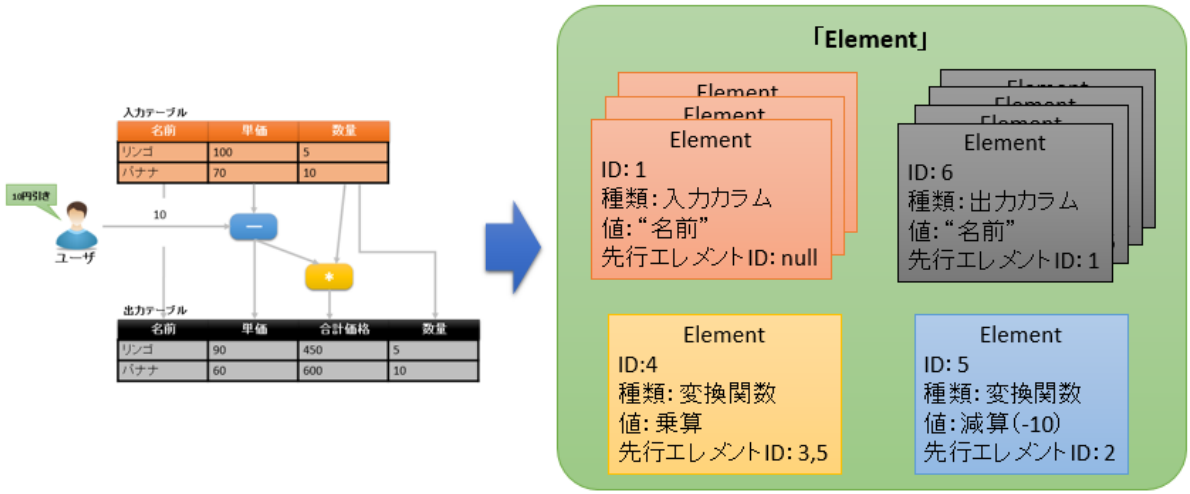


図 6.5 「Element」の定義

入力カラムを入力カラム Element、出力カラムを出力カラム Element、減算や乗算ファン

クションを変換関数 Element として管理する。

Element についてのクラス図は図 6.6 に示す。

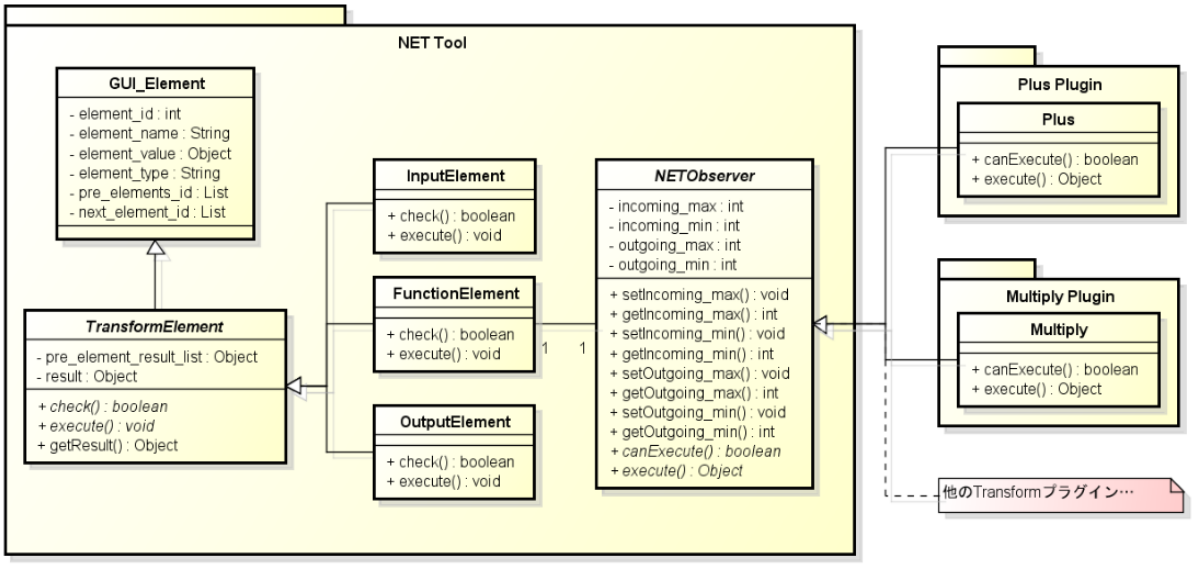


図 6.6 Element のクラス図

GUI Element は Transform 作業を編集する時に生成されたエレメントであり、Element の基本情報(ID、種類、バリュー、先行 ID リスト、後継 ID リスト)を持っている。Transform 機能を実行する時に、GUI から取得した GUI Element を拡張して、Transform 用の Transform Element を作成する、Transform Element は 3 種類のエレメントを含んでいて、入力カラム処理、出力カラム処理、変換関数処理などを担当している。

GUI Element と Transform Element の構造を図 6.7 に示す。

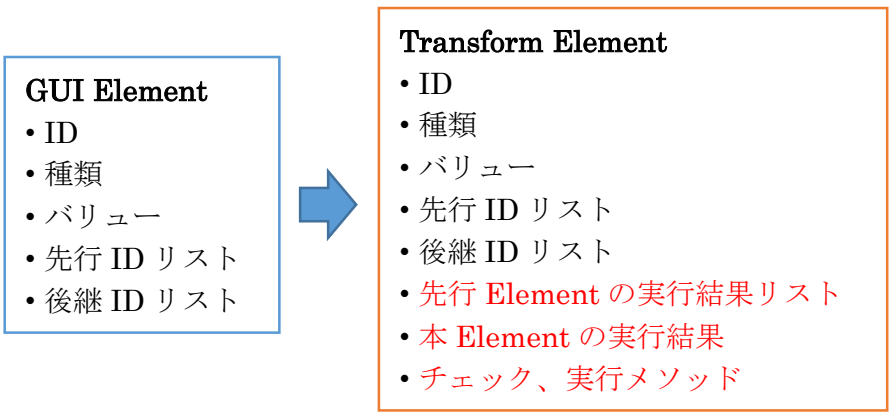


図 6.7 GUI Element と Transform Element の構成

すべての Transform Element はチェックメソッドと実行メソッドを持っている。

チェックメソッドは該当 Element の結果が算出できるかどうか確かめるメソッドである。算出できる場合に、該当 Element の実行メソッドを呼び出す。算出できない場合に、Transform 作業を停止する。Element が変換関数の場合に、該当する Transform プラグインを呼び出し、プラグイン内の canExecute 関数を利用することでチェックを行う。チェックメソッドのイメージ図を図 6.8 に示す。

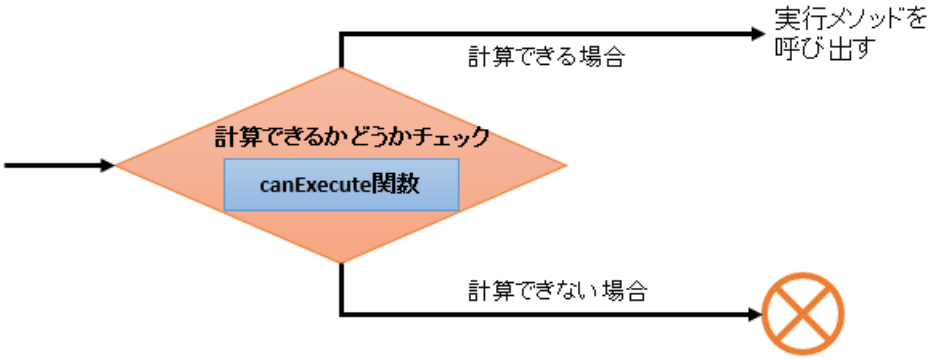


図 6.8 チェックメソッドのイメージ図

実行メソッドの内容は Element の種類によって異なる。入力カラム Element の実行内容は入力カラムのデータを結果として生成するだけだが、出力カラム Element の実行内容はデータ型の変換である。また、変換関数 Element の実行内容はユーザに指定された Transform プラグインを呼び出し、定義された変換を行うことである。

例えば、図 6.3 の場合に、以下の GUI Element が作られている。(バリューの値に関しては、入力カラムか出力カラムの場合は、バリューはカラム名であり、変換関数の場合は，使用する Transform プラグインの ID である。)

表 6.1 図 6.3 の Transform 作業での GUI Element 一覧

ID	Element タイプ	バリュー	先行 ID	後継 ID	備考
1	入力カラム	名前	null	6	
2	入力カラム	単価	null	5	
3	入力カラム	数量	null	4, 9	
4	変換関数	function_multiply	3, 5	8	乗算関数
5	変換関数	function_minus10	2	4, 7	減算関数
6	出力カラム	名前	1	null	
7	出力カラム	単価	5	null	
8	出力カラム	合計価格	4	null	
9	出力カラム	数量	3	null	

6.1.3 データ変換の流れ

Transform 作業を実行する場合、実行済みリストと実行待ち循環リストという 2 つの

Element リストが作られる。(ここの Element は Transform Element を指す。)

初期状態で、実行済みリストは空リストであり、Element がすべて実行待ち循環リストに格納されている。データ変換が始まると、実行待ち循環リストにある各 Element が順に処理される。

実行待ち循環リストの初期状態例を図 6.9 に示す。

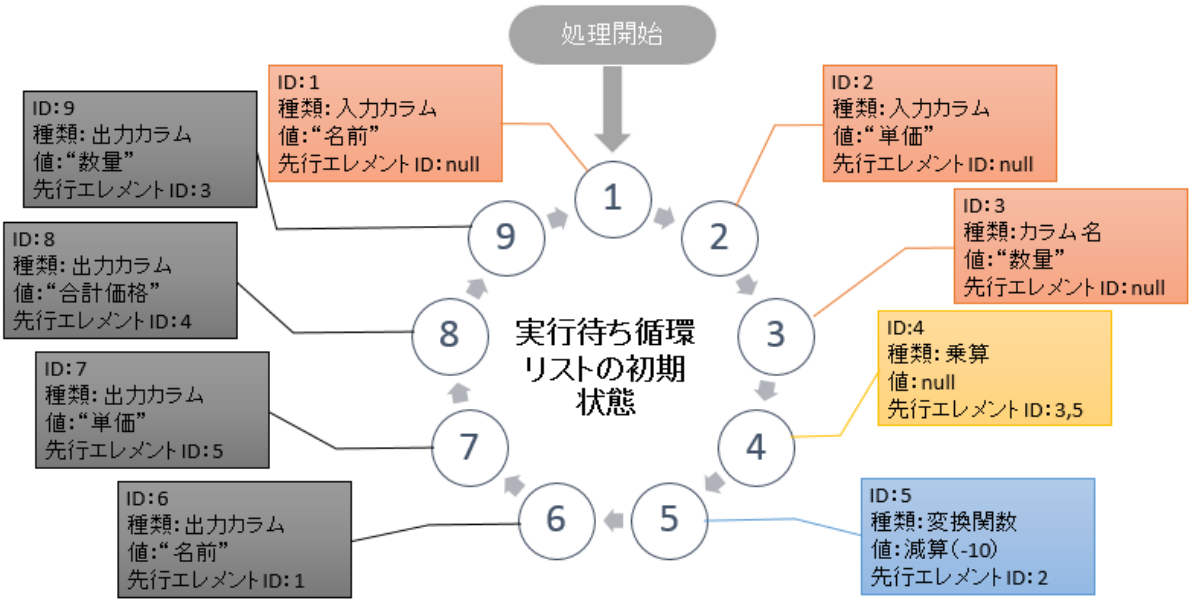


図 6.9 実行待ち循環リストの初期状態例

処理時、まず実行済みリストを照会することで、先行 ID に対応する先行 Element の実行が済んだかどうかチェックする。

先行する Element がすべて実行済みの場合、現 Element 自身のチェックと実行メソッドを実行して、Element の結果が算出される。Element 実行後、この Element を実行待ち循環リストから実行済みリストに移動し、実行待ち循環リストにある次の Element を処理する。

例えば、表 6.3 にある、ID が 1 の Element を処理する場合、先行する Element が存在しないので、この Element のチェックメソッドと実行メソッドを実行し、Element を実行済みリストに移動する。この場合実行待ち循環リストと実行済みリストの状態変更を図 6.10 に示す。

図 6.10 実行待ち循環リストと実行済みリストの状態変更例(a)

先行する Element に未実行の Element がある場合、この Element の処理ができないため、当 Element をスキップして、次の Element を処理する。この未実行 Element はまだ循環リスト（実行待ち循環リスト）に格納されていて、次の番を待つ。この期間に、もし先行する Element の処理が済んだら、この Element が処理できるようになる。

例えば、表 6.3 にある、ID が 4 の Element を処理する場合、先行する ID が 5 の Element がまだ実行されていないため、この Element をスキップして、次の Element(ID は 5 である)を処理する。この場合実行待ち循環リストと実行済みリストの状態変更例を図 6.11 に示す。

図 6.11 実行待ち循環リストと実行済みリストの状態変更例(b)

Element 処理の全体流れを図 6.12 に示す。

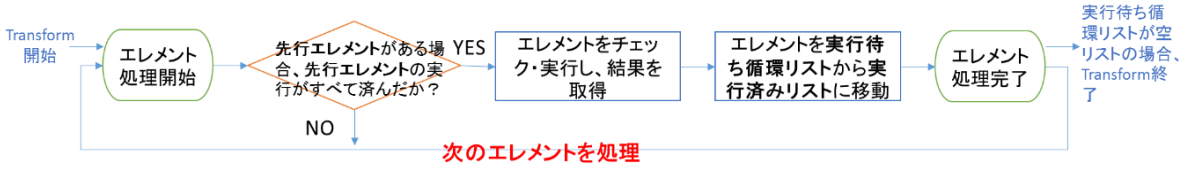


図 6.12 Element 処理の全体流れ

実行待ち循環リストが空リストになると、1 レコードの Transform 過程が完了したことが

分かる。

例えば、図 6.3 にある入力テーブルの 1 行目<リンゴ, 100, 5>を対象にデータ変換を行う場合、表 6.1 にある Element を処理する。処理の流れを表 6.2 に示す(初期の実行待ち循環リストにあるエレメント ID は 1,2,3,4,5,6,7,8,9 である)。

表 6.2 表 5.1 にある Element の処理流れ

処理順番	処理 Element ID	先行 Element ID	先行 Element の実行結果	処理概要	実行結果	処理後の実行待ち循環リストにある Element ID
1	1	null	null	入力カラム処理(名前)	リンゴ	2,3,4,5,6,7,8,9
2	2	null	null	入力カラム処理(単価)	100	3,4,5,6,7,8,9
3	3	null	null	入力カラム処理(数量)	5	4,5,6,7,8,9
4	4	3, 5	-	未実行の先行 Element(5 番)があるため，スキップ	-	5,6,7,8,9,4
5	5	2	100	減算処理(2 番の結果-10)	90	6,7,8,9,4
6	6	1	リンゴ	出力カラム処理(名前)	リンゴ	7,8,9,4
7	7	5	90	出力カラム処理(単価)	90	8,9,4
8	8	4	-	未実行の先行 Element(4 番)があるため，スキップ	-	9,4,8
9	9	3	5	出力カラム処理(数量)	5	4,8
10	4	3, 5	5,90	乗算処理(3 番の結果*5 番の結果)	450	8
11	8	4	450	出力カラム処理(合計価格)	450	空

最後、実行済みリストから出力カラム Element を抽出し整理すると、Transform 処理の結果が分かる。上述処理の場合、名前、単価、合計価格、数量の出力カラム Element を抽出し整理して、Transform 処理結果<リンゴ, 90, 450, 5>が得られる。

もし実行待ち循環リストのサイズがいつも変わらない場合、Element の対応関係指定に問題があるためである可能性が高い。データの変換に問題が発生する場合、エラーメッセージが表示され、Transform 作業が停止する。

6.1.4 データ型の変換

IRS のデータ型と PostgreSQL のデータ型に差異があるので、データを PostgreSQL に挿入する前に、データ型を変換する機能が必要である。NET ツールのデータ型変換機能は出力カラム Element で行われている。

NET ツールが対応しているデータ型変換を表 6.3 に示す。

表 6.3 対応しているデータ型変換

IRS データ型	PostgreSQL データ型	内部動作 Java データ型
INTEGER, INT	int4, serial	java.lang.Integer
BIGINT	int8	java.lang.Long
DECIMAL	decimal	java.math.BigDecimal
FLOAT	float4	java.lang.Float
DOUBLE	float8	java.lang.Double
CHAR, VARCHAR, TEXT	char, varchar, text	java.lang.String
DATE	date	java.sql.Date
TIME	time	java.sql.Time
TIMESTAMP, DATETIME	timestamp	java.sql.Timestamp
BOOLEAN, BOOL	boolean	java.lang.Boolean
BINARY, VARBINARY	bytea	java.lang.Byte の配列 byte[]

データ変換の場合、基本的に表 6.3 の通り対応しているデータ型間の変換ができる。例えば、IRS から INTEGER 型のデータを int4 型の PostgreSQL カラムに挿入する場合、INTEGER から int4 の変換ができる。それ以外の場合は対応できない。

対応できないデータ型が検出された場合、またはデータ変換ができない場合、エラーメッセージのポップアップが表示され、Transform 作業が停止する。データ型が変換できない場合のエラーメッセージ例を図 6.13 に示す。

図 6.13 データ型変換ができない場合のエラーメッセージ例

6.2 データマッピング

データマッピングとは、GUI でカラムや変換関数の対応関係を指定する機能である。前述の GUI Element はここで作成されている。

Transform 作業は Transform Window という画面で定義されている。Transform Window を開いた時、Extract 部分から取得されたテーブルとユーザに選択された PostgreSQL テーブルはすでに表示されていて、テーブル情報をもとに、そのテーブルの各カラムを対象に GUI Element が作られる。

Transform Window 初期状態のイメージ図を図 6.14 に示す。

図 6.14 Transform Window 初期状態

Function ビューにある変換関数を Transform Window にドラッグ&ドロップすると、変換関数の追加ができる。変換関数のドラッグ&ドロップ例を図 6.15 に示す。

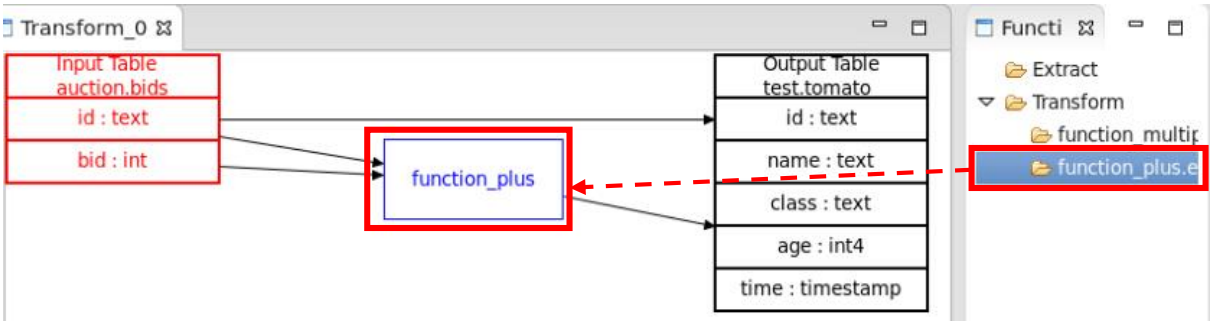


図 6.15 変換関数のドラッグ&ドロップ例

ユーザは各要素の間に線を引くことで、入力カラム、出力カラム、変化関数の間の対応関係を指定する。対応関係指定のイメージ図を図 6.16 に示す。

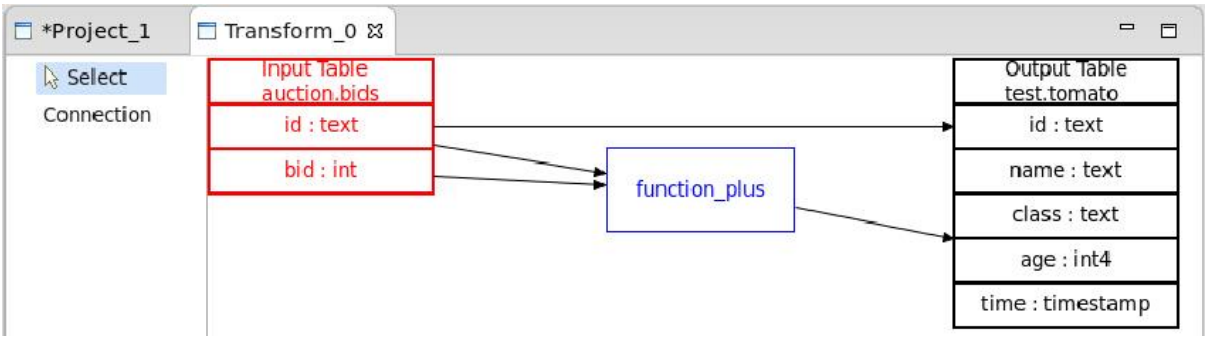


図 6.16 対応関係を指定

線が引かれる際に、関連する GUI Element の先行 ID リストか後継 ID リストは修正される。

また，各要素の特徴をもとに、線が引かれる時に、以下の制限が設けられた。

1) テーブルを対象とする変換がないため、テーブルを連結対象とすることができない。

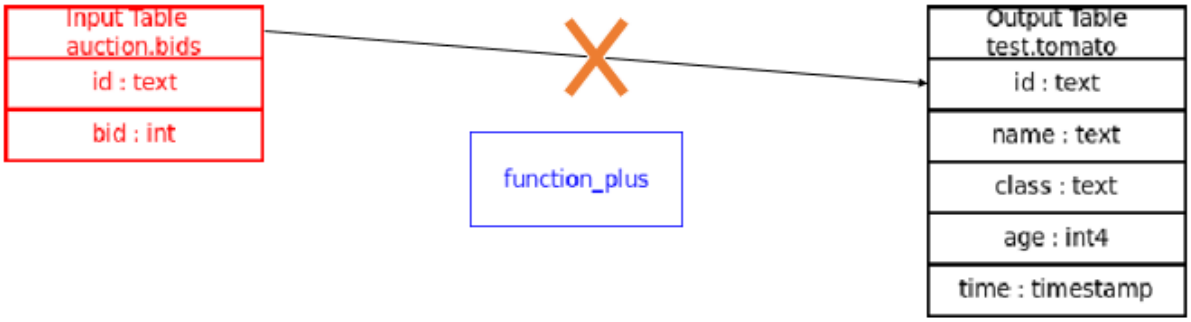


図 6.17 線を引く場合の制限(1)

2) 入力カラムに対して、先行する Element が存在しないため、入力カラムを線の終点とすることができない。

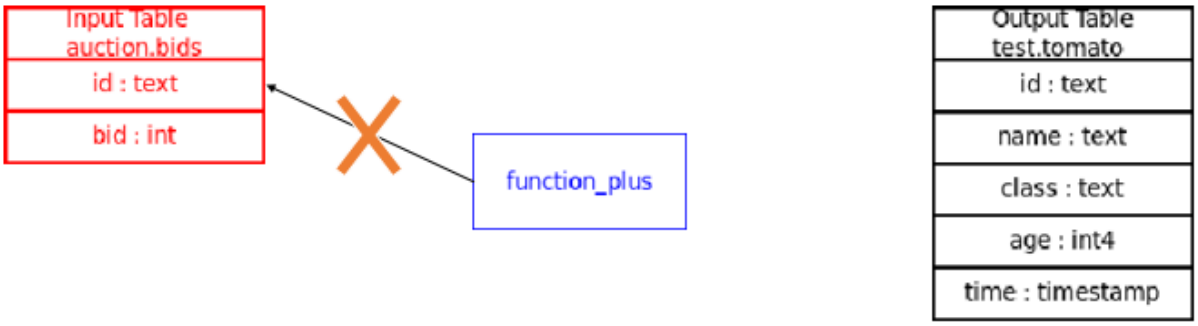


図 6.18 線を引く場合の制限(2)

3) 出力カラムに対して、後継する Element が存在しないため、出力カラムを線の始点とすることができない。

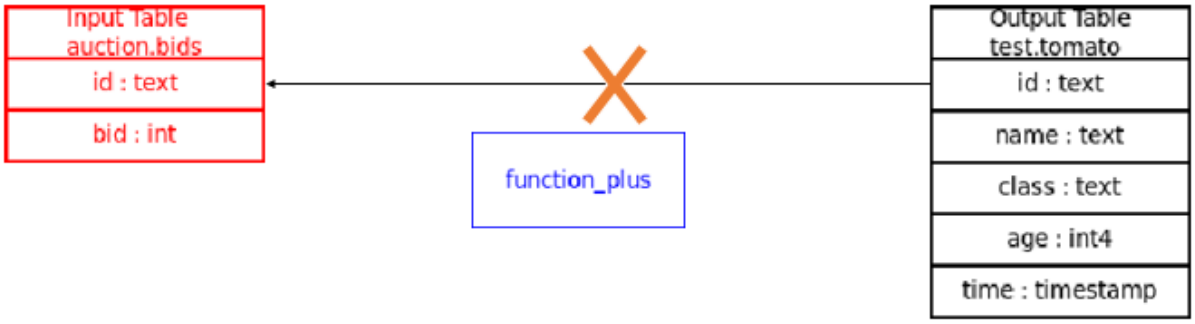


図 6.19 線を引く場合の制限(3)

4) 出力カラムへの入力は 2 つ以上存在すると、出力カラムの処理はできなくなってしまうため、出力カラムとつながる線は 1 本しかない。

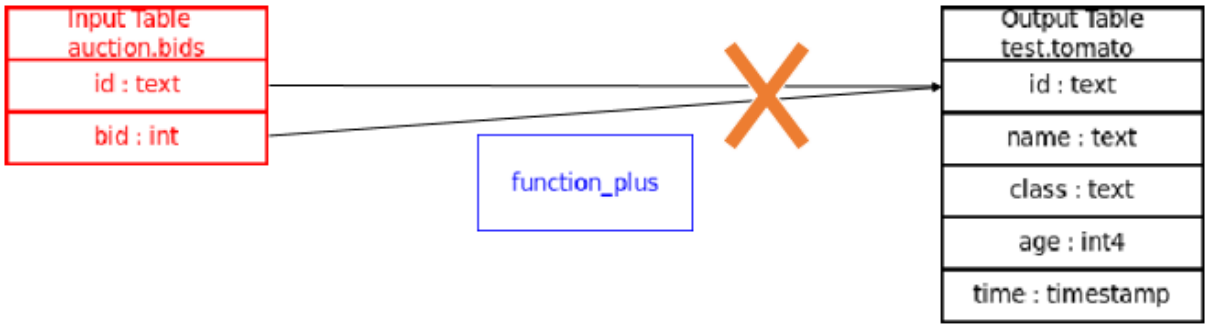


図 6.20 線を引く場合の制限(4)

5) 変換関数の入力矢印数と出力矢印数は Transform プラグインの設定値に制限される。(Transform プラグイン作成時、入力矢印数と出力矢印数の上限・下限が作成者に設定された。設定されていない場合、上限を Integer.MAX_VALUE、下限を 0 とする。)

Transform Window を実装する時に、Eclipse に搭載されている GEF を利用した。GEF を活用することで、テーブル・カラムと変換関数をアイコンとして表示する機能、線での連結機能、およびテーブル、変換関数および線の削除、Undo、Redo 機能を実現した。

また、テーブルや線が修正される同時に、GUI Element の修正も自動に行われている。

6.3 Transform 機能のプラグイン化

6.3.1 Transform 機能プラグイン化の概要

顧客の要求として、いろいろな変換が考えられる。例えば、固定数をたす加算や文字列の置換など、それらを NET ツールに簡単に追加できることが期待されている。顧客の要求に応じて、我々は Transform 機能をプラグイン化して NET ツールから呼び出せるようにすることを提案した。顧客と相談した結果、Transform 機能のプラグイン化に関する調査を着手した。調査結果として、この機能のプラグイン化が可能であり、Transform 機能をプラグイン化することを決めた。

Transform 機能プラグイン化のイメージを図 6.21 に示す。

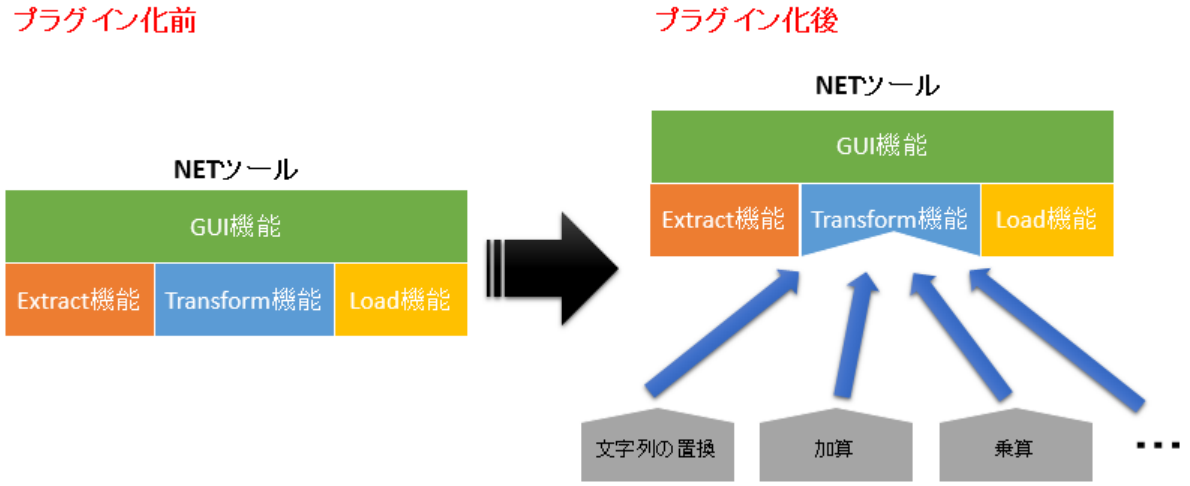


図 6.21 Transform 機能プラグイン化のイメージ図

NET ツール開発の第 2 イテレーションでは、Transform 機能のプラグイン化をした。Transform プラグインを作成することで、自由に変換方法を追加することができるようになった。

NET ツールの Transform プラグインを作成するため、拡張ポイントを作る必要がある。新しい拡張ポイントを作成し、利用するに必要な手順は以下である [19]。

1) 拡張ポイントの宣言
2) 拡張ポイント・スキーマ・ファイルの作成
3) 拡張オブジェクト用のクラスまたはインターフェースの作成

4) 拡張の宣言
5) 拡張をロードするコードの作成
6) ロードした拡張を利用するコードの作成

作成したプラグインを jar ファイルとしてエクスポートし、Eclipse インストール先の plugins フォルダに置くと、このプラグインが使える。

NET ツールでは、拡張ポイント、スキーマを作り、拡張オブジェクト用のクラスを作成した。また、サンプル Transform プラグインを作成し、顧客に提供した。

6.3.2 拡張オブジェクト用クラスの用意

NET ツール側で作成した拡張オブジェクト用のクラスでは NETObserver と呼ぶ。NETObserver クラスで、変換関数入力矢印数と出力矢印数の制限を設定するメソッドおよび canExecute、execute という抽象メソッドが用意されている。先行エレメントの実行結果リストを canExecute と execute の引数とする。

NET ツールが Transform プラグインを実行する時に、まず canExecute メソッドで変換関数が実行できるかどうか判断して、もし実行できると判断された場合、execute メソッドで具体的な変換を実行する。

6.3.3 Transform プラグインの作成

Transform プラグインを作成する際に、基本的には拡張を作って、NETObserver を継承するクラスを作成する必要がある。

このクラスのコンストラクタで、本 Transform ファンクションへの入力数と本 Transform ファンクションからの出力数の上限と下限が設定できる。設定しない場合、デファルト値にみなす（デフォルト値として、上限は Integer.MAX_VALUE、下限は 0 である）。

NETObserver を継承するクラスでは、canExecute と execute メソッドを実装する必要がある。

canExecute メソッドは当 Transform が実行できるかどうかチェックするメソッドである。List<Object> preResults はこのメソッドの引数であり、先行要素の計算結果リストで、当ファンクションへの入力引数リストとする。

例えば、乗算の Transform プラグインを作成する時、canExecute では、乗算の条件として、すべての引数は数字であるかどうかチェックする。数字でない引数が存在する場合には false を返す。存在しない場合に true を返す。canExecute メソッドのサンプルコードを表 5.6 に示す。

表 6.4 canExecute メソッドのサンプルコード

```
public boolean canExecute(List<Object> preResults) {

  for(Object preResult:preResults){
    String strPreResult=preResult.toString(); //引数をString型に変換
    if(!strPreResult.matches("-?¥¥d+¥¥.?¥¥d*")) //正規表現で数字であるかどうかチェック
```

```
        return false;
    }

    return true;
}
```

execute メッソドは当 Transform が実行するメッソドである。メソッドのパラメータは同じく List<Object> preResults である。

例えば、乗算の Transform プラグインを作成する時、execute では、引数の乗算を行い、計算結果を返す。execute メソッドのサンプルコードを表 5.7 に示す。

表 6.5 execute メソッドのサンプルコード

```
public Object execute(List<Object> preResults) {

    Double sum=1.0;

    for(Object obj:preResults){
        double x=Double.parseDouble(obj.toString()); //引数をDouble型に変換
        sum=sum*x;
    }

    result=sum.toString();
    return result;
}
```

データの変換作業を行う時、Transform プラグインは前述の変換関数 Element に使われている。

6.4 Transform 機能のテスト

Transform 機能の設計開発が完了した後、開発成果物に対するテストを行った。NET ツール開発の第 1 イテレーションと第 2 イテレーションで、筆者は Transform 機能の単体テストと機能テストを担当した。また、第 1 イテレーションで、筆者は第 1 イテレーション開発成果物の総合テストを担当した。

6.4.1 単体テスト

Eclipse 上の Junit とインストールした EclEmma プラグインを利用して、Transform 部分におけるデータ変換の単体テストを行った。単体テストで、テストケースを 49 件作成した。単体テストで、Date、Time のデータ型変換ができないなどのバグを発見し、バグを修正

して再テストする。単体テスト完了後、単体テストのテストケース、テスト結果、バグ対応が記載されている単体テスト報告書(Transform 機能)を作成した。

6.4.2 機能テスト

IRS から PostgreSQL 用にデータを変換する機能、テーブル間の対応関係の定義を行う(マッピング)機能、および Transform プラグイン化に関する機能の機能テストを行った。機能テストの形式としては、テストケースを作成し、事前にテスト結果を想定する。操作後、実際の結果と想定結果と比較して、合否を判断する。機能テストで、Transform 関数 Element に関連する矢印数の制限問題などのバグを発見し、バグを修正して再テストする。機能テスト完了後、機能テストのテストケース、テスト結果、バグ対応が記載されている機能テスト報告書(Transform 機能)を作成した。

6.4.3 総合テスト(第 1 イテレーション)

NET ツール第 1 イテレーション全般の総合テストを行った。総合テストの段階でバグを発見した場合、チームに報告した。総合テスト完了後、総合テスト報告書を作成した。

6.5 マニュアルの作成

NET ツールの第 1 イテレーションと第 2 イテレーションの納品フェーズで、筆者は一部のマニュアル作成を担当した。

6.5.1 Transform プラグイン作成マニュアル

ユーザが自分で変換方式を定義し、Transform プラグインが作成できるようにするため、Transform プラグイン作成方法が記載されているマニュアルを作った。また、Transform プラグイン作成マニュアルを分かりやすくために、サンプル Transform プラグインを提供した。

6.5.2 一部のユーザズマニュアル

NET ツール第 1 イテレーションの納品フェーズで、筆者は他のメンバーと分担して、一部ユーザズマニュアルを作成した。筆者の担当部分はデータベース接続、テーブル情報表示、Transform 作業定義、ETL 作業編集(Working Window)、およびプロジェクト実行のマニュアルである。(詳細は付録を参照)

第 2 イテレーションでは、修正・追加開発の実績に基づいて、一部のユーザズマニュアルを修正した。

6.5.3 データ変換ルール

第 1 イテレーション終了後、データ変換ルールが顧客に要求された。第 2 イテレーション

では、ユーザズマニュアルの補足として、データ変換ルールを作成した。データ変換ルールでは、NET ツールの Transform 機能が対応しているデータ型について説明した。

第7章　プロジェクトの振り返り

本章では、本研究開発プロジェクト第 1 イテレーションと第 2 イテレーション終了後の振り返り、および筆者担当部分の振り返りを述べる。

7.1　第 1 イテレーションの振り返り

第 1 イテレーション終了後、Keep-Problem-Try 法(以下、KPT 法)を用いて振り返りを行った。第 1 イテレーションに対して、振り返りの結果として、問題点と対策を以下に示す。

まず、見積もりや進捗などの管理が個人裁量になっており、プロジェクト全体で積極的に管理できていなかった。対策として、見積もりや進捗について他人からチェックをされる機会を設け、自身が説明責任を持つルールを定めるという方法を考えた。

次に、Git や Redmine が活用できていないため、資料の共有不足や進捗、バージョン管理が効率にできていなかった。それに対して、プロジェクト開始時に Git や Redmine の使用ルールを明確化する対策を考えた。

そして、コアタイムを設けなかったため、メンバー間で円滑なコミュニケーションがとりにくかった。それに対して、連絡方式についてルールを定め、第 2 イテレーションでコアタイムを導入する対策を考えた。

筆者の担当部分について、Redmine を活用しなかったため、設計資料の共有が不十分で、自分の考え方を他のチームメンバーに伝えなかった。第 2 イテレーションで、チームが定めた Git や Redmine の使用ルールに従って情報を共有し、コアタイムを守り、よりよいコミュニケーションをとる必要がある。

まとめると、第 1 イテレーションで不足していたのはメンバー間のコミュニケーションであると考えられる。第 1 イテレーションの振り返り成果と考えた対策を第 2 イテレーションに適用した。

7.2　第 2 イテレーションの振り返り

第 1 イテレーションと同様に、第 2 イテレーション終了後 KPT 法を用いて振り返りを行った。第 2 イテレーションに対して、振り返りの結果として、問題点と対策を以下に示す。

まず、開発の現状に応じて見直しを行う機会はなかった。それに対して、現状の見直しを行う機会を作り、プロジェクトの進行に合わせてコアタイムを調整する対策を考えた。

次に、設計資料の共有不足があって、設計を見直す機会が少ない。対策として、Redmine のチケット作成ルールの詳細化や設計の見直し機会の増加などを考えた。

また、最終の受け入れテストでバグが発見されていて、マニュアル不備に関する指摘があった。それに対して、マニュアルの検証を含むテストを強化する意見を提出した。

筆者の担当部分について、総合テストで他機能との連携問題に関連するバグが発見された。その原因はチームメンバーとのコミュニケーション不足や第 1 イテレーションの情報共有不

十分である。それに対して、遅滞なく情報を共有し、他メンバーの資料に疑問がある場合早めに聞くのは非常に重要であると考える。

第1イテレーションと比べると、第2イテレーションは全体として順調に進捗しているが、プロジェクト管理上の経験不足はまだ存在している。本研究開発プロジェクトの振り返り結果を将来に活かすことができる。

7.3 筆者担当部分の振り返り

本プロジェクトで、筆者は IRS サーバ管理と Transform 機能の設計開発を担当した。

IRS サーバの管理担当として、IRS の障害が発生する時毎回ただちに復旧したが、ただ IRS を再インストールだけで、効率的に解決できなかった。対策として、システムのログを重視する必要がある。

Transform 機能の設計開発担当として、機能をほとんど完成していたが、他チームメンバーとのコミュニケーション不足のため、他機能との連携問題に関連するバグが発見されることが多かった。それに対して、チームワークをより重視し、コミュニケーション力を高めるべきである。また、テストを強化し、レビューに関する意識を上げることが必要である。

本プロジェクトで、筆者は Transform 機能の設計に工夫した。これを通じて Eclipse プラグインの開発技術やアルゴリズムの設計技術を磨いた。また、チームワークを通じて、プロジェクト管理に関する知識を学んで体験した。この貴重な経験を今後のプロジェクトに活かすことができる。そして、コミュニケーション能力を改善していく必要がある。

第8章　　終わりに

本研究開発プロジェクトで、我々は NEC の要求に基づき、ETL ツールの特長を参考にして、IRS から他 DB やデータウェアハウスへデータを移行する NET ツールを開発した。12 月 3 日に納品を行い、プロジェクトは終了した。

本プロジェクトの準備期間で、筆者は NET ツールの開発環境とする IRS のインストールを行った。NET ツールの開発で、筆者は主に Transform 機能担当として、データ変換、データマッピング機能、および Transform プラグイン化機能の設計・開発・テストを行い、一部のマニュアル作成も担当した。また、IRS サーバの管理担当として、IRS の運用・保守を担った。

本研究開発プロジェクトを通じて、IRS、ETL ツールに関する知識を学んで、Eclipse プラグイン開発とアルゴリズムの設計技術を磨いた。また、プロジェクト管理、品質保証に関する知識を学び、顧客やチームメンバーとコミュニケーションする能力を高めた。プロジェクトの成功にとって、コミュニケーション力は非常に重要であることを体験できた。本プロジェクトの経験を将来に活かすことができる。

謝辞

本研究開発プロジェクトを行うにあたり、貴重な時間を割いてご指導いただいている日本電気株式会社の白石様、佐野様、並木様に心より感謝いたします。

委託元教員である天笠准教授から大変貴重なご助言、ご指導をいただきました。深く感謝いたします。

指導教員である田中教授には、様々なご指摘とご助言をいただきました。大学院から指導してくださった二年間、本当にお世話になりました。深く感謝しています。

また、チームの構成員である斎藤君、成澤君、唐君からも多くの助力と意見をいただきました。ありがとうございます。

最後には、自分が日本に来てから、ずっと支えてくれた両親、すべての友人に心より感謝いたします。

参考文献

[1] 近藤 悟, 宮城 安敏, 金子 雅志, 福元 健, 上田 清志, 多様な検索と効率的な冗長化を実現するためのデータ配置方式に関する一検討, 信学技報, Vol. 111, No. 196, pp.11-16, 2011.

[2] 堀江 光, 浅原 理人, 山田 浩史, 河野 健二, 複数のデータセンタを跨ぐ伸縮性を備えたキーバリューストレージの実現手法, 情報処理学会研究報告, Vol. 2012-OS-122, No. 7, pp.1-7, 2012.

[3] NEC, InfoFrame Relational Store, http://www.nec.co.jp/pfsoft/irs/.

[4] 北川 博之, “データベースシステムにおけるデータ管理,” データベースシステム, pp.4-7, 株式会社昭晃堂, 2007.

[5] 清田 陽司, ビッグデータ時代の情報インフラのあり方を考える RDBMS と分散型コンピューティングシステム, 情報の科学と技術, Vol. 62, No. 11, pp.484-489, 2012.

[6] Ralph Kimball, Joe Caserta, “The Data Warehouse ETL Toolkit”, pp.10-11, Wiley Publishing, Inc., Indianapolis, 2004.

[7] 麻野 洋, 岩井 毅, インテックの BI ソリューション・フレームワークの紹介とその適用事例について, INTEC Technical Journal, 2004 年第 3 号, pp.10-15, 2004.

[8] 清崎 大輔, 大久保 弘崇, 粕谷 英人, 山本 晋一郎, 拡張可能ソフトウェアの動作情報を用いたプラグイン開発支援, ソフトウェア工学研究会報告, ソフトウェア工学研究会報告 2008(29), pp.25-32, 2008.

[9] 祐成 光樹, 田村 稔, ビッグデータの活用に最適なスケールアウト型 新データベース「InfoFrame Relational Store」, NEC 技報, Vol. 65, No. 2, pp.30-33, 2013.

[10] 日本電気株式会社, InfoFrame Relational Store V3.1 移行ガイド, pp.1-2, 2014.

[11] 日本電気株式会社, InfoFrame Relational Store V3.1 ユーザーズガイド, p.13, 2014.

[12] 日本電気株式会社, InfoFrame Relational Store V3.1 データ定義設計ガイド, pp.10-16, 2014.

[13] Eclipse Foundation, Eclipse, https://eclipse.org/.

[14] 竹添 直樹, 志田 隆弘, 奥畑 裕樹, 里見 知宏, 野沢 智也, "GEF," Eclipse 3.4 プラグイン開発 徹底攻略, pp.423-427, 株式会社毎日コミュニケーションズ, 2009.

[15] Lu Jun, Yang Deren, Wang Yong, Techniques for GEF-based Graphical Editor, Value Engineering, Vol. 2011-3, pp.181-182, 2011.

[16] Huang YongWen, Li Guangjian, Review on the Application of ETL in Digital Library, New Technology of Library and Information Service, Vol. 158, pp.1-5, 2007.

[17] 堀田 健太郎, 柴田 朋子, 国分 利直, 新規 DWH /データマート構築時におけるデータ加工テンプレートの再利用性の検討, 電子情報通信学会総合大会講演論文集, 2003 年_情報・システム(1), p.153, 2003.

[18] 日本電気株式会社, InfoFrame Relational Store V3.1 インストレーションガイド, pp.8-9, 2014.

[19] 竹添 直樹, 志田 隆弘, 奥畑 裕樹, 里見 知宏, 野沢智也, “拡張ポイントの定義,” Eclipse 3.4 プラグイン開発 徹底攻略, p.359, 株式会社毎日コミュニケーションズ, 2009.

付録　成果物

- 第 1 イテレーション機能一覧
- 第 2 イテレーション機能一覧
- Transform 機能のクラス図
- 単体テストテストケース一覧 S1(Transform)
- 単体テストテストケース一覧 S2(Transform)
- 機能テスト S1
- 機能テスト S2
- 総合テスト報告書 S1
- ユーザズマニュアル
- データ型の変換ルール
- Transform プラグイン作成マニュアル

第１イテレーション機能一覧

第１イテレーション機能一覧

機能番号	機能名
1	IRS に接続する
2	IRS からテーブル定義データの取得
3	IRS から取得したテーブル定義データの表示
4	データを抽出するテーブルの指定
5	IRS からデータを抽出する機能
6	抽出する際に抽出条件(フィルター)を指定する機能
7	PostgreSQL との接続
8	PostgreSQL からテーブル定義データの取得
9	PostgreSQL から取得したテーブル定義データの表示
10	テーブル作成
11	データを挿入するテーブルを指定する機能
12	データ挿入機能
13	IRS から PostgreSQL 用にデータを変換する
14	テーブル間の対応関係の定義を行う(マッピング)機能
15	アイコン(データ移行元やデータ移行先、条件追加)を表示する機能
16	アイコン(データ移行元やデータ移行先、条件追加)同士を線で結ぶ機能

機能番号	1
機能名	IRS に接続する
実装理由	IRS からテーブル定義を入手するため。データを抽出するため。
要検討項目	● IRS に接続するために必要な情報 ● IRS に接続するための方法 ● GUI の詳細
実装方針	1. ユーザから入力された IRS 接続に必要な情報を受け取る

	2. 指定された IRS に接続を行う 3. ログインの可否を表示する
担当者	唐

調査内容

● **IRS に接続するために必要な情報**

ストレージサーバの URL(必須)
トランザクションサーバの URL(必須)
ストレージサーバ側のフィルタリングの有無(任意)
フィルタリングのクラスファイルを生成するディレクトリ(任意)
ストレージサーバの状態のチェックと動作フラグ(必須)
RSMasterDB の接続 URL(ストレージサーバの状態のチェックと動作フラグに’no’以外を指定した場合)
RSMasterDB の接続ユーザ名(通常はこのパラメータを指定する必要はない)
RSMasterDB の接続パスワード(通常はこのパラメータを指定する必要はない)

● **IRS に接続するための方法（マニュアルより引用）**

まず、 Configuration オブジェクトを作成する。次に作成した Configuration オブジェクトを org.apache.hadoop.mapreduce.Job のコンストラクタに指定して、Hadoop ジョブを生成する。Hadoop ジョブで対象とするデータベース名とテーブル名を指定する。Hadoop 連携機能では、対象とするデータベース名とテーブル名の指定は必須である。
下記に例を示す。この例では、"auction"データベースの"user"テーブルを対象にした Hadoop ジョブを作成している。

```
Configuration conf = new Configurator().storage(props)
.database("auction")
.table("user")
.build();
Job job = new Job(conf);
```

● **GUI の詳細**

GUI からの入力
ストレージサーバの URL(必須)
トランザクションサーバの URL(必須)
ストレージサーバ側のフィルタリングの有無
フィルタリングのクラスファイルを生成するディレクトリ
(ストレージサーバの状態のチェックと動作)
(RSMasterDB の接続 URL)

- 追加調査
  1. 実際に動かしてみる

機能番号	2
機能名	IRS からテーブル定義データの取得
実装理由	指定された IRS 内に格納されているテーブル情報を GUI 上に表示するため
要検討項目	● 入手するテーブル定義データの内容 ● テーブル定義データの入手方法
実装方針	1. 起動時点での IRS のテーブル情報を IRS から 1 度だけ取得する 2. 一時ファイルを作るなどして一時的に保存する
担当者	唐

調査内容

- **入手するテーブル定義データの内容**

  INFORMATION_SCHEMA についてさらなる調査を行う必要がある。
- 追加調査

  2. 実際に入手してみる必要がある

機能番号	3
機能名	IRS から取得したテーブル定義データの表示
実装理由	GUI 上に指定された IRS 内に格納されているデータを表示し、ユーザの操作難度を下げるため
要検討項目	● 表示するテーブル定義データの内容 ● GUI の詳細
実装方針	1. 表示用に保存されているデータを読み込む 2. テーブル情報を GUI 上に表示する
担当者	成澤
備考	**表示されるテーブル情報は「カラム名」と「データ型」のみである。**

	「長さ」や「NOTNULL」、サンプルデータなどは表示されない仕様である。

機能番号	4
機能名	データを抽出するテーブルの指定
実装理由	視覚的にどのテーブルに移行するかを指定できるようにするため
要検討項目	● GUI の詳細
実装方針	1. IRS のテーブル一覧を表示する 2. 表示されたテーブルの中から、データを抽出するテーブルを指定する 3. 指定された情報を保存する
担当者	成澤

機能番号	5
機能名	IRS からデータを抽出する機能
実装理由	ETL ツールとして必要不可欠な機能のため
要検討項目	● IRS からデータを抽出する際の HadoopAPI の詳細
実装方針	1. データを抽出するテーブル、条件を受け取る 2. 条件に基づいてデータを抽出する 3. 抽出したデータを HDFS に保存する
担当者	唐
備考	IRS から抽出したデータを一度 HDFS に保存するため、’/tmp/irs2db_<Centos のユーザ名>’というフォルダを作成する。

調査内容

● **IRS からデータを抽出する際の HadoopAPI の詳細(マニュアルより引用)**

Configuration オブジェクトを作成する。次に作成した Configuration オブジェクトを org.apache.hadoop.mapreduce.Job のコンストラクタに指定して、Hadoop ジョブを生成する。

Hadoop ジョブで対象とするデータベース名とテーブル名を指定する。Hadoop 連携機能では、対象とするデータベース名とテーブル名の指定は必須である。

下記に例を示す。この例では、"auction"データベースの"user"テーブルを対象にした Hadoop ジョブを作成している。

```
Configuration conf = new Configurator().storage(props)
.database("auction")
.table("user")
```

```
.build();
Job job = new Job(conf);
```

InputFormat クラスの設定
Hadoop ジョブで利用する InputFormat クラスとして、PartiqleInputFormat クラスを指定する。
下記の例のように Job オブジェクトに対し、setInputFormatClass()メソッドで指定する。

```
job.setInputFormatClass(PartiqleInputFormat.class);
```

Map クラスの定義
Map クラスの入力形式は、キーが KeyWritable 型、バリューが TupleWritable 型のオブジェクトになる。下記に例を示す。

```
class Map extends Mapper<KeyWritable, TupleWritable, T1, T2> {
@Override
protected void map
(KeyWritable key, TupleWritable value, Context context)
throws IOException, InterruptedException {
...(省略)...
}
}
```

行オブジェクト(Tuple)の取得

```
Tuple tuple = value.getValue();
```

列オブジェクト(AtomicValue)の取得

```
AtomicValue atomic1 = tuple.get(0);
```

- 追加調査

3. 実際に動かしてみる

機能番号	6
機能名	抽出する際に抽出条件(フィルター)を指定する機能
実装理由	射影や選択を行えるようにするため

要検討項目	● 実装する「条件」の項目 ➢ IN ➢ 比較演算子(<, >, =, ≦, ≧, ≠) ● GUI の詳細
実装方針	1. 条件を扱えるような GUI を作成する 2. ユーザが入力した条件を受けとる 3. 条件をデータ抽出機能に渡す
担当者	唐
備考	第 1 イテレーションでは本機能は実装しない

調査内容

フィルターに対応する抽出条件は、比較述語及び IN 述語である。
一つのカラムに対して、指定できる抽出条件は 1 つである。

比較述語

演算子	説明
<	x< y は、x が y 未満
>	x> y は、x が y を超える
<=	x<= y は、x が y 以下
>=	x>= y は、x が y 以上
=	x= y は、x と y が等しい
!=	x!= y は、x と y が等しくない

IN 述語

Expression IN (expr1[, expr2]...)の式で、左辺の式 expression の値が右辺の式 expr の値のいずれかと等しい場合、IN 述語の結果は"真" になる。

複数のカラムにそれぞれの抽出条件が定義される場合、AND 演算子で結びつく。

機能番号	7
機能名	PostgreSQL との接続
実装理由	テーブル定義の取得。データ挿入のため。
要検討項目	● 接続に必要な情報 ● GUI の詳細
実装方針	1. PostgreSQL 接続に必要な情報をユーザから入力される

	2. 指定された PostgreSQL にログインを行う 3. ログインの可否を表示する
担当者	斎藤

調査内容

● 接続に必要な情報

・データベースへの接続

JDBC を使用する場合、データベースは URL 表される。PostgreSQL は次の形式のいずれか。

jdbc:postgresql:database

jdbc:postgresql://host/database

jdbc:postgresql://host:port/database

host：サーバのホスト名。デフォルトは localhost である。IPv6 アドレスを指定するためには、以下のように host を角括弧で括る必要がある。

jdbc:postgresql://[::1]:5740/accounting

port：PostgreSQL のポート番号。デフォルト (5432)。

database：データベース名。

username：ログインするユーザネーム。

pasqqord:ログインするユーザに対応したパスワード。

以上の情報で指定された PostgreSQL のデータベースに入力されたユーザとパスワードを用いて接続を試みる。

```
Connection db = DriverManager.getConnection(url, username, password);
```

＊注意：ログインするユーザカウントは予め PostgreSLQ に作成しておく必要がある。

● GUI の詳細

GUI からの入力

- ・サーバのホスト名(必須)
- ・サーバのポート(必須)(デフォルト：5432)
- ・データベース名(必須)
- ・ユーザネーム(必須)
- ・パスワード(必須)

GUI への出力

- ・ログインの可否

*実際に JDBC から PostgreSQL に接続し、実験を行った。

機能番号	8
機能名	PostgreSQL からテーブル定義データの取得
実装理由	GUI 上に指定された PostgreSQL 内に格納されているテーブル情報を表示するため
要検討項目	● 入手するテーブル定義データの内容
実装方針	1. GUI 上に表示するためなので、接続時点での PostgreSQL のテーブル情報を１度だけ取得する 2. テーブル情報を抜き出す
担当者	斎藤

調査内容

● テーブル定義データの入手方法
コネクションのメタデータ入手関数を用いることで入手可能

● 入手できるテーブル定義データの内容
- テーブル名
- カラム名
- カラム型種類
- カラム型の桁数
- NULL 値を許容するかどうか

*実際に JDBC で PostgreSQL からテーブル定義データを取得する実験を行った。

機能番号	9
機能名	PostgreSQL から取得したテーブル定義データの表示
実装理由	GUI 上に指定された PostgreSQL 内に格納されているデータを表示し、ユーザの操作難度を下げるため
要検討項目	● 表示するテーブル定義データの内容 ● GUI の詳細
実装方針	1. 表示用に格納されているデータを受け取る 2. GUI 上にテーブル情報を表示する(表示するテーブル情報は要検討)

担当者	斎藤

調査内容

- 表示するテーブル定義データの内容
  - プライマリキー
  - テーブル名
  - カラム名
  - カラム型種類
  - カラム型の桁数
  - NULL 値を許容するかどうか

- GUI の詳細
  データ移行作業時のどのタイミングでテーブル定義データが表示される必要があるのか、GUI 担当者と検討する。

機能番号	10
機能名	データを挿入するテーブルを作成する
実装理由	テーブル作成を PostgreSQL を直接操作せずに行えるようにするため
要検討項目	● テーブル作成に必要な情報 ● GUI の詳細 ● テーブル作成のタイミング
実装方針	1. テーブル作成 GUI を表示する 2. 入力された情報に基づいてテーブル作成の準備を行う 3. 実際にテーブルを作成する 4. 作成したテーブル情報をデータ挿入機能に渡す
担当者	斎藤

調査内容

- PostgreSQL テーブル作成するのに必要な情報
  以下、データベースに接続した後を前提とする。
  コネクション con に対して
  `Statement stmt = con.createStatement ();`
  を実行し、
  `stmt.executeQuery(sql);`
  で sql を実行できる
  SQL で TABLE CREATE に必要な情報

(http://www.postgresql.jp/document/pg653doc/j/user/sql-createtable.htm)

テーブル名

カラム名

カラム型

主キー

(以下、オプションもある)

例：EMP という名前のテーブルを作成する。

```
"create table EMPL ( "+
        "EID NUMERIC(6) primary key, " +
        "ENAME VARCHAR(20), " +
        "MANAGERID NUMERIC(6), " +
        "HIREDATE DATE, " +
        "SALARY INTEGER )"
```

- GUI の詳細

  テーブル作成時にユーザが入力する情報

  - ✧ テーブル名(必須)
  - ✧ カラム名(必須)
  - ✧ カラム型(必須)
  - ✧ 主キー指定(任意)
  - ✧ NOT NULL 制約、UNIQUE 制約、DEFAULT 初期値設定（任意）

- テーブル作成のタイミング

  データ移行開始ボタンを押したら作成する

- テーブル削除について：エラー発生時に、データ移行が中断した場合、作成したテーブルをどうするか。

  データ移行が中断した際に、{テーブル削除、テーブルを空にする、なにもしない（テーブルはそのまま放置)} の三択をユーザに聞く　を提案する。

  テーブルを空にする方法

  テーブルごと削除して、同じカラムを持ったテーブルを作成しなおしたほうが早い

  http://homepage2.nifty.com/sak/w_sak3/doc/sysbrd/psql_k05.htm

- テーブル作成時の注意点

- テーブル名が重複してはならない

テーブル作成時に指定できるカラム型の一覧は以下の通りである

カラム型名	カラム方の長さを指定できるか	説明
integer	×	4 バイト符号付き整数
serial	×	自動増分 4 バイト整数
bigint	×	8 バイト符号付き整数
bigserial	×	自動増分 8 バイト整数
decimal	○*	精度の選択可能な高精度数値
numeric	○*	精度の選択可能な高精度数値
real	×	単精度浮動小数点（4 バイト）
double precision	×	倍精度浮動小数点（8 バイト）
varchar	○	可変長文字列
char	○	固定長文字列
text	×	可変長文字列
date	○	暦の日付（年月日）
time	○	時刻（時間帯なし）
timetz	○	時間帯付き時刻
timestamp	○	日付と時刻（時間帯なし）
timestamptz	○	時間帯付き日付と時刻
boolean	×	論理（ブール）値（真/偽）
bytea	×	バイナリデータ（"バイトの配列（byte array）"）

*第 1 イテレーション段階では対応していないので、varchar とともに対応する。
decimal,numeric はデータ型の長さの入力ボックスが 1 つなので、「整数桁数,小数点桁数」 と入力する。

機能番号	11
機能名	データを挿入するテーブルを指定する機能
実装理由	GUI を用いてデータ挿入するテーブルを指定できるようにし、ユーザの負担を軽減する
要検討項目	● GUI の詳細
実装方針	1. 接続した PostgreSQL のテーブルの中から、データを抽出するテーブルを指定する
担当者	成澤

機能番号	12
機能名	データ挿入機能
実装理由	効率的なデータ挿入を行うため
要検討項目	● データ挿入に必要な情報の調査 ● JDBC の調査
実装方針	1. 挿入するデータと挿入先情報を受け取る。 2. 指定された挿入先にデータを挿入する 3. 挿入の可否を表示する
担当者	斎藤

調査内容

JDBC の COPY 機能を使用する。

機能番号	13
機能名	IRS から PostgreSQL 用にデータを変換する
実装理由	IRS から抽出したデータがそのまま PostgreSQL に挿入できるとは限らないので、挿入できる形に変換する
要検討項目	● IRS から抽出したデータの詳細 ● PostgreSQL に挿入できるデータの詳細
実装方針	1. IRS から抽出したデータから PostgreSQL に挿入できるデータに変換す

	る
担当者	コ

調査内容

IRS と PostgreSQL のデータ型対応関係は以下の表の通りである。

IRS データ型	PostgreSQL データ型	内部動作 Java データ型
INTEGER, INT	int4, serial	java.lang.Integer
BIGINT	int8	java.lang.Long
DECIMAL	decimal	java.math.BigDecimal
FLOAT	float4	java.lang.Float
DOUBLE	float8	java.lang.Double
CHAR, VARCHAR, TEXT	char, varchar, text	java.lang.String
DATE	date	java.sql.Date
TIME	time	java.sql.Time
TIMESTAMP, DATETIME	timestamp	java.sql.Timestamp
BOOLEAN, BOOL	boolean	java.lang.Boolean
BINARY, VARBINARY	bytea	java.lang.Byte の配列 byte[]

データ変換ができる場合、表 1 の通り対応しているデータ型間の変換ができる。データ変換ができない場合、エラーメッセージのポップアップが表示される。

例えば、IRS から INTEGER 型のデータを int4 型の PostgreSQL カラムに挿入する場合、INTEGER から int4 の変換ができる。

機能番号	14
機能名	テーブル間の対応関係の定義を行う(マッピング)機能
実装理由	GUI を用いてテーブル間の対応関係の定義を行えるようにし、ユーザの支援を行うため
要検討項目	● IRS から抽出したデータの詳細 ● PostgreSQL に挿入できるデータの詳細 ● GUI の詳細
実装方針	1. IRS から抽出したデータから PostgreSQL に挿入できるデータに変換する
担当者	コ

調査内容

- GUI の詳細

  GUI からの入力

  ・移動元のテーブル名、カラム名

  ・移動先のテーブル名、カラム名

  GUI への出力

  ・対応関係表示（線）

機能番号	15
機能名	アイコン(データ移行元やデータ移行先、条件追加)を表示する機能
実装理由	アイコンを作業ウィンドウに表示し、アイコンをクリックすることでアイコンの詳細情報にアクセスしやすくするため
要検討項目	● アイコンの詳細情報に表示する項目 ● GUI の詳細
実装方針	1. アイコンを作業ウィンドウに表示する 2. アイコンがクリックされたら詳細情報をウィンドウに表示する
担当者	コ

調査内容

- アイコンの詳細情報に表示する項目

  データベース（input と output）：データベース名

  テーブル：テーブル名、カラム名、データ型、長さ、Null は許可するかどうか

  mapping：マッピングの操作ウィンドウ

  between：カラム名、引数

in：カラム名、引数

>, >=, <, <=, =, !=：カラム名、引数

● GUI の詳細

GUI からの入力

・アイコンの種類

・ドラッグ先の位置（座標）

・項目との連結関係

GUI への出力

・大きいアイコン、アイコンについて簡単な説明

・連結線

機能番号	17
機能名	アイコン(データ移行元やデータ移行先、条件追加)同士を線で結ぶ機能
実装理由	アイコンを線で結ぶことで、データの抽出から挿入まで一連の流れとして表示するため。
要検討項目	● GUI の詳細
実装方針	データの抽出や挿入の際に進捗を示すデータを受け取り、進捗を表示する
担当者	成澤

第 2 イテレーション機能一覧

第２イテレーション機能一覧

機能番号	機能名
1	PostgreSQL のテーブル作成時に特定のデータ型の長さを指定できる機能
2	PostgreSQL の COPY コマンドを使用したデータの挿入機能
3	ETL 作業が実行中であることを示す機能
4	IRS からのテーブル情報取得状況の表示機能
5	作成テーブルの表示更新機能
6	IRS のテーブル表示の修正
7	Transform のプラグイン化
8	Hadoop API を用いた条件抽出
9	ユーザが入力した抽出条件を扱う機能
10	ETL 作業の中止機能

機能番号	1
機能名	PostgreSQL のテーブル作成時に特定のデータ型の長さを指定できる機能
機能概要	PostgreSQL にテーブルを作成する時に、特定のデータ型のデータ型の長さを指定できる
実現理由	PostgreSQL の varchar 型はデータ型の長さを指定することができる。適切なデータ型の長さの指定を行うことでデータ格納の効率が上がるため、NET ツールでも varchar 型のデータ型の長さを指定できるようにする。 decimal や numeric もデータ型の長さを指定できるようにする。
他書類への影響	あり デーブル作成時に指定できるカラム型の一覧を資料に追加する
担当者	斎藤

調査結果

- **現在の NET ツールのコーディング**
  データ型を用いて条件分岐を行い、データ型の長さを指定できるデータ型の場合にデータ型の長さを設定した SQL 文を作成している
  decimal や numeric でのデータ型の長さを指定できるようにする
- **実現可能性**
  データ型の長さを用いて条件分岐を行うように変更する。データ型の長さが””（空白文字列）ではなかった場合に、長さを指定した SQL 文を作成するように、コーディング内容を変化させる。

備考

テーブル作成時に指定できるカラム型の一覧は以下の通りである

カラム型名	カラム方の長さを指定できるか	説明
integer	×	4 バイト符号付き整数
serial	×	自動増分 4 バイト整数
bigint	×	8 バイト符号付き整数
bigserial	×	自動増分 8 バイト整数
decimal	○*	精度の選択可能な高精度数値
numeric	○*	精度の選択可能な高精度数値
real	×	単精度浮動小数点（4 バイト）
double precision	×	倍精度浮動小数点（8 バイト）
varchar	○	可変長文字列
char	○	固定長文字列
text	×	可変長文字列
date	○	暦の日付（年月日）
time	○	時刻（時間帯なし）
timetz	○	時間帯付き時刻
timestamp	○	日付と時刻（時間帯なし）
timestamptz	○	時間帯付き日付と時刻
boolean	×	論理（ブール）値（真/偽）
bytea	×	バイナリデータ（"バイトの配列（byte array）"）

*第 1 イテレーション段階では対応していないので、varchar とともに対応する。
decimal,numeric はデータ型の長さの入力ボックスが 1 つなので、「整数桁数,小数点桁数」 と入力してもらう。

PostgreSQL　8.4 のデータ型の詳細については以下のドキュメントを参照ください
https://www.postgresql.jp/document/8.4/html/datatype.html

機能番号	2
機能名	PostgreSQL の COPY コマンドを使用したデータの挿入機能

機能概要	ETL 作業の際に PostgreSQL の COPY コマンドを使用してデータの挿入を行う機能(PreparedStatement を使用している部分と取替、PreparedStatement のクラスは残しておく)
実現理由	PostgreSQL へのデータの挿入は COPY コマンドを使用した方が早いとの指摘を受けて、PreparedStatement と比較した結果、COPY の方が早く動作することが判明したため。
他書類への影響	あり PreparedStatement の代わりに COPY を使用してデータの挿入を行っている旨の説明に各種書類を訂正する。
担当者	斎藤

調査結果

- **JDBC での COPY コマンドの使用方法**

  JDBC の CopyManager,CopyIn クラスを使用することで COPY コマンドを用いてデータを挿入することができる。

  データベースに接続した後(Connection が成功した後)はテーブル名カラム名が必要であるが、現在の PreparedStatement を用いた実装でも使用しているため、追加の情報は必要ない。

- **実現可能性**

  コーディング例を示す。

```
String values = "San Francisco insert" + rnd.nextInt(9999)
            + ", 43, 57, 0 ,'1994-11-29' ¥n";
Connection con = DriverManager.getConnection(url, user, password);
Statement st = null;
st = con.createStatement();
  CopyManager cm = new CopyManager((BaseConnection) con);
  CopyIn cpIN=null;
cpIN=cm.copyIn( "COPY "+tablename+"(city,temp_lo,temp_hi,prcp,date) FROM STDIN
WITH CSV" );
cpIN.writeToCopy(values.getBytes(),0,values.getBytes().length);
  cpIN.endCopy();
```

  上記のように作成した CopyIn クラスに writeToCopy メソッドで挿入するデータを登録することで本機能は実現可能である。

  複数行データを挿入する場合は、挿入するデータの末尾に改行をいれる必要がある。

備考

実装終了後、COPY を用いたデータの挿入速度についての調査実験を行う必要がある

機能番号	3
機能名	ETL 作業が実行中であることを示す機能
機能概要	ETL 作業を開始した後、GUI が固まらないようにする
実現理由	第 1 イテレーションでは、ETL 作業の実行中では、ETL 作業が終了するか、エラーで終了するまで GUI が固まっていた。 これでは、作業が進んでいるのかわからないので作業中であることを示すように GUI を変更する。
他書類への影響	あり GUI が固まる仕様から、作業中であることを示している。　という仕様に記述を変更する
担当者	斎藤

調査結果

- **現 EclipseGUI での進捗表示の方法**

  org.eclipse.core.runtime.jobs.Job　を用いることで表示することが可能。

  Job を継承したクラスを作成し、その中で ETL 作業を行う。

  終了ダイアログの表示は

```
this.showInformation("Progress Sample", "完了");
while(!WindowFlag){
        wait(1000);
}
volatile private Boolean windowFlag = false;
        public void showInformation(final String title, final String message) {
                Display.getDefault().syncExec(new Runnable() {
```

```
                @Override
                public void run() {
MessageDialog dialog = new MessageDialog(null, title, null,
                message, MessageDialog.ERROR, new String[] { "OK",
            }, 0);
dialog.create();
while(dialog.open()!=MessageDialog.OK){
                try {
                        wait(1000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
            }
windowFlag=true;
                }
            });
            }
```

のようにして行う

- **実現可能性**

  上記のように　org.eclipse.core.runtime.jobs.Job　を用いれば可能である。

備考

イメージ画面を以下にのせる

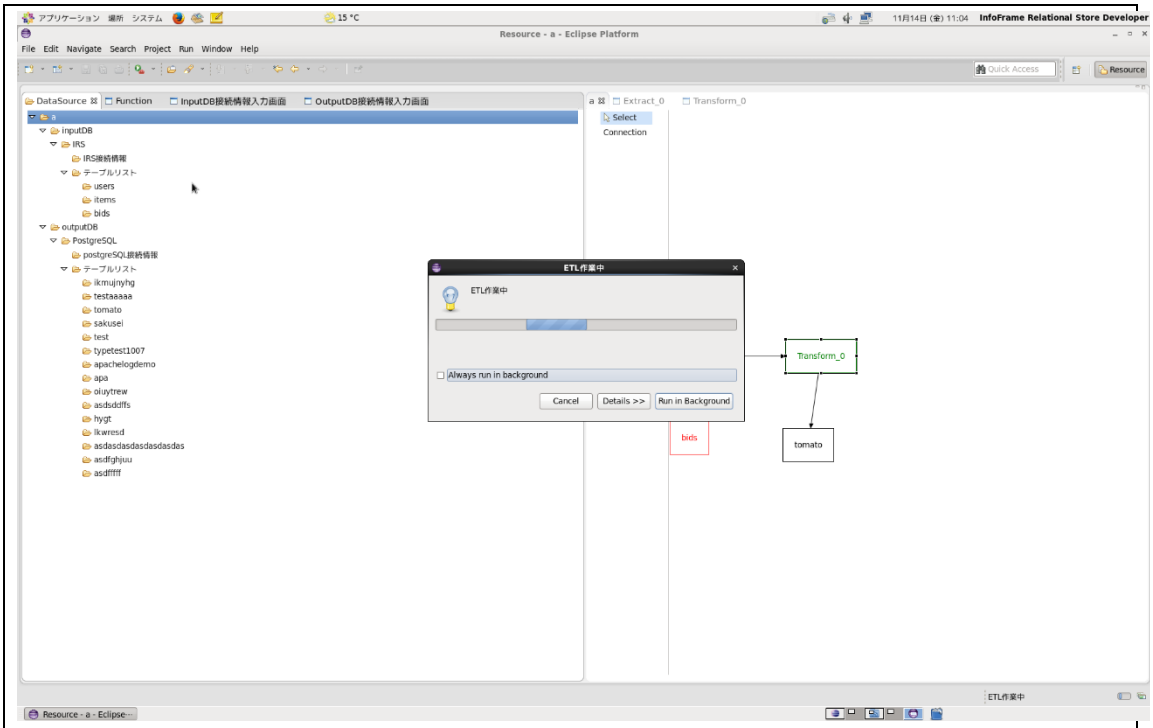


図 1

このように ETL 作業中ポップアップが表示され、中央のバーが左右に振れることで、作業中であることを示す。

作業が完了した場合の画面を以下に示す

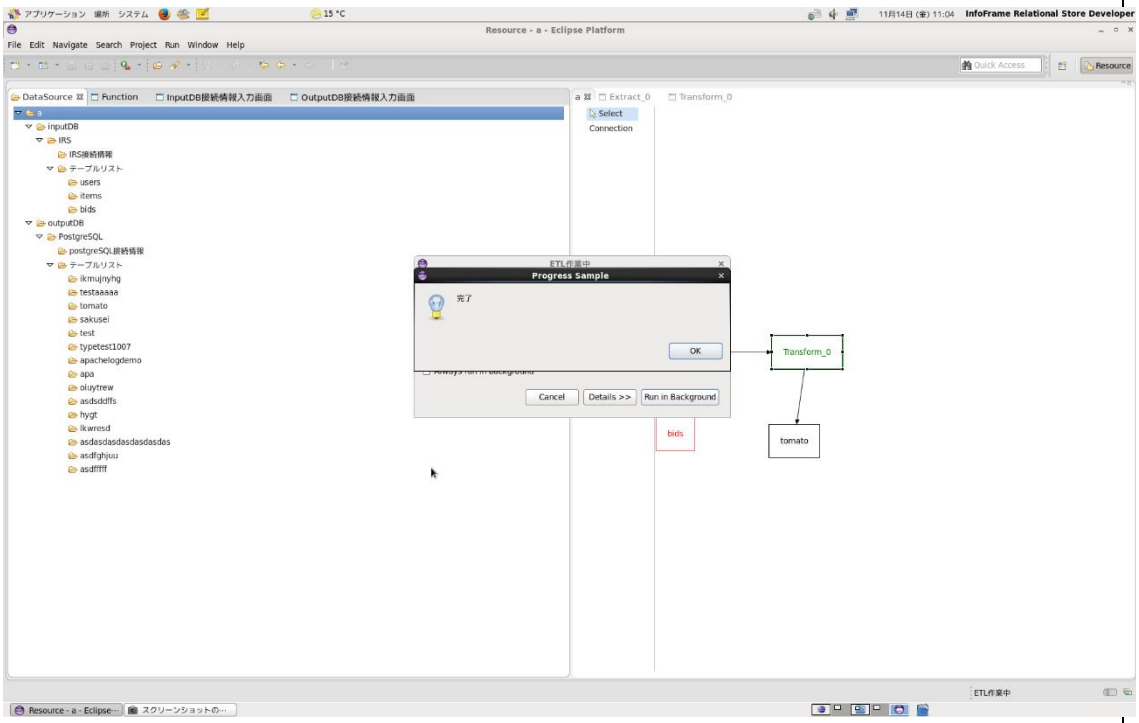


図 2

作業が完了すると、ETL 作業中ポップアップの上に完了ポップアップが表示される。

OK ボタンを押すと ETL 作業中ポップアップと共に完了ポップアップが消える。

機能番号	4
機能名	IRS からのテーブル情報取得状況の表示機能
機能概要	IRS に接続した際の結果を詳細に表示する 1. IRS に接続することができない場合 2. IRS に接続できたが、取得したデータベース情報が 0 個である場合 3. IRS、データベースに接続でき、取得したテーブル情報が 0 個である場合 4. 1 つ以上のテーブル情報を取得できた場合 の 4 段階で結果の表示を行う
実現理由	第 1 イテレーションの受け入れテストで頂いた 「IRS にデータベースがひとつも定義されていないとき、「DB への接続が失敗しました」とエラーが表示され、先に進むことができない。実用上問題はないが、不親切である。」 に対応するため。
他書類への影響	あり メッセージウィンドウの表示とテーブル情報の取得状況の対応付けをマニュアルに記述する。
担当者	斎藤

調査結果

- **現在の NET ツールのコーディング**
  IRS に接続した際の結果を詳細に区別せずにメッセージを表示している。
- **実現可能性**
  以下のように接続の状況を区別して表示する
  - IRS に接続することができない(SQLExeption が返ってきた時)
  - IRS に接続できたが、取得したデータベース情報が 0 個である場合(データベースの HashMap が空の時)
  - IRS、データベースに接続でき、取得したテーブル情報が 0 個である場合（テーブルの HashMap が空の時)
  - 1 つ以上のテーブル情報を取得できた場合(正常に情報が取得できた場合)

備考

なし

機能番号	5
機能名	作成テーブルの表示更新機能
機能概要	・テーブル情報機能が動作しないバグを修正する ・サンプルデータが存在しない場合は空白を表示する
実現理由	・バグにより作成したテーブルに対してテーブル情報機能を利用できないため、これを修正する。 ・修正箇所は少なく、例外処理を埋め込むことで修正が可能である。
他書類への影響	あり ・サンプルデータが存在しない場合の仕様は空白を表示する
担当者	成澤

調査結果

● バグの原因 バグの原因は例外処理の抜けであった。 サンプルデータがないテーブルに対しての読み込みを行い、エラーが発生していた。 例外処理を付け加え、サンプルデータがない場合の表示仕様を決定する必要がある。 ● 実現可能性 上記から、変更範囲が狭く特定されているため実現は可能である。

備考

特になし

機能番号	6
機能名	IRS のテーブル表示の修正
機能概要	IRS のテーブル表示を「テーブル名」から「DB 名.テーブル名」とする
実現理由	・DB についての表記がないため、同名のテーブルを見分けられない。 ・変更箇所は少なく、引数の変更だけで修正が可能である。 ・副作用として全ての IRS のテーブル名の表示が「DB 名.テーブル名」となる。
他書類への影響	あり ・IRS のテーブル名の仕様について変更が必要である。

担当者	成澤

調査結果

- IRS からの接続時の振る舞い

現在、IRS からテーブルデータを取得し、テーブル名を引数として作成している。そのため、引数を「DB 名.テーブル名」とすることで変更が可能である。

- 影響度

影響度は低く、当該メソッドの引数変更だけで修正が可能である。
変更の副作用として、カラムの対応関係定義画面のテーブル名が「DB 名.テーブル名」となる。
理由としては、IRS と PostgreSQL のクラスは分けられており、引数の変更が PostgreSQL や他メソッドに影響を及ぼすことはないからである。

- 実現可能性

上記から、変更範囲が狭く特定されているため実現は可能である。

備考

特になし

機能番号	7
機能名	Transform のプラグイン化
機能概要	Transform 機能をプラグイン化する。プラグイン化することで、Transform の種類を追加しても、既存の NET ツールを変更する必要がなくなる。
実現理由	Eclipse プラグインの拡張ポイントを利用して本機能の実現ができる。 既存 NET ツールへの影響が小さいことが判明した。 工数は第２イテレーションで取り組める範囲である。
他書類への影響	あり Transform 機能が別のプラグインとして存在することを各種書類に記述する
担当者	コ

調査結果

- **Eclipse プラグイン間の通信**

  プラグイン A は既存プラグイン(NET ツール)で、プラグイン B は拡張プラグイン(Transform 機能)であると過程した場合、プラグイン A はプラグイン B に引数を渡し、B の関数で実行されて、結果を取得することができる。
  以下は実装手順：

(1)プラグイン A でプラグイン B 向けのインターフェース、アブストラクトクラスを作成する。

(2)プラグイン A の Extension Point で拡張ポイントを作る。拡張ポイントのスキーマで新規 Element を追加し、Element の下に Type が Java の属性を作成する。この属性の Implements、Extends で(1)で作成したインターフェースを選択する。

(3)プラグイン A の Runtime でインターフェースの所在パッケージを追加する。

(4)プラグイン B で A のインターフェースを実装するクラスを書く。

(5)プラグイン B の Extension で(2)で作成した拡張ポイントを追加する。下の属性の引数として、(4)で作成したクラスを選択する。以上で A に B を使わせることができる。

(6)プラグイン A が B を呼び出す時に、まず A の拡張ポイントに関係する拡張をすべて取得する。取得した拡張から拡張名で B の所属拡張を探して、B のクラスを使う。

- **E,L,GUI とのインターフェースおよび既存コードへの影響**

  NET ツールの拡張ポイントを作成して、Transform 向けのインターフェース,
  またはアブストラクトクラスを提供する。既存の画面設計を少し変更すればよい。
  E と L への影響はない。

  変更が必要なもの：

  Transform の function tree

  Transform Window に使われている model と controller(一部項目の統合)

  Transform 用 Element の構造(加算、乗算などのエレメントの統合)

- **実現可能性**

  実現可能。

  理由：

  (1)技術点から見ると機能の実現ができる。

  (2)既存 NET ツールへの影響が小さいことが判明した。

  (3)工数は第二イテレーションで取り組める範囲である。

- **備考**

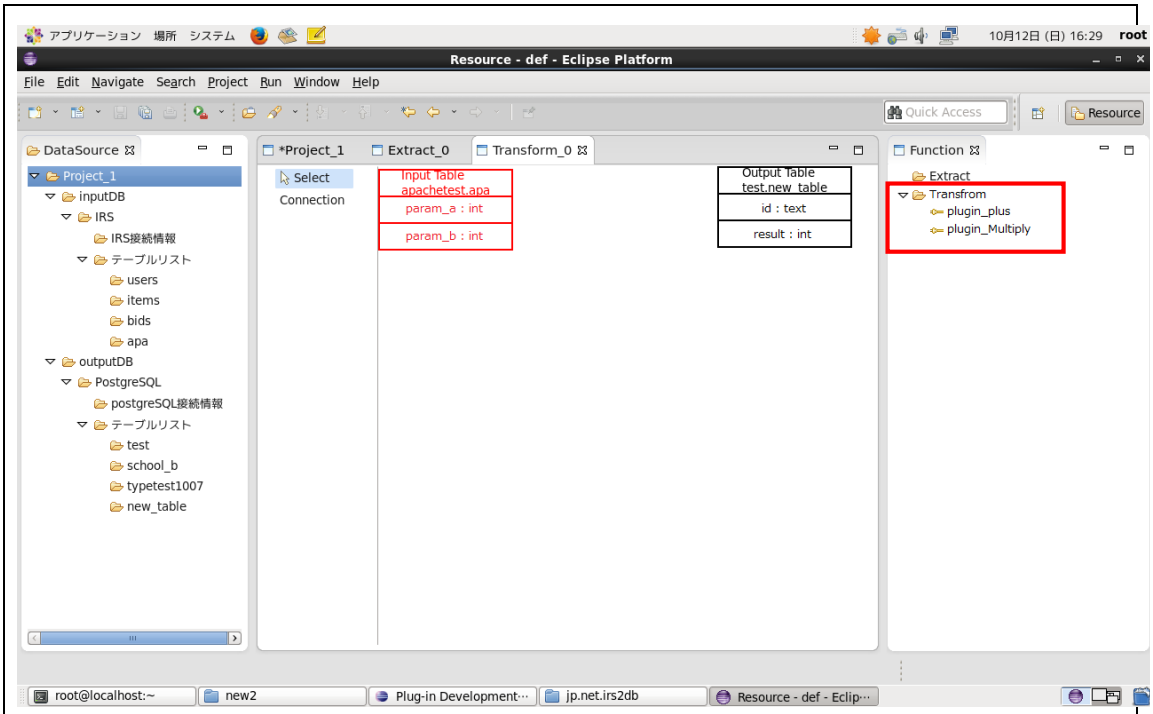
  プラグイン作成マニュアル　を作成する。

備考

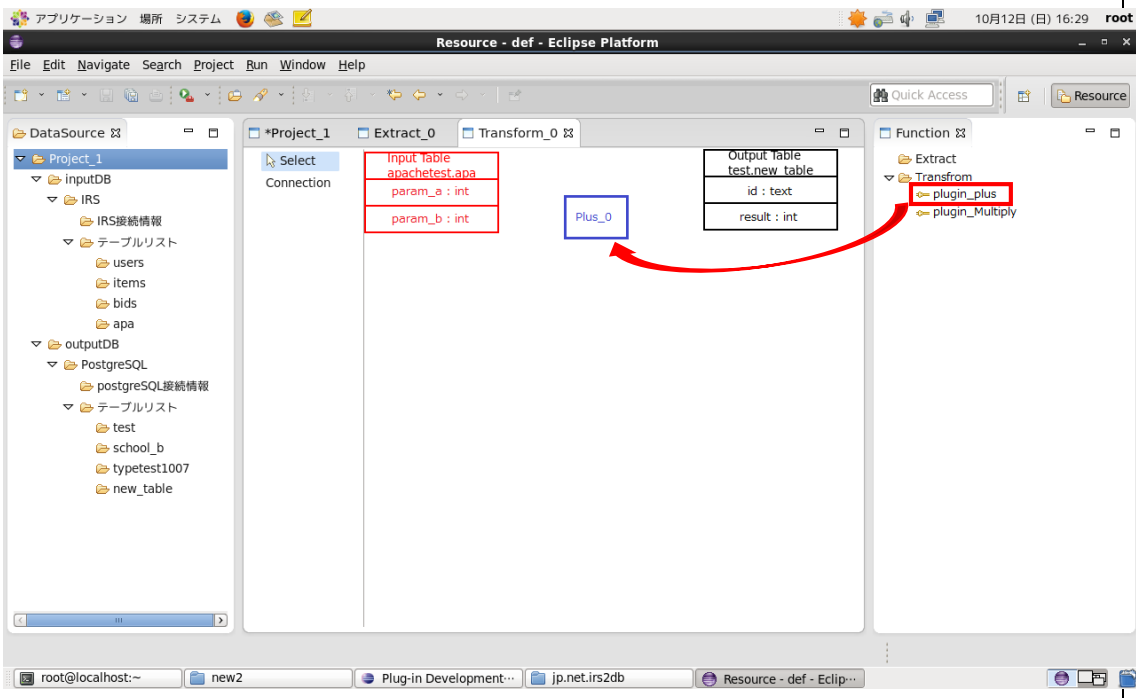
プラグイン作成マニュアルを作成する。

画面イメージ

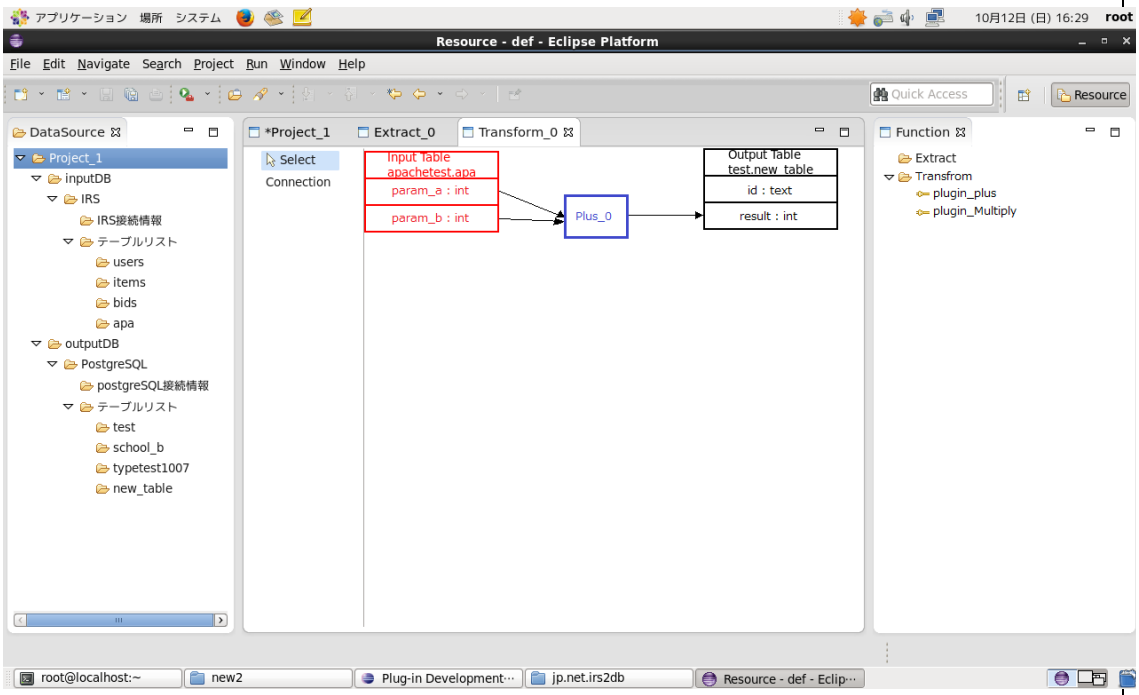
1. NET ツールが起動する時、プラグインを自動的に読み込んで、Function ビューにある Transform のツリーを生成する。



2. 使用するプラグインを選択し、Transform 編集ウィンドウにドラッグアンドドロップする。



3.　入力カラム、出力カラムとの対応関係を指定する。



機能番号	8
機能名	Hadoop API を用いた条件抽出
機能概要	Hadoop API で SQL フィルターを用いた条件抽出
実現理由	第１イテレーションで実装しなかった機能の実装 分析のためデータ抽出する際、抽出するデータの量を限定する必要があるケースが存在する
他書類への影響	あり フィルターの仕様に関する資料を追加する
担当者	トウ
備考	IRS から抽出したデータを一度 HDFS に保存するため、’/tmp/irs2db_<Centos のユーザ名>’というフォルダを作成する。

調査結果

- **Hadoop API を用いた条件抽出**

  納品時の NET ツールで、

  Hadoop API の設定ファイル中に filter.push=false を設定し、Configurator.filter()で比較文（>,=など）のフィルターを挿入し、サンプルの

auction データベースに対する抽出を行った。結果として、抽出したデータを限定することに成功した。

- **実現可能性**

  可能

  納品時の NET ツールの実装で動作が可能であったことが判明した。

  filter.push=false の設定を仕様とする

備考

フィルターに対応する抽出条件は、比較述語及び IN 述語である。
一つのカラムに対して、指定できる抽出条件は１つである。

比較述語

演算子	説明
<	x< y は、x が y 未満
>	x> y は、x が y を超える
<=	x<= y は、x が y 以下
>=	x>= y は、x が y 以上
=	x= y は、x と y が等しい
!=	x!= y は、x と y が等しくない

IN 述語

Expression IN (expr1[, expr2]...)の式で、左辺の式 expression の値が右辺の式 expr の値のいずれかと等しい場合、IN 述語の結果は"真" になる。

複数のカラムにそれぞれの抽出条件が定義される場合、AND 演算子で結びつく。

機能番号	9
機能名	抽出条件設定 GUI との機能連携
機能概要	GUI からの抽出条件を Extractor に反映する機能
実現理由	第１イテレーションで実装しなかった機能の実装 分析のためデータ抽出する際、抽出するデータの量を限定する必要があるケースが存在する
他書類への影響	あり GUI からの抽出条件の仕様に関する資料を追加する
担当者	唐

調査結果

- **GUI からの抽出条件の仕様**

  ユーザーが入力する抽出条件を扱うクラスを作成した。詳細は以下のとおりである。
  GUI でカラムごとにフィルターを指定する場合、ユーザーが入力した抽出条件がリストに保存される。Extractor はその抽出条件を受け取り、HadoopAPI に設定する。
  SQL 文を直接入力するフィルターを使用する場合は、別に保存し管理する。
  抽出条件を扱うクラスから入力された抽出条件を読み込むメソッドが存在しない。

- **Extractor 側の仕様**

  Hadoop API の Configurator.filter()でカラム名、比較文 (>,=など)、値を入力することで、フィルターが可能となる。カラムの index からカラム名を読み込むことが可能である。

- **実現可能性**

  可能
  GUI 側の抽出条件の getter メソッドを作れば、条件を Extractor に渡すことができる。Extractor 側で抽出条件を解析し、String 型に変換して、HadoopAPI に渡せば条件抽出ができる。

備考

GUI 側の抽出条件仕様は以下である。
図 1 に抽出条件入力画面の全体を示す。

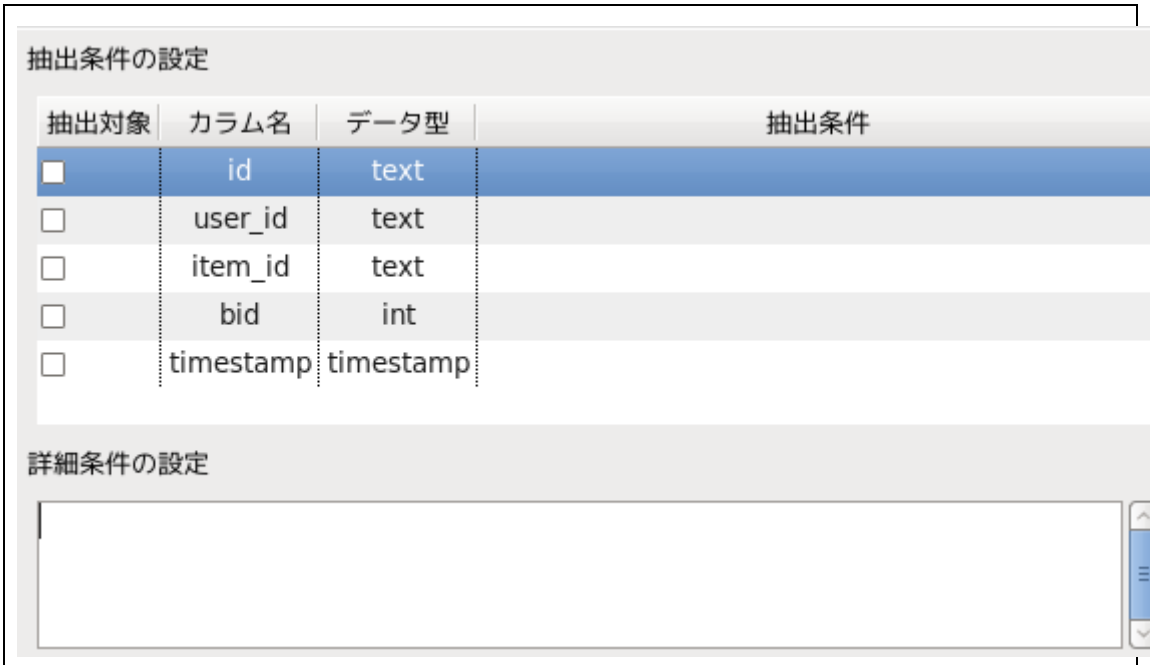


図 3

カラムごとに、抽出条件を指定することができる。対応している抽出条件は、比較述語と IN 述語である。一つのカラムに対して、入力できる抽出条件はひとつである。

図 4

図 2 のように左の入力欄に比較する値を入力し、右側のコンボボックスから条件を選択することで、抽出条件が指定される。

空文字を入れると既に登録されている抽出条件を削除できる。

コンボボックスにない抽出条件を利用する場合、図 1 の詳細条件の設定のテキストボックスに入力する必要がある。詳細条件の設定のテキストボックスに文字が場合、抽出条件として入力された文字列を利用する。その場合、カラムごとに設置された抽出条件は無視される。

指定した文字列は直接 HadoopAPI の Configurator.filter(String filterExpression)に渡される。filterExpression には SQL の述語表現（比較述語、BETWEEN 述語、IN

述語、LIKE 述語）が使用できる。詳細は「InfoFrame Relational Store SQL リファレンス」を参照してください。

対応する比較術後

演算子	説明
<	x< y は、x が y 未満
>	x> y は、x が y を超える
<=	x<= y は、x が y 以下
>=	x>= y は、x が y 以上
=	x= y は、x と y が等しい
!=	x!= y は、x と y が等しくない

IN 述語

Expression IN (expr1[, expr2]...)の式で、左辺の式 expression の値が右辺の式 expr の値のいずれかと等しい場合、IN 述語の結果は"真" になる。

ユースケース記述

ユースケース記述

事前条件：テーブルの情報が正しく抽出されており、抽出元データベースと Extract アイコンが作業ウィンドウで関連付けられていること。

Extract アイコンをダブルクリックすると抽出条件指定画面に移動する。

1.id カラムに対して 1 以上のデータを抽出するとき

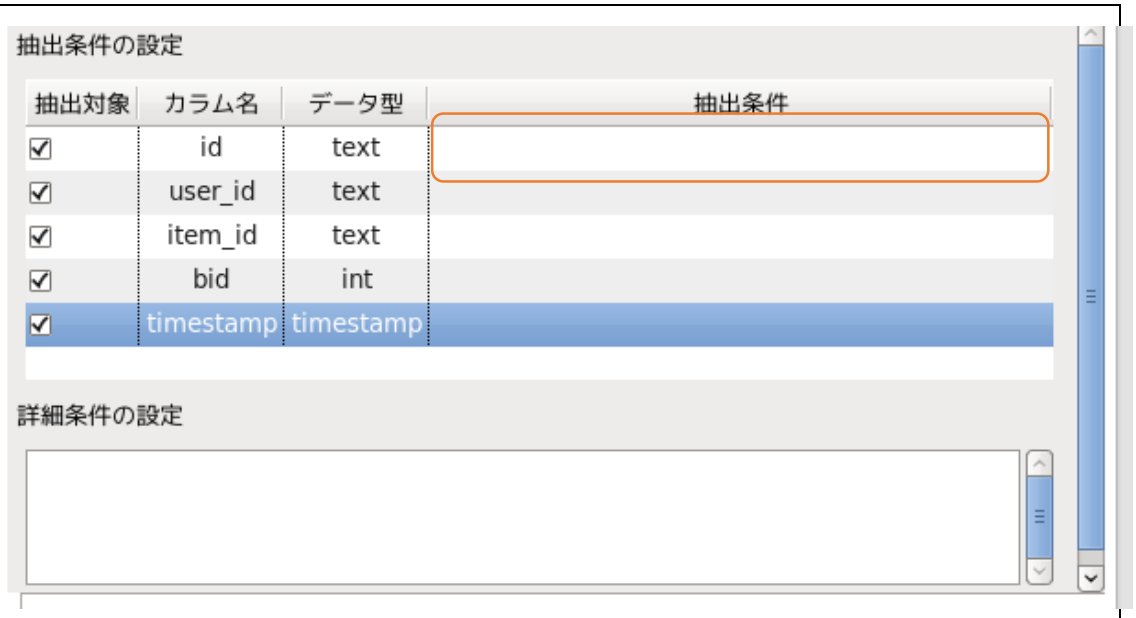


図１

図１の赤い枠線で囲った、カラムの抽出条件をクリックする。

図２

図２のテキストボックスに「1」を入力して、コンボボックスで「以上」を選択する。そして OK をクリックする。

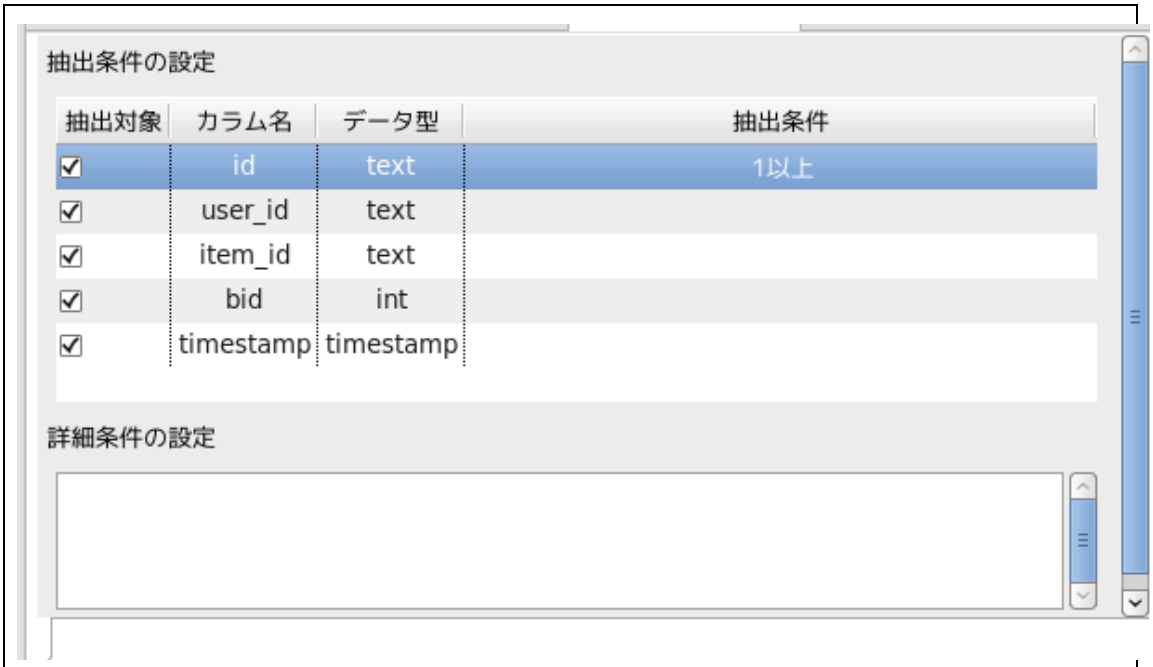


図３

図３のように、抽出条件が設定される。

2. 詳細条件入力機能を使うときの方法

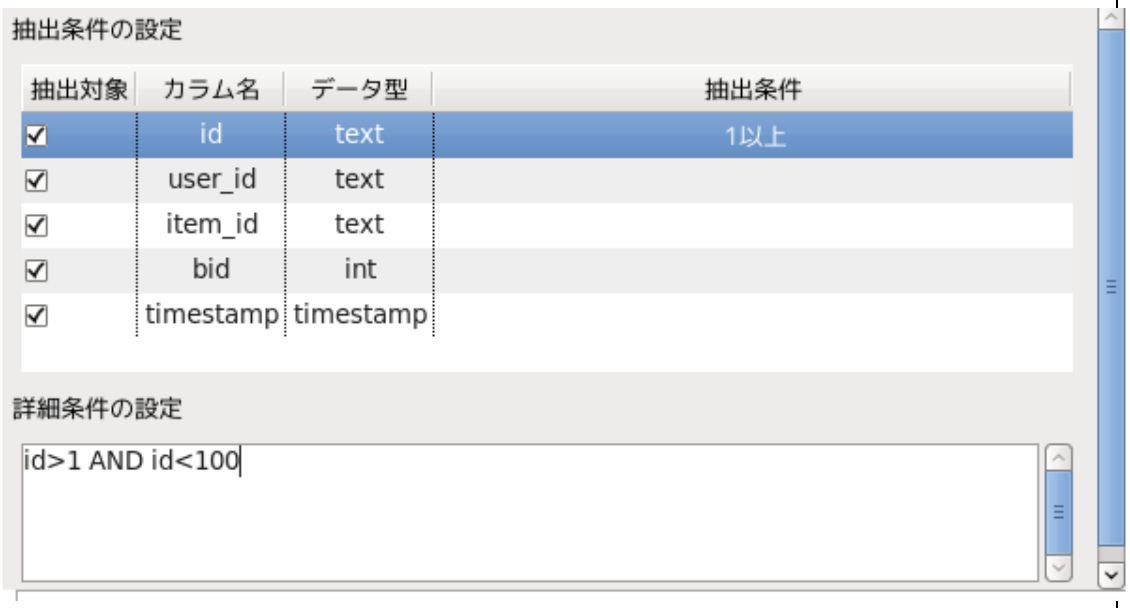


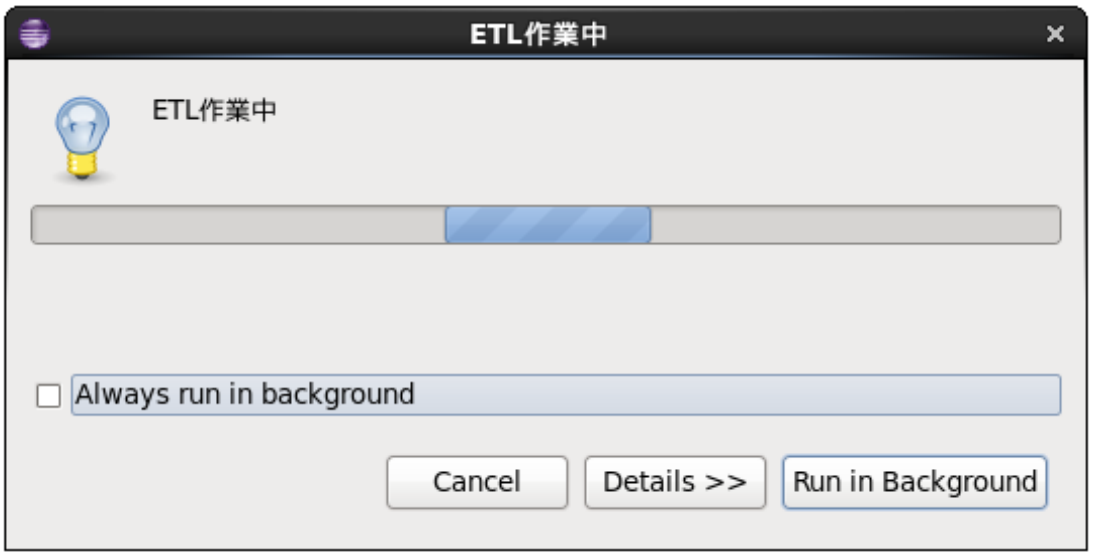
図４

詳細条件の設定のテキストボックスに抽出条件を入力する。この場合、カラムごとの抽出条件がされていてもカラムごとの抽出条件は無視され、図４のように入力されたテキストが抽出条件として使用される。

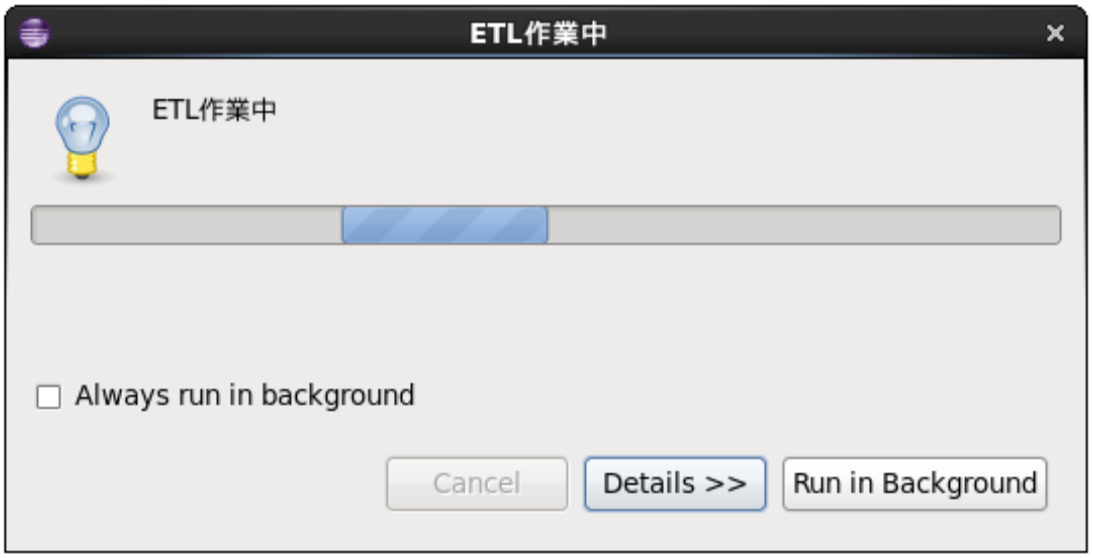
機能番号	10
機能名	ETL 作業の中止機能
機能概要	ETL 作業を中止する機能
実現理由	ETL 機能は時間のかかる作業であることが多いため、間違った ETL 作業を行っていると気づいた時に作業を中断できることで、ユーザの待ち時間を減らすことができるため
他書類への影響	なし
担当者	斎藤

備考

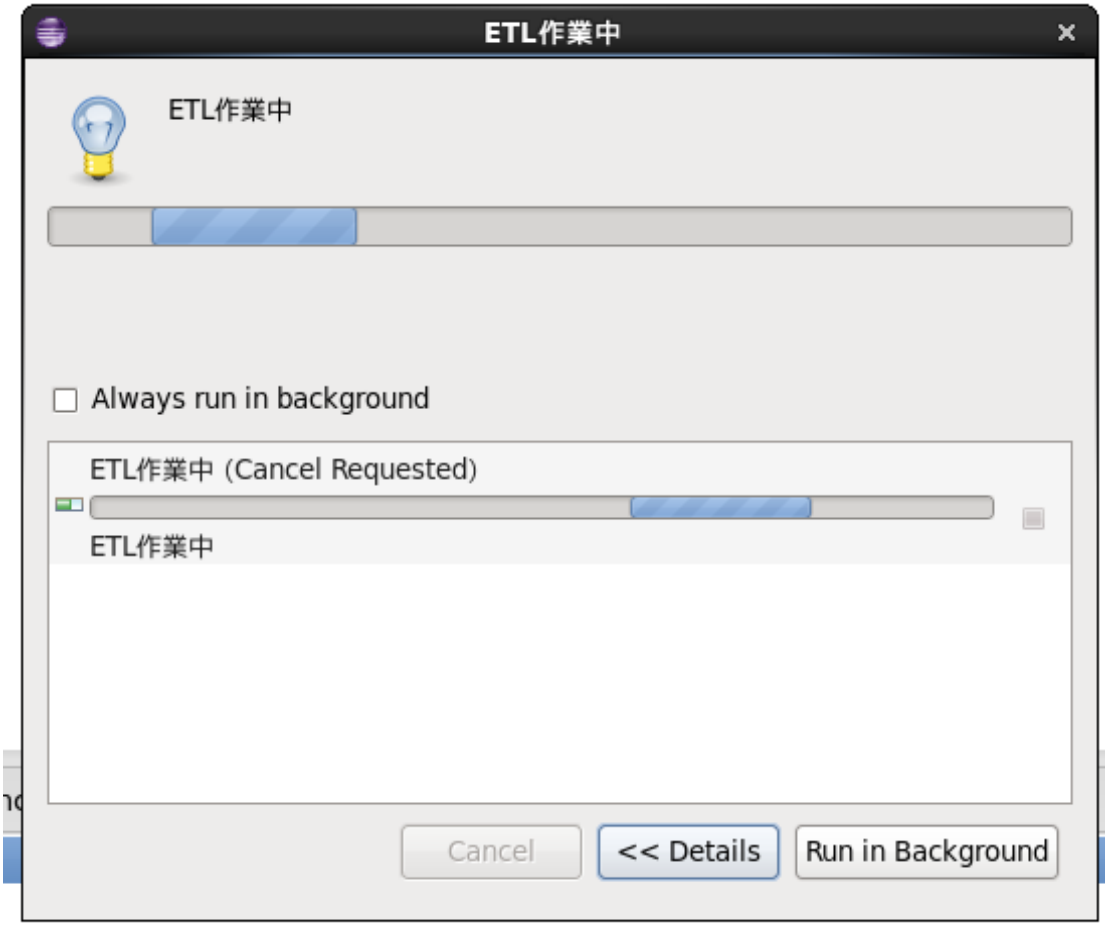
ETL 作業中では以下のようなポップアップが表示される。
ポップアップの右上の×ボタンは押しても何も変化はない。



このキャンセルボタンを押すと ETL 作業を中止する。
Cancel ボタンが押されると以下の画面のように Cancel ボタンが押せなくなる。

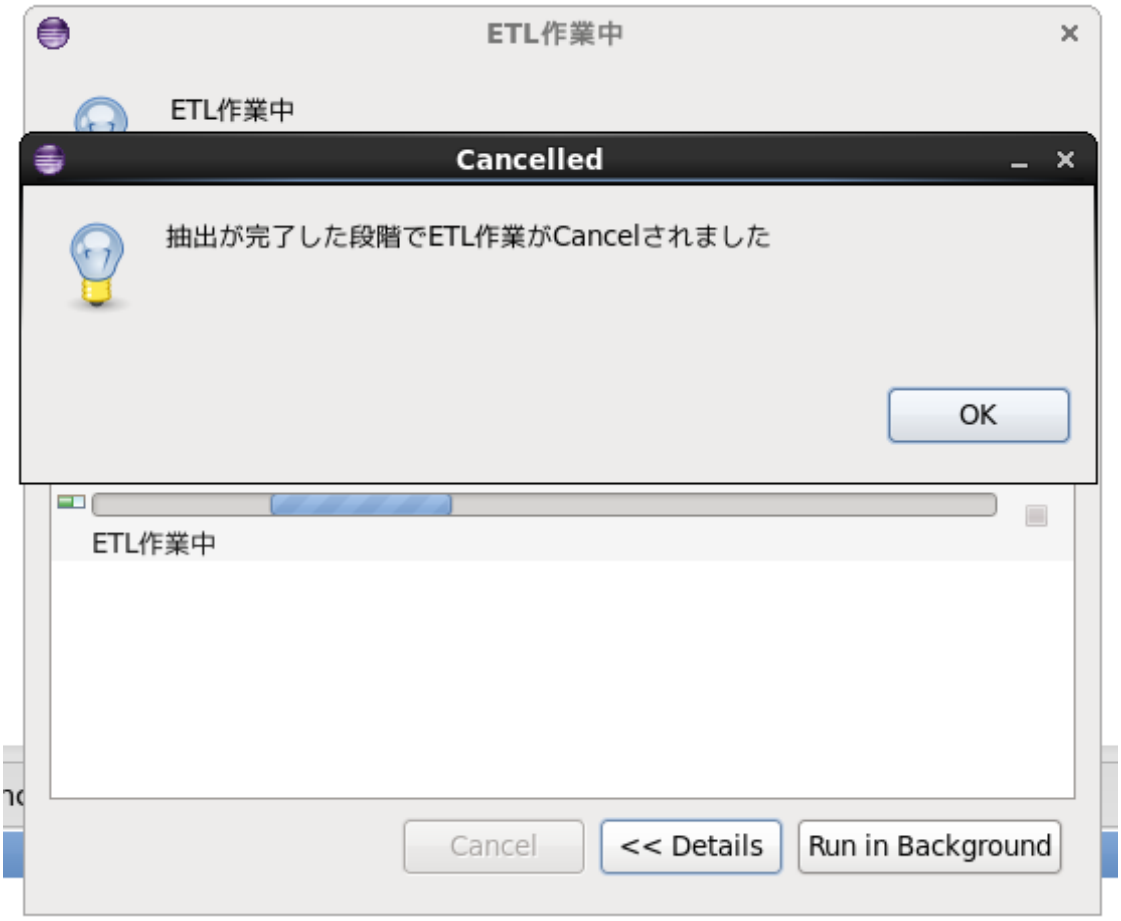


この状態で Detail ボタンを押すと以下のように ETL 作業が中断準備中であることが表示される。



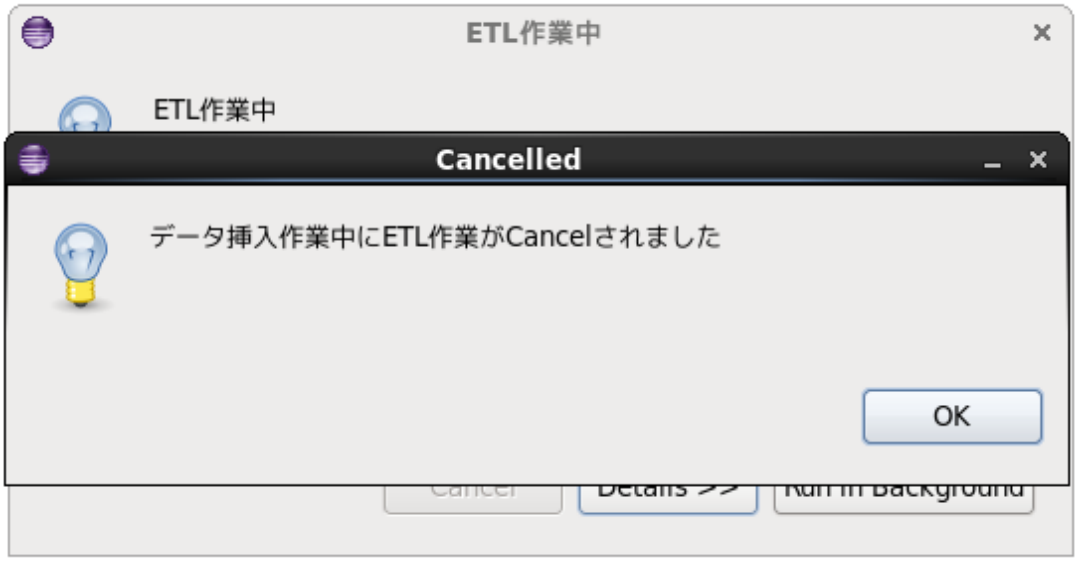
中断するタイミングは２つあり、

1.HadoopAPI でのデータ抽出中に Cancel ボタンが押された場合：データ抽出が終了するまで待機し、抽出が終了した段階で ETL 作業を中断する。



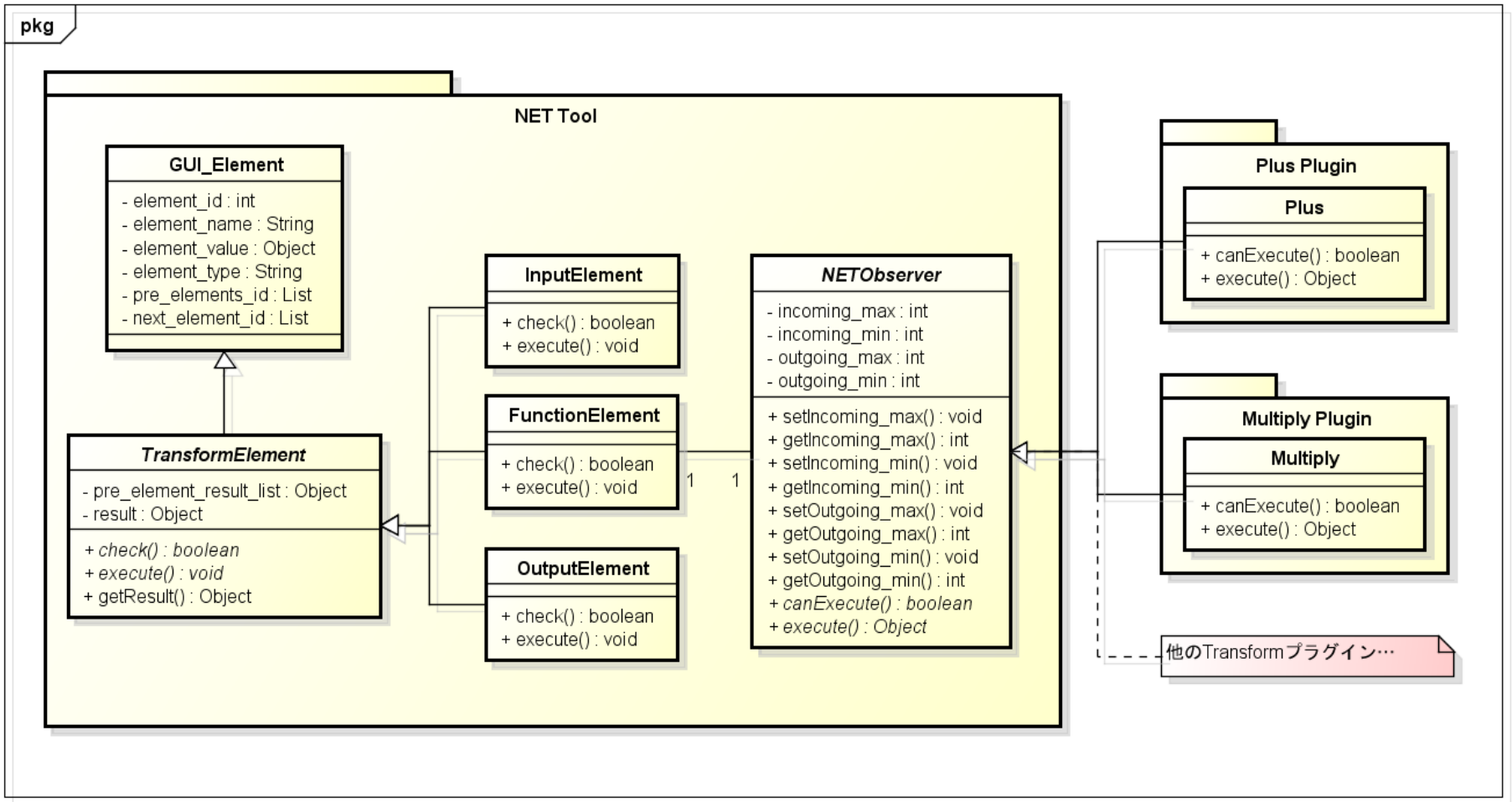
2.データ挿入中に Cancel ボタンが押された場合：Cancel ボタンが押されたタイミングでデータ挿入と ETL 作業を中断する。

Cancel される前にデータ移行先データベースに挿入されたデータは、データベースに残る。



Cancel が行われた後の NET ツールの表示は ETL 作業開始前に戻る。

Transform 機能のクラス図


pkg
NET Tool
GUI_Element
- element_id : int
- element_name : String
- element_value : Object
- element_type : String
- pre_elements_id : List
- next_element_id : List
TransformElement
- pre_element_result_list : Object
- result : Object
+ check() : boolean
+ execute() : void
+ getResult() : Object
InputElement
+ check() : boolean
+ execute() : void
FunctionElement
+ check() : boolean
+ execute() : void
OutputElement
+ check() : boolean
+ execute() : void
1
1
NETObserver
- incoming_max : int
- incoming_min : int
- outgoing_max : int
- outgoing_min : int
+ setIncoming_max() : void
+ getIncoming_max() : int
+ setIncoming_min() : void
+ getIncoming_min() : int
+ setOutgoing_max() : void
+ getOutgoing_max() : int
+ setOutgoing_min() : void
+ getOutgoing_min() : int
+ canExecute() : boolean
+ execute() : Object
Plus Plugin
Plus
+ canExecute() : boolean
+ execute() : Object
Multiply Plugin
Multiply
+ canExecute() : boolean
+ execute() : Object
他のTransformプラグイン…

単体テストテストケース一覧 S1(Transform)

単体テストテストケース一覧　作成者：HU BEIQI

番号	対象クラス	対象メソッド	テスト内容	想定結果	テスト結果	合否	実行日	対処	対処日
1	GUI_Element	add_pre_id	適切な引数を与え、先行ElementリストにElement IDが追加されるか確かめ	先行ElementリストにElement IDが追加されている	先行ElementリストにElement IDが追加されていた	○	2014/9/25	なし	
2		add_next_id	適切な引数を与え、後継ElementリストにElement IDが追加されるか確かめ	後継ElementリストにElement IDが追加される	後継ElementリストにElement IDが追加された	○	2014/9/25	なし	
3		getType	Elementの種類番号を正しく取得できるか確かめる	Elementの種類番号を返す	Elementの種類番号が返された	○	2014/9/25	なし	
4	TransformElement	add_pre_result	適切な引数を与え、先行Elementの実行結果リストにElementの実行結果を追加されるか確かめる	先行Elementの実行結果リストにElementの実行結果が追加される	先行Elementの実行結果リストにElementの実行結果が追加された	○	2014/9/26	なし	
5		add_pre_type	適切な引数を与え、先行Elementのタイプリストに Elementの種類番号を追加されるか確かめる	先行Elementのタイプリストに Elementの種類番号を追加される	先行Elementのタイプリストに Elementの種類番号を追加された	○	2014/9/26	なし	
6		add_next_type	適切な引数を与え、後継Elementのタイプリストに Elementの種類番号を追加されるか確かめる	後継Elementのタイプリストに Elementの種類番号を追加される	後継Elementのタイプリストに Elementの種類番号を追加された	○	2014/9/26	なし	
7	InputColumnElement	check	正しいフォーマットを持っている InputColumnElementを対象にメソッドを実行し、チェック結果が正しいか確	trueを返す	trueを返した	○	2014/9/26	なし	
8			不正なフォーマット（冗長Elementがある場合）を持っている InputColumnElementを対象にメソッドを実行し、チェック結果が正しいか確かめる	falseを返す	falseを返した	○	2014/9/26	なし	
9		execute	メソッドを実行し、正しいElement実行結果が取得できるか確かめる	正しいElement実行結果が取得できる	正しいElement実行結果が取得できた	○	2014/9/26	なし	
10	OutputColumnElement	check	正しいフォーマットを持っている OutputColumnElementを対象にメソッドを実行し、チェック結果が正しいか確かめる	trueを返す	trueを返した	○	2014/9/27	なし	
11			不正なフォーマット（後継Elementがある場合）を持っている InputColumnElementを対象にメソッドを実行し、チェック結果が正しいか確	falseを返す	falseを返した	○	2014/9/27	なし	
12			不正なフォーマット（冗長Elementが2個以上ある場合）を持っている InputColumnElementを対象にメソッドを実行し、チェック結果が正しいか確かめる	falseを返す	falseを返した	○	2014/9/27	なし	
13		execute	メソッドを実行し、intの変換ができるか確かめる	intの変換ができて、正しい結果が取得される	intの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
14			メソッドを実行し、bigintの変換ができるか確かめる	bigintの変換ができて、正しい結果が取得される	bigintの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
15			メソッドを実行し、decimalの変換ができるか確かめる	decimalの変換ができて、正しい結果が取得される	decimalの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
16			メソッドを実行し、floatの変換ができるか確かめる	floatの変換ができて、正しい結果が取得される	floatの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
17			メソッドを実行し、doubleの変換ができるか確かめる	doubleの変換ができて、正しい結果が取得される	doubleの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
18			メソッドを実行し、varcharの変換ができるか確かめる	varcharの変換ができて、正しい結果が取得される	varcharの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
19			メソッドを実行し、dateの変換ができるか確かめる	dateの変換ができて、正しい結果が取得される	java.util.Dateのメソッドをつかっているため、NGになった	×	2014/9/27	java.sql.Dateが利用できる変換方式に修正した	2014/9/27
20			メソッドを実行し、timeの変換ができるか確かめる	timeの変換ができて、正しい結果が取得される	java.util.Dateのメソッドをつかっているため、NGになった	×	2014/9/27	java.sql.Dateが利用できる変換方式に修正した	2014/9/27
21			メソッドを実行し、timestampの変換ができるか確かめる	timestampの変換ができて、正しい結果が取得される	timestampの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
22			メソッドを実行し、booleanの変換ができるか確かめる	booleanの変換ができて、正しい結果が取得される	booleanの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
23			メソッドを実行し、byteaの変換ができるか確かめる	byteaの変換ができて、正しい結果が取得される	byteaの変換ができて、正しい結果が取得された	○	2014/9/27	なし	
24	TransformThread	run	適切なElementを作り、メソッドを実行して、実行結果が正しいか確かめる	正しいスレッド実行結果が取得できる	正しいスレッド実行結果が取得した	○	2014/9/27	なし	

単体テストテストケース一覧 S2(Transform)

単体テストテストケース一覧　　作成者：HU BEIQI

番号	対象クラス	対象メソッド	テスト内容	想定結果	テスト結果	合否	実行日	対処	対処日
1	OutputColumnElement	execute	メソッドを実行し、対応していないデータ型に変換する対応できるか確かめ	実行結果として0を返す	実行結果として0が返された	○	2014/11/27	なし	
2			メソッドを実行し、データ型変換失敗の場合対応できるか確かめる	実行結果として0を返す	実行結果として0が返された	○	2014/11/27	なし	
3	TransformThread	run	Elementが実行不可の場合対応できるか確かめる	実行結果としてfalseを返す	実行結果としてfalseが返された	○	2014/11/27	なし	
4			Elementが実行失敗の場合対応できるか確かめる	実行結果としてfalseを返す	実行結果としてfalseが返された	○	2014/11/27	なし	
5	Transform	transformRun	適切なElementを作り、メソッドを実行して、実行結果が正しいか確かめる	実行結果のDataSetを返す	実行結果のDataSetを返された	○	2014/11/27	なし	
6			Elementが実行失敗の場合対応できるか確かめる	実行結果のDataSetを返す	実行結果のDataSetを返された	○	2014/11/27	なし	
7	CreateConnectionCommand	execute	Connectionを作成する場合、GUI_Elementの先行/後継Element IDリストが更新できるか確かめる	ターゲットElementのIDがソースElementの後継Element IDリストに追加されて、ソースElementのIDがターゲットElementの先行Element IDリストに追加されている	ターゲットElementのIDがソースElementの後継Element IDリストに追加されて、ソースElementのIDがターゲットElementの先行Element IDリストに追加されも	○	2014/11/27	なし	
8		canExecute	ソースとターゲットが同じ場合、チェック結果が正しいか確かめる	falseを返す	falseが返された	○	2014/11/27	なし	
9			ソースが出力カラムの場合、チェック結果が正しいか確かめる	falseを返す	falseが返された	○	2014/11/27	なし	
10			ターゲットが入力カラムの場合、チェック結果が正しいか確かめる	falseを返す	falseが返された	○	2014/11/27	なし	
11			作成Connectionがすでに存在している場合、チェック結果が正しいか確か	falseを返す	falseが返された	○	2014/11/27	なし	
12		undo	Undoを実行して、ソースとターゲットのGUI_Elementが復旧されるか確か	GUI_Elementが復旧される	GUI_Elementが復旧された	○	2014/11/27	なし	
13		redo	Redoを実行して、ソースとターゲットのGUI_Elementが復旧されるか確か	GUI_Elementが復旧される	GUI_Elementが復旧された	○	2014/11/27	なし	
14	CreateNodeCommand	execute	メソッドを実行し、要素がDiagramに追加されるか確かめる	要素がDiagramに追加される	要素がDiagramに追加された	○	2014/11/27	なし	
15		undo	Undoを実行して、Diagramが復旧されるか確かめる	Diagramが復旧される	Diagramが復旧された	○	2014/11/27	なし	
16		redo	Redoを実行して、Diagramが復旧されるか確かめる	Diagramが復旧される	Diagramが復旧された	○	2014/11/27	なし	
17	DeleteConnectionCommand	execute	メソッドを実行し、Connectionが削除される場合、GUI_Elementが正しく修正されるか確かめる	ターゲットElementのIDがソースElementの後継Element IDリストから削除され、ソースElementのIDがターゲットElementの先行Element IDリストから削除されている	ターゲットElementのIDがソースElementの後継Element IDリストから削除され、ソースElementのIDがターゲットElementの先行Element IDリストから削除されていた	○	2014/11/27	なし	
18		undo	Undoを実行して、ソースとターゲットのGUI_Elementが復旧されるか確か	GUI_Elementが復旧される	GUI_Elementが復旧された	○	2014/11/27	なし	
19		redo	Redoを実行して、ソースとターゲットのGUI_Elementが復旧されるか確か	GUI_Elementが復旧される	GUI_Elementが復旧された	○	2014/11/27	なし	
20	DeleteNodeCommand	execute	メソッドを実行し、要素がDiagramから削除されるか確かめる	要素がDiagramから削除される	要素がDiagramに追加された	○	2014/11/28	なし	
21		undo	Undoを実行して、Diagramが復旧されるか確かめる	Diagramが復旧される	Diagramが復旧された	○	2014/11/28	なし	
22		redo	Redoを実行して、Diagramが復旧されるか確かめる	Diagramが復旧される	Diagramが復旧された	○	2014/11/28	なし	
23	MoveNodeCommand	execute	メソッドを実行し、要素の座標が正しく修正されるか確かめる	要素の座標が正しく修正される	要素の座標が正しく修正された	○	2014/11/28	なし	
24		undo	要素の座標は復旧されるか確かめる	要素の座標は復旧される	要素の座標は復旧された	○	2014/11/28	なし	
25		redo	要素の座標は復旧されるか確かめる	要素の座標は復旧される	要素の座標は復旧された	○	2014/11/28	なし	

機能テスト S1

機能テスト

目的：「第一イテレーション機能詳細一覧」で合意した機能について相違なく実現できているか確かめるため実施する。

方法：各機能について、担当者が「テスト内容」に記述された操作を行い、想定した結果と実際の結果を比較し、テストの合否を判断する。

機能番号：1

機能名：IRS に接続する

担当：唐可

番号	テスト内容	想定結果	実際の結果	合否	実施日
1-1	正常なログイン情報を入力し、正常にログインできるか確かめる	正常にログインができ、その旨のポップアップが表示される	左に同じ	合	2014/10/13
1-2	URL を正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/10/13
1-3	ユーザー名を正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/10/13
1-4	パスワードを正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/10/13

1-5	ポート番号を正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/10/13
1-6	何も入力しない状態で接続を押します	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/10/13

機能番号：2

機能名：IRS からテーブル定義データの取得

担当：唐可

番号	テスト内容	想定結果	実際の結果	合否	実施日
2-1	テーブルが存在する IRS データへ接続	テーブルリストが正しく表示されます	左に同じ	合	2014/10/13
2-2	テーブルにカラムが存在する時、テーブル情報を取ります	テーブル情報画面でカラム名、データタイプが正しく表示されます	左に同じ	合	2014/10/13
2-3	テーブルにカラムが存在しない時、テーブル情報を取ります	テーブル情報画面でカラム名、データタイプが正しく表示されます	左に同じ	合	2014/10/13

機能番号：3

機能名：IRS から取得したテーブル定義データの表示

備考:機能 1,2 で正常にテーブルを取得できたことを前提とする。

担当：成澤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
3-1	IRSから正常にテーブルを取得した後に、テーブルリストに入っているテーブルを右クリックし、「テーブル情報」と右クリックメニューに表示されるか確認する。	右クリック時にテーブル情報が表示される。	左に同じ	合	2014/10/11
3-2	右クリック時に表示された「テーブル情報」をクリックし、テーブル定義データを表示するダイアログが表示されるか確認する。	テーブル定義データを表示するダイアログが表示される。	左に同じ	合	2014/10/11

機能番号：4

機能名：データを抽出するテーブルの指定

担当：成澤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
4-1	データソースウィンドウから作業ウィンドウへ抽出するテーブルをドラックする。	作業ウィンドウにドラッグしたテーブル名が表記されたアイコンが描画される。	左に同じ	合	2014/10/11
4-2	作業ウィンドウ内で「Connect」を選択し、抽出アイコンの「Extract」と線をつなぐ。	Extractとの間に線が引かれ、抽出対象のテーブルとなる。	左に同じ	合	2014/10/11

機能番号：5
機能名：IRS からデータを抽出する機能
担当：唐可

番号	テスト内容	想定結果	実際の結果	合否	実施日
5-1	正しく入力を行い、任務を実行し、IRS からデータを抽出します	IRS からのデータが正しく抽出されます	左に同じ	合	2014/10/13
5-2	walStorageVoldemortBootstrapUrls の入力値が間違えた状態で抽出を行います	抽出できないポップアップが表示され、データベース情報が表示されます	左に同じ	合	2014/10/13
5-3	kvstoreStorageVoldemortBootstrapUrls の入力値が間違えた状態で抽出を行います	抽出できないポップアップが表示され、データベース情報が表示されます	左に同じ	合	2014/10/13

機能番号：6
機能名：抽出する際に抽出条件(フィルター)を指定する機能
担当：唐可
備考：未実装

機能番号：7
機能名：PostgreSQL との接続
担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
7-1	正常なログイン情報を入力し、正常にログインできるか確かめる	正常にログインができ、その旨のポップアップが表示される	左に同じ	合	2014/09/25

7-2	URLを正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/09/25
7-3	ユーザー名を正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/09/29
7-4	パスワードを正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/09/29
7-6	データベース名を正しくない情報、その他を正常なログイン情報を入力しログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/09/29
7-7	ポート番号を正しくない情報、その他を正常なログイン情報を入力し	ログイン出来ないエラーを占めるポップ	左に同じ	合	2014/09/29

	ログイン出来ないエラーが表示されるか確かめる	アップが表示される。ログインに使用したログイン情報が表示される。			
7-8	不正なポート番号(65536)、その他を正常なログイン情報を入力しポート番号が不正であることを示すエラーが表示されるか確かめる	ポート番号が不正であることを示すエラーが表示される	ログイン出来ないエラーが表示される	否	2014/09/29 ユーザーが入力したポート番号をチェックし、適切なエラーを表示させる(対処済み)
7-9	不正なポート番号(port_error)、その他を正常なログイン情報を入力しポート番号が不正であることを示すエラーが表示されるか確かめる	ポート番号が不正であることを示すエラーが表示される	ログイン出来ないエラーが表示される	否	ユーザーが入力したポート番号をチェックし、適切なエラーを表示させる(対処済み)
7-10	PostgreSQL を停止した状態で、正常なログイン情報を入力し、ログイン出来ないエラーが表示されるか確かめる	ログイン出来ないエラーを占めるポップアップが表示される。ログインに使用したログイン情報が表示される。	左に同じ	合	2014/09/29

機能番号：8

機能名：PostgreSQL からテーブル定義データの取得

担当：斎藤

備考：機能 7 で正常にログインできたことを前提とする

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
8-1	テーブルが正しく取得	正常にテーブ	左に同じ	合	2014/09/25

	できているか確かめる。	ルが取得でき、テーブルリストに表示されている			
8-2	テーブルが存在しないデータベースからテーブルを取得する。	接続できたが、テーブルが１つも取得できないことを示す警告が表示される	左に同じ	合	2014/09/25
8-3	テーブルを正しく取得した後、再度同じデータベースからテーブルを取得する	テーブルリストに変化はない	テーブルリストに重複してテーブルが登録される	否	2014/09/28 テーブル取得はデータベースにつき、一度しかできないようにした。再度取得する場合は、データベース情報を削除する必要がある。(対処済み)
8-4	テーブルを正しく取得した後、データベースでテーブル情報を変更し、再度テーブル情報を取得する	テーブルリストが更新される	テーブルリストは更新され、テーブルリストに重複してテーブルが登録される	否	2014/09/28 テーブル取得はデータベースにつき、一度しかできないようにした。再度取得する場合は、データベース情報を削除する必要がある(対処済み)

機能番号：9
機能名：PostgreSQL から取得したテーブル定義データを表示する機能
担当：斎藤
備考：機能 8 で正常にテーブルリストを取得できていることを前提とする

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
9-1	テーブルを右クリックしテーブル情報を選択し、テーブル情報が表示されるか確かめる	テーブル情報が表示される	左に同じ	合	2014/09/28
9-2	データベースを右クリックする。テーブル情報が右クリックメニューにないことを確認する。	テーブル情報が右クリックメニューにない	テーブル情報が右クリックメニューにある。	否	2014/09/29 右クリックする対象によって表示される右クリックメニューを変更する。(対処済み)

機能番号：10
機能名：PostgreSQL に新規テーブルを作成する機能
担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
10-1	正しいテーブル作成情報を入力し、テーブルが作成される	新しくテーブルが作成され、テーブルリストが更新さ、作成	左に同じ	合	2014/10/08

	か確かめる。	されたテーブル情報がポップアップで表示される			
10-2	誤ったテーブル作成情報(カラムが0)を入力し、入力が不正であることが表示されるか確かめる。	入力が不正であることが表示される。	カラムが 0 個の場合は「OK」ボタンが押せないようになっている	否	2014/10/07 ポップアップが表示されるよりは、「OK」ボタンが押せないほうがユーザーにとって親切だと判断し、修正しない(対処済み)
10-3	テーブルをクリックし、右クリックメニューにテーブル作成が表示されないことを確かめる。	テーブル作成が右クリックメニューに表示されない	左に同じ	合	2014/09/30
10-4	接続情報を取得する前にテーブル作成を行おうとする	接続情報を入力するまでは、右クリックメニューにテーブル作成を表示させない	左に同じ	合	2014/10/11
10-5	プライマリキーを複数設定しようとする	複数のプライマリキーは設定できないことを表示する	１つプライマリキーが登録されると、プライマリキーを設定するチェックボック	否	2014/10/08 対処：現在の仕様の方がユーザーにとって親切だと判断し、修正せず。

			スが押せなくなる		
10-6	テーブル名を入力せずに OK ボタンを押そうとする	OKボタンが押せないようになっている	OKボタンが押せる	否	2014/10/08

機能番号：11
機能名：データを挿入するテーブルを指定する機能
担当：成澤
備考：機能 8~10 でデータソースウィンドウにテーブルが表示されていることを前提とする。

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
11-1	データソースウィンドウから作業ウィンドウへデータを挿入されるテーブルをドラックする。	作業ウィンドウにドラッグしたテーブル名が表記されたアイコンが描画される。	左に同じ	合	2014/10/11
11-2	作業ウィンドウ内で「Connect」を選択し、変換アイコンの「Transform」と線をつなぐ。	Transform との間に線が引かれ、データの挿入対象のテーブルとなる。			

機能番号：12
機能名：データ挿入機能
担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
12-1	対応予定全ての型数をインサートする	用意したテーブルに適切にインサー	time,timetz, timestamp,timestampz の結果でエラーが発生した	否	2014/10/06 InsertThread.java の条件分岐を修正(2014/10/06 修正済み)

		トが行われている。			
12-2	型とINSERTするデータの型が不整合なデータをINSERTする	型変換が不正であるポップアップが表示される。	左に同じ	合	2014/10/06

機能番号：13
機能名：IRS から PostgreSQL 用にデータを変換する
担当：HU

番号	テスト内容	想定結果	実際の結果	合否	実施日
13-1	正常な Element 情報を入力し、正しいトランスフォーム結果が取得できるか確かめる	Element 情報に対応する正しいトランスフォーム結果が取得できる	左に同じ	合	2014/10/01
13-2	不正な変換（int から Date など）を実行し、エラーが表示されるか確かめる	“＜データ型＞への変換はできませんでした。”が表示される	左に同じ	合	2014/10/12

機能番号：14
機能名：テーブル間の対応関係の定義を行う(マッピング)機能
担当：HU

番号	テスト内容	想定結果	実際の結果	合否	実施日
14-1	Input Column から Output Column へ線を引くことを試す	線と矢印が正しく表示される	左に同じ	合	2014/10/01
14-2	Input Column から	線が引けない	左に同じ	合	2014/10/01

	Input Column / Input Table のタイトル / Output Table のタイトルへ線を引くことを試す				
14-3	Output Column から他のエレメントへ線を引くことを試す	線が引けない	左に同じ	合	2014/10/01
14-4	Output Column をある線の終点とする場合、Output Column へ線をもう一本引くことを試す。	線が引けない	左に同じ	合	2014/10/12
14-5	画面に存在する線を選択し、Delete する	該当する線が消される	左に同じ	合	2014/10/01
14-6	14-4 の直後、画面を右クリックして、Undo を選択する	消された線が書き直される	左に同じ	合	2014/10/01

機能番号：17
機能名：アイコン（データ移行元やデータ移行先、条件追加）同士を線で結ぶ機能
担当：成澤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
17-1	作業ウィンドウ内に既にアイコンがある状態で、「Connect」を選択する。	Connect が選択され、マウスカーソルが変化する。	左に同じ	合	2014/10/11
17-2	線を結ぶために始点となるアイコン、終点となるアイコンの順番でクリックする。	始点と終点が線を引くことができる組み合わせであった場合に線が引かれる。※	左に同じ	合	2014/10/11

17-3	線を結ぶことができない組み合わせに対して線を引こうとし、線がひけないことを確認する。	線が引けず、副作用はない。	左に同じ	合	2014/10/11

※データ移行元から抽出機能、抽出機能から変換機能、変換機能からデータ移行先へのみ線を引くことができる。

機能テスト S2

機能テスト

目的:「第一イテレーション機能詳細一覧」で合意した機能について相違なく実現できているか確かめるため実施する。

方法:各機能について、担当者が「テスト内容」に記述された操作を行い、想定した結果と実際の結果を比較し、テストの合否を判断する。

第二イテレーション実装機能

機能番号	機能名
1	PostgreSQL のテーブル作成時に特定のデータ型の長さを指定できる機能
2	PostgreSQL の COPY コマンドを使用したデータの挿入機能
3	ETL 作業が実行中であることを示す機能
4	IRS からのテーブル情報取得状況の表示機能
5	作成テーブルの表示更新機能
6	IRS のテーブル表示の修正
7	Transform のプラグイン化
8	Hadoop API を用いた条件抽出
9	ユーザが入力した抽出条件を扱う機能
10	ETL 作業の中止機能

機能番号:1

機能名:PostgreSQL のテーブル作成時に特定のデータ型の長さを指定できる機能

担当:斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
1-1	データ型の長さを指定して(2,2) decimal 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。データ型の長さが 2,2 の decimal 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
1-2	データ型の長さを指定して(2) decimal 型のカラムを登録し	テーブル作成成功のポップアップが表示	左に同じ	合	2014/11/27

	たテーブルを作成する	される。 データ型の長さが 2,0 の numeric 型のカラムを持ったテーブルが作成される			
1-3	データ型の長さを指定せず decimal 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが設定されていない numeric 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
1-4	データ型の長さを指定して (2,2) numeric 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが 2,2 の numeric 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
1-5	データ型の長さを指定して(2) numeric 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが 2,0 の numeric 型のカラムを持っ	左に同じ	合	2014/11/27

		たテーブルが作成される			
1-6	データ型の長さを指定せず numeric 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが設定されていない numeric 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
1-7	データ型の長さを指定して(2) varchar 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが 2 の varchar 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
1-8	データ型の長さを指定せず varchar 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが指定されていない varchar 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
1-9	データ型の長さを指定して(2) char 型の	テーブル作成成功のポップ	左に同じ	合	2014/11/27

	カラムを登録したテーブルを作成する	アップが表示される。 データ型の長さが2のchar型のカラムを持ったテーブルが作成される			
1-10	データ型の長さを指定せず　char 型のカラムを登録したテーブルを作成する	テーブル作成成功のポップアップが表示される。 データ型の長さが指定されていないchar 型のカラムを持ったテーブルが作成される	左に同じ	合	2014/11/27
	データ型の長さに不適切な値を入力し、テーブルを作成する	テーブル作成を試みたが、テーブルを作成できなかった旨のエラーと SQL エラー詳細が表示される	左に同じ	合	2014/11/27

機能番号：2

機能名：PostgreSQL の COPY コマンドを使用したデータの挿入機能

担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
2-1	対応予定全ての型に対して正しい	用意したテーブルに適切にインサ	左に同じ	合	2014/11/28

	データをインサートする	ートが行われている。			
2-1	対応予定全ての型に対して NULL をインサートする	用意したテーブルに NULL がインサートされている。	NULL がインサートされず Nullpointer 例外が発生する。	否	2014/11/28 PostgresqlInsertThreadCopy の makeValues メソッドを修正し、再テストの結果 合格 （2014/11/28）
2-2	型と INSERT するデータの型が不整合なデータを INSERT する	型変換が不正であるポップアップが表示される。			

機能番号：3

機能名：ETL 作業が実行中であることを示す機能

担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
3-1	ETL 作業を正常に実行し、ETL 作業が実行中であることを示すポップアップが表示されることを確かめる。	用意したテーブルに適切にインサートが行われている。	左に同じ	合	2014/11/27
3-2	ETL 作業を正常に実行し、ETL 作業が終了した時点で、終了	正常に ETL 作業が終了した時点で終了を告知するポップ	左に同じ	合	2014/11/27

	を告知するポップアップが表示されることを確かめる。	アップが表示される			

機能番号：4

機能名：IRS からのテーブル情報取得状況の表示機能

担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
4-1	正しい IRS に接続し、1つ以上のデータベース、１つ以上のテーブル情報を取得し、取得できているか確かめる	テーブル情報が取得されたことを告知するポップアップが表示される	左に同じ	合	2014/11/21
4-2	IRS に接続を失敗する情報を入力し、接続が失敗した旨のポップアップが表示されるか確かめる	IRS への接続が失敗した旨のポップアップが表示される	左に同じ	合	2014/11/21
4-3	データベースが存在しない IRS に接続し、データベースが存在しない旨のポップ	IRS への接続は成功したが、データベースが存在しないためデータベース情報が	左に同じ	合	2014/11/21

	アップが表示されるか確かめる。	取得できなかった旨のポップアップが表示される			
4-4	データベースが存在するが、データベース内にテーブルが存在しない IRS に接続し、テーブルが存在しない旨のポップアップが表示されるか確かめる	IRS、データベースへの接続は成功したが、テーブルが存在しないためテーブル情報が取得できなかった旨のポップアップが表示される	左に同じ	合	2014/11/21

機能番号：5

機能名：作成テーブルの表示更新機能

担当：成澤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
5-1	NET ツール上で作成したテーブルに対してテーブル情報機能を利用する。	テーブル情報がダイアログで表示される。	左に同じ。	合	2014/12/02
5-2	作成したテーブルや、空のテーブルに対してテーブル情報機能を利用する。	テーブル定義は表示されるが、TableExample は空欄で表示される。	左に同じ。	合	2014/12/02
5-3	IRSのテーブルに対してテーブル表示機能を利用する。	テーブル定義情報は表示されるが、Table Example は空	左に同じ。	合	2014/12/02

		欄で表示される。			

機能番号：6

機能名：IRS のテーブル表示の修正

担当：成澤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
6-1	IRS に接続し、テーブル情報を取得する。	テーブルリストに「DB 名.テーブル名」と表記される。	左に同じ。	合	2014/12/02
6-2	作業ウィンドウに IRS のテーブルをドラックアンドドロップする。	「DB 名.テーブル名」のアイコンが作業ウィンドウに生成される。	左に同じ。	合	2014/12/02
6-3	対応付けを行い、Extract 条件を入力した後に Transform ウィンドウを開く	「DB 名.テーブル名」のカラムが最上部になっている、テーブルが生成されている。	左に同じ。	合	2014/12/02

機能番号：7

機能名：Transform のプラグイン化

担当：HU

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
7-1	Function ビューの Transform にすべての Transform プラグインが正常に表示されるか確かめる	すべての Transform プラグインが表示される	左に同じ	合	2014/12/01

7-2	Function ビューの Transform にあるプラグインを Transform Window にドラグ・ドロップできるか確かめる	プラグインを Transform Window にドラグ・ドロップできる	左に同じ	合	2014/12/01
7-3	Transform プラグインへの入力数が設定値を超える場合、線が引けないことを確かめる	線が引けない	線が入力数上限+1 まで引ける	否	2014/12/01 CreateConnectionCommand.java を修正することで解決した
7-4	Transform プラグインからの出力数が設定値を超える場合、線が引けないことを確かめる	線が引けない	線が出力数上限+1 まで引ける	否	2014/12/01 CreateConnectionCommand.java を修正することで解決した
7-5	Transform プラグインへの入力数が不足の場合、エラーメッセージが表示するか確かめる	エラーメッセージのポップアップが表示される	Transform プラグインが見つかれなかった	否	2014/12/01 FunctionElement を修正することで解決した
7-6	Transform プラグインからの出力数が不足の場合、エラーメッセージが表示するか確かめる	エラーメッセージのポップアップが表示される	左に同じ	合	2014/12/01
7-7	正常な Transform を編集し、Transform プラグインの実行結果が正しい	正しい実行結果が出る	左に同じ	合	2014/12/01

	か確かめる				

機能番号：8

機能名：Hadoop API を用いた条件抽出

担当：トウ

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
8-1	比較述語なしで抽出を行う	すべてのデータが抽出される	左に同じ	合	2014/12/2
8-2	一つのカラムに対して、比較述語を用いて抽出条件を設置する	対応するデータが抽出される	左に同じ	合	2014/12/2
8-3	一つのカラムに対して、IN 述語を用いて抽出条件を設置する、抽出を行う	対応するデータが抽出される	左に同じ	合	2014/12/2
8-4	複数のカラムに対して、比較述語を用いて抽出条件を設置する、AND 演算子で結びつく、抽出を行う	対応するデータが抽出される	左に同じ	合	2014/12/2
8-5	複数のカラムに対して、IN 述語を用いて抽出条件を設置する、AND 演算子で結びつく、抽出を行う	対応するデータが抽出される	左に同じ	合	2014/12/2
8-6	不正確のフィルター条件を設置して、抽出を行う	データが抽出されない	左に同じ	合	2014/12/2

機能番号：9
機能名：抽出条件設定と GUI の機能連携
担当：トウ

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
9-1	「詳細条件の設定」で抽出条件が存在する、カラムに抽出条件が存在しない場合、抽出を行う	「詳細条件の設定」で設置された条件で抽出を行う	左に同じ	合	2014/12/2
9-2	「詳細条件の設定」で抽出条件が存在する、カラムに抽出条件も存在する場合、抽出を行う	「詳細条件の設定」で設置された条件で抽出を行う	左に同じ	合	2014/12/2
9-3	「詳細条件の設定」で抽出条件が存在しない、一つのカラムに抽出条件が存在する場合、抽出を行う	カラムで設置された抽出条件で抽出を行う	左に同じ	合	2014/12/2
9-4	「詳細条件の設定」で抽出条件が存在しない、複数のカラムに抽出条件が存在する場合、抽出を行う	複数のカラムで設置された抽出条件で抽出を行う	左に同じ	合	2014/12/2
9-5	選択されないカラムに抽出条件が存在する場合、抽出を行う	該当カラムで設置された抽出条件が無視され、抽出を行う	左に同じ	合	2014/12/2

機能番号：10
機能名：ETL 作業の中止機能
担当：斎藤

番号	テスト内容	想定結果	実際の結果	合否	実施日と対処
10-1	HadoopAPI を用いたデータの抽出中にキャンセルボタン押し、データの抽出が終了した段階で ETL 作業が中断し、中断したことを報告するポップアップが表示されることを確かめる。	データの抽出が終了するまで待機し、データの抽出が終了し次第 ETL 作業を中断する。 データの抽出が終了した段階で ETL 作業が中断したことを報告するポップアップが表示される。	左に同じ	合	2014/11/27
10-2	データの挿入作業中にキャンセルボタンを押し、ETL 作業が中断されることを確かめる。また、中断したことを報告するポップアップが表示されることを確かめる。	キャンセルボタンが押された段階で ETL 作業を中断する。 データの挿入作業の途中で ETL 作業が中断したことを報告するポップアップが表示される。		合	2014/11/27

総合テスト報告書 S1

総合テスト報告書

目次

1. 目的

総合テストでは「GUI 画面検討書類」に基づき操作を行い、機能が実現されているか確かめます。

2. テスト環境

VMware Workstation 10.0.3 で本テストを行っている。テスト環境については下表に示します。

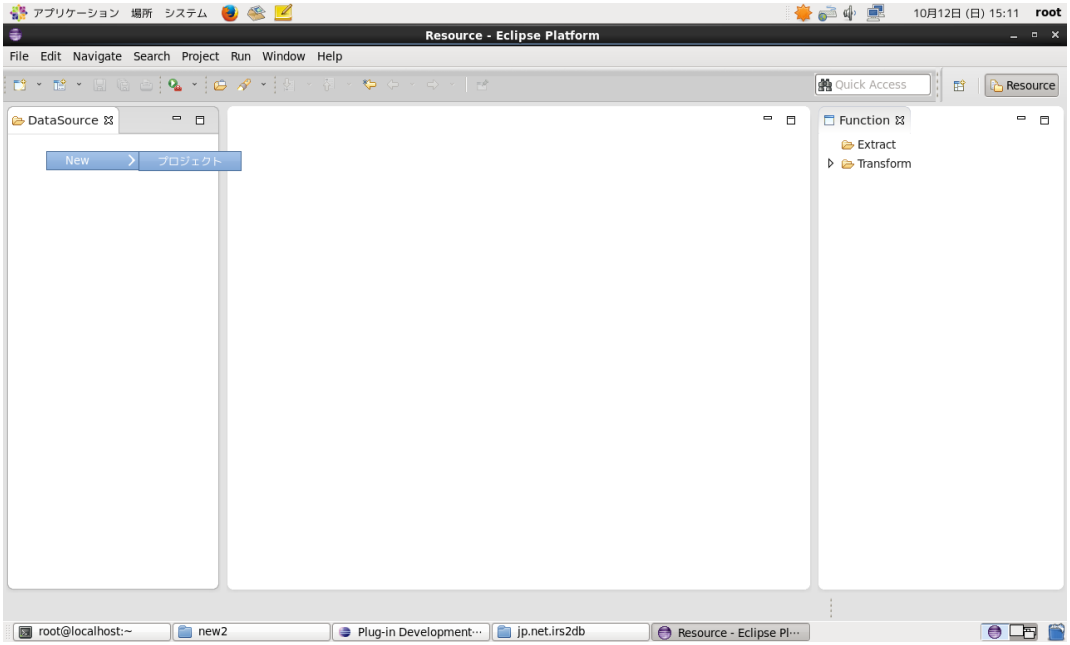
表 1 テスト環境

ハードウェア環境	
メモリ	2 GB
ハードディスク	40 GB
ネットワーク	ホストの IP アドレスを共有して使用
ソフトウェア環境	
OS	CentOS 6.5
データベース	InfoFrame Relational Store 3.1、PostgreSQL 8.4.20
その他	Eclipse 4.3.2

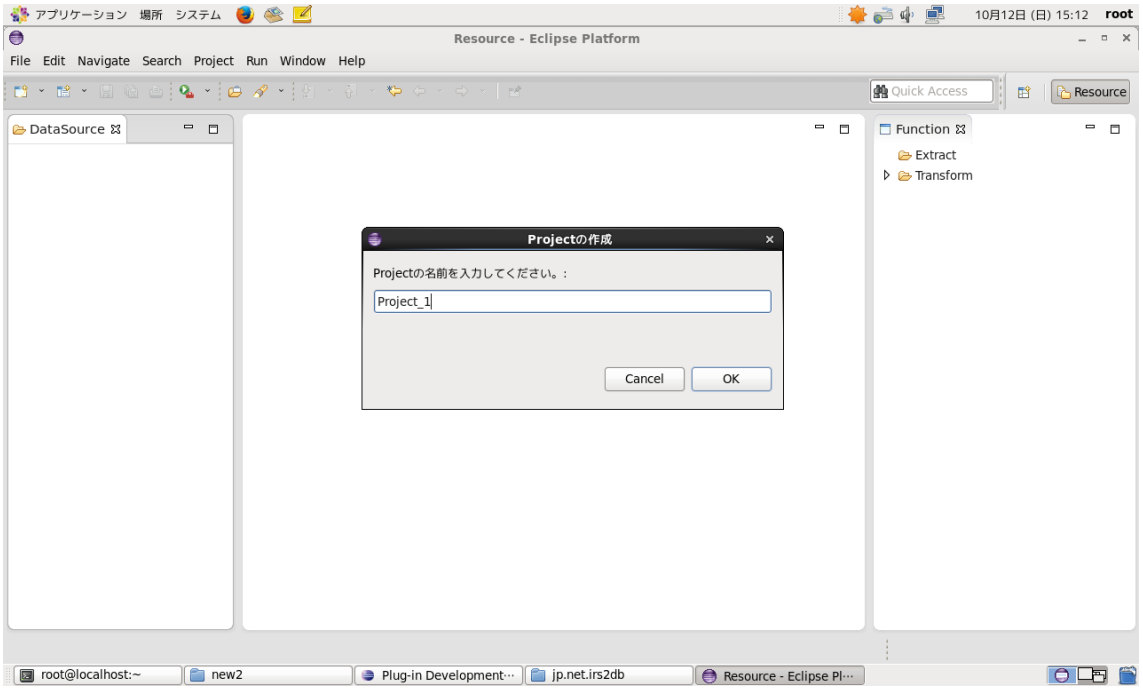
3. テスト操作と結果

3.1. プロジェクト作成

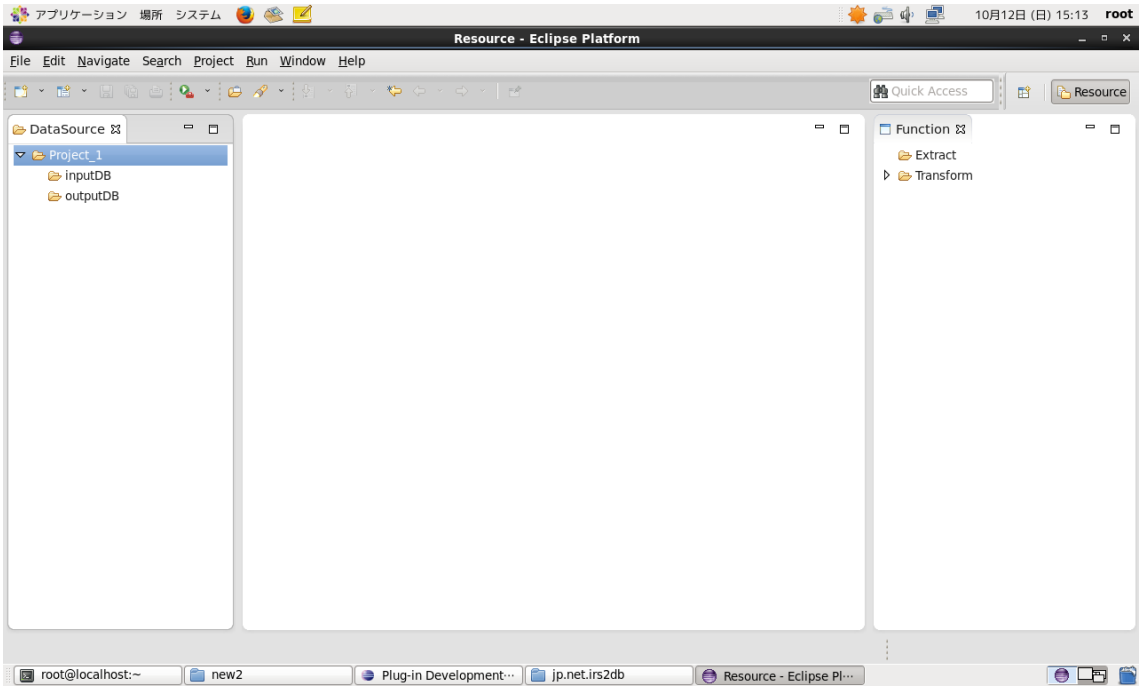
操作 1：DataSource ビューで右クリックして、プロジェクトを選択します。



操作 2：プロジェクト名入力画面で「Project_1」を入力して、「OK」ボタンを押します。

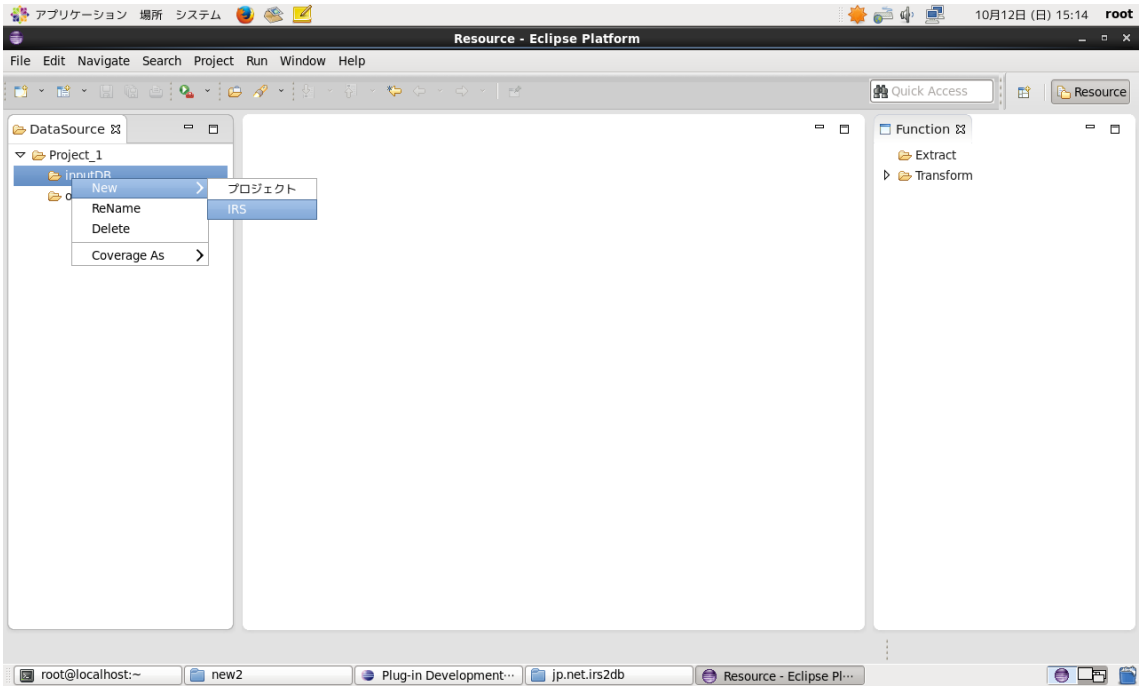


結果：「Project_1」というプロジェクトが新規作成され、「inputDB」と「outputDB」が表示されています。

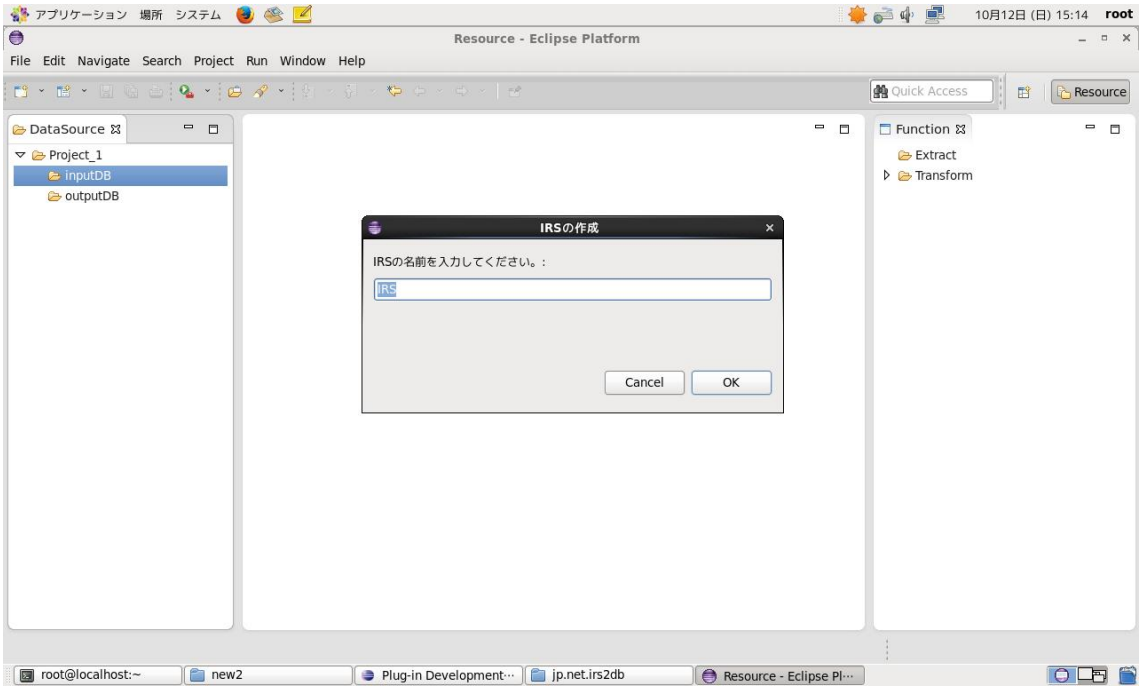


3.2. IRS 接続

操作 1：「inputDB」を右クリックして、「New」→「IRS」を選択します。



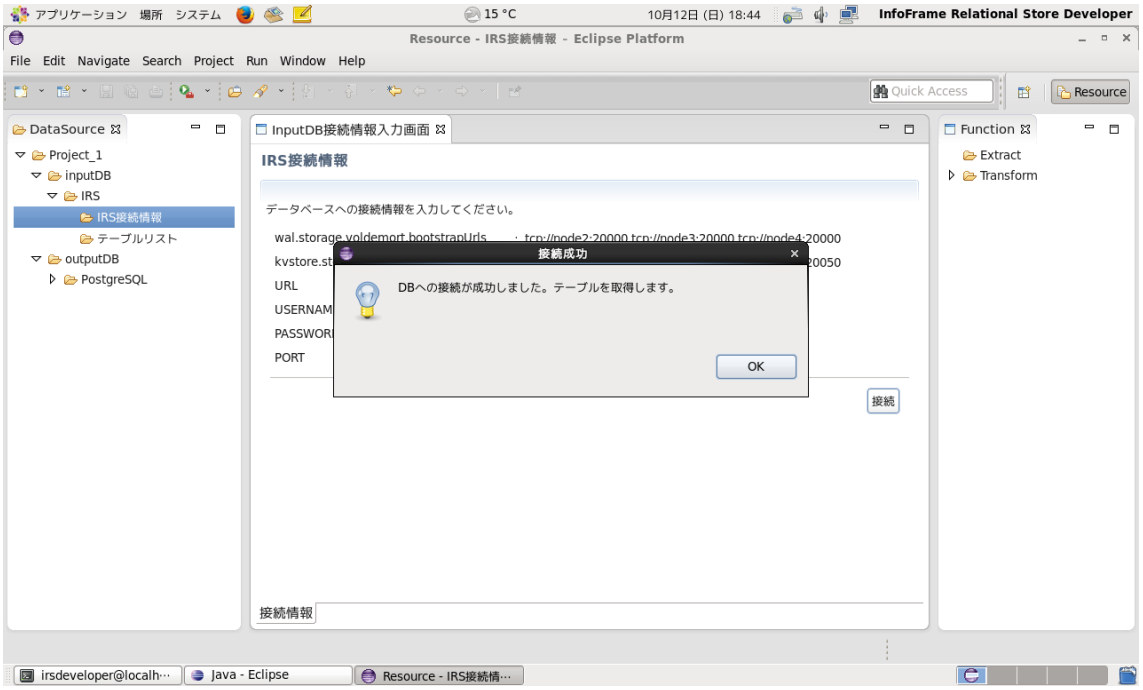
操作 2：「IRS 作成画面」で IRS の名前を入力して、「OK」ボタンを押します。



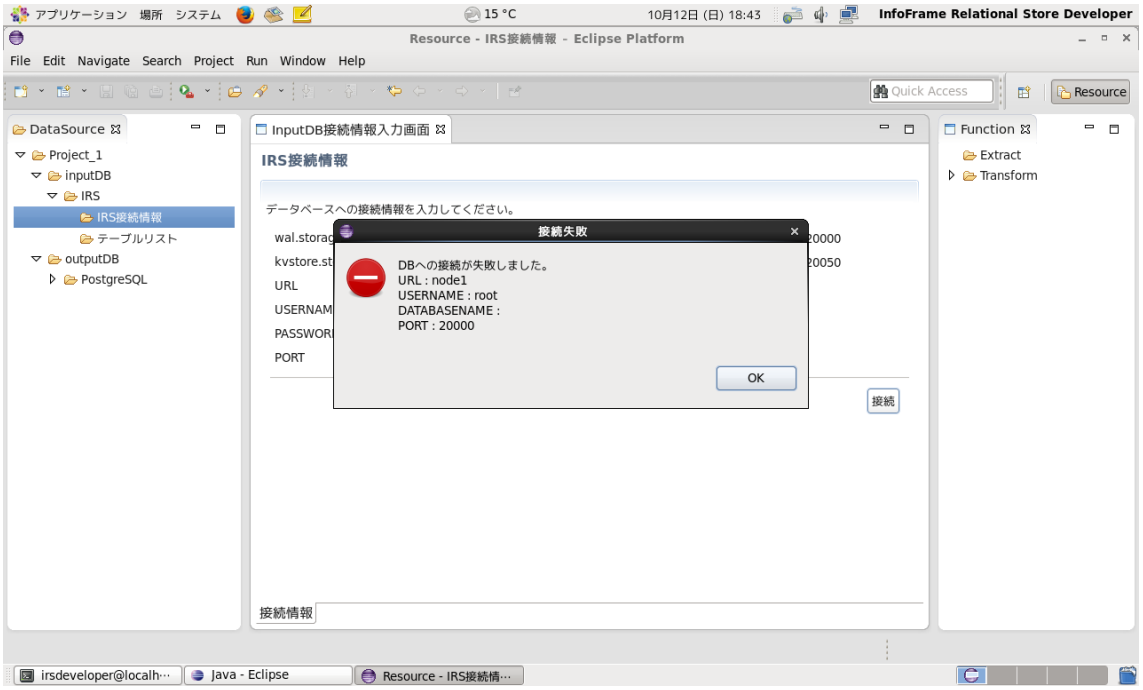
操作 3：作成した「IRS」を選択し、「IRS 接続情報」をダブルクリックします。

操作 4：「InputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押します。

結果 A：正しく接続できる場合、「接続成功」ポップアップが表示され、DataSource のテーブルリストに取得されたテーブルが表示されます。

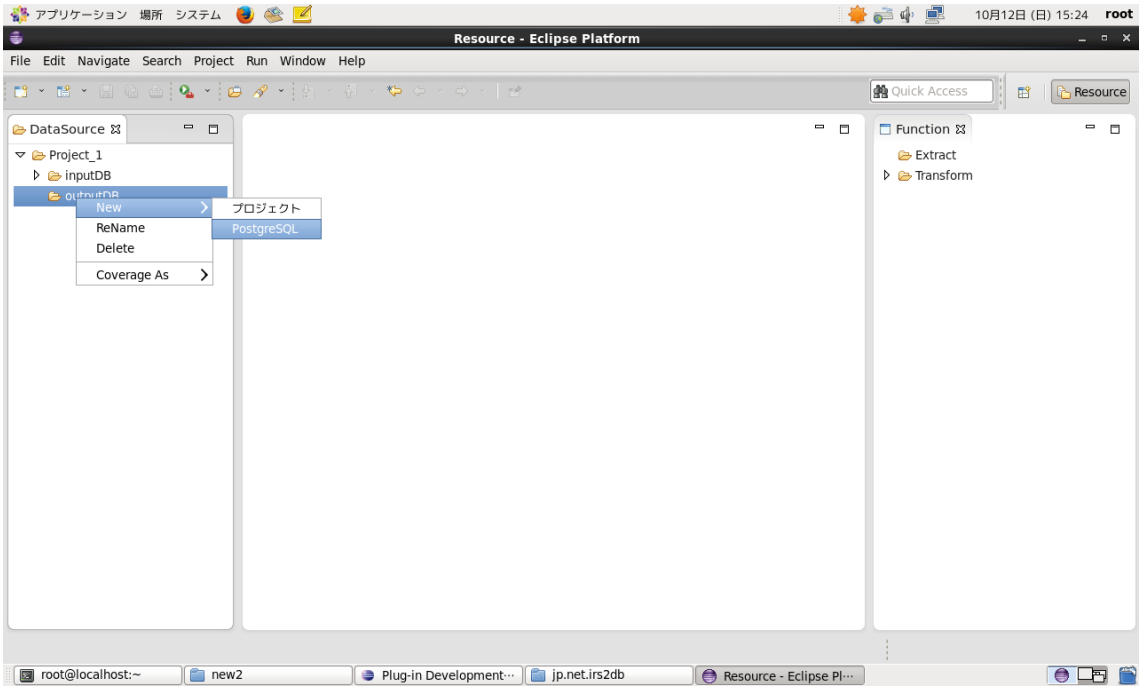


結果 B:接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示されます。

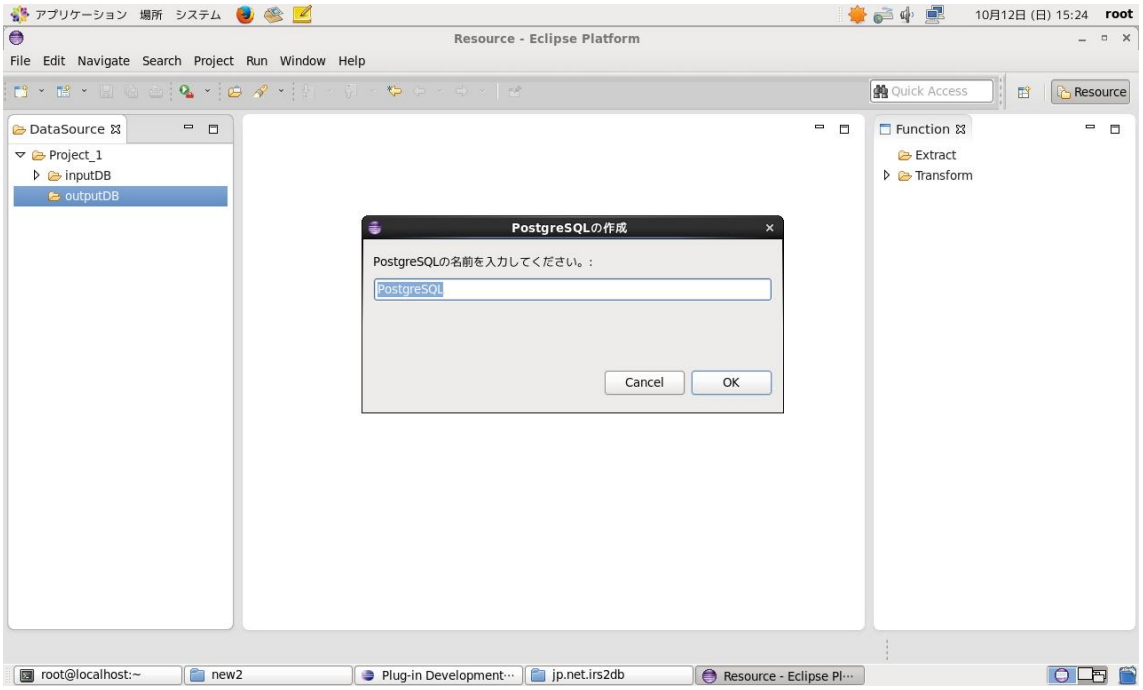


3.3. PostgreSQL 接続

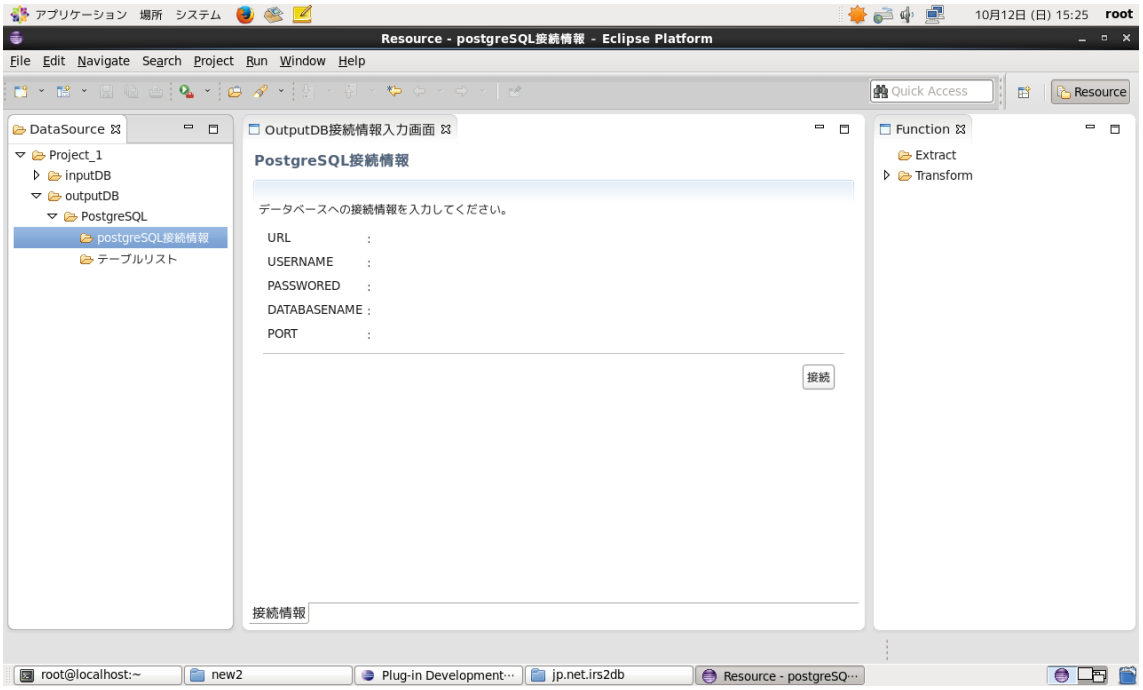
操作 1:「outputDB」を右クリックして、「New」→「PostgreSQL」を選択します。



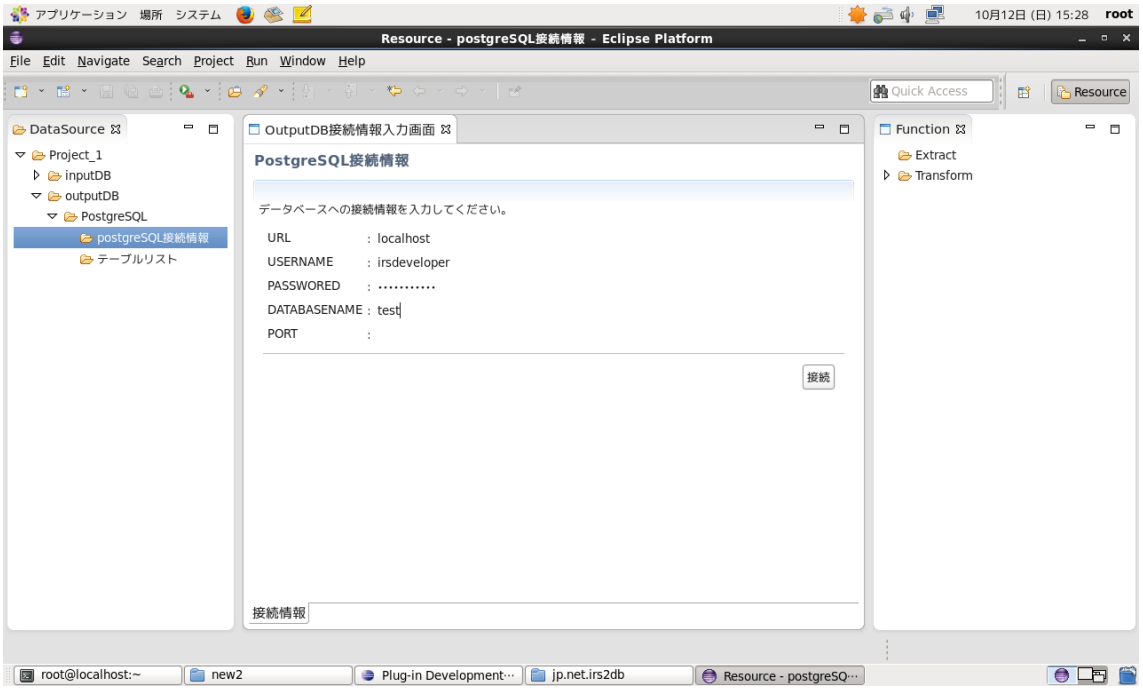
操作 2：「PostgreSQL 作成画面」で PostgreSQL の名前を入力して、「OK」ボタンを押します。



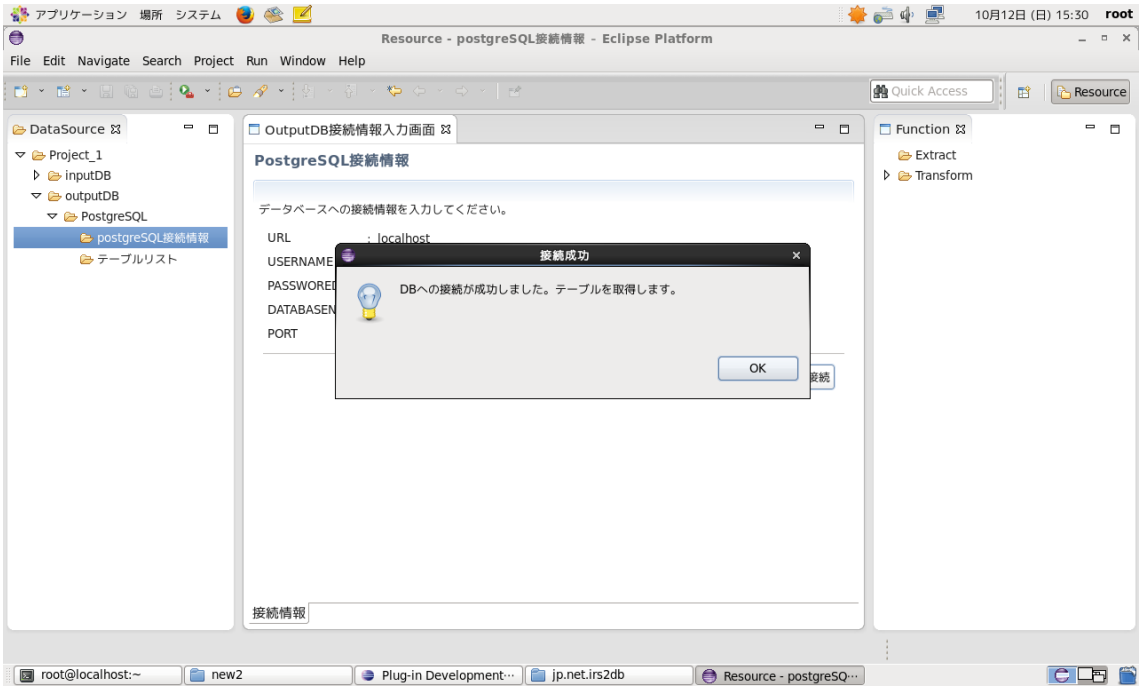
操作 3：作成した「PostgreSQL」を選択し、「postgreSQL 接続情報」をダブルクリックします。



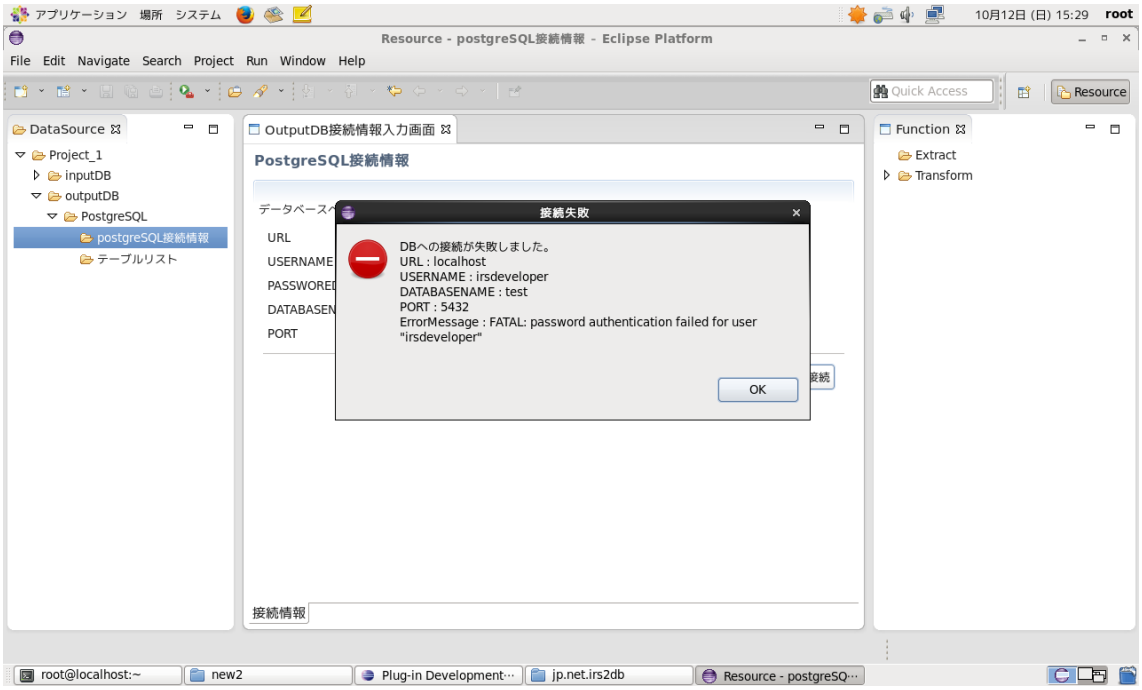
操作 4：「OutputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押します。



結果 A：正しく接続できる場合、「接続成功」ポップアップが表示され、DataSource のテーブルリストに取得されたテーブルが表示されます。

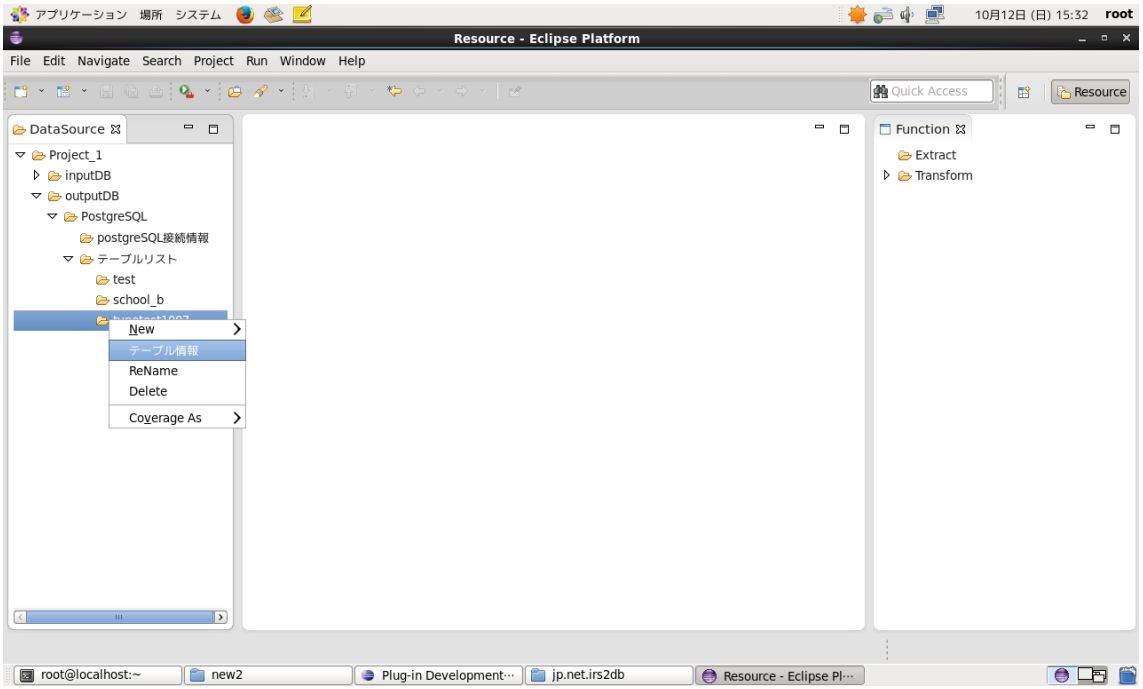


結果 B:接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示されます。

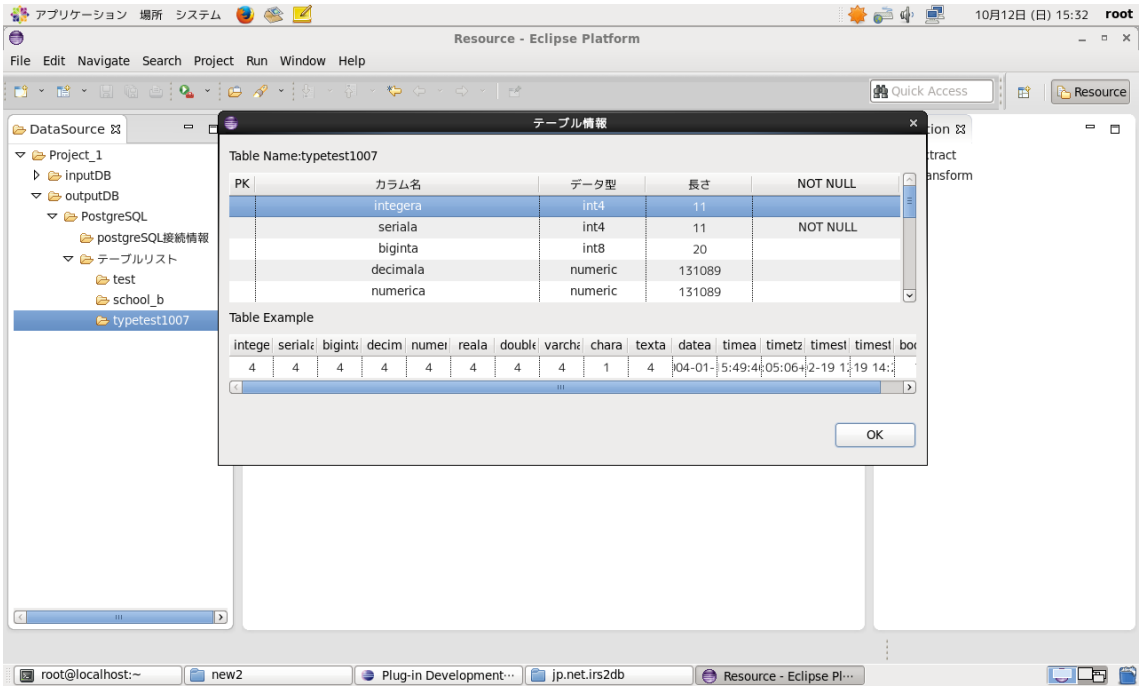


3.4. テーブル情報の表示

操作:テーブルリストにあるテーブル名を右クリックして、「テーブル情報」を選択します。

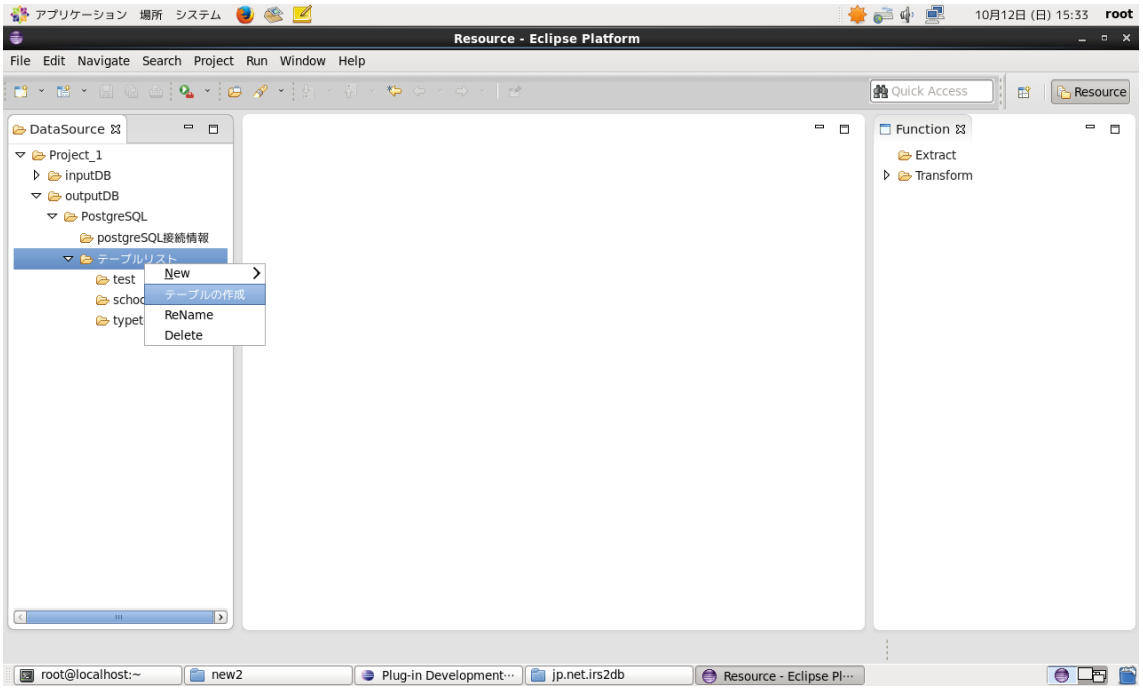


結果：テーブル情報が表示されます。

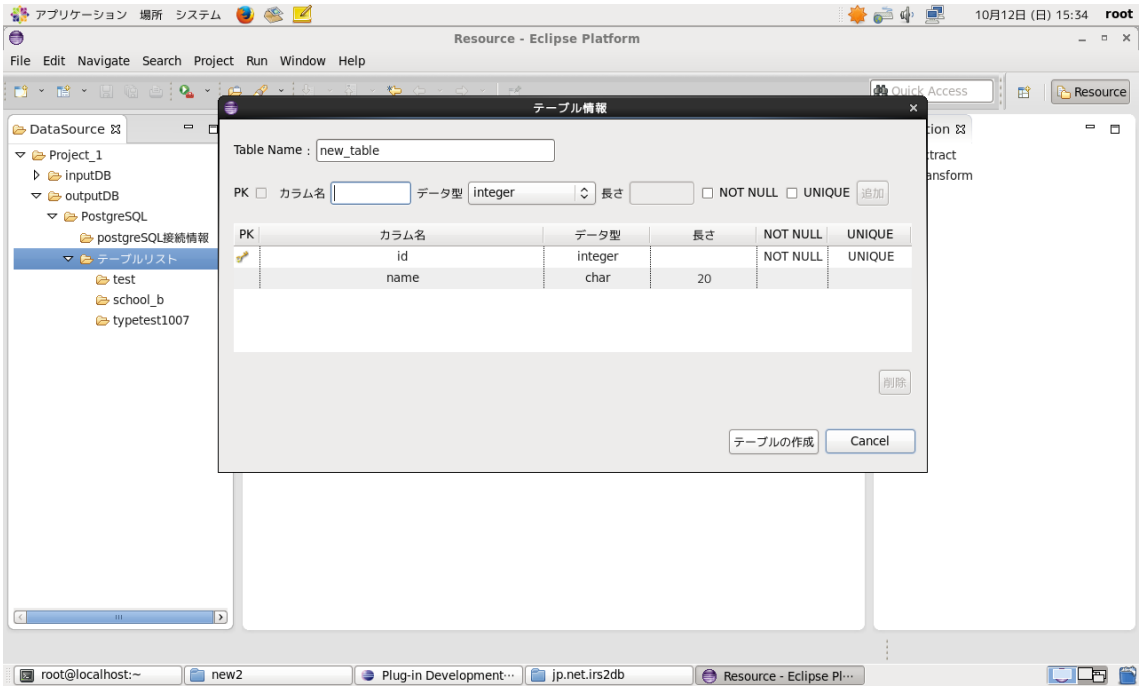


3.5. PostgreSQL に新規テーブルを作成

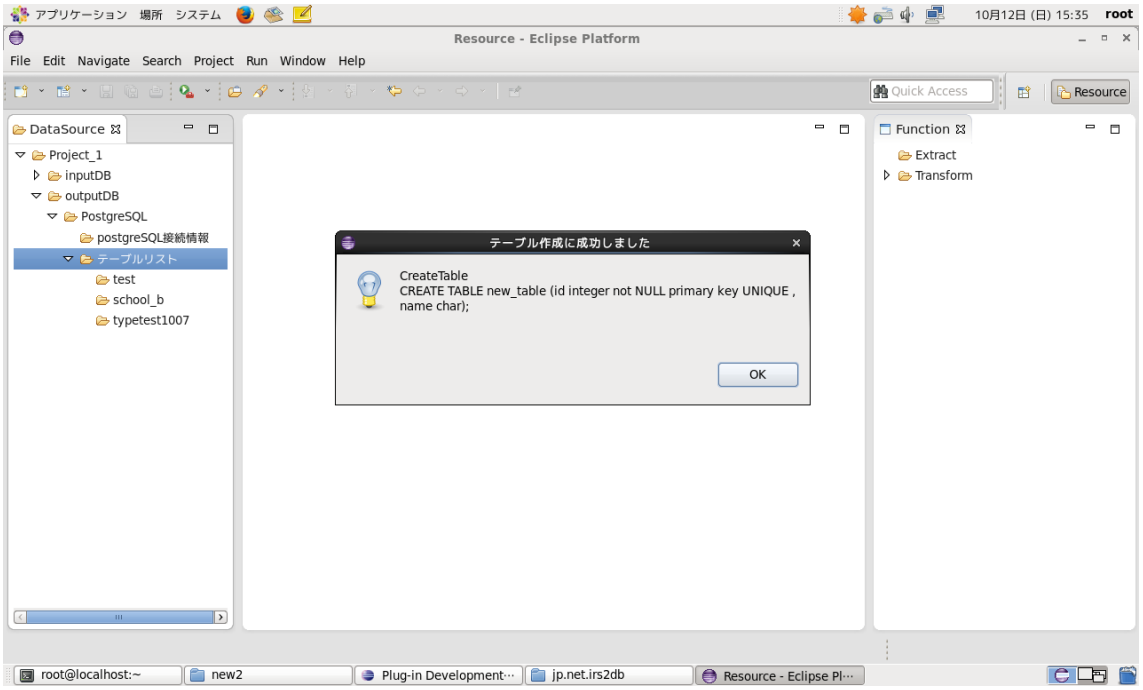
操作１：「テーブルリスト」を右クリックして、「テーブルの作成」を選択します。

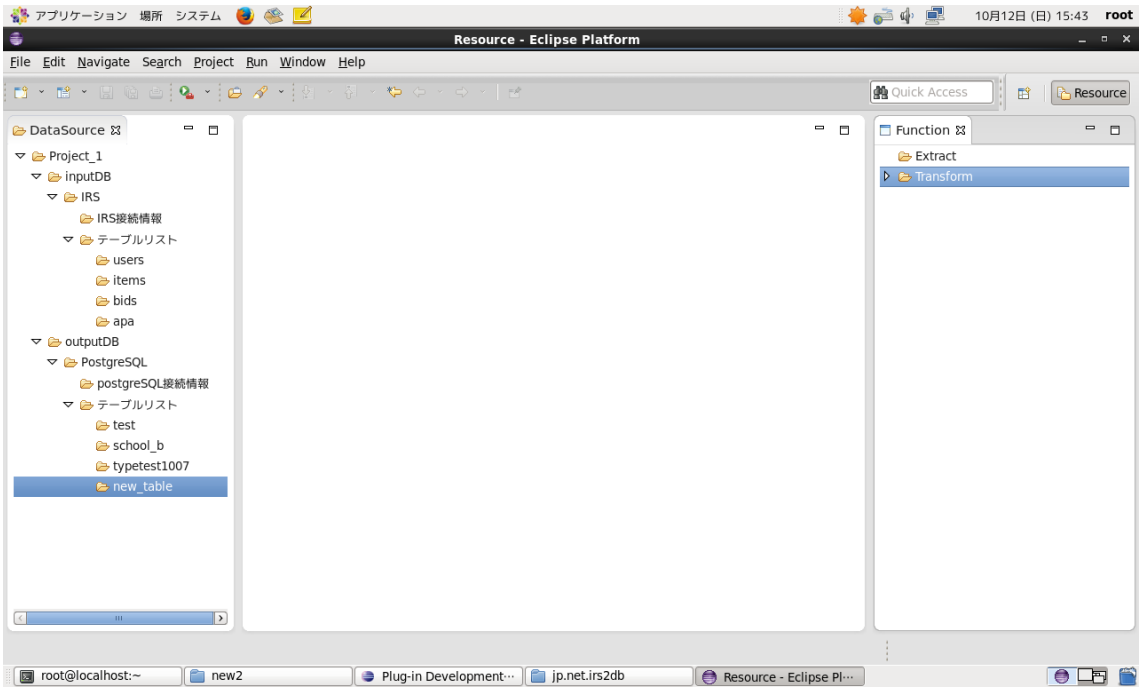


操作２：「テーブル情報」で作成するテーブルの情報を入力して、「テーブルの作成」ボタンを押します。

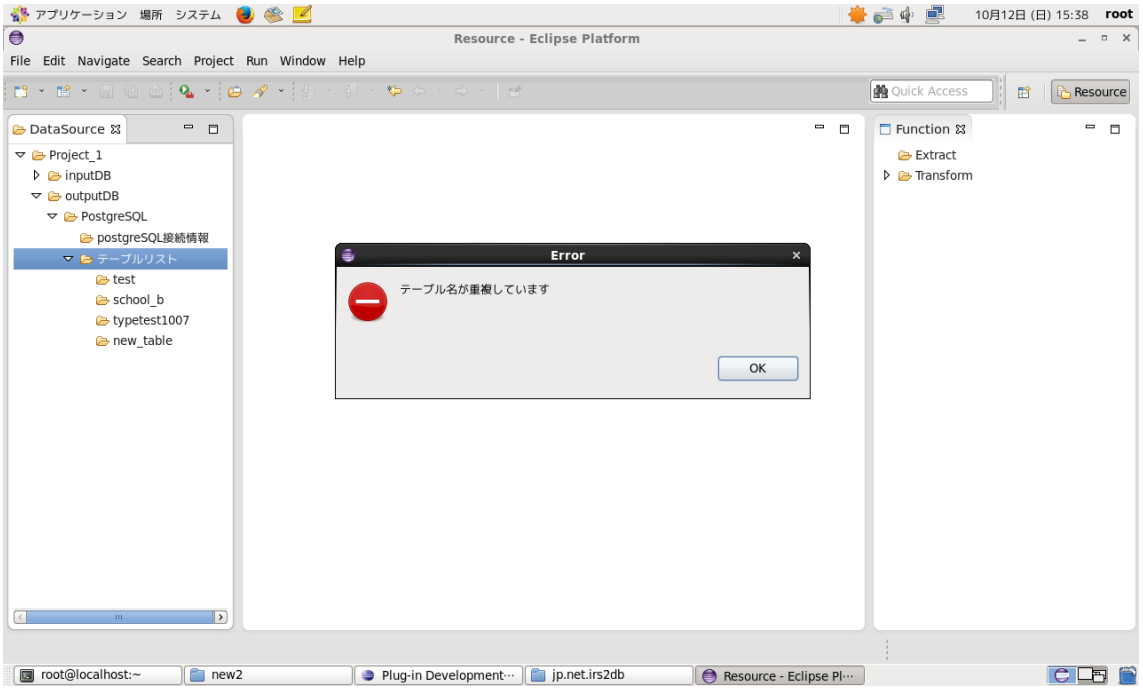


結果Ａ：正しく作成できた場合、「テーブルの作成に成功しました」ポップアップが表示されています。再接続すると、テーブルリストにテーブル名が追加されます。



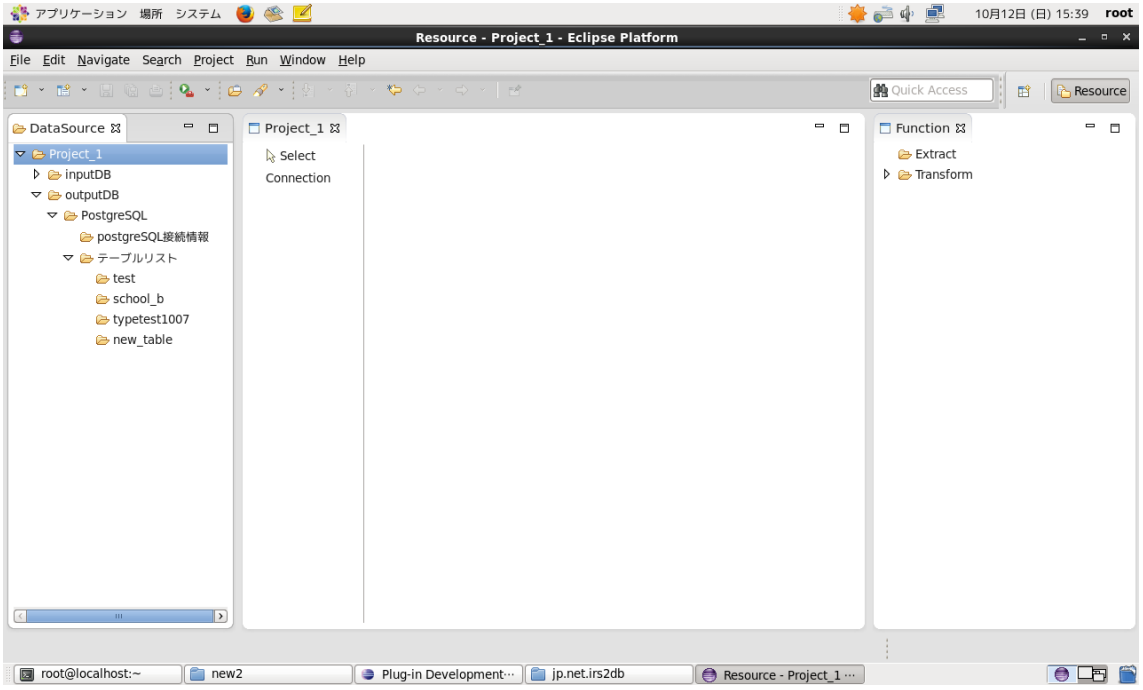


結果 B：デーブル作成が失敗した場合は、エラー画面が表示されます。



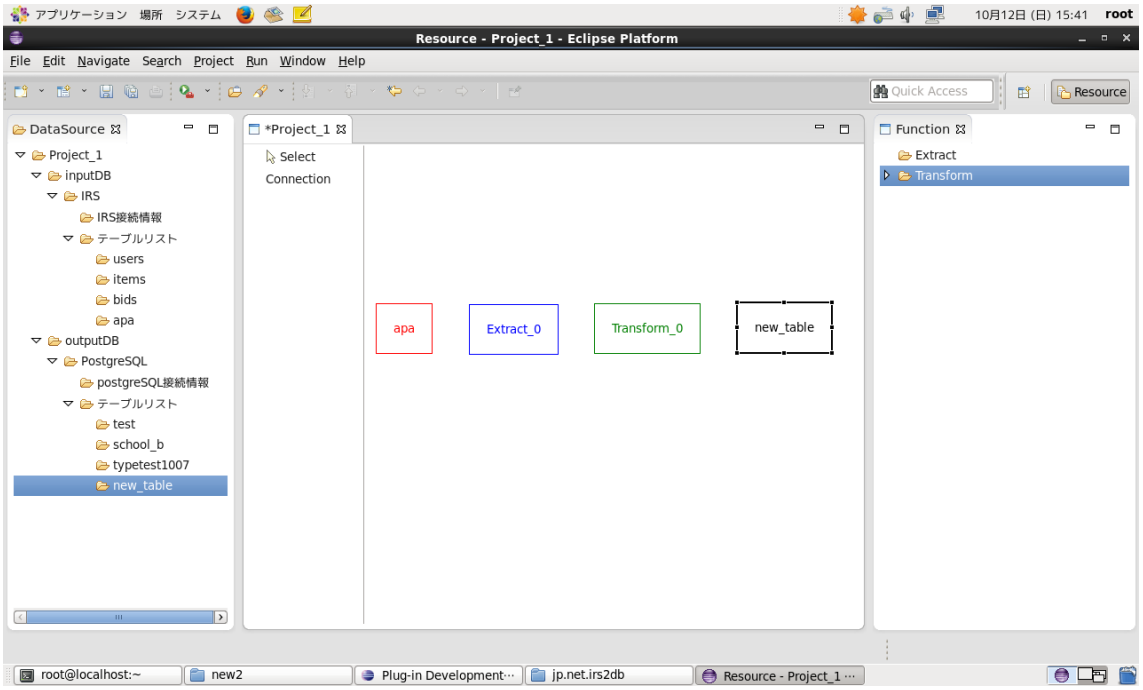
3.6. ETL 作業の編集

操作 1：プロジェクト名をダブルクリックすることで、作業ウィンドウを開きます。



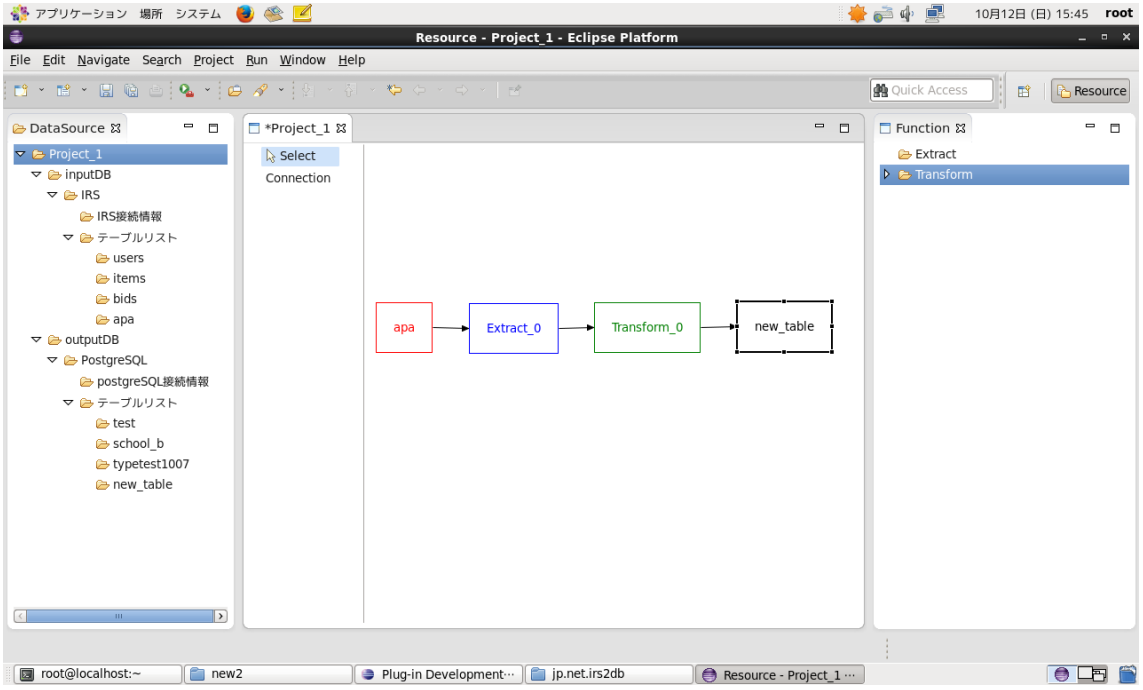
操作 2：使用するテーブルやファンクションなどを作業ウィンドウにドラグします。

結果：作業ウィンドウにテーブルと各ファンクションのアイコンが表示されています。

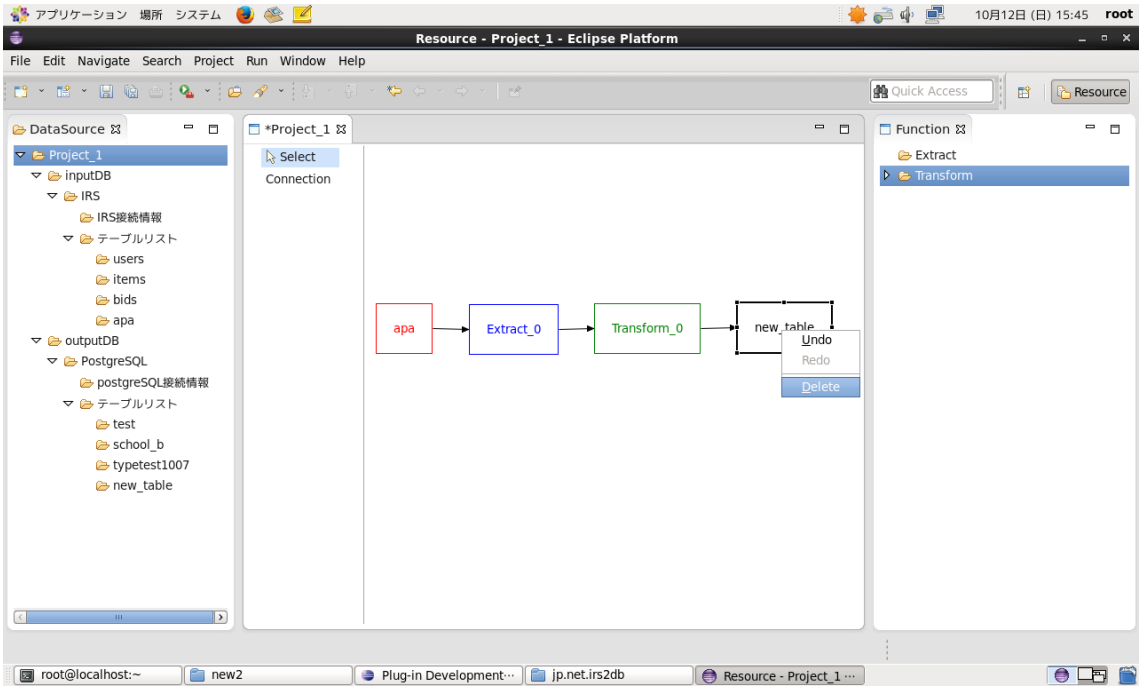


操作 3：左側の「Connection」を選択して、作業ウィンドウ内のアイコンを紐付けします。

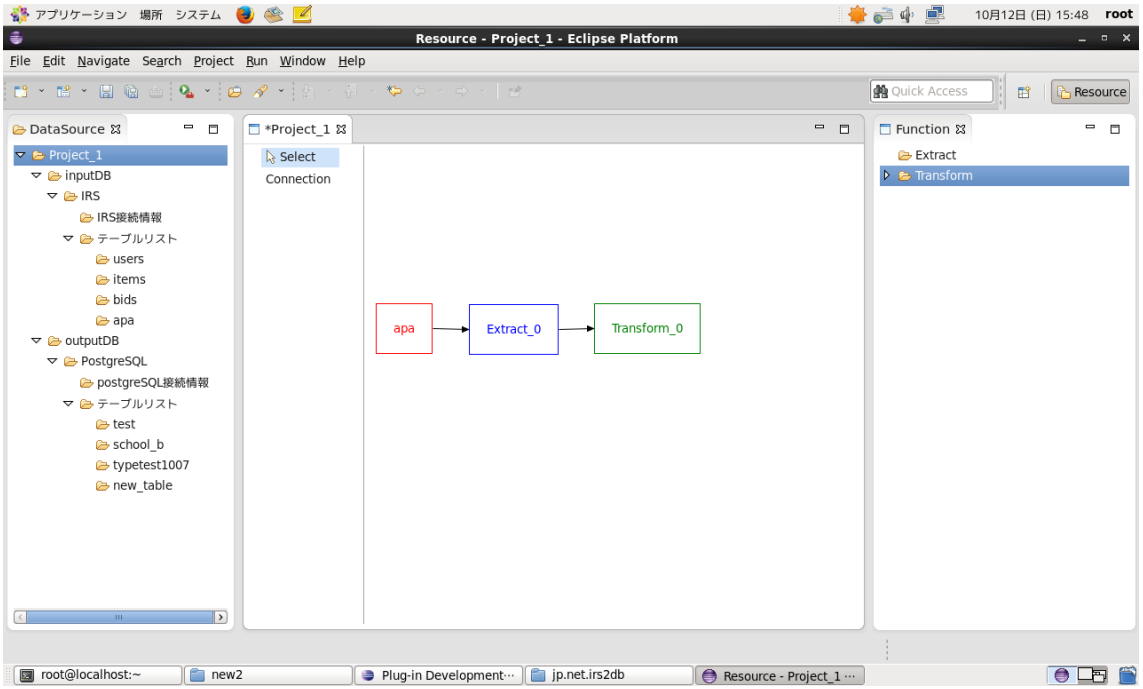
結果：テーブルと各ファンクションの対応関係が指定されています。



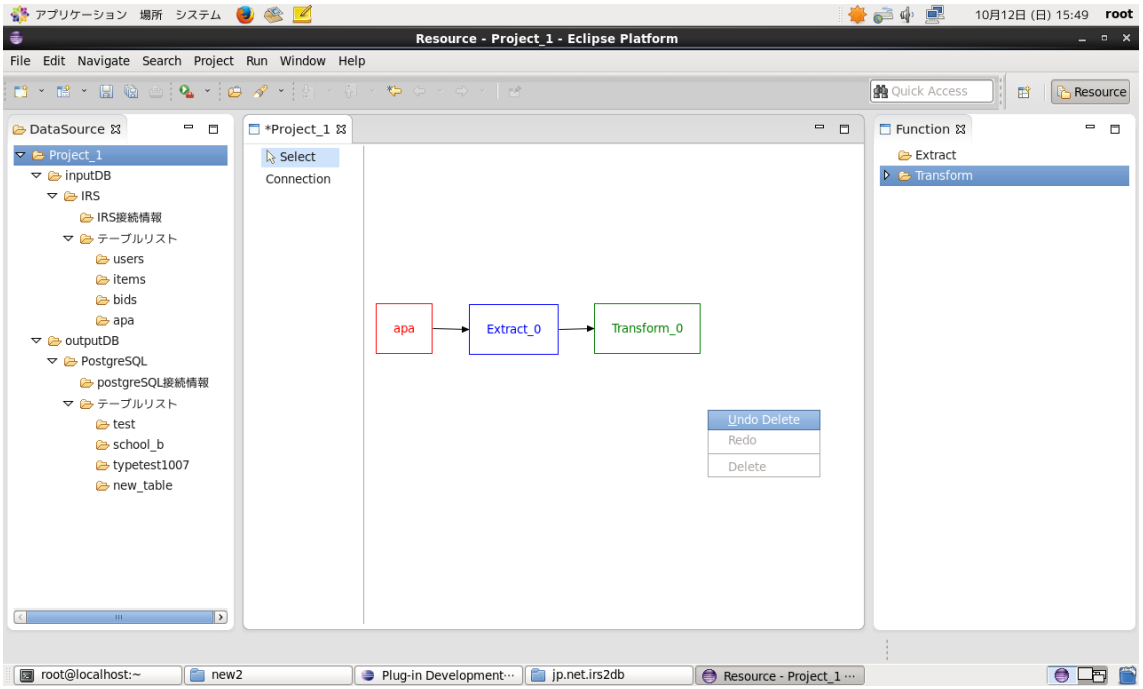
操作 4：アイコンを右クリックして、「Delete」を選択します。



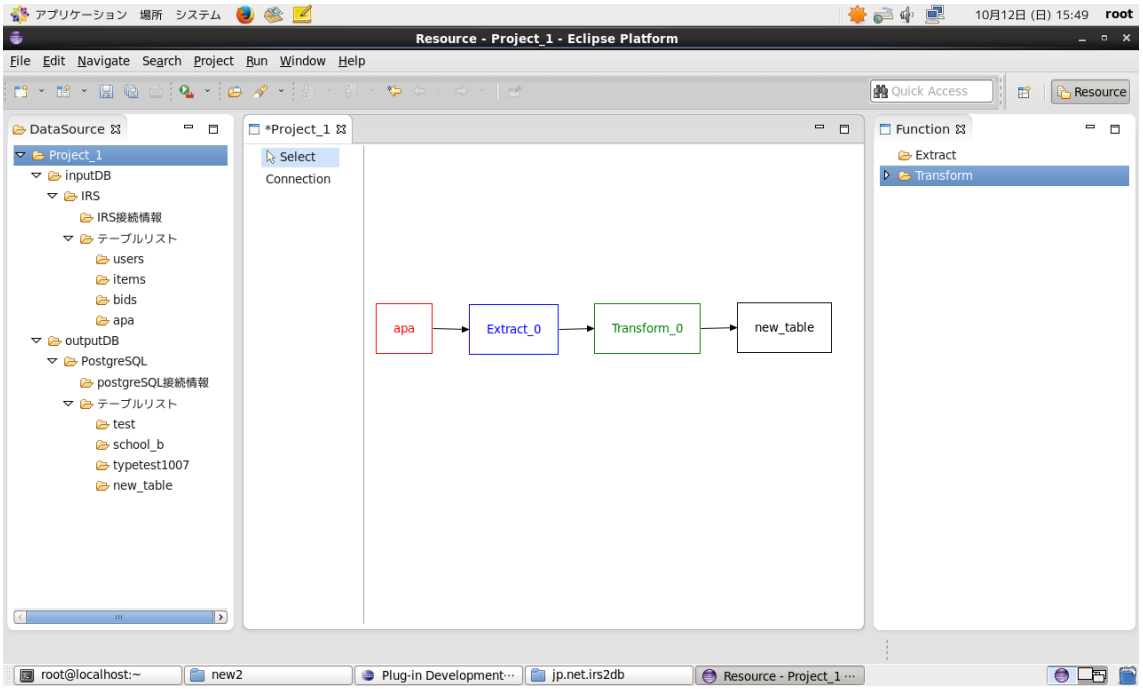
結果：該当アイコンが削除されました。



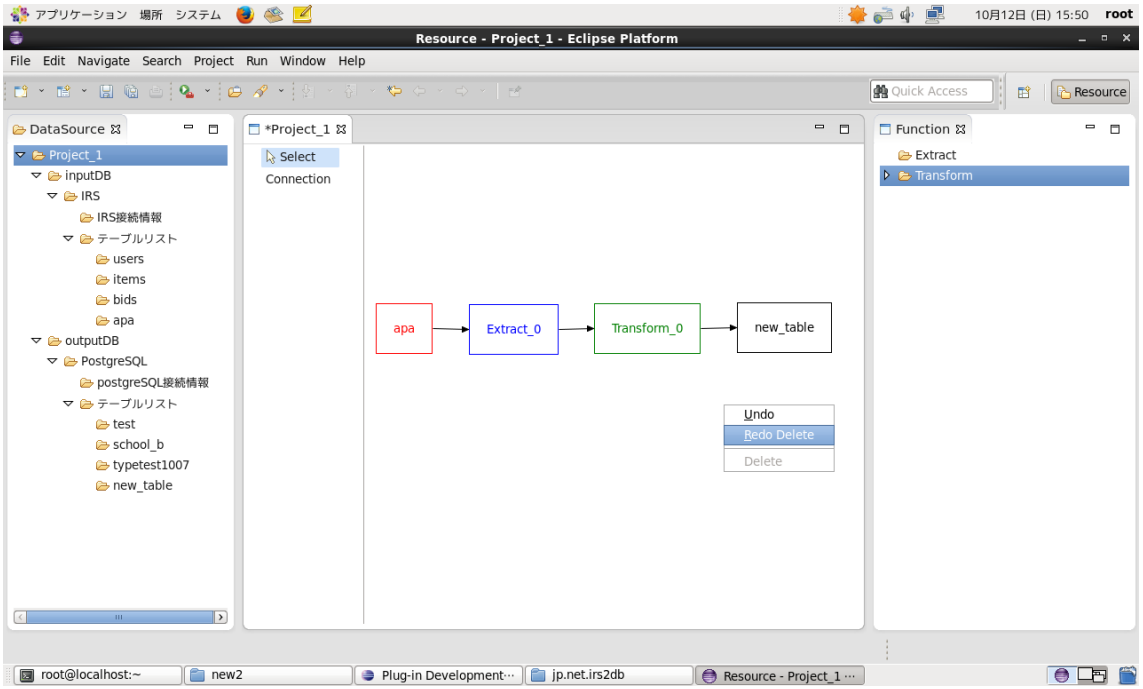
操作 5：作業ウィンドウを右クリックして、「Undo」を選択します。



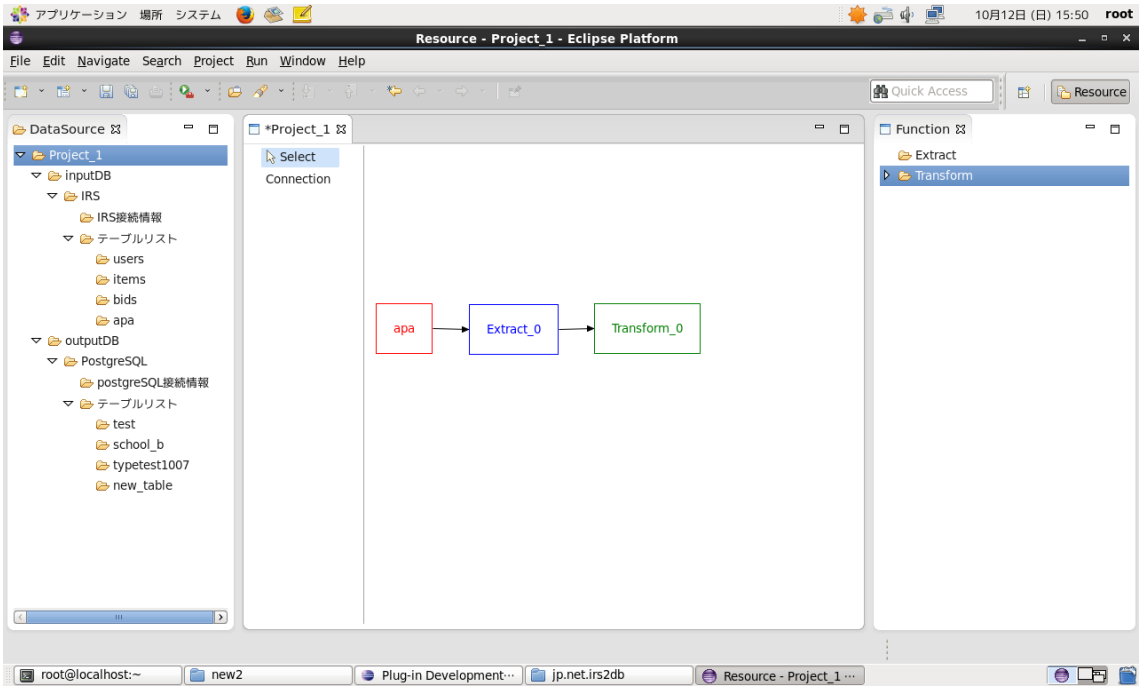
結果：削除したアイコンが回復されました。



操作 6：作業ウィンドウを右クリックして、「Redo」を選択します。



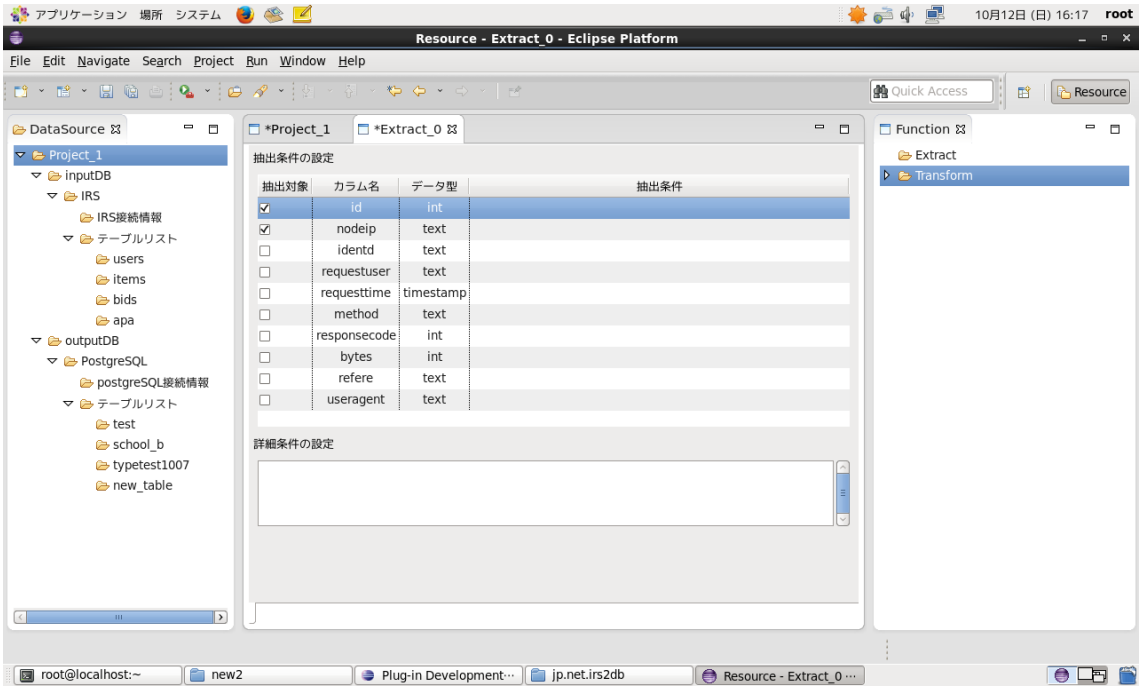
結果：回復されたアイコンはもう一度削除されています。



3.7. Extract 作業の編集

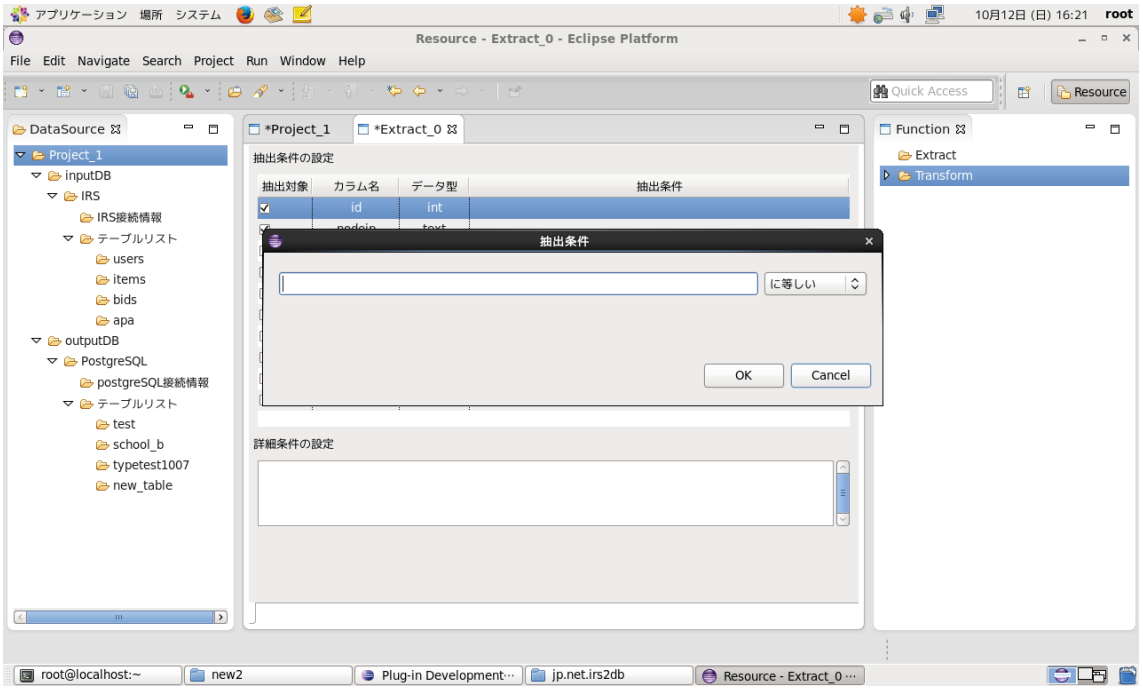
操作 1：Extract アイコンをダブルクリックします。

結果：Extract 条件編集画面が表示され、Extract 条件の編集・保存ができます。

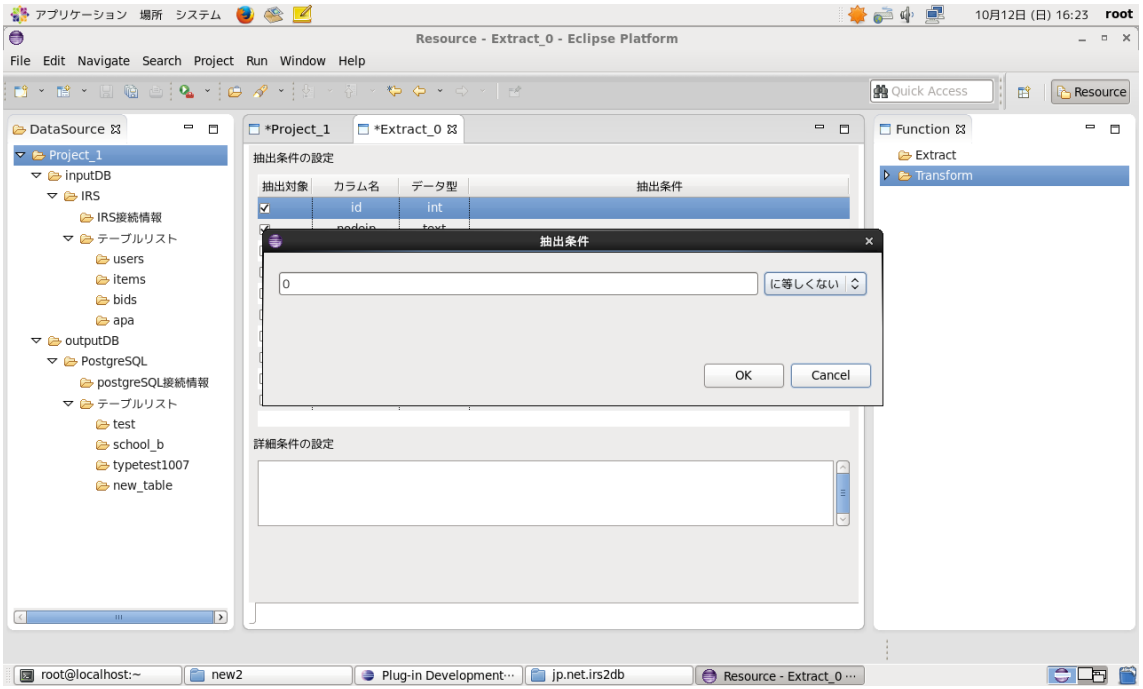


操作２：カラムの「抽出条件」をクリックします。

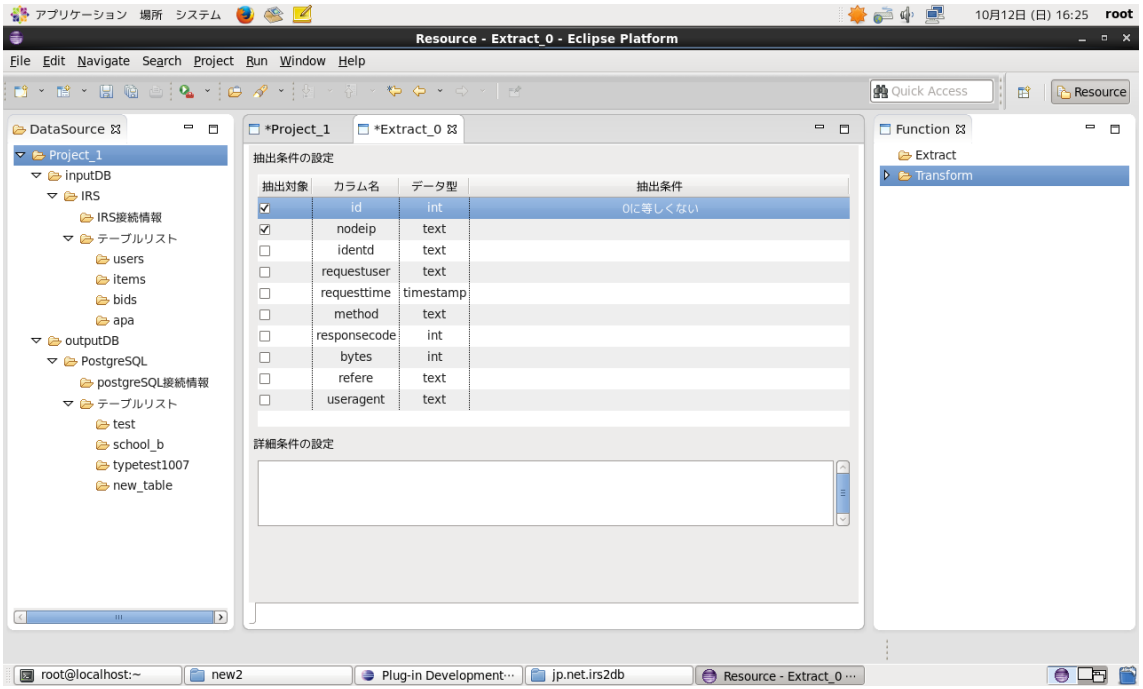
結果：抽出条件編集画面が表示されます。



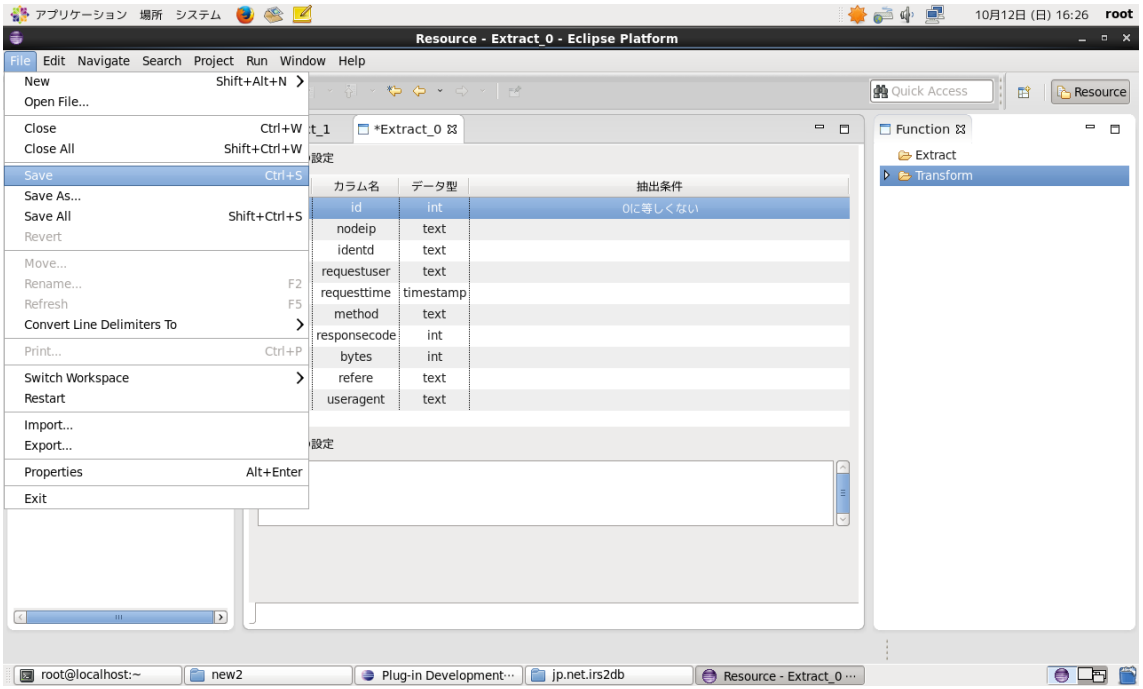
操作３：抽出条件を入力して、「OK」ボタンを押します。



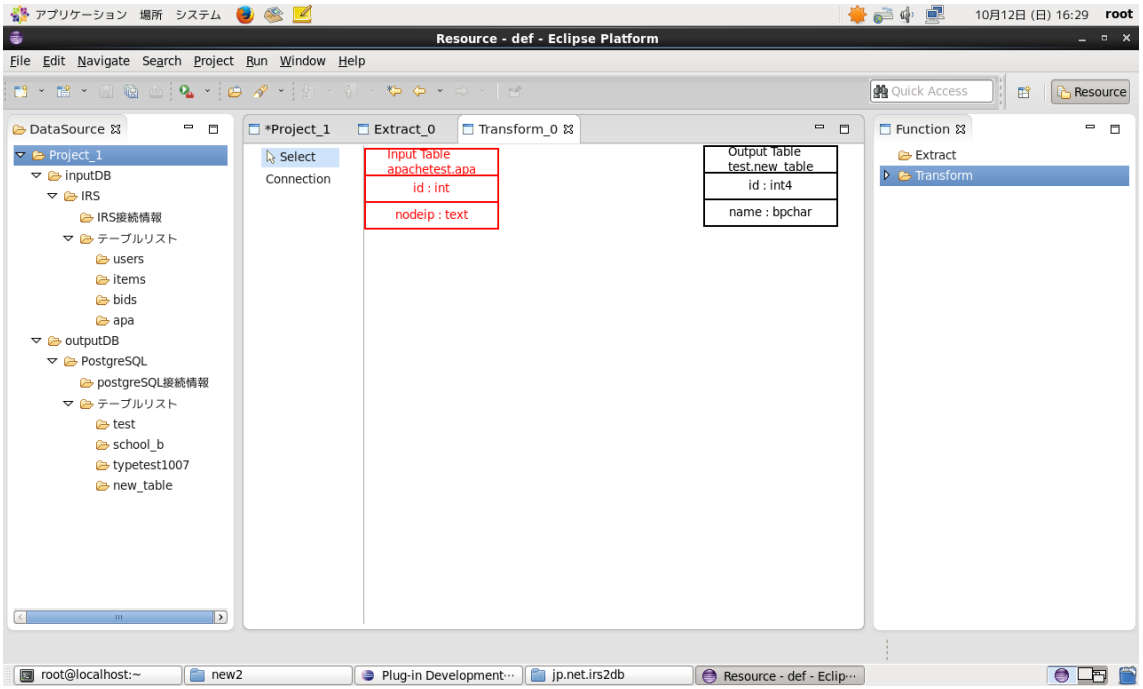
結果：Extract 編集画面に該当カラムの抽出条件が表示されます。



操作 4：Extract 条件を保存して、作業ウィンドウで Transform アイコンをダブルクリックします。



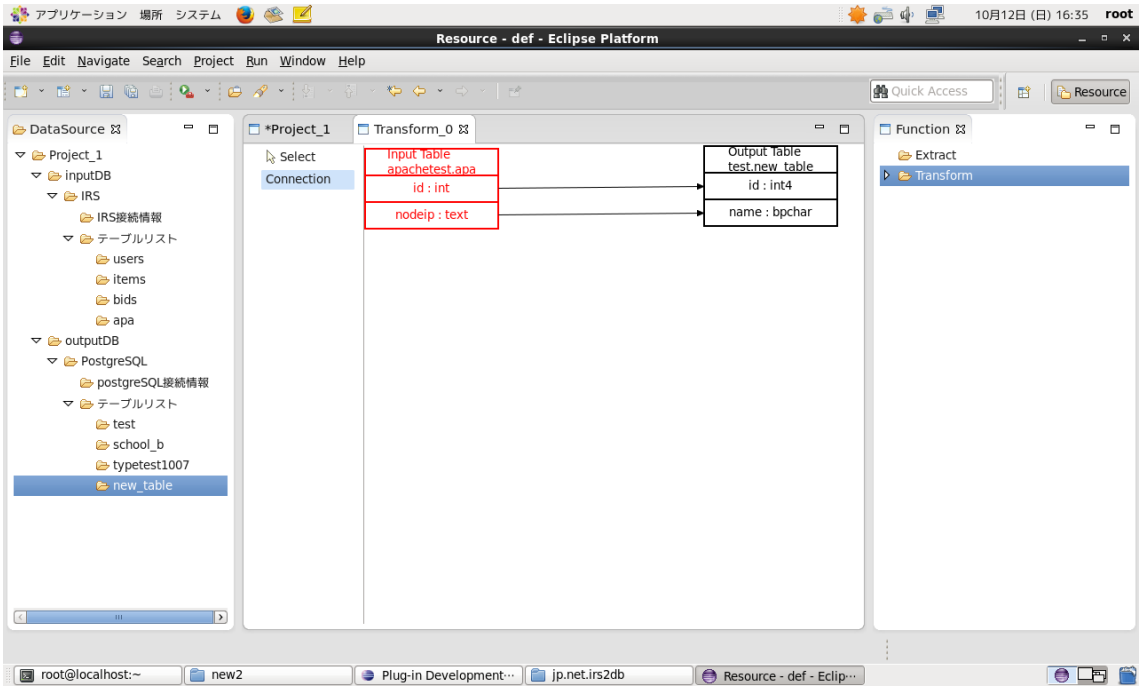
結果：Transform 編集画面に抽出後 Input Table が表示されています。



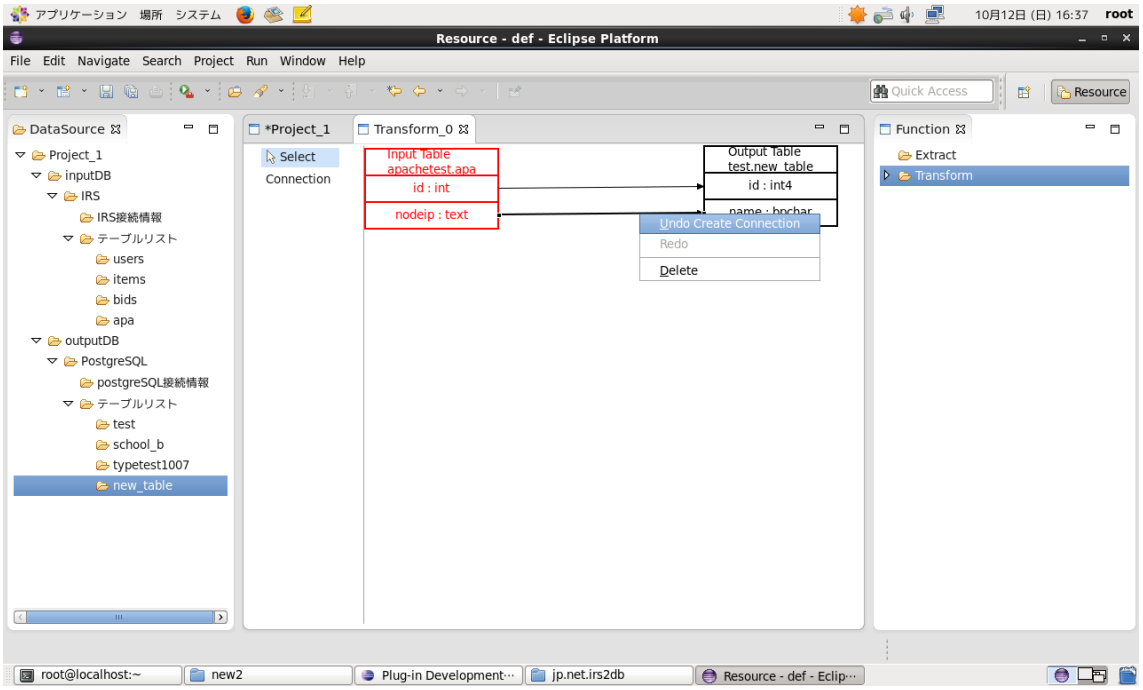
3.8. Transform 作業の編集

操作 1：Transform 編集画面を開いて、入力カラムと出力カラムの対応関係を紐付けします。

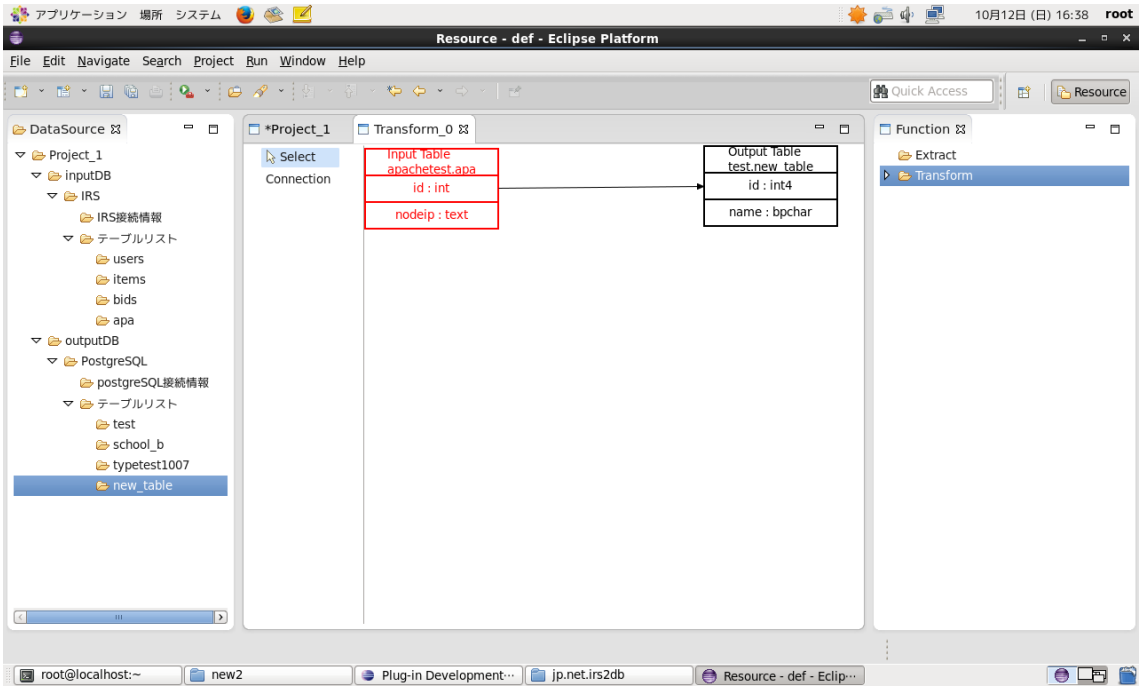
結果：対応関係を指定している線が表示されています。



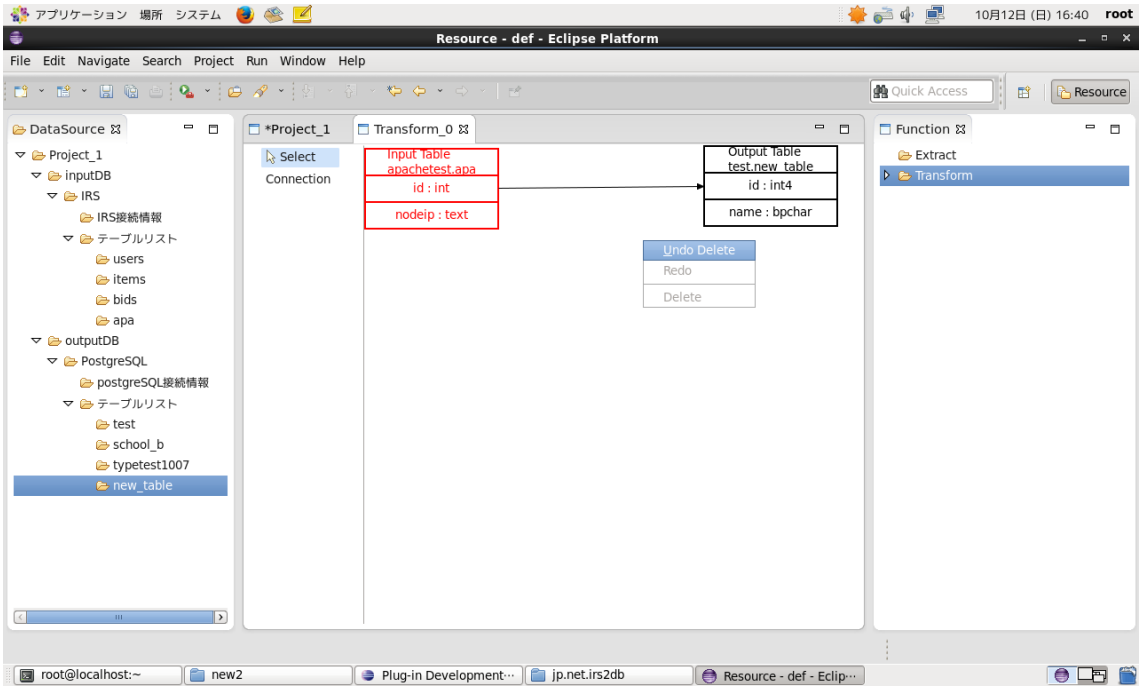
操作 2：線を右クリックして、「Delete」を選択します。



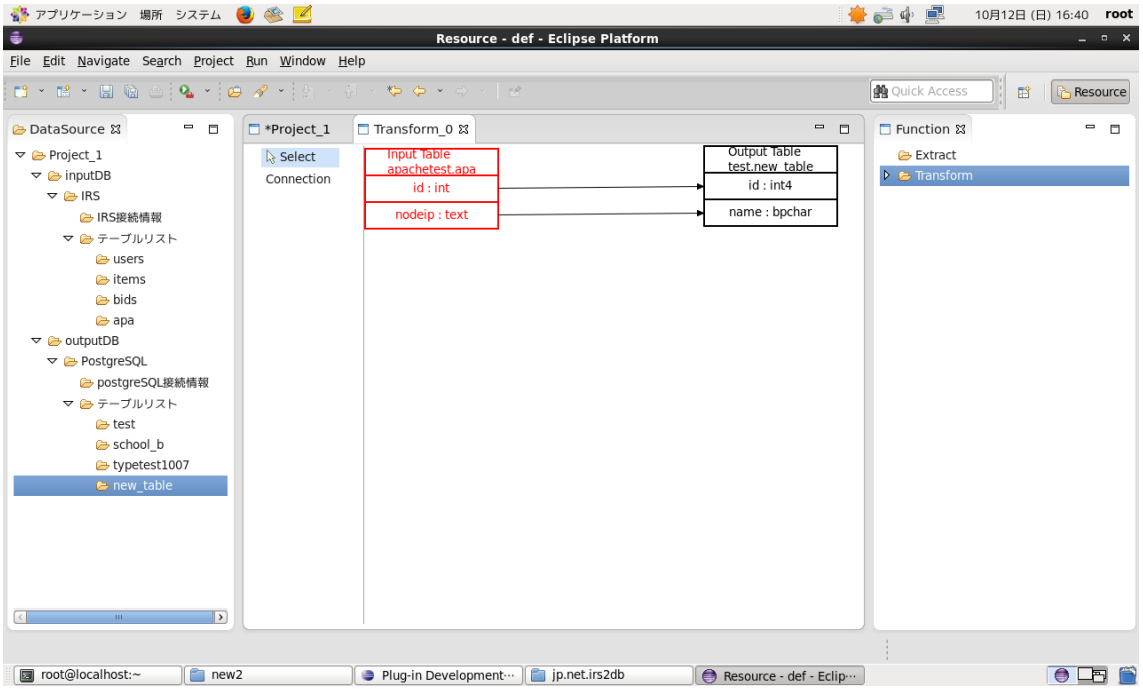
結果：選択されている線が消されました。



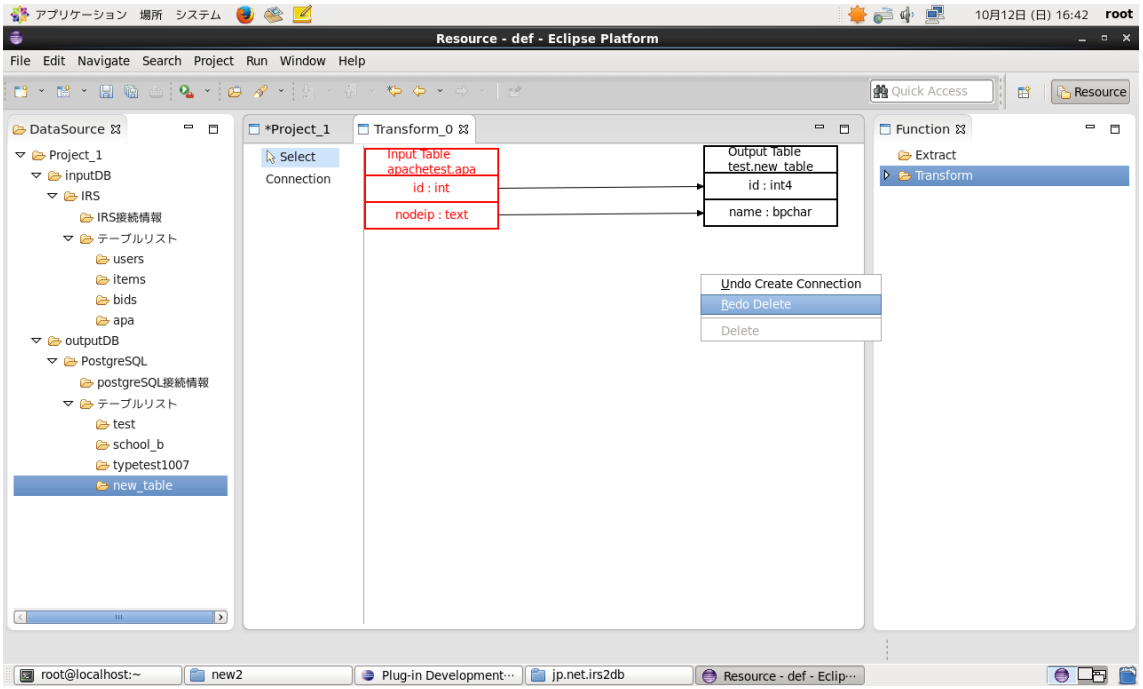
操作 3：Transform 編集画面を右クリックして、「Undo」を選択します。



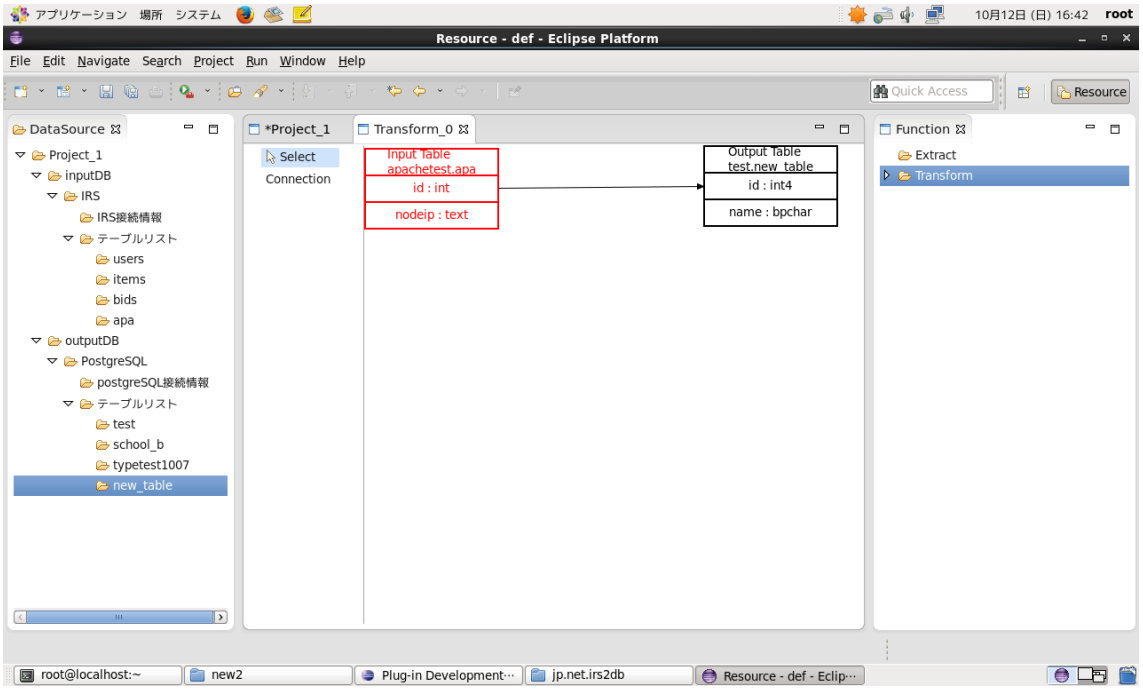
結果：削除された線が回復されました。



操作 4：Transform 編集画面を右クリックして、「Redo」を選択します。



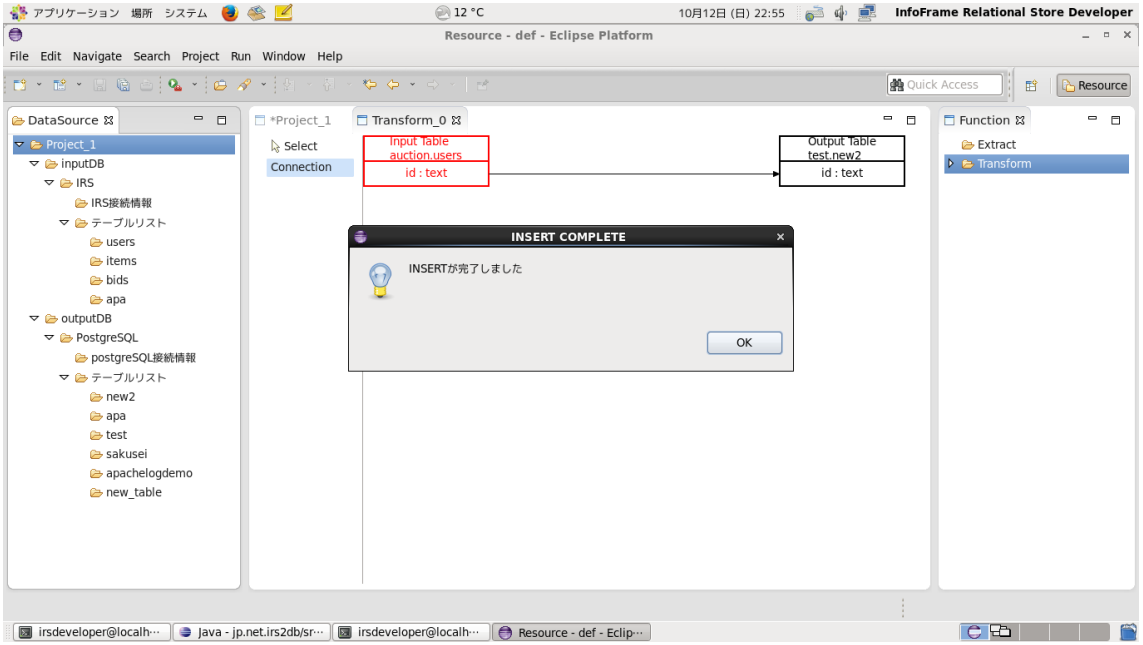
結果：回復された線がもう一度消されました。



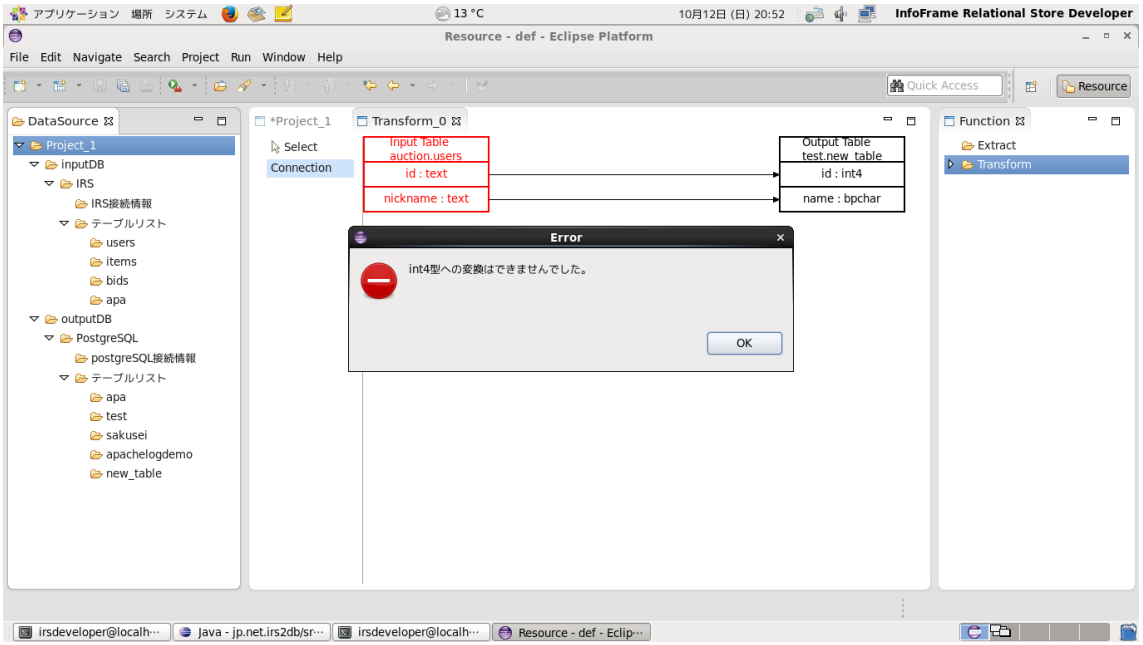
3.9. ETL の実行

操作：プロジェクト名を右クリックして、「実行」を選択します。

結果 A：実行が成功した場合、「Insert が完了しました」ポップアップが表示されます。



結果 B：実行（text から int4 に変換）が失敗した場合、エラーメッセージのポップアップが表示されます。


InfoFrame Relational Store Developer
Resource - def - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Resource
DataSource
Project_1
inputDB
IRS
IRS接続情報
テーブルリスト
users
items
bids
apa
outputDB
PostgreSQL
postgreSQL接続情報
テーブルリスト
apa
test
sakusei
apachelogdemo
new_table
*Project_1
Transform_0
Select
Connection
Input Table
auction.users
id : text
nickname : text
Output Table
test.new_table
id : int4
name : bpchar
Function
Extract
Transform
Error
int4型への変換はできませんでした。
OK
irsdeveloper@localh…
Java - jp.net.irs2db/sr…
irsdeveloper@localh…
Resource - def - Eclip…

ユーザズマニュアル

内容

1. プロジェクトを作成する

1. DataSource ビュー内で右クリックするとポップアップが表示されます。

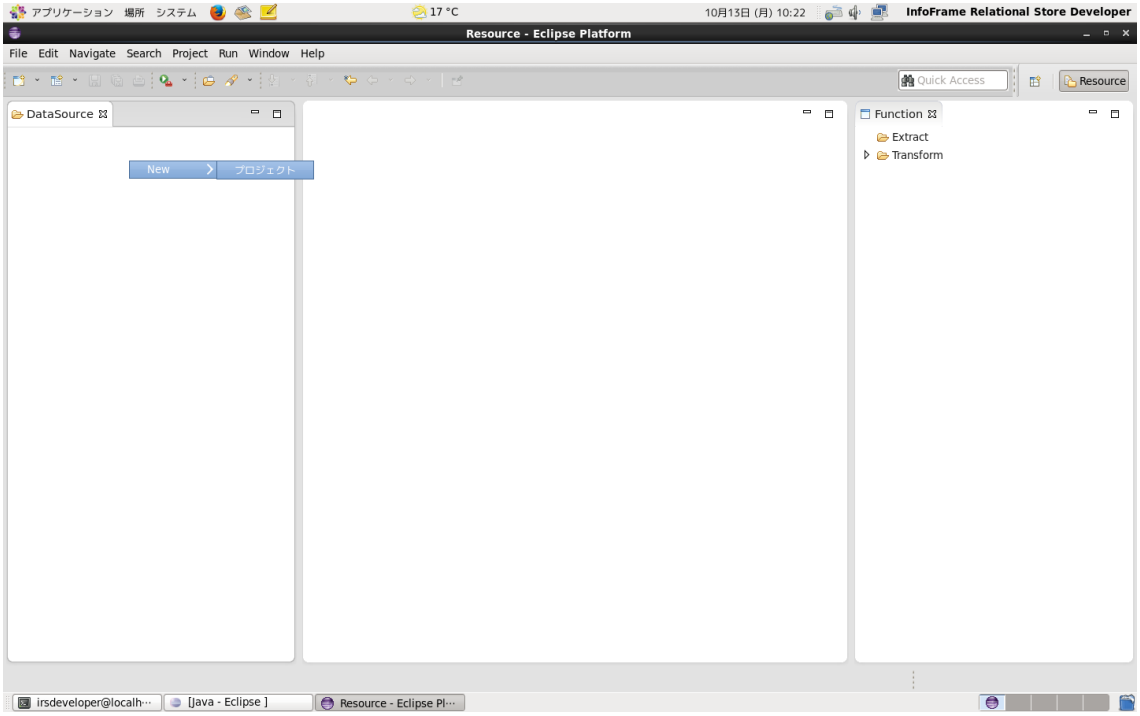


図 1　データソースウィンドウ内で右クリックする

2. 「New」→「プロジェクト」と選択することでプロジェクトを作成するダイアログが表示されます。

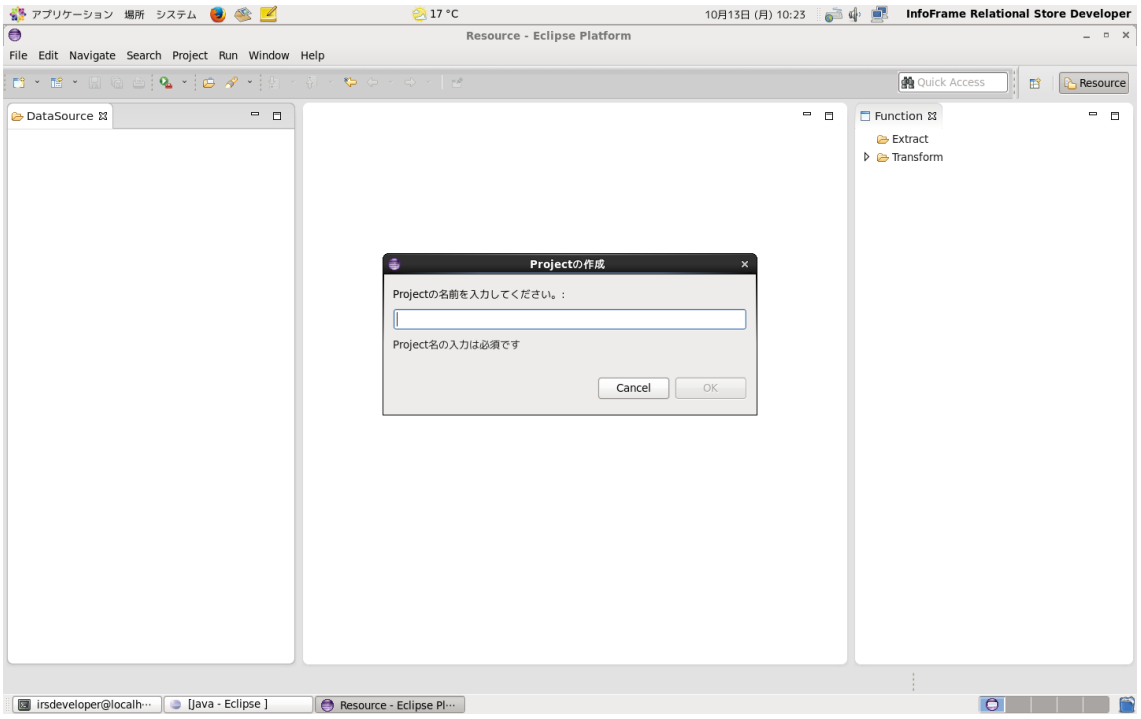


図 2　プロジェクト作成ダイアログ

3.　プロジェクトの名前を入力すると、OK ボタンが有効になり、プロジェクトを作成することができます。

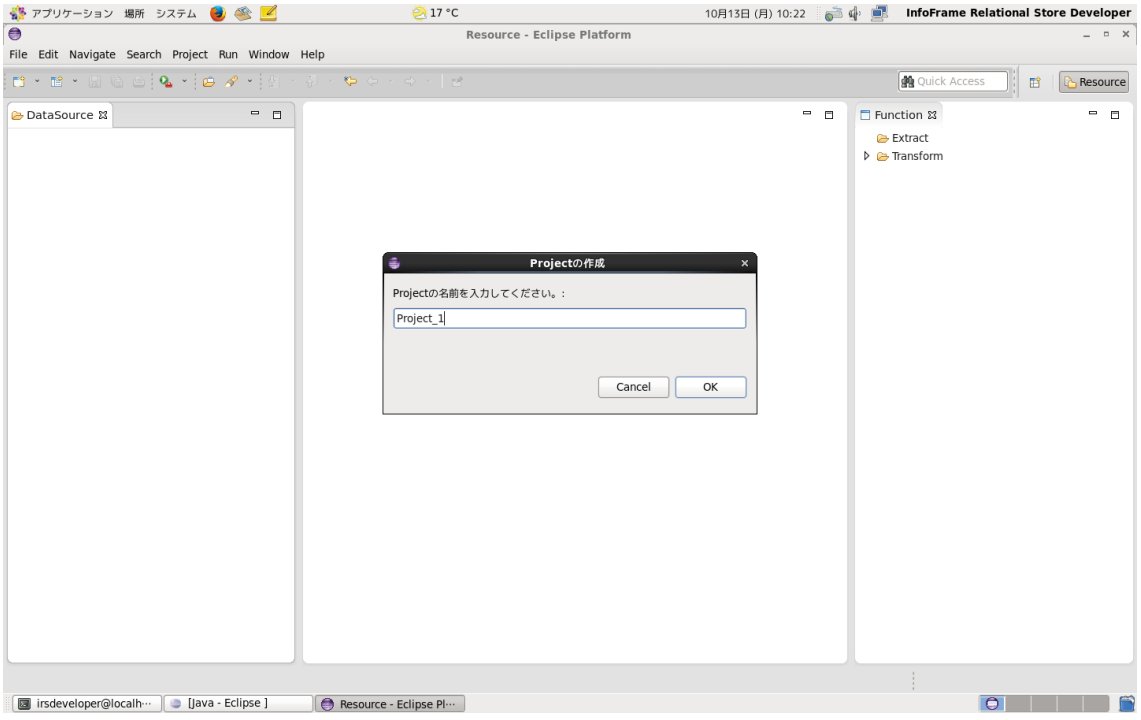


図 3　プロジェクト名の入力

4.　プロジェクト作成後はデータソースウィンドウにプロジェクトが表示されます。

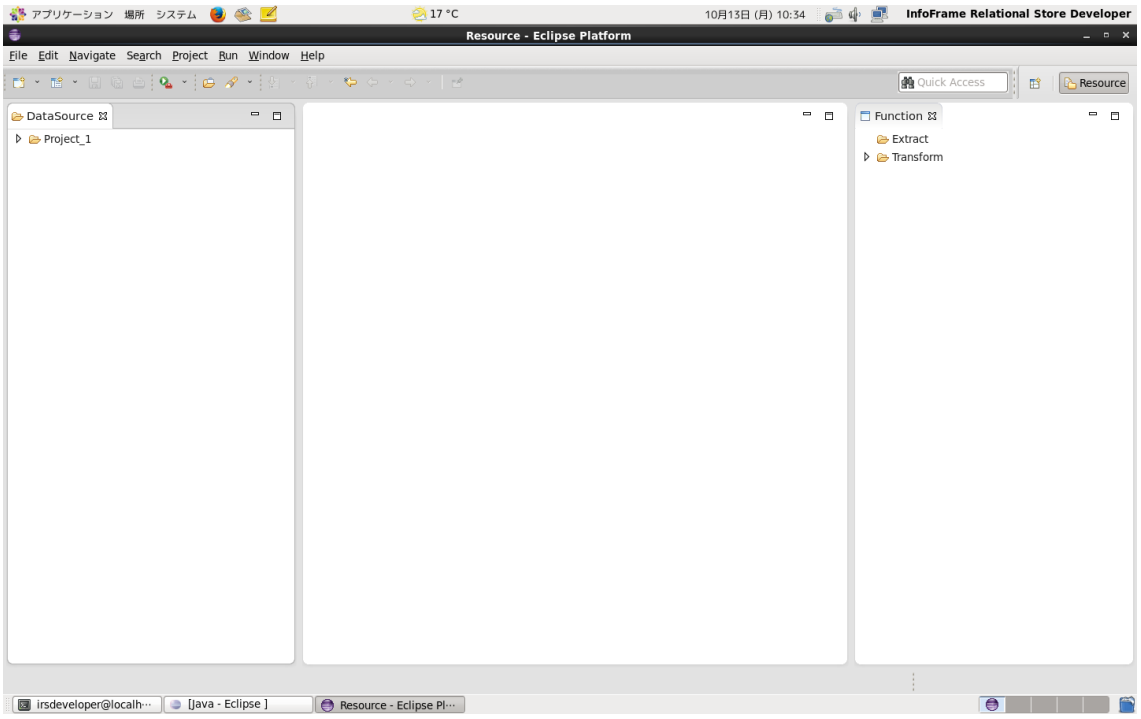


図 4 プロジェクト作成後

5. また、プロジェクトを展開すると「inputDB」と「outputDB」があることを確認できます。

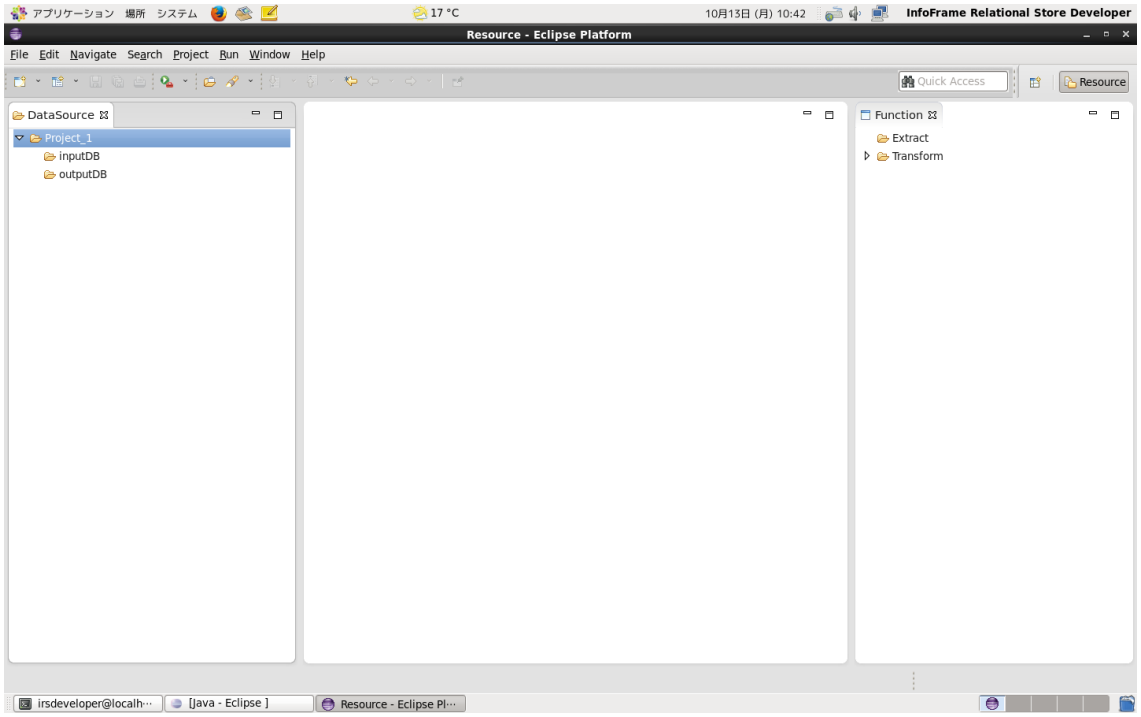


図 5　intputDB と outputDB の確認

2. IRS(InfoFrame Relational Store)を作成する

1. プロジェクトの作成後にプロジェクトを展開し、「inputDB」を右クリックします。
2. ポップアップメニューから「New」→「IRS」を選択します。

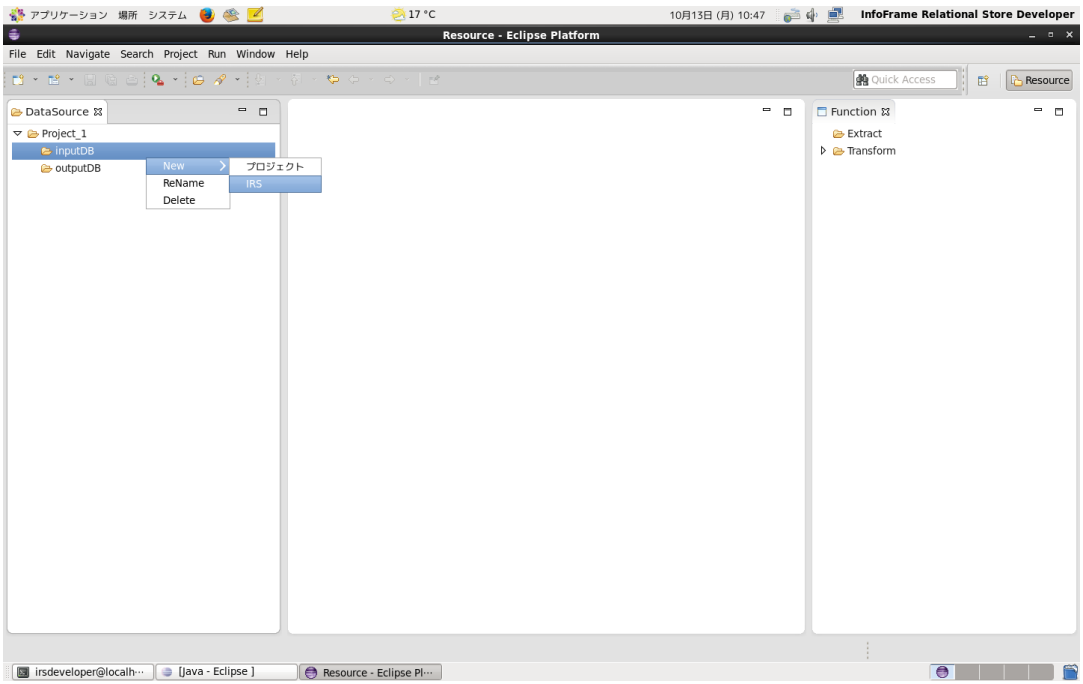


図 6　IRS を選択

3. IRS を作成するダイアログが表示され、名前の入力後に OK ボタンを押すことで

IRS を作成することができます。

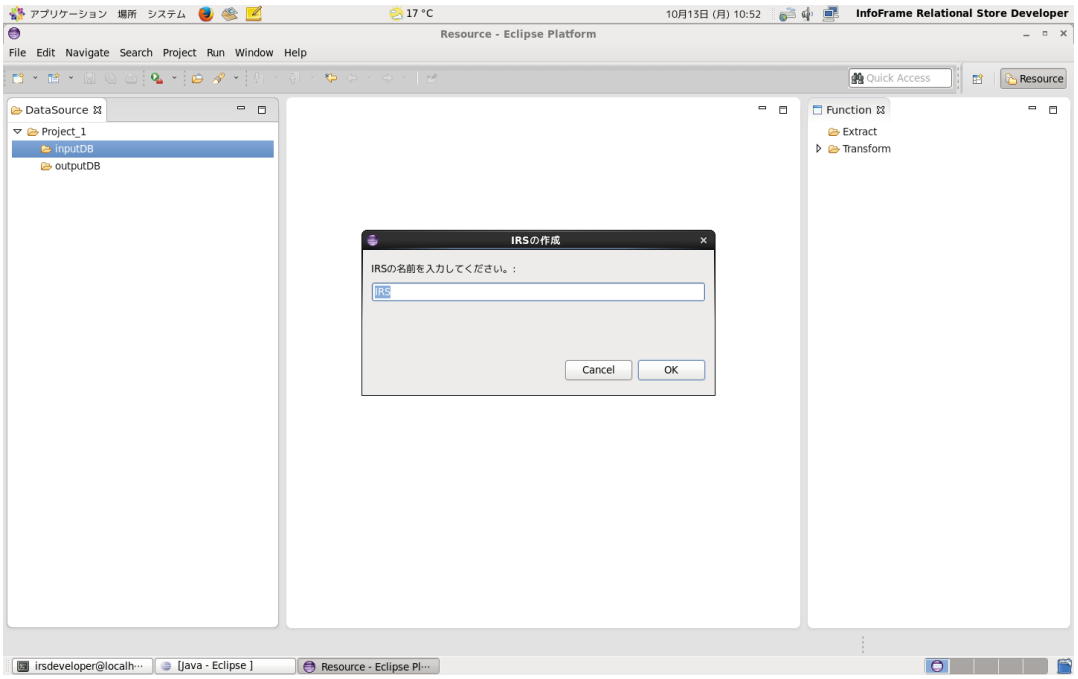


図 7 IRS の作成ダイアログ

4. IRS を作成し、「inputDB」を展開すると「IRS」が格納されており、「IRS」を展開すると、「IRS 接続情報」、「テーブルリスト」が格納されていることが確認できます。

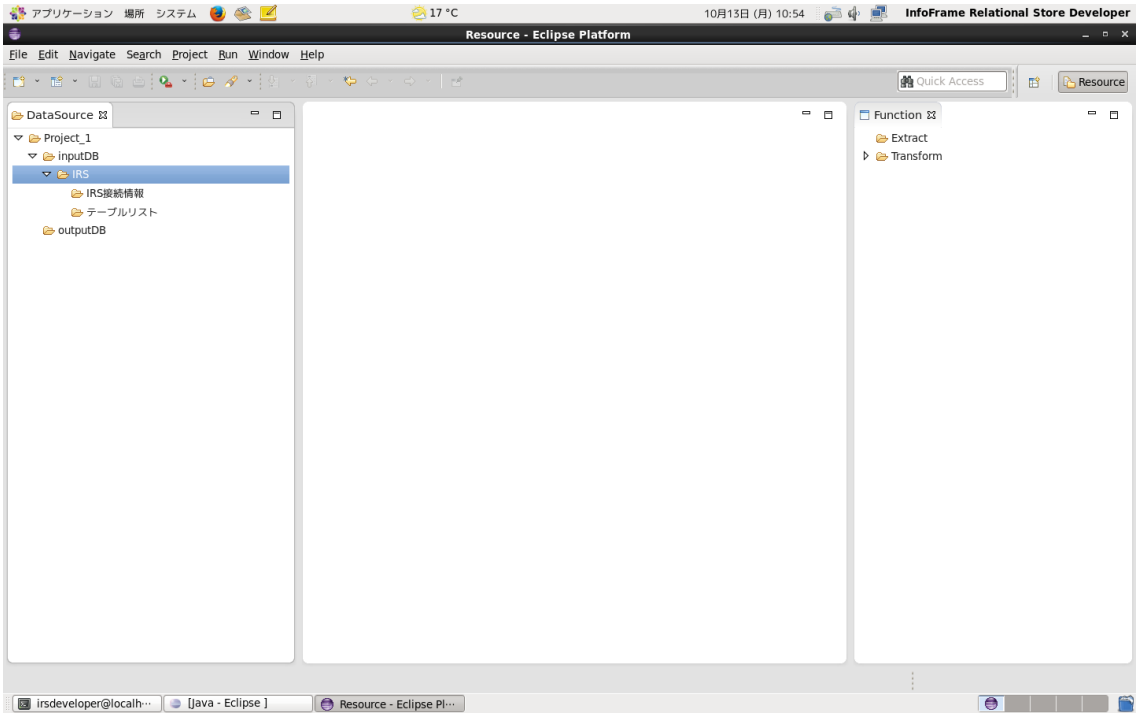


図 8 IRS 作成後の画面例

3. IRS に接続する

1. IRS の作成後に、「inputDB」にある IRS 名を展開し、「IRS 接続情報」をダブルクリックすると、「InputDB 接続情報入力画面」が表示されます。

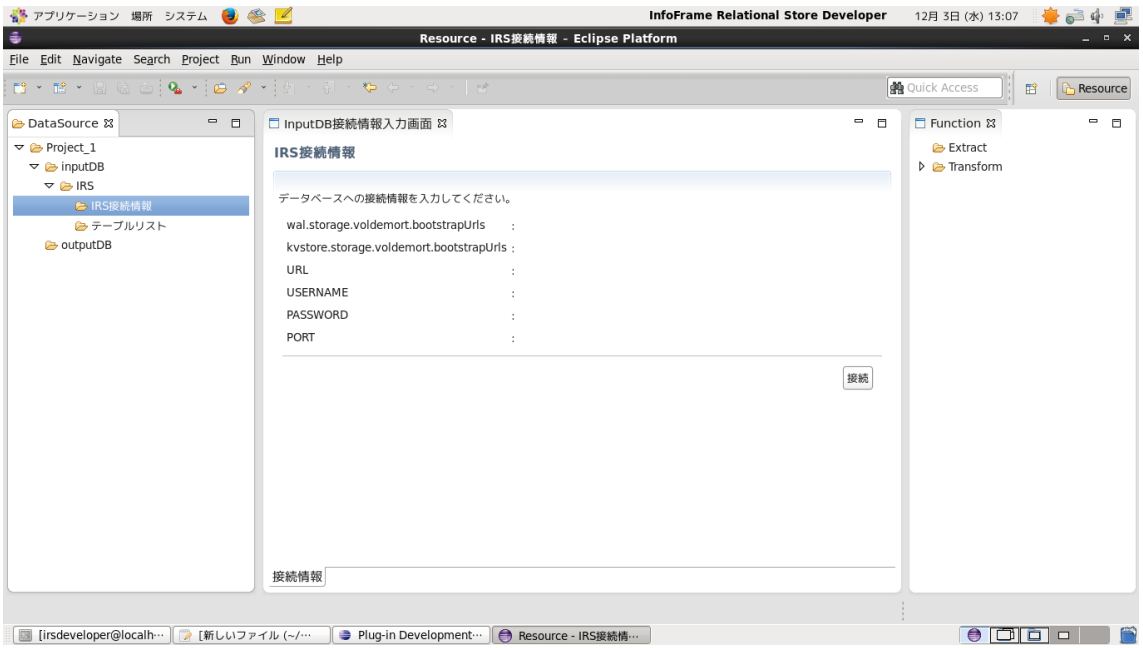


図 9　InputDB 接続情報入力画面

2. 「InputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押すと、IRS への接続が始まります。

① 「walStorageVoldemortBootstapUrls」に RSUserDB 各トランザクションサーバの URL を入力します。入力形式は tcp://<ホスト名または IP アドレス>:<ポート番号>であり、URL 間の区切りは半角のスペースとなります。

② 「kvstoreStorageVoldemortBootstapUrls」に RSUserDB 各ストレージサーバの URL を入力します。入力形式は tcp://<ホスト名または IP アドレス>:<ポート番号>であり、URL 間の区切りは半角のスペースとなります。

③ 「URL」に構成管理サーバのホスト名を入力します。

④ 「USERNAME」に IRS のユーザ名を入力します。「PASSWORD」にユーザ登録用のパスワードを入力します。

⑤ 「PORT」に接続ポート番号（既定値：20000）を入力します。

入力例：

node1(192.168.11.44)：

構成管理サーバ、RSMasterDB の Partiqle サーバ、RSUserDB の Partiqle サーバを担っています。

node2(192.168.11.36)、node3(192.168.11.43)、node4(192.168.11.128)：

RSMasterDB の Transaction サーバ、RSUserDB の Transaction サーバと Storage サーバを担っています。

walStorageVoldemortBootstapUrls	tcp://node2:20000 tcp://node3:20000 tcp://node4:20000
kvstoreStorageVoldemortBootstapUrls	tcp://node2:20050 tcp://node3:20050 tcp://node4:20050
URL	node1
USERNAME	root
PASSWORD	
PORT	20000

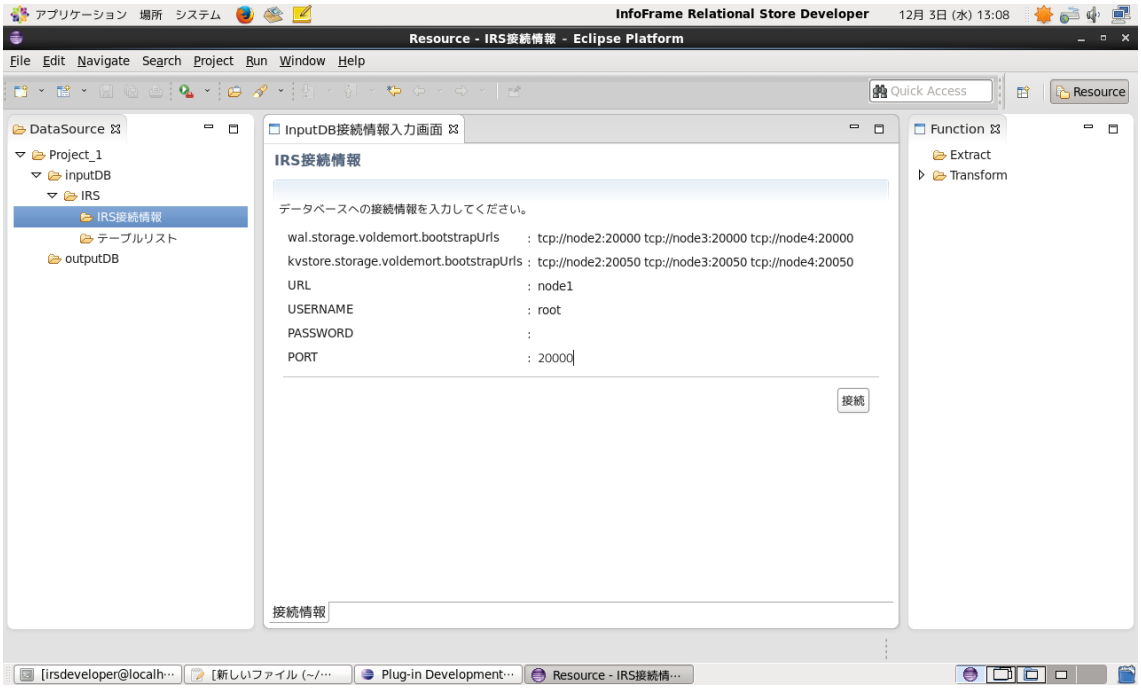


図 10 入力例

3. 正しく接続できる場合、「接続成功」ポップアップが表示され、DataSource のテーブルリストに取得されたテーブルが表示されます。

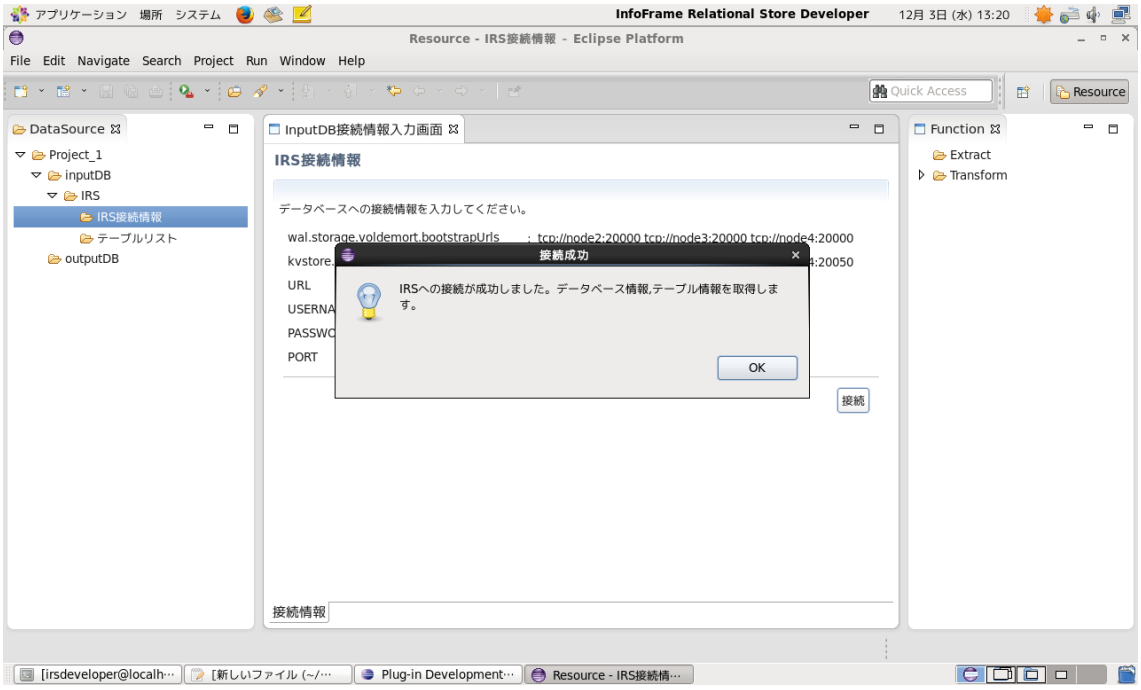


図 11 接続の成功

4. 接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示されます。

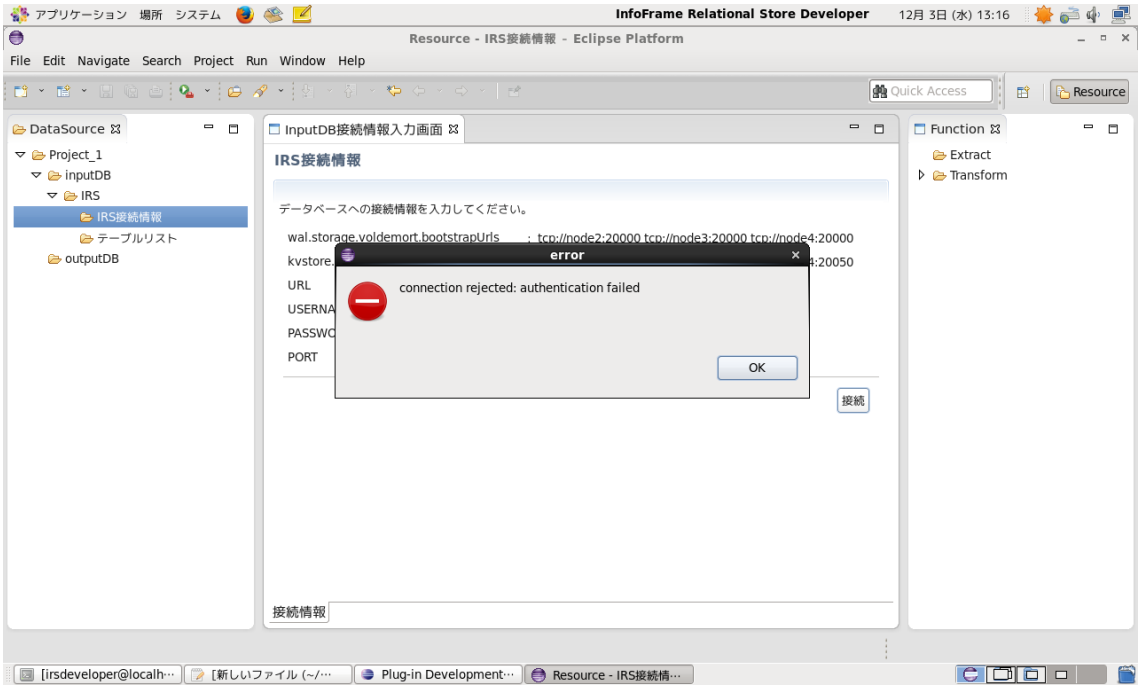
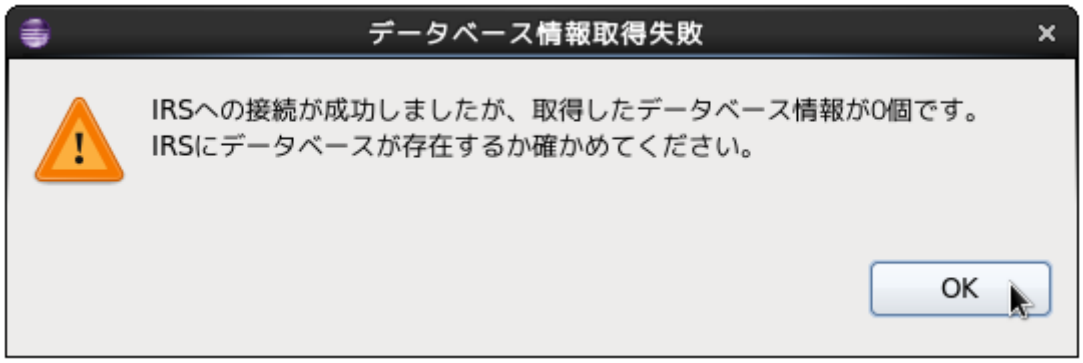


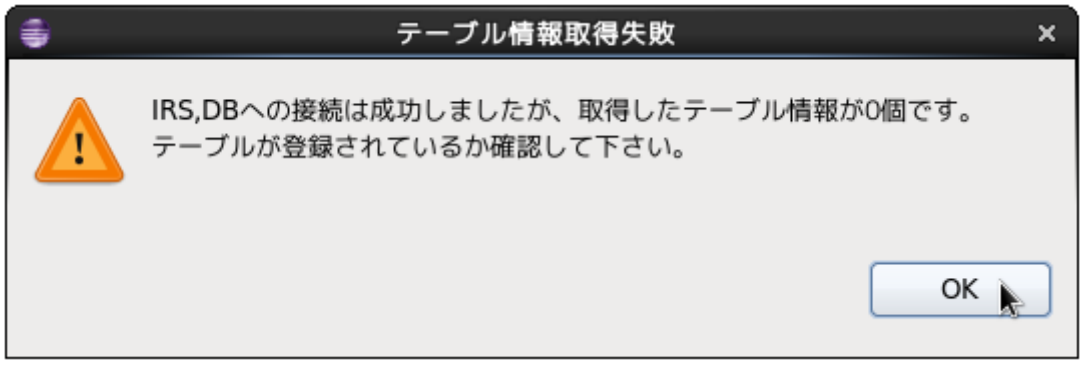
図 12 接続の失敗

5. 接続が成功した場合でも以下の条件をみたす場合は警告ダイアログが表示されます。

① IRS への接続に成功したが、データベースが登録されておらず、データベースの情報が取得できなかった場合



② IRS、データベースに接続でき、取得したテーブル情報が 0 個である場合



4. PostgreSQL を作成する

1. プロジェクトの作成後にプロジェクトを展開し、outputDB を右クリックします。
2. ポップアップメニューから New→PostgreSQL を選択します。

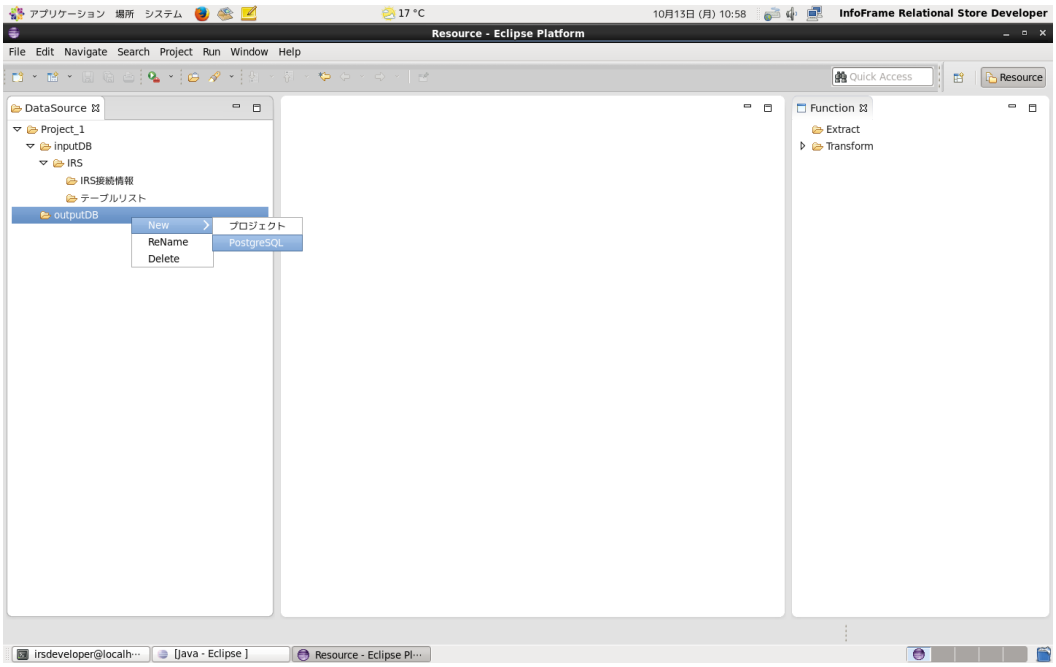


図 13 PostgreSQL を選択

3. PostgreSQL を作成するダイアログが表示され、名前の入力後に OK ボタンを押すことで PostgreSQL を作成することができます。

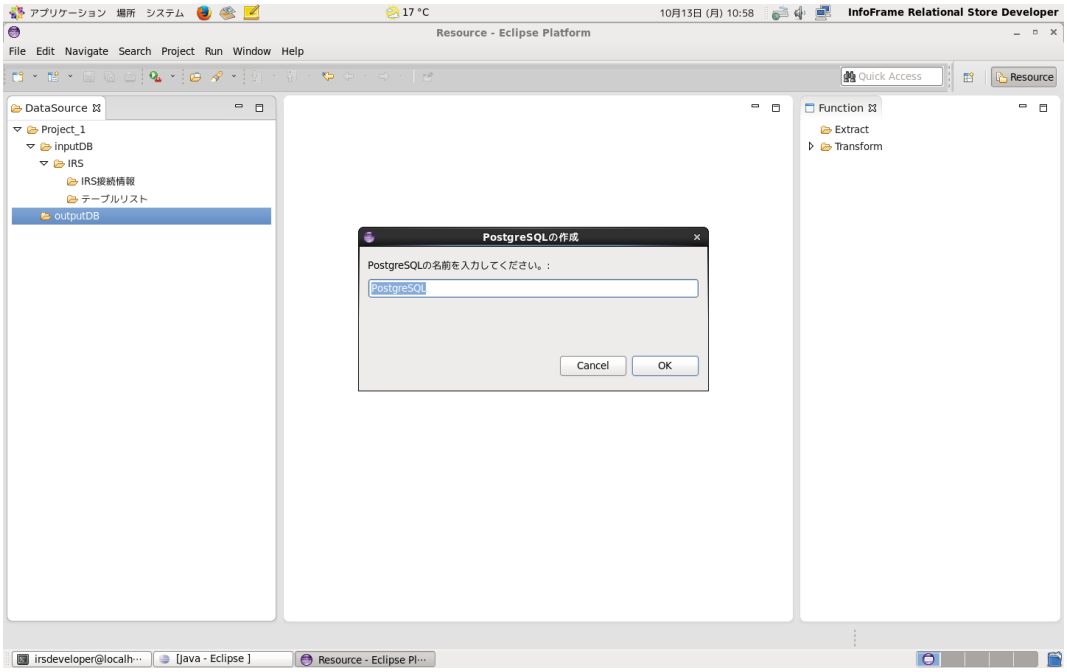


図 14 PostgreSQL 作成ダイアログ

4. PostgreSQL を作成し、outputDB を展開すると PostgreSQL が格納されており、PostgreSQL を展開すると、PostgreSQL 接続情報、テーブルリストが格納されていることが確認できます。

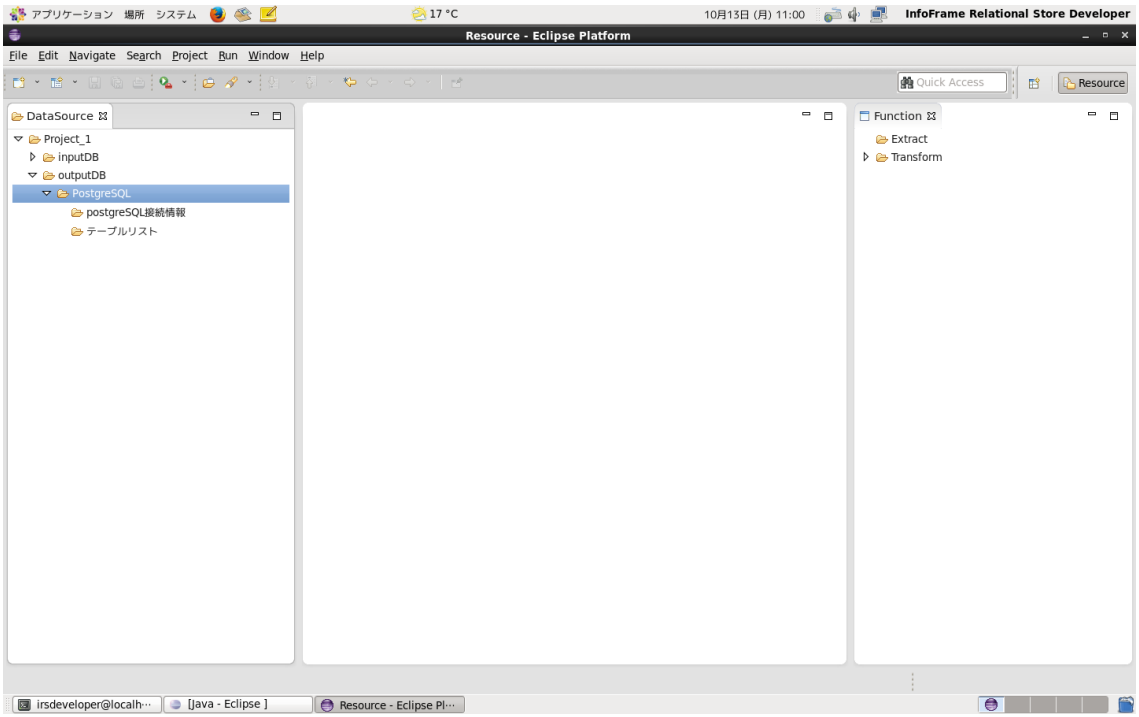


図 15 PostgreSQL 接続情報、テーブルリストの確認

5. PostgreSQL に接続する

1. PostgreSQL の作成後に「outputDB」にある IRS 名を展開し、「postgreSQL 接続情報」をダブルクリックすると、「OutputDB 接続情報入力画面」が表示されます。

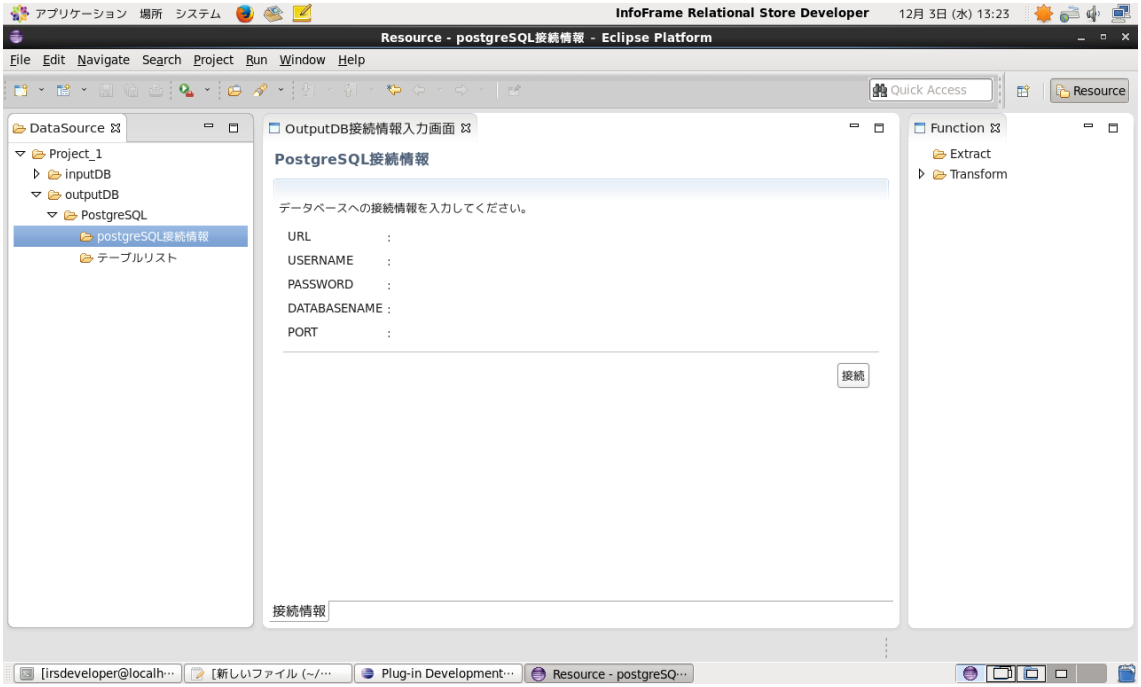


図 16　OutputDB 接続情報入力画面

2. 「OutputDB 接続情報入力画面」で接続情報を入力して、「接続」ボタンを押すと、PostgreSQL への接続が始まります。

① 「URL」に PostgreSQL サーバのホスト名を入力します。

② 「USERNAME」に PostgreSQL のユーザ名を入力します。「PASSWORD」にユーザ登録用のパスワードを入力します。

③ 「DATABASENAME」に使用するデータベース名を入力します。

④ 「PORT」に接続ポート番号を入力します。

入力例：

URL	localhost
USERNAME	irsdeveloper
PASSWORD	
DATABASENAME	test
PORT	

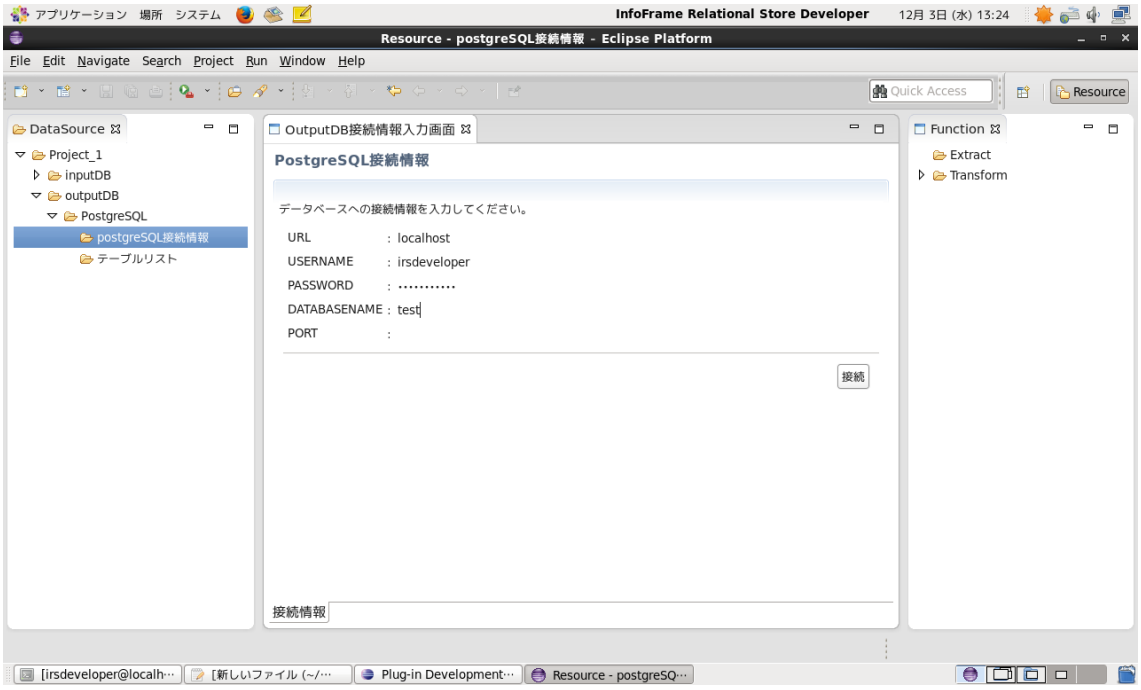


図 17　入力例

3. 正しく接続できる場合、「接続成功」ポップアップが表示され、DataSource のテーブルリストに取得されたテーブルが表示されます。

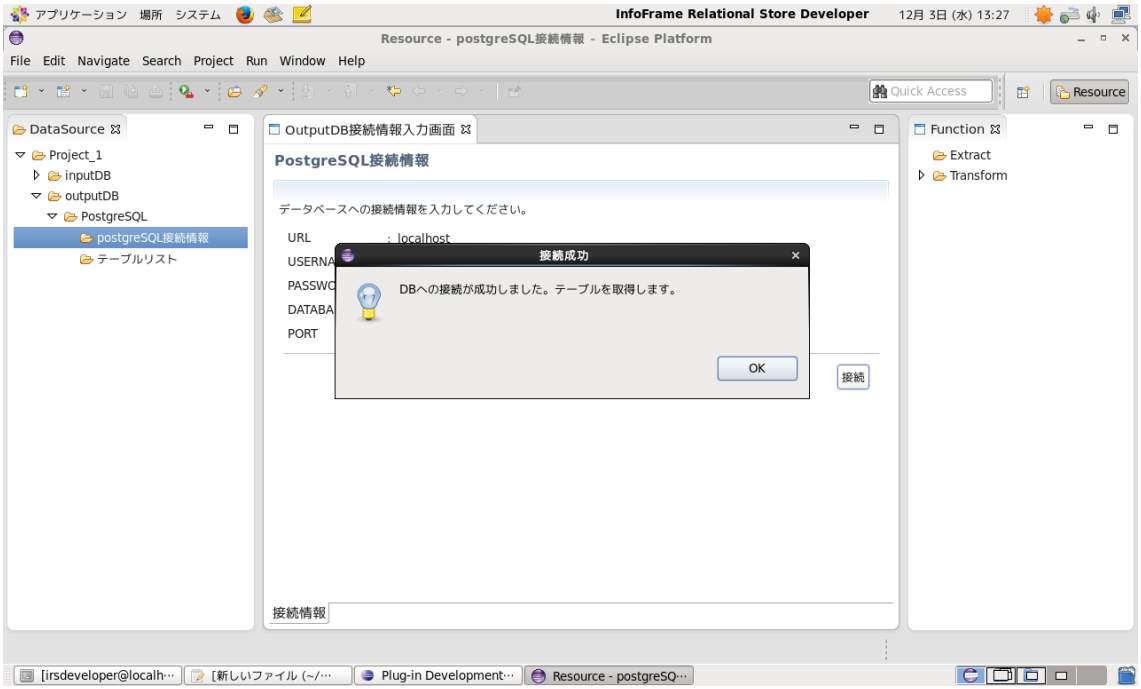


図 18 接続の成功

4. 接続情報に誤りがあり、接続が失敗した場合は、エラー画面が表示されます。

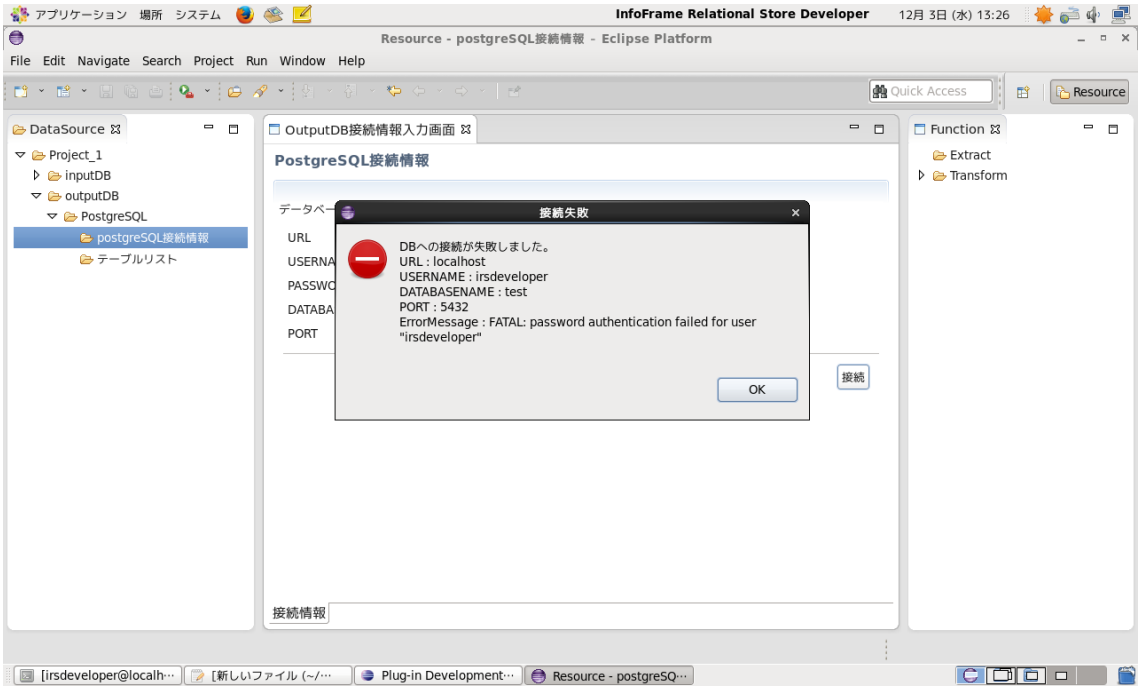


図 19 接続の失敗

6. テーブル情報を確認する

1. IRS に接続する、PostgreSQL に接続するを行った後にテーブルリストにあるテーブル名を右クリックして、「テーブル情報」を選択すると、テーブル情報画面が表示されます。

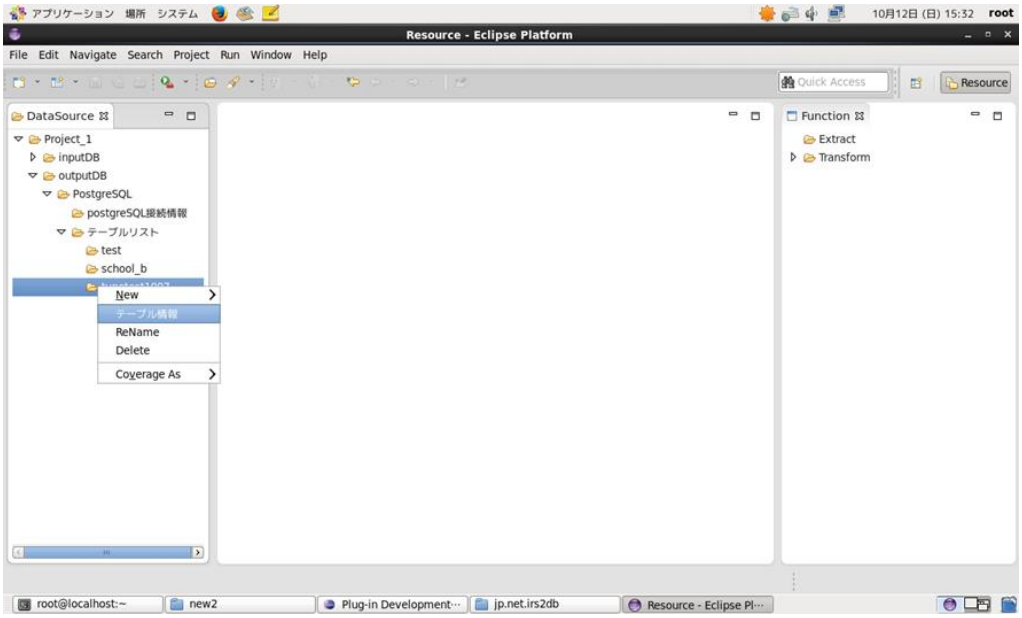


図 20　テーブル情報の選択

2. テーブル情報表示で表示されるテーブル情報はIRSとPostgreSQLで異なります。
   IRS：カラム名,データ型のみ（プライマリキー,長さ,NOT NULL,Table Exampleは表示されません）
   PostgreSQL：プライマリキー,カラム名、データ型,長さ,NOT NULL,Table Example
   *プライマリキーはカラム名の隣に「鍵」のマークが表示されます。

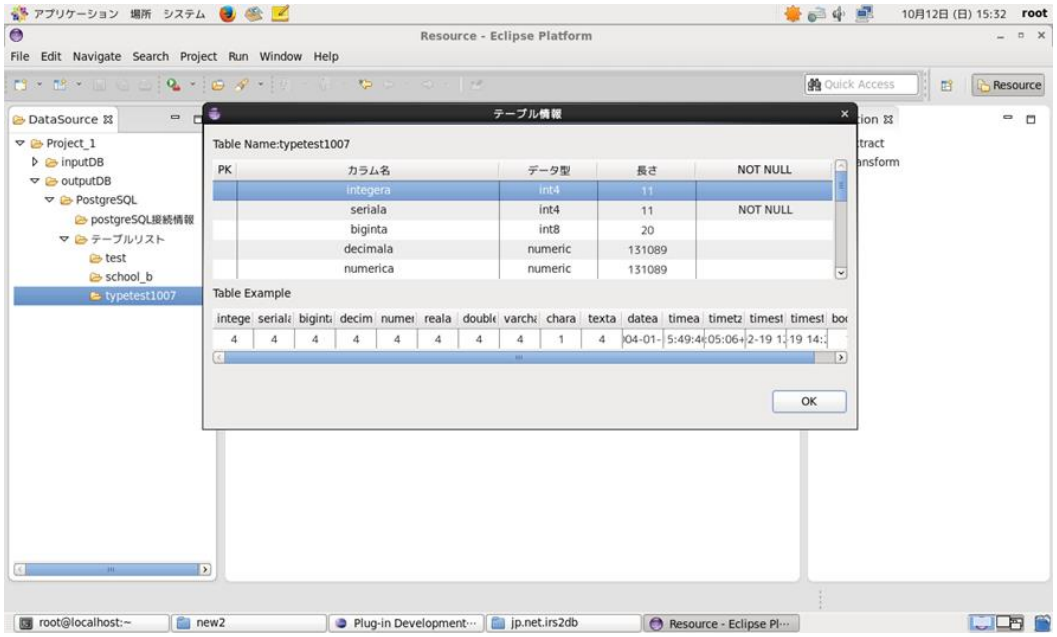


図 21　テーブル情報の確認

7. テーブルを作成する

1. PostgreSQL に接続後、「テーブルリスト」を右クリックして、テーブルの作成をクリックします。
2. テーブル作成を選択するとテーブル作成ダイアログが表示され、テーブル名の設定及び各カラムに対して PK、カラム名、データ型、長さ、NOT NULL、UNIQUE を設定することでテーブルを作成することができます。

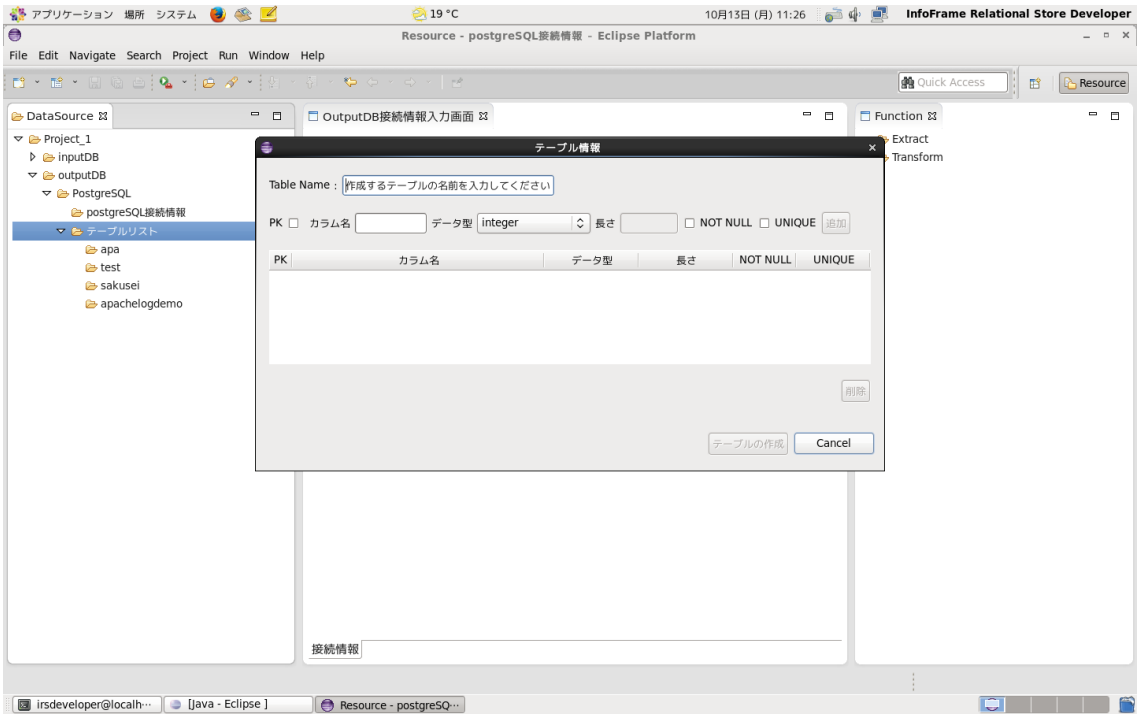


図 22　テーブル作成ダイアログ

3. PK、カラム名、データ型、長さ、NOT NULL、UNIQUE を設定後、追加ボタンを押すことで作成するテーブルにカラムを追加することができます。テーブル作成時に指定できるカラム方の一覧は以下の通りです。

カラム型名	カラム方の長さを指定できるか	説明
integer	×	4 バイト符号付き整数
serial	×	自動増分 4 バイト整数
bigint	×	8 バイト符号付き整数
bigserial	×	自動増分 8 バイト整数
decimal	○	精度の選択可能な高精度数値
numeric	○	精度の選択可能な高精度数値
real	×	単精度浮動小数点（4 バイト）
double precision	×	倍精度浮動小数点（8 バイト）
varchar	○	可変長文字列
char	○	固定長文字列
text	×	可変長文字列
date	○	暦の日付（年月日）
time	○	時刻（時間帯なし）

timetz	○	時間帯付き時刻
timestamp	○	日付と時刻（時間帯なし）
timestamptz	○	時間帯付き日付と時刻
boolean	×	論理（ブール）値（真/偽）
bytea	×	バイナリデータ（"バイトの配列（byte array）"）

decimal,numeric はデータ型の長さの入力ボックスが１つなので、「整数桁数,小数点桁数」と入力してもらう。
decimal と numeric は PostgreSQL のデータ型的には同じものです。

PostgreSQL　8.4 のデータ型の詳細については以下のドキュメントを参照ください
https://www.postgresql.jp/document/8.4/html/datatype.html

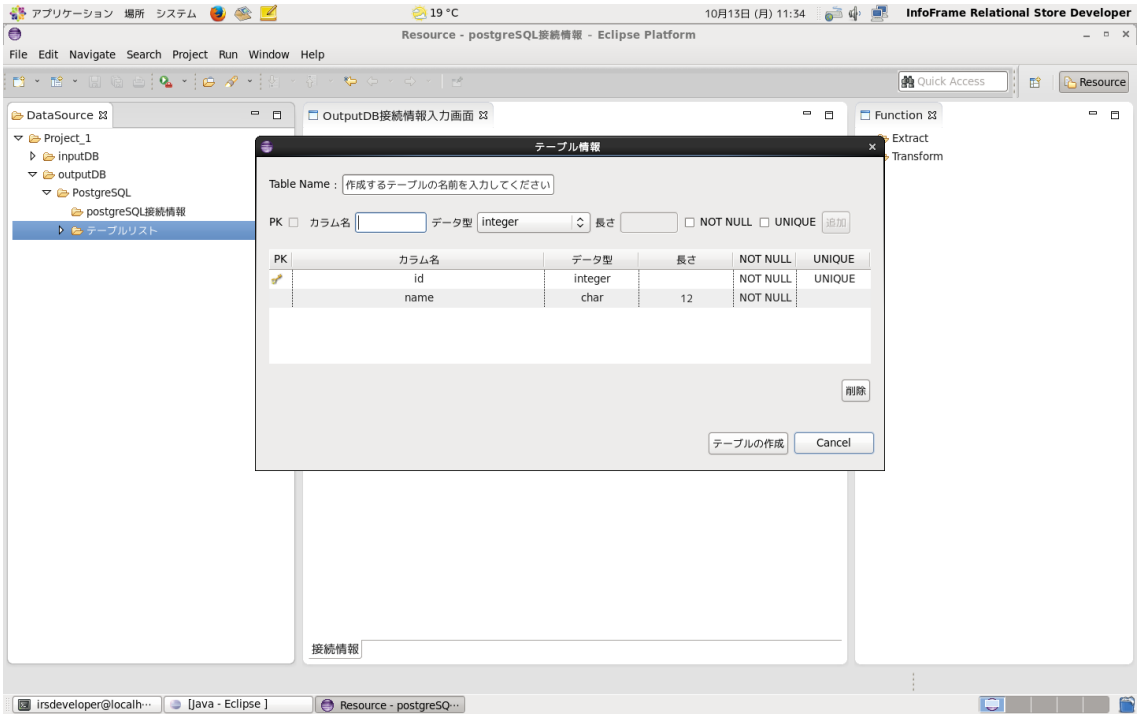


図 23 カラムの追加

4. 作成したカラムをクリックし、削除ボタンを押すことでカラムを削除することができます。

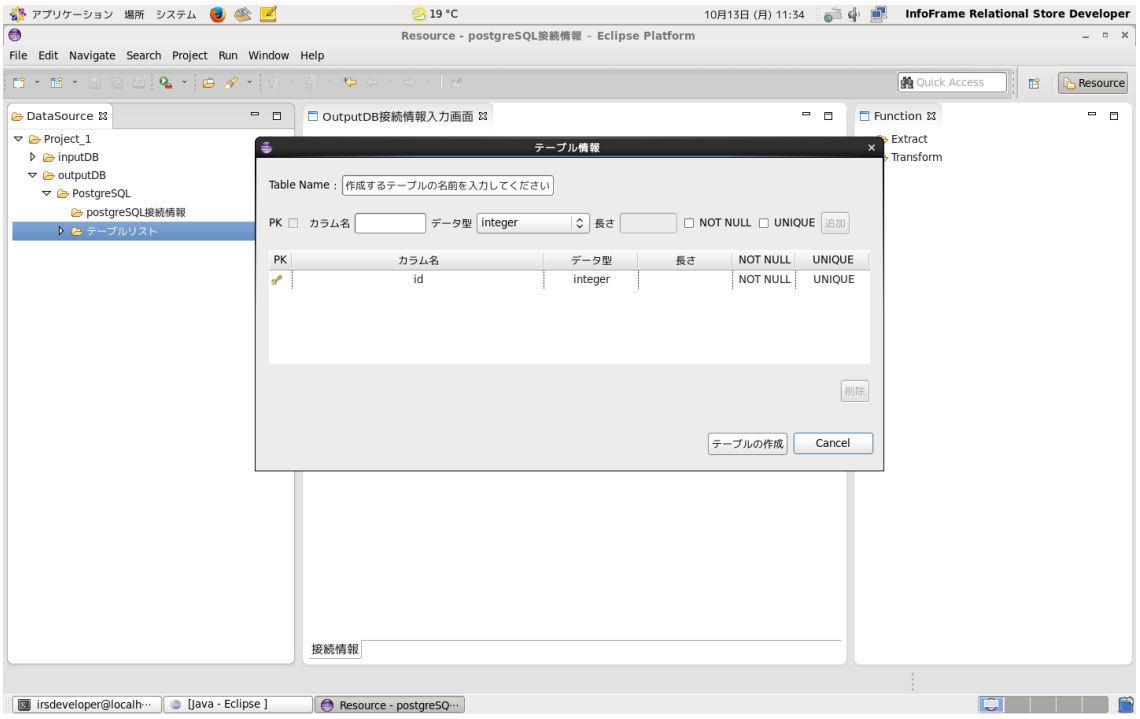


図 24 カラムの削除

5. 最後にテーブル、カラムの設定後にテーブル作成ボタンを押すとデータベースにテーブルが作成され、作成に成功するとダイアログが表示されます。

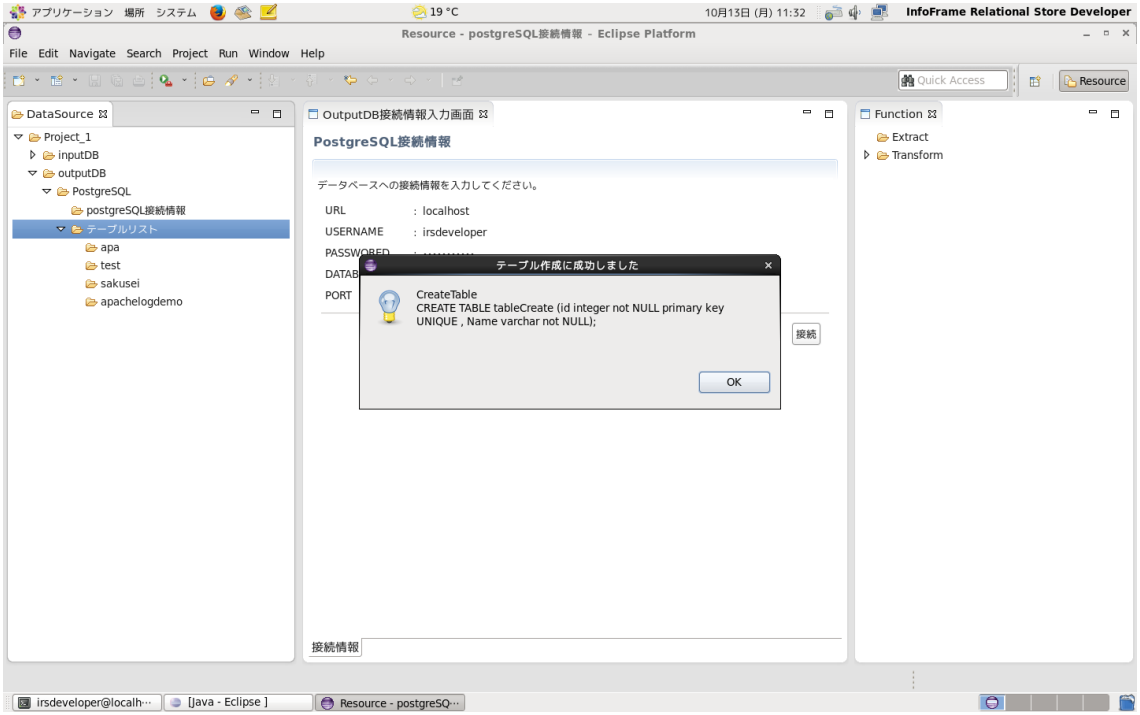


図 25 テーブル作成成功

8. 作業ウィンドウで条件を設定する。

1. プロジェクト名をダブルクリックすることで、作業ウィンドウを開きます。

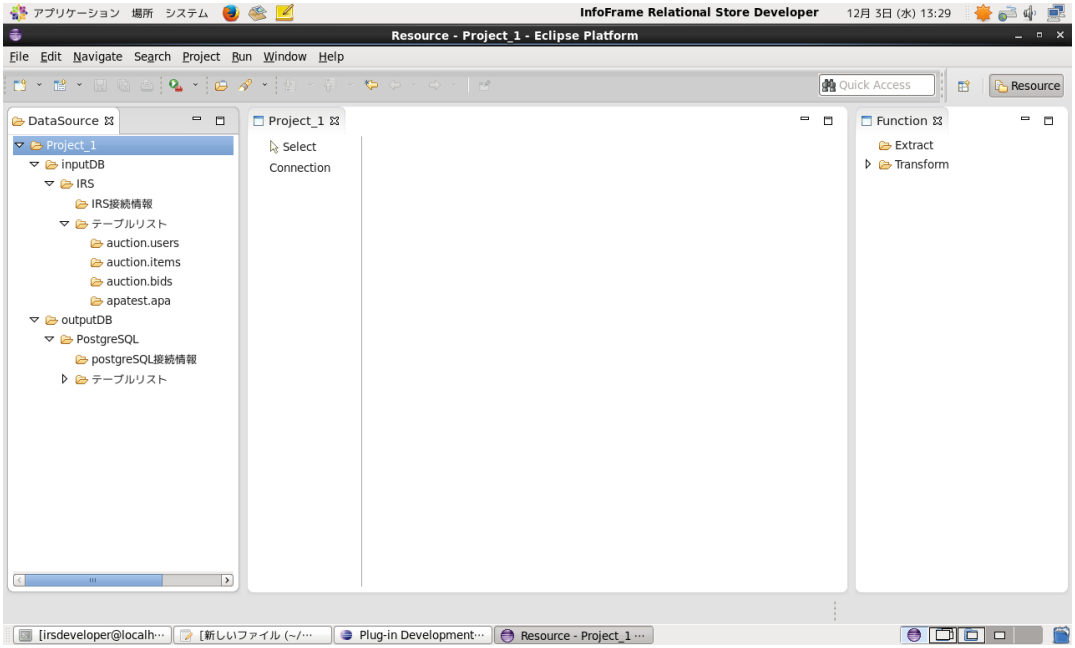


図 26 作業ウィンドウを開く

2. DataSource ビューにあるテーブルや Function ビューファンクション（Extract または Transform）を選択して、作業ウィンドウへドラッグすることができます。使用するテーブルやファンクションなどを作業ウィンドウにドラッグすると、作業ウィンドウにテーブルと各ファンクションのアイコンが表示されます。

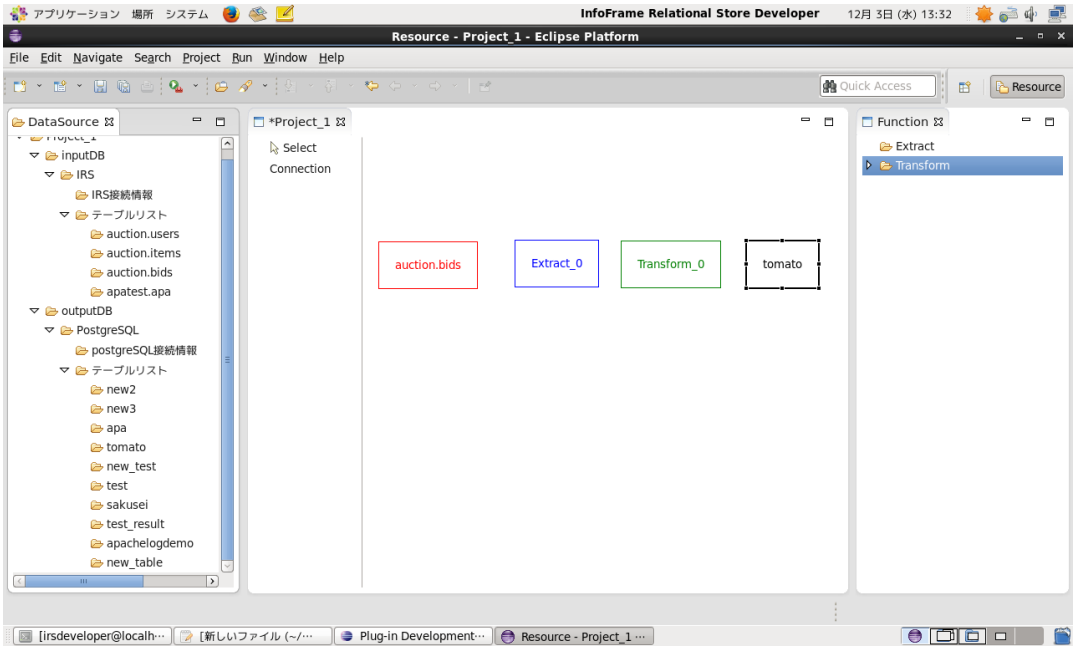


図 27　作業ウィンドウにドラッグアンドドロップする

3. 作業ウィンドウの左側に「Select」と「Connection」というツールがあります。「Select」を選択した場合に、アイコンや線の選択ができます。「Connection」を選択した場合に、アイコン間の紐付けができます。

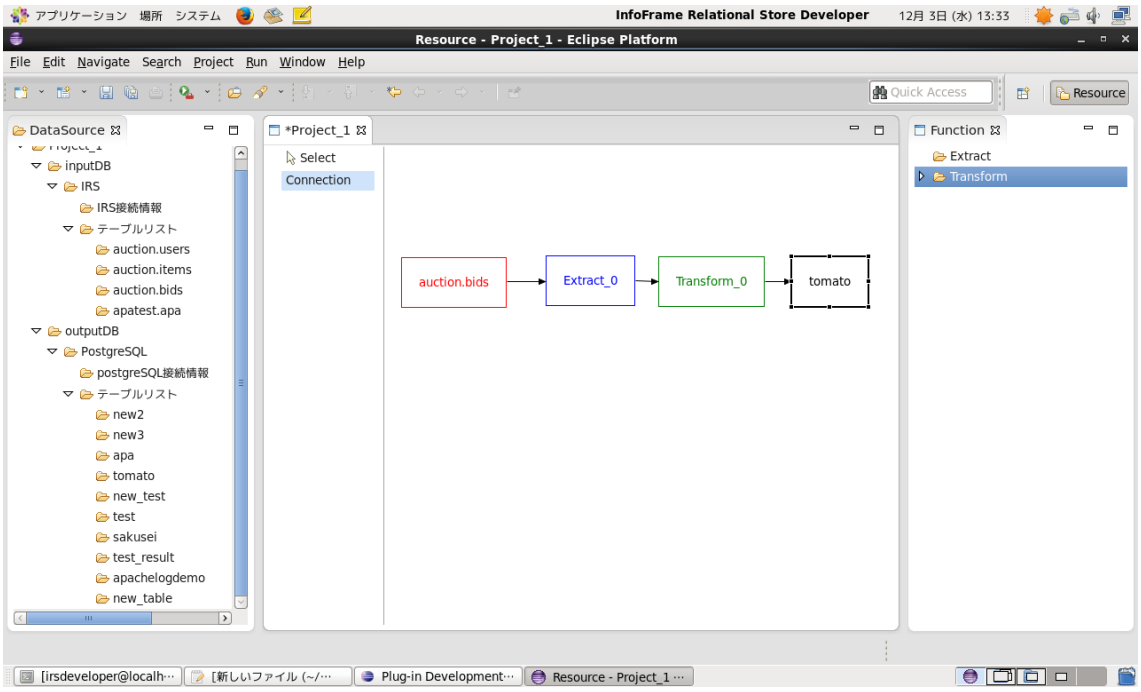


図 28 アイコンの紐付け

4. 左側の「Connection」を選択して、作業ウィンドウ内のアイコンを紐付けすることで、テーブルと各ファンクションの対応関係が指定できます。

注意：プロジェクトに対して、Extract ファンクションと Transform は省略できません。データの流れは必ず Input Table → Extract → Transform → Output Table になります。Extract アイコンをダブルクリックすると、Extract 条件の編集ができます。Transform アイコンをダブルクリックすると、Transform 条件の編集ができます。

5. 左側の「Select」を選び、アイコンを選択してドラッグすることで、アイコンの位置が調整できます。
6. 選択したアイコンや線を右クリックして、「Delete」を選択するまたはキーボードの Delete キーをことで、アイコンの削除ができます。アイコンが削除される際に、関連する線も一括削除されます。

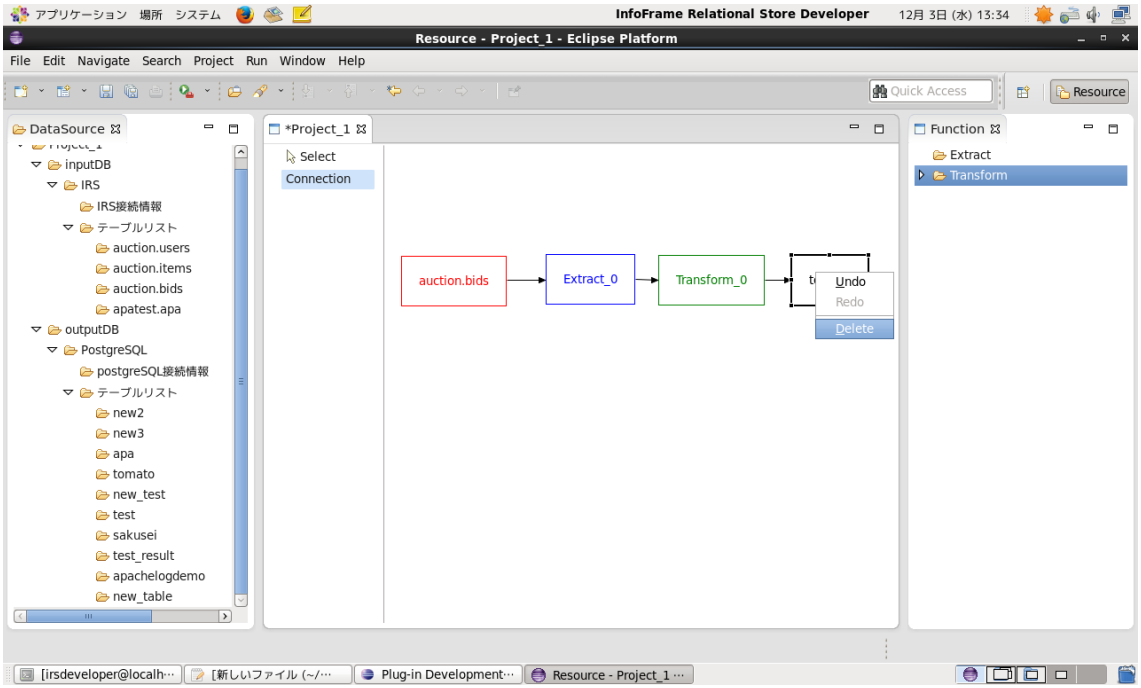


図 29 アイコンの削除（1）

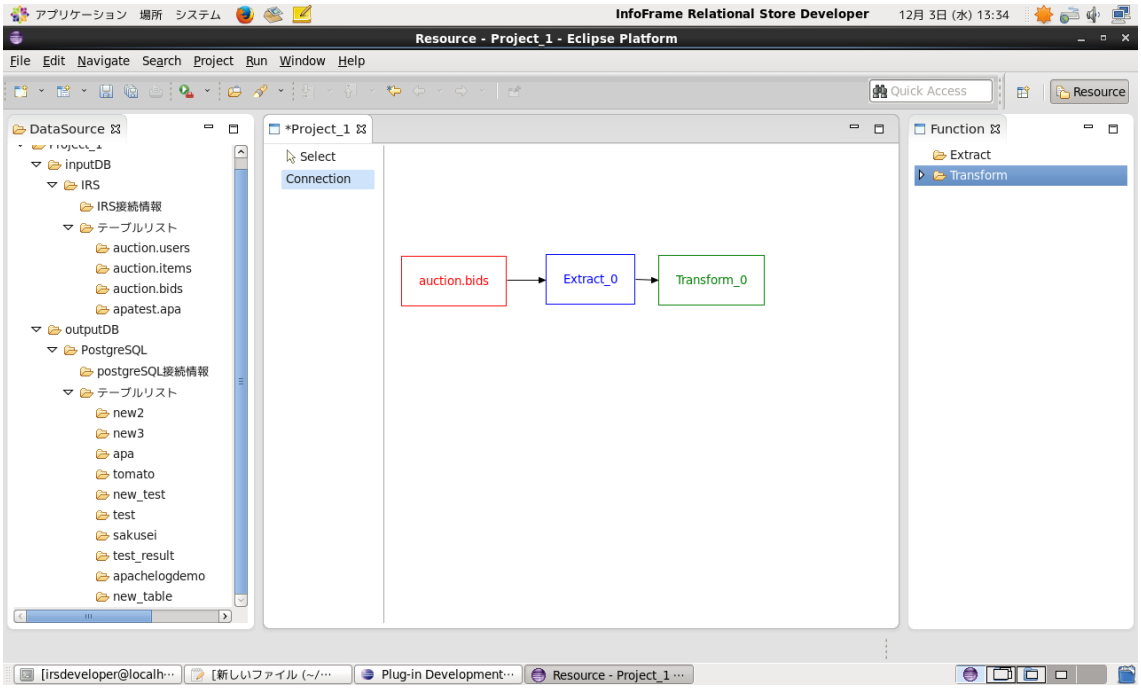


図 30 アイコンの削除（2）

7. 作業ウィンドウを右クリックして、「Undo」を選択するまたはキーボードの Ctr+Z を押すことで、実行済みの操作が取り消せます。

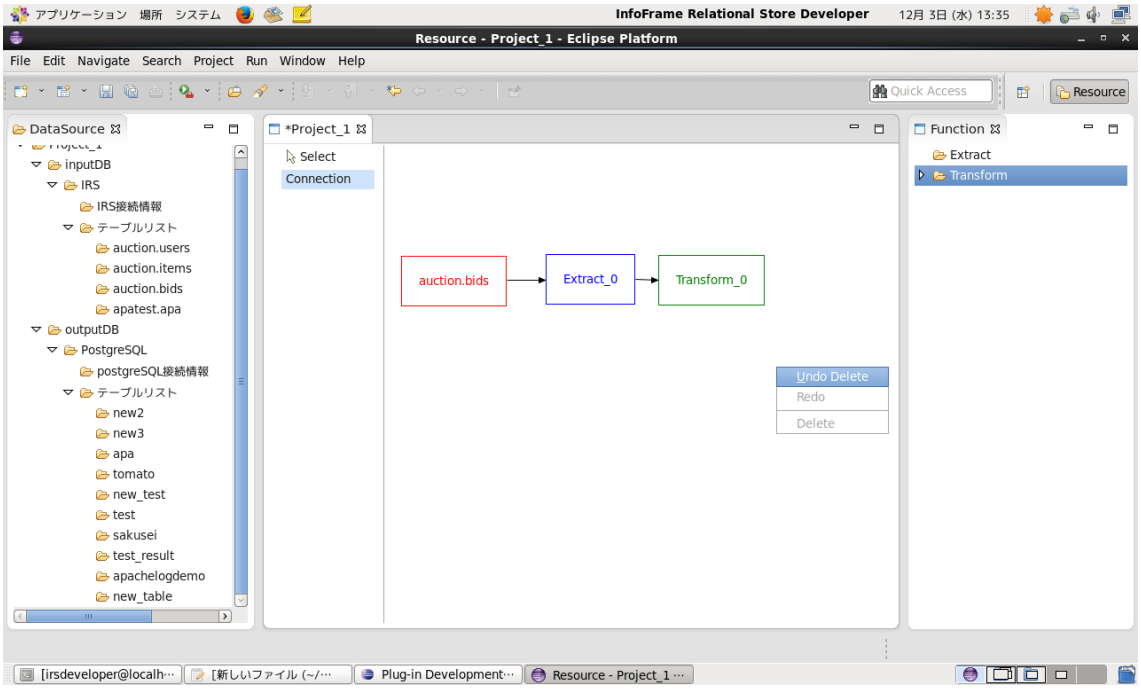


図 31 Undo 操作

8. 作業ウィンドウを右クリックして、「Redo」を選択することで、取り消された操作がやり直せます。

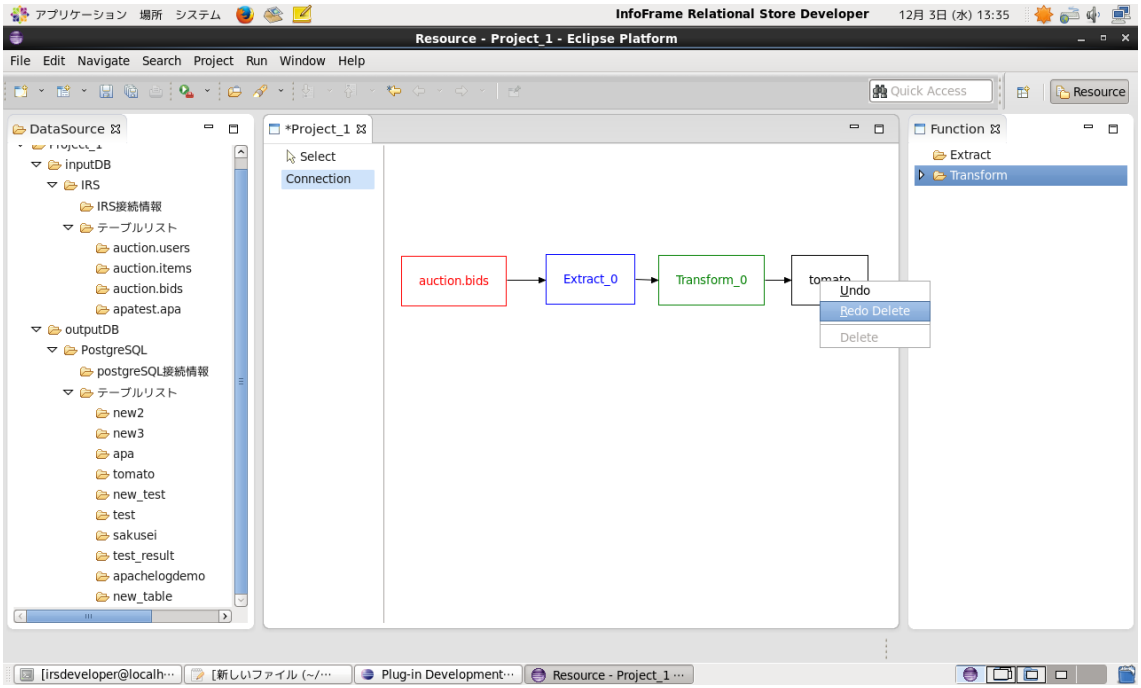


図 32 Redo 操作

9. Extract 条件を設定する

1. 線を引いた状態で Extract をダブルクリックすると、ウィンドウが開かれます。

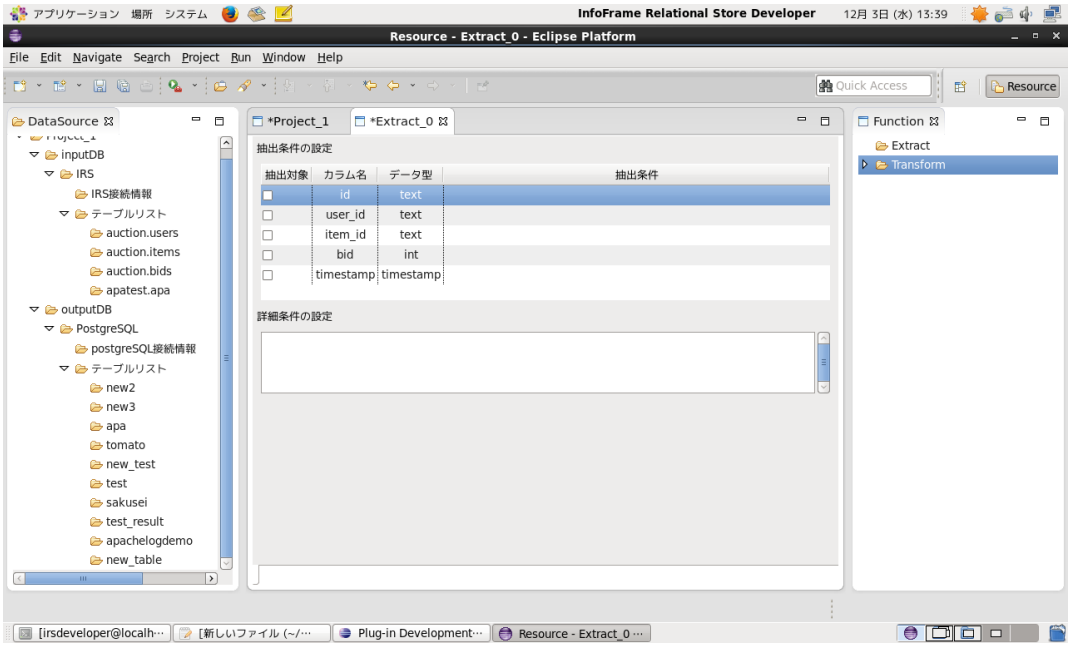


図 33 Extract ウィンドウ

2. ウィンドウには Extract と紐づけられている移行元テーブルのテーブル情報が表

示されます。

3. 抽出条件の設定

① 抽出対象の欄にチェックをいれると、そのカラムを抽出対象とすることができます。

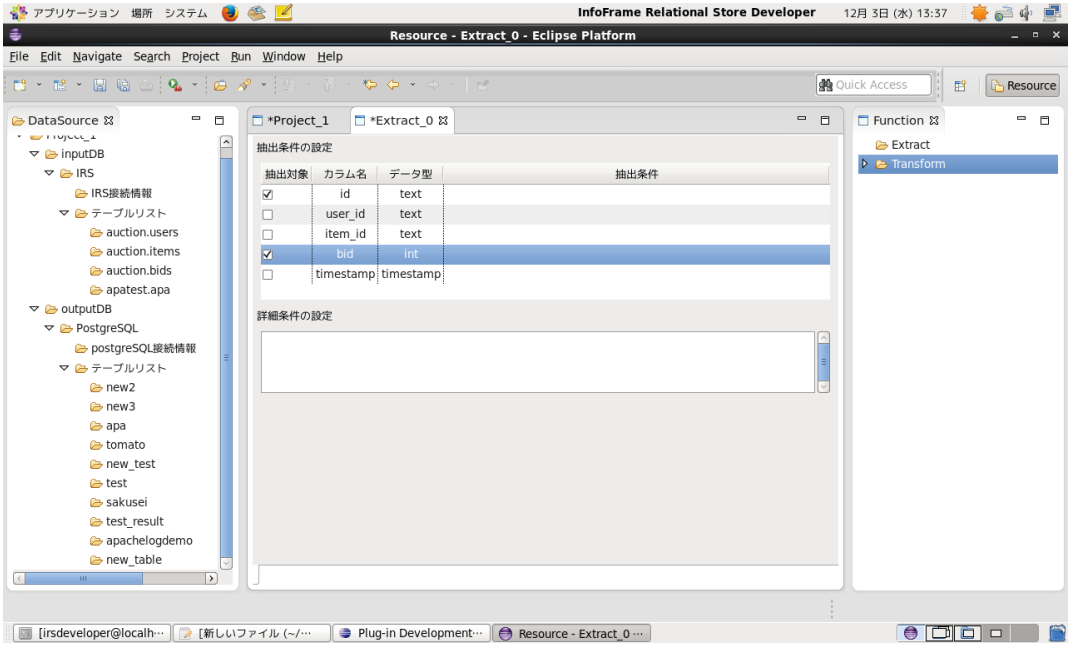


図 34 抽出対象とする

② 抽出条件のセルをクリックすると、そのカラムの抽出条件を設定するダイアログが表示されます。

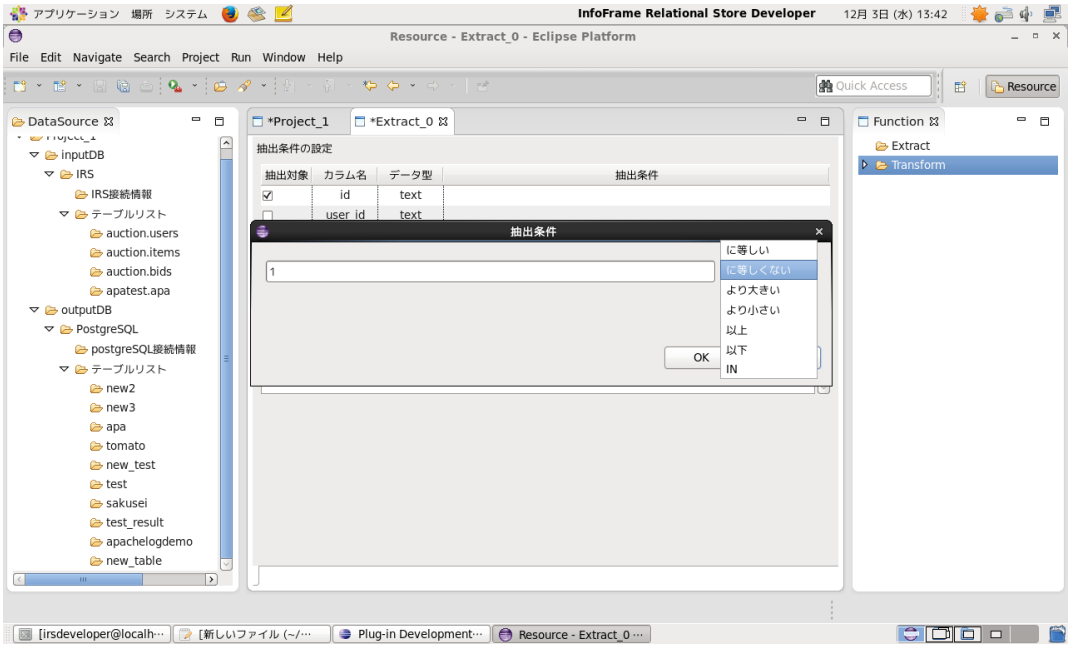


図 35 条件抽出設定画面

③ 値を入力していただき、条件を選択することで任意のカラムに対して抽出条件を設定することができます。空文字を入力すると一度登録した条件を削除することができます。また、TEXT 型などの文字列を扱うデータ型の条件を入力する際は、シングルクォーテーションやダブルクォーテーションで囲む必要はありません。

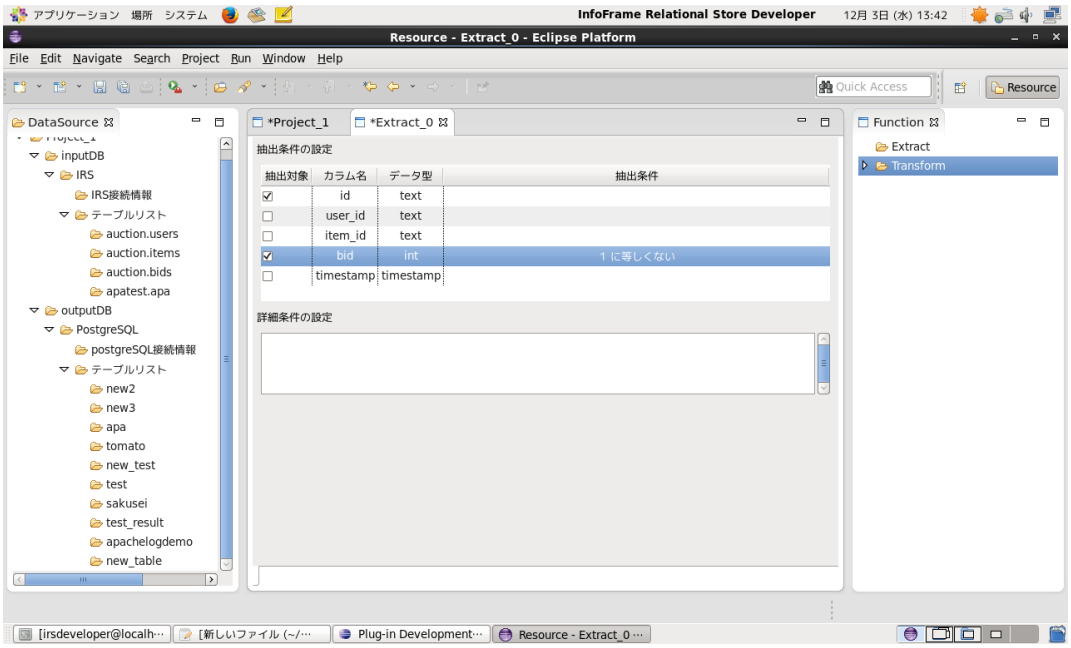


図 36 条件設定後

4. 詳細条件の設定

① テーブルの下のウィンドウに条件文を入力することで IRS に対して直接抽出条件を与えることができます。ここの入力は直接 IRS の HadoopAPI の Configurator. filter()に入力されます。

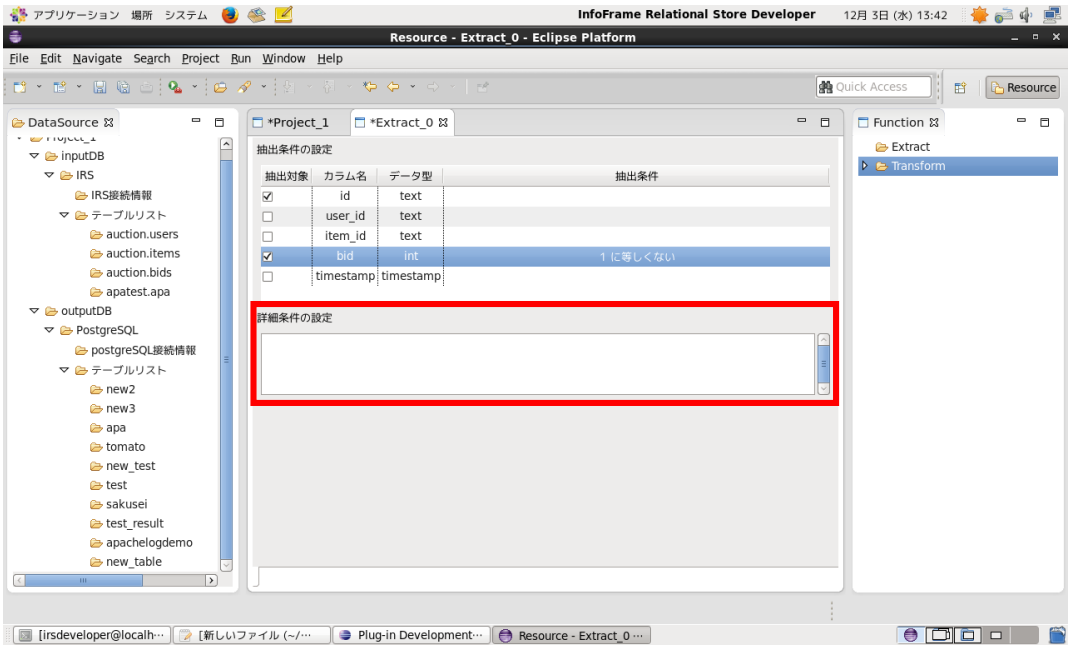


図 37 詳細条件の設定画面

② この場合は 3 で設定した抽出条件は IRS からデータを抽出する際に無視されます。そのため、詳細条件を設定しない場合は詳細条件を入力しないでください。

5. 最後に条件設定後タブを閉じて保存する、もしくは Ctr+S で保存すると設定した条件が反映されます。

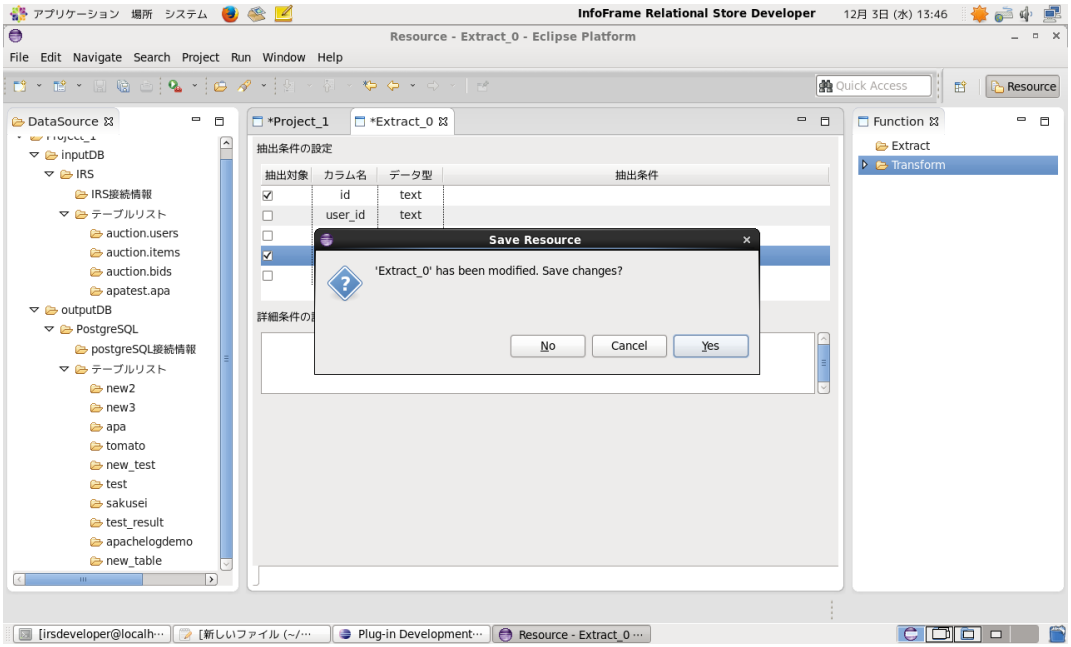
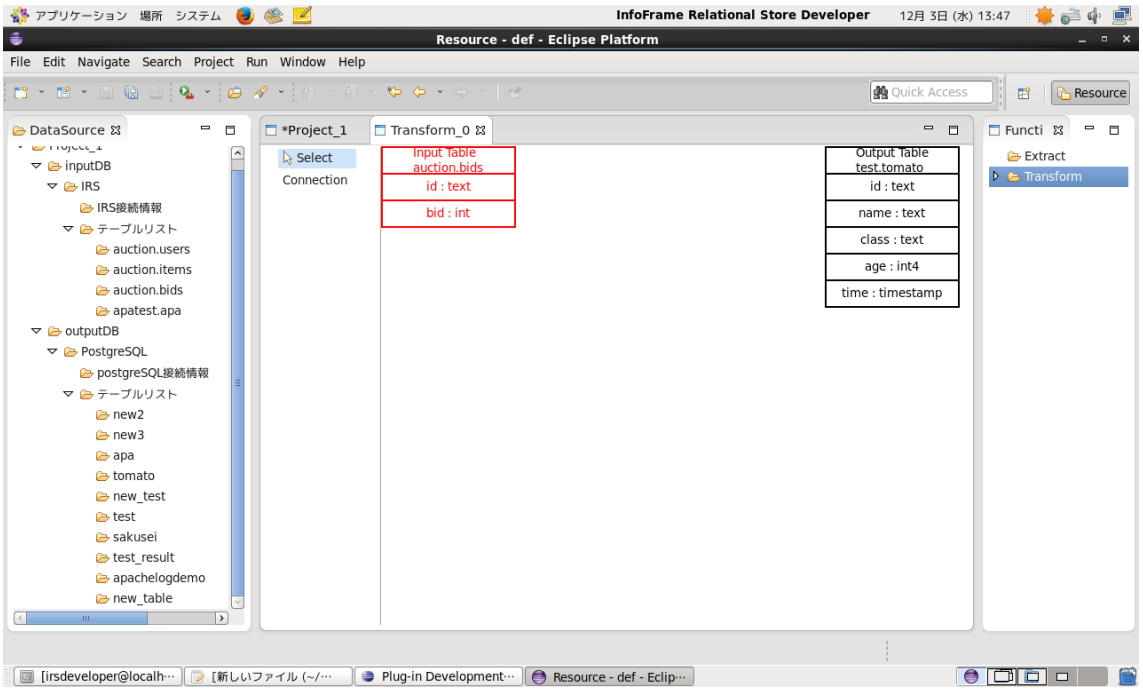


図 38 セーブ確認画面

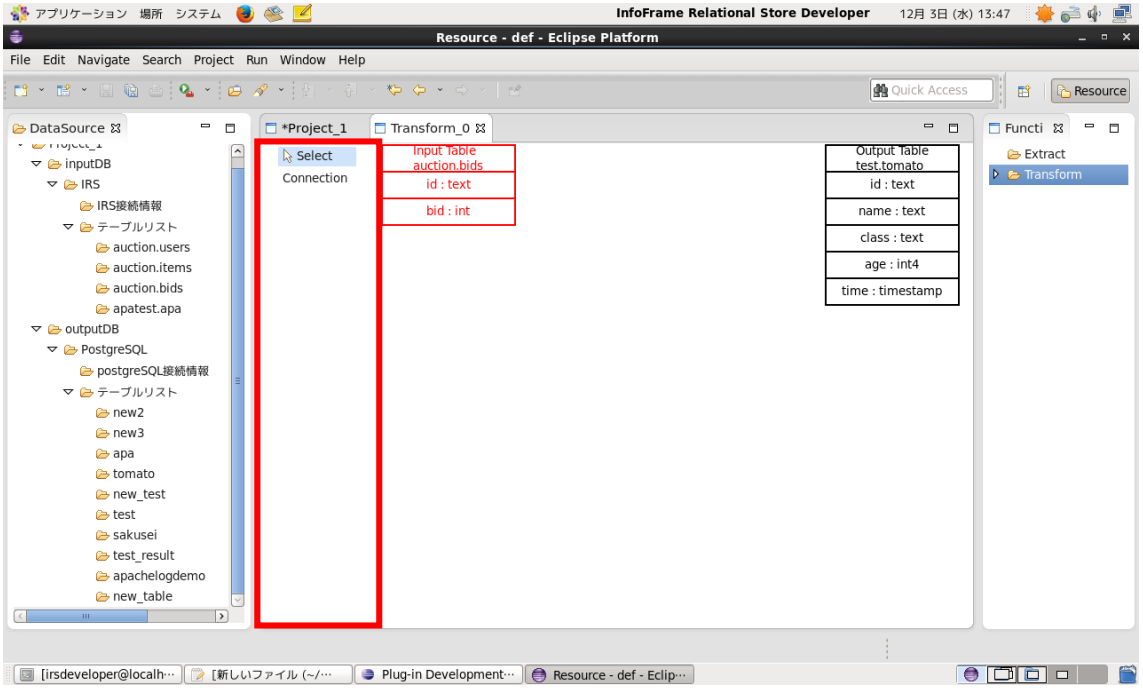
10. Transform 条件を設定する

1. Transfrom アイコンをダブルクリックすると、Transform 編集画面が表示されます。Transform 編集画面には Output Table と Extract から出力される Input Table が表示されます。



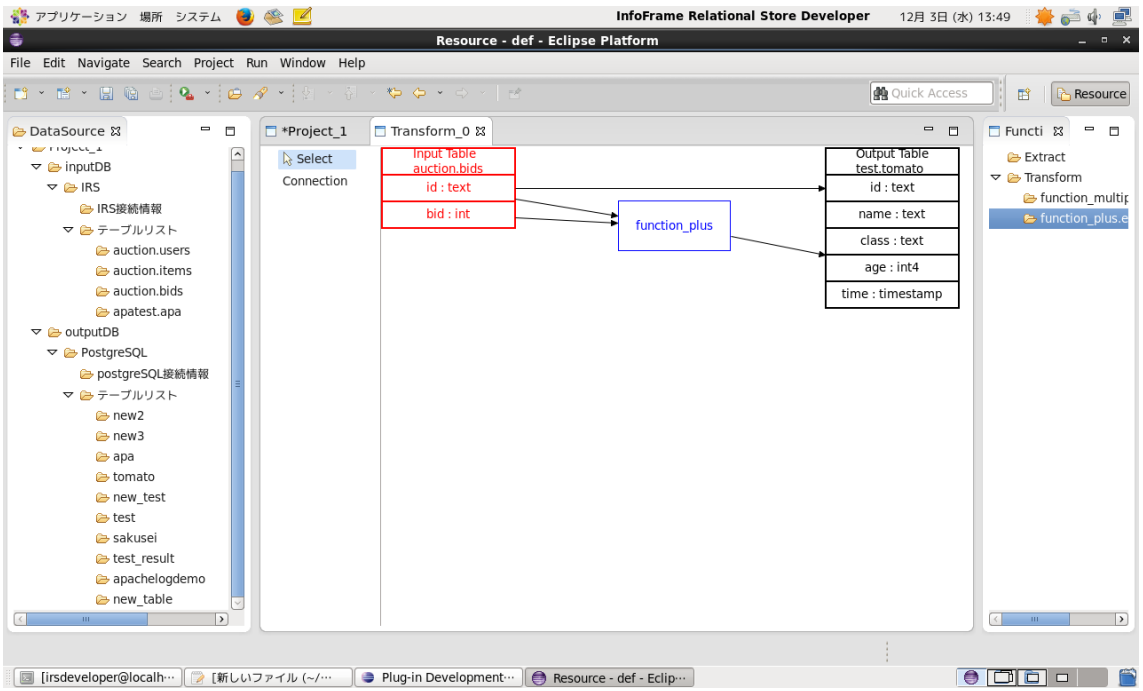
2. 作業ウィンドウの左側に「Select」と「Connection」というツールがあります。

「Select」を選択した場合に、カラムや線の選択ができます。「Connection」を選択した場合に、カラム間の紐付けができます。



3. 入力カラムと出力カラムを線で結ぶことで、入力カラムと出力カラムの対応関係が指定できます。

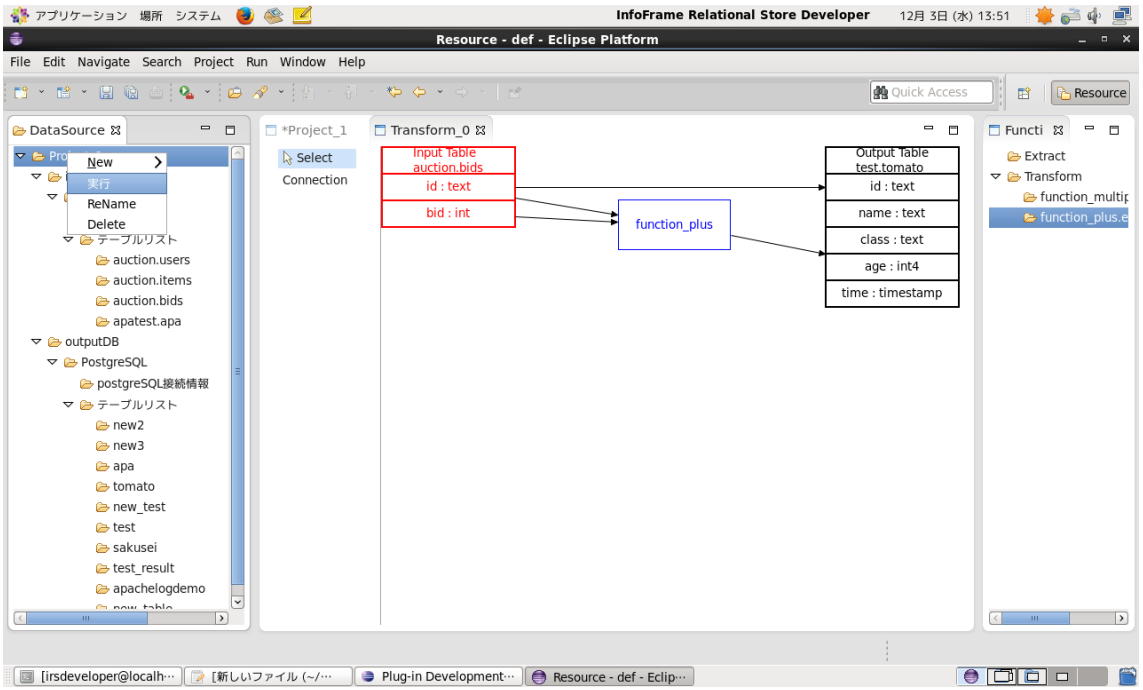
   注意：Input Table のカラムを線の終点とすることができません。同様に、Output Column を線の始点とすることができません。



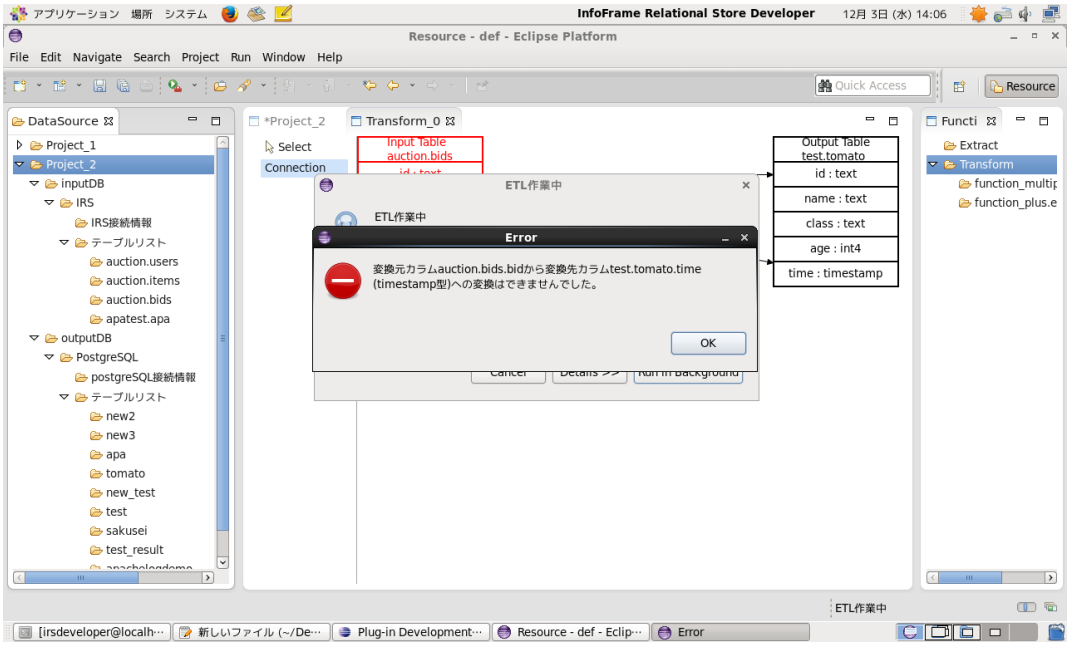
4. 左側の「Select」を選び、テーブルを選択してドラグすることで、テーブルの位置が調整できます。
5. 選択した線を右クリックして、「Delete」を選択することで、線の削除ができます。
6. Transform 編集画面を右クリックして、「Undo」を選択することで、実行済みの操作が取り消せます。
7. Transform 編集画面を右クリックして、「Redo」を選択することで、取り消された操作がやり直せます。

11. プラグインを実行する

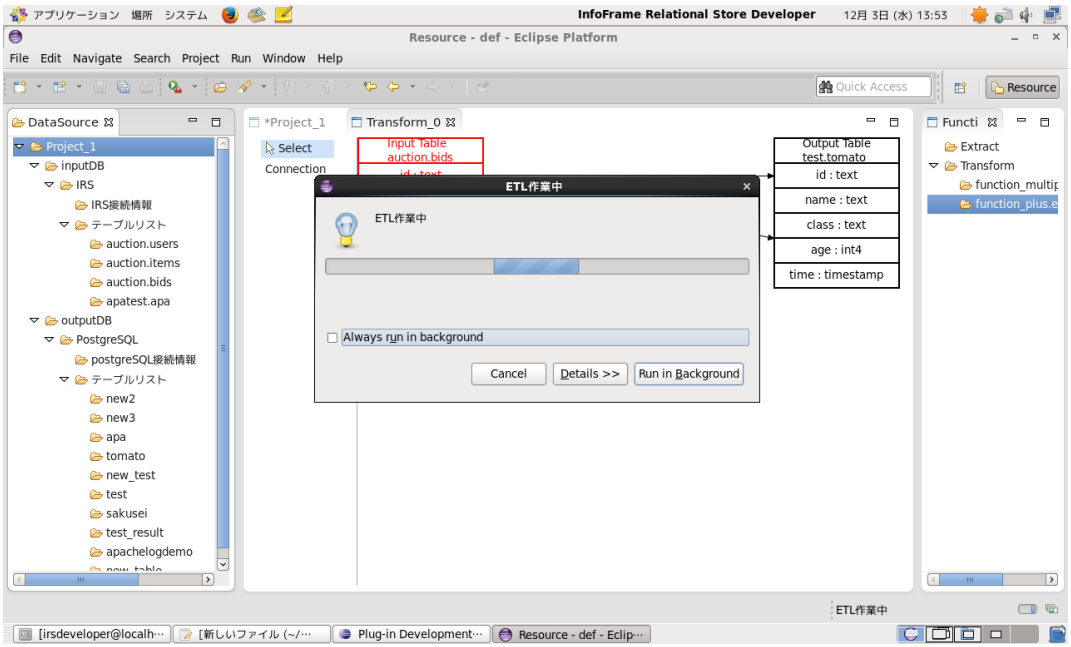
1. Extract 条件編集がすべて終わったら、プロジェクトの実行ができます。プロジェクト名を右クリックして、「実行」を選択すると、プロジェクトが実行されます。



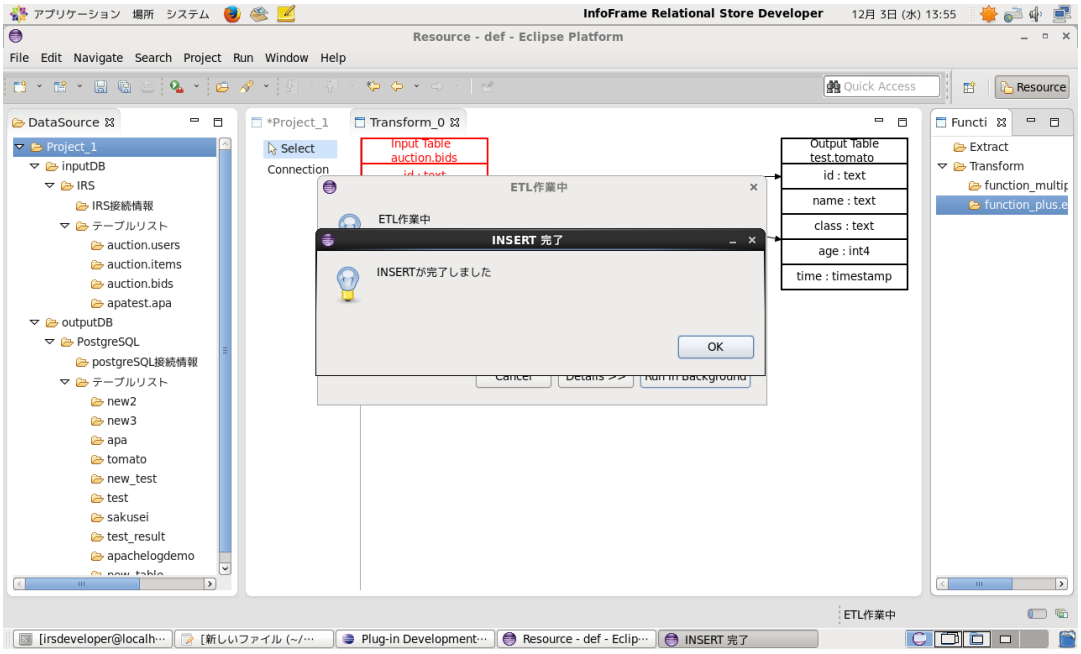
2. プロジェクトの実行が失敗した場合、エラーメッセージが表示されます。



3. データの移行が始まると以下の様なダイアログが表示され、データの移行中であることを表示します。



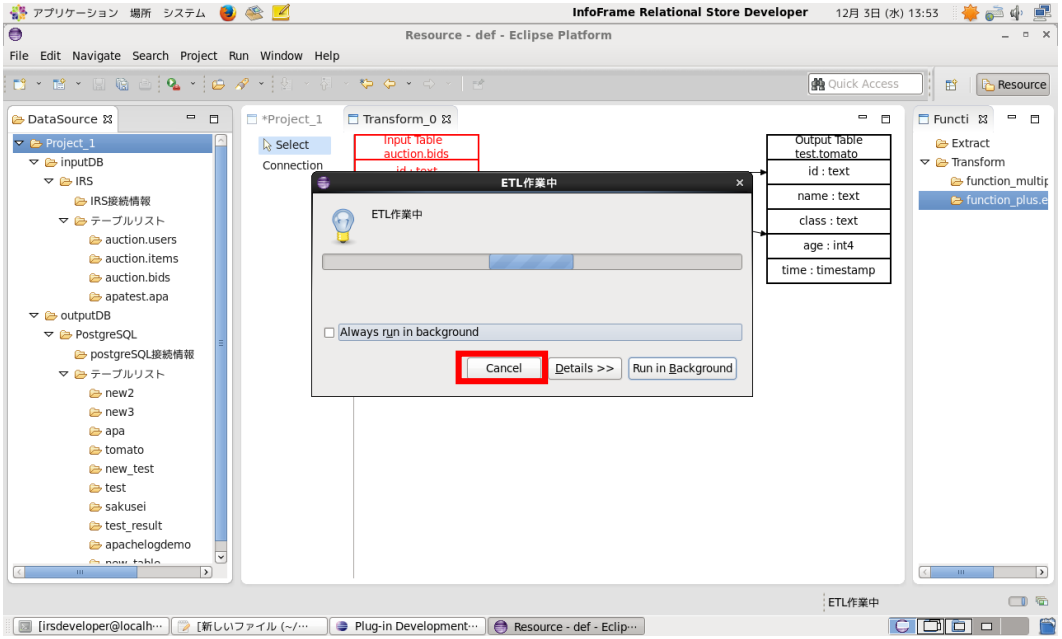
4. 作業が完了すると以下のように完了した旨の表示がされます。OK ボタンを押すと作業中ダイアログと共に完了ダイアログが消えます



12. ETL 作業を中断する

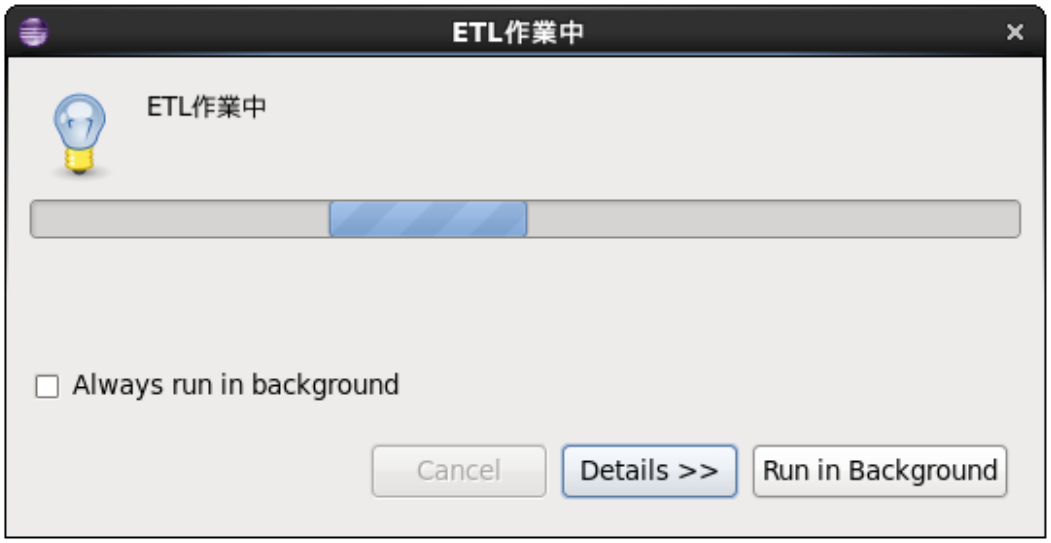
ETL 作業は中断することができる

1. ETL 作業中に表示されるダイアログのキャンセルボタンを押すと ETL 作業を中断することができます。

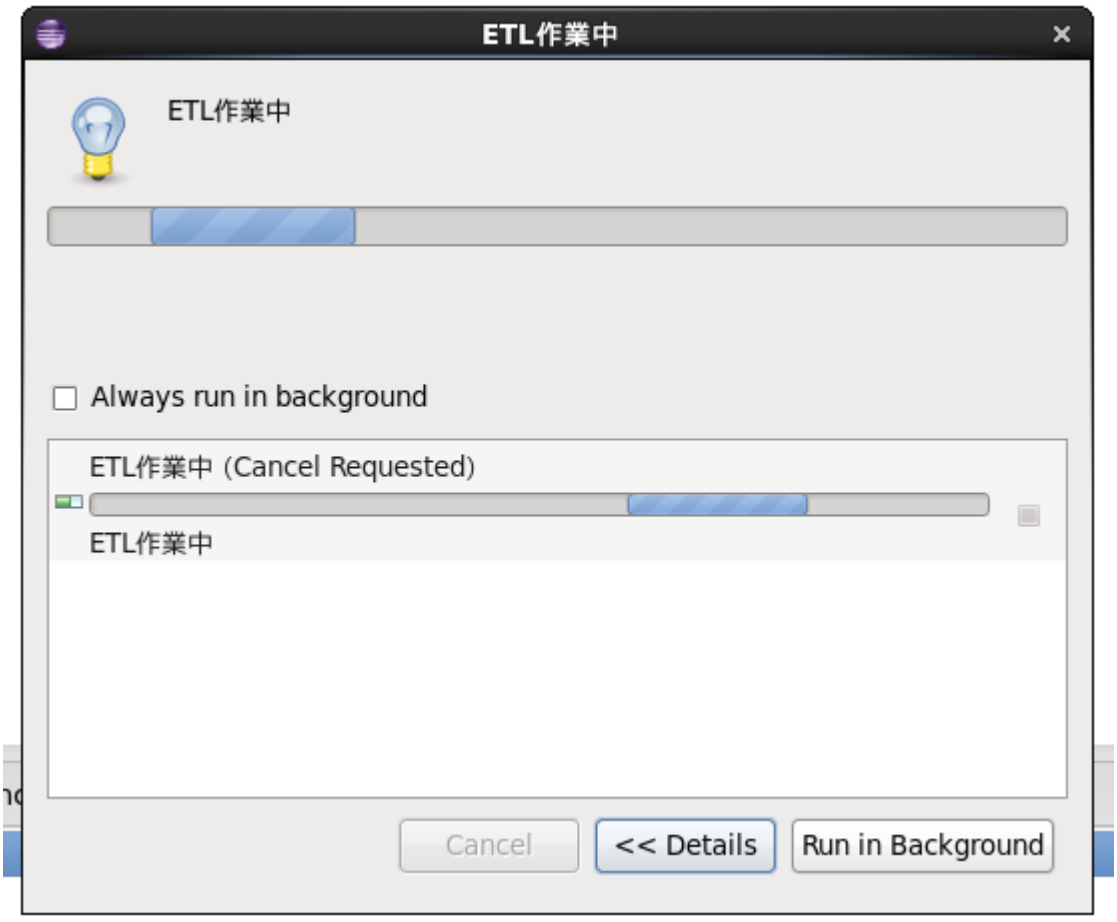


これ移行はダイアログを拡大した画像用いて説明を行います。

2. キャンセルボタンが押されると以下の画面のようにキャンセルボタンが押せなくなります。

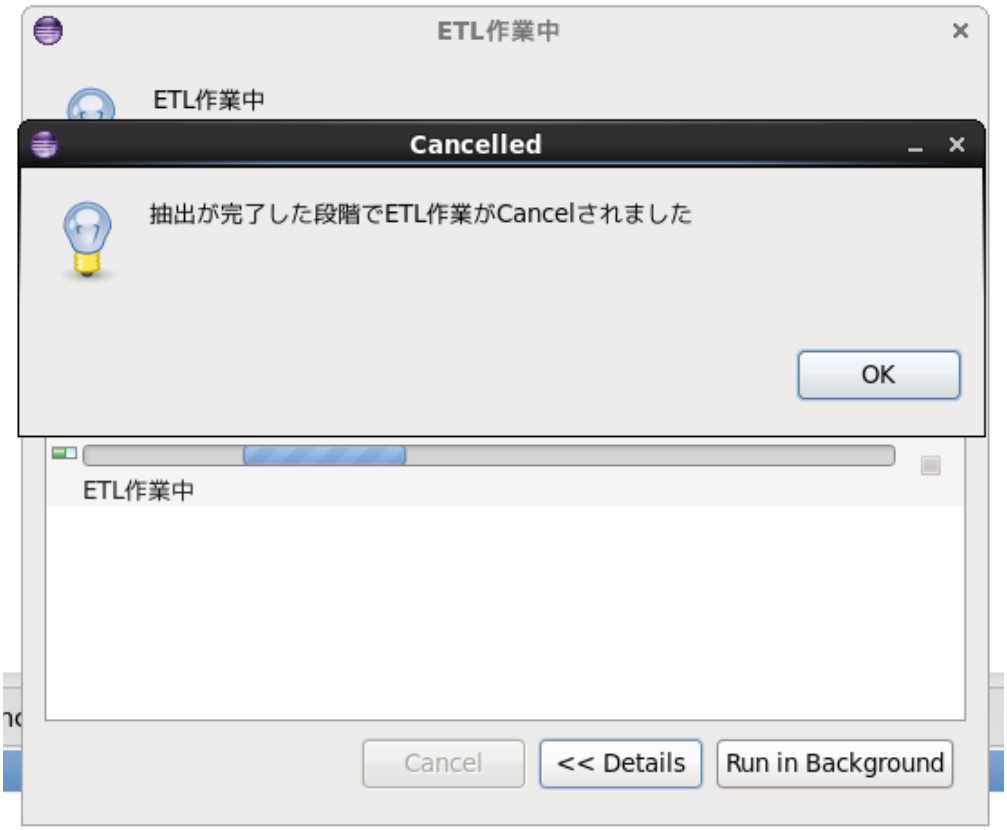


3. キャンセルボタンを押した後に、Detail ボタンを押すと以下のように ETL 作業が中断準備中であることが表示されます。

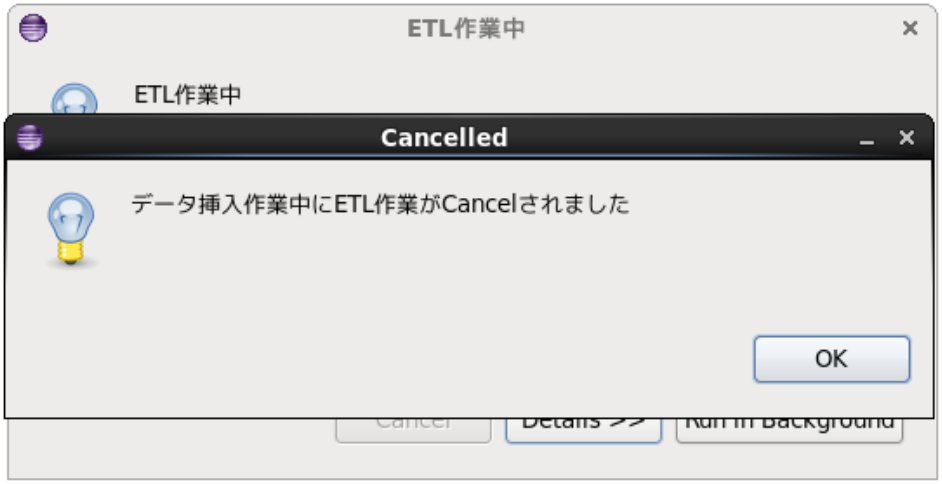


4. 中断するタイミングは２つあります。

(ア) HadoopAPI でのデータ抽出中にキャンセルボタンが押された場合：データ抽出が終了するまで待機し、抽出が終了した段階で ETL 作業を中断します。



(イ) データ挿入中にキャンセルボタンが押された場合：キャンセルボタンが押されたタイミングでデータ挿入と ETL 作業を中断します。キャンセルされる前にデータ移行先データベースに挿入されたデータは、データベースに残ります。



5. キャンセルが行われた後の NET ツールの表示は ETL 作業開始前に戻ります。
6. キャンセルを行った場合は、一度 Eclipse を再起動させてください。

13. 終了する

1. 本プラグインを終了する際には、現在開かれているすべてのウィンドウを閉じてから終了してください。ウィンドウを右クリックし、「Close all」を選択することですべてのウィンドウを閉じることができます。

データ型の変換ルール

データ型の変換ルール

IRS と PostgreSQL のデータ型対応関係は以下となります。

IRS データ型	PostgreSQL データ型	内部動作 Java データ型
INTEGER, INT	int4, serial	java.lang.Integer
BIGINT	int8	java.lang.Long
DECIMAL	decimal	java.math.BigDecimal
FLOAT	float4	java.lang.Float
DOUBLE	float8	java.lang.Double
CHAR, VARCHAR, TEXT	char, varchar, text	java.lang.String
DATE	date	java.sql.Date
TIME	time	java.sql.Time
TIMESTAMP, DATETIME	timestamp	java.sql.Timestamp
BOOLEAN, BOOL	boolean	java.lang.Boolean
BINARY, VARBINARY	bytea	java.lang.Byte の配列 byte[]

データ変換の場合、基本的に上表の通り対応しているデータ型間の変換ができます。

例えば、IRS から INTEGER 型のデータを int4 型の PostgreSQL カラムに挿入する場合、INTEGER から int4 の変換ができます。それ以外の場合は変換できません。データ変換ができない場合、エラーメッセージのポップアップが表示されます。

Transform プラグイン作成マニュアル

Transform プラグイン作成マニュアル

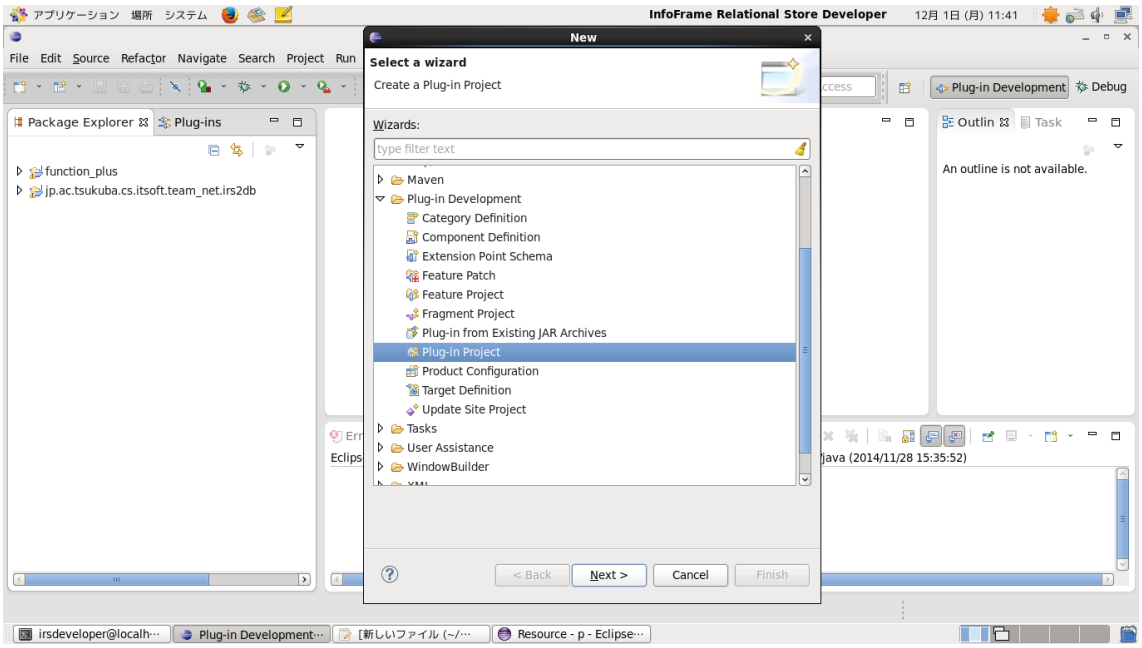
本マニュアルで使った Eclipse のバージョン情報を表 1 に示す。

表 1

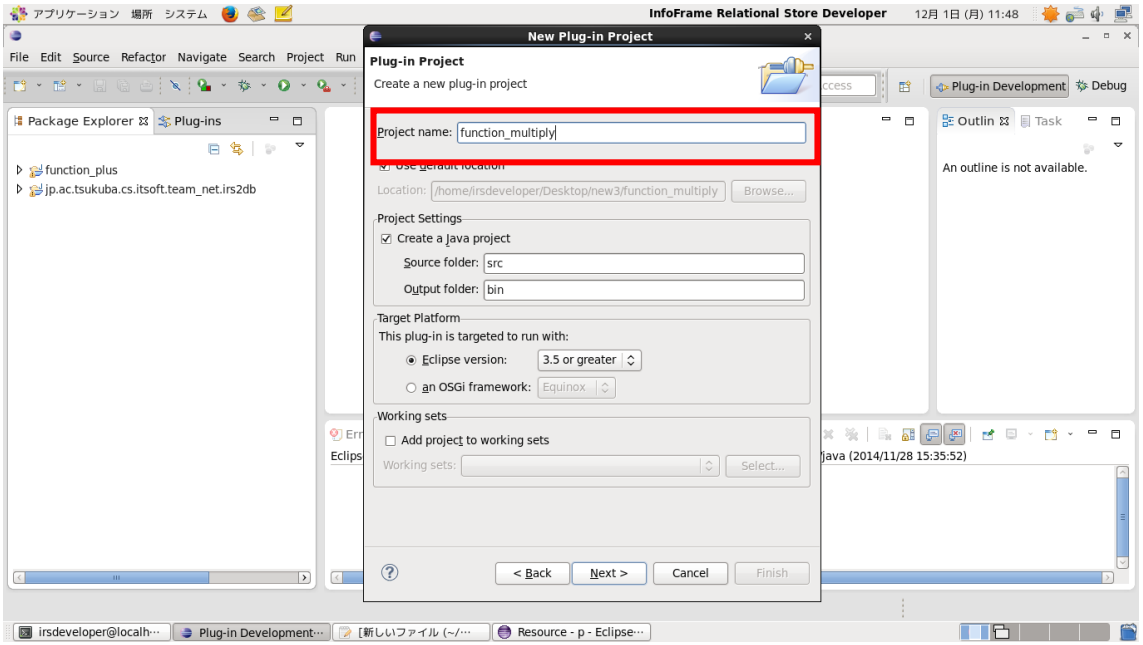
Eclipse for RCP and RAP Developers	
Version	Kepler Service Release 2
Build id	20140224-0627

以下は Transform プラグインの作成手順である。

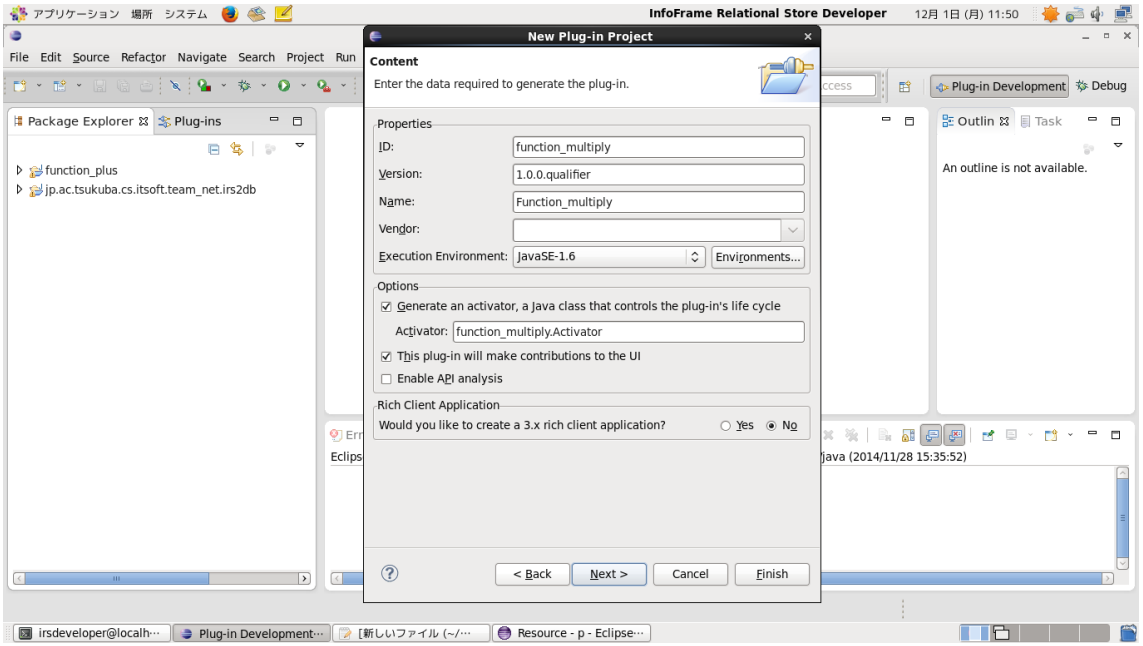
1. File → New → Other → Plug-in Development の Plug-in Project を選択 → Next



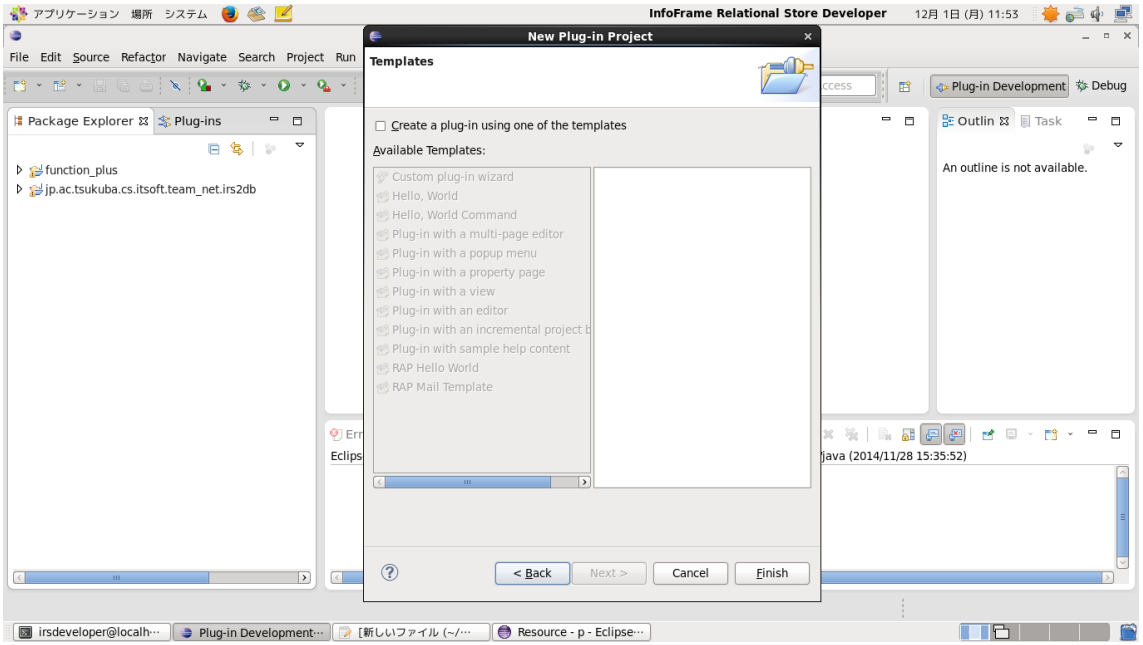
2. プラグインの名前を入力 → Next



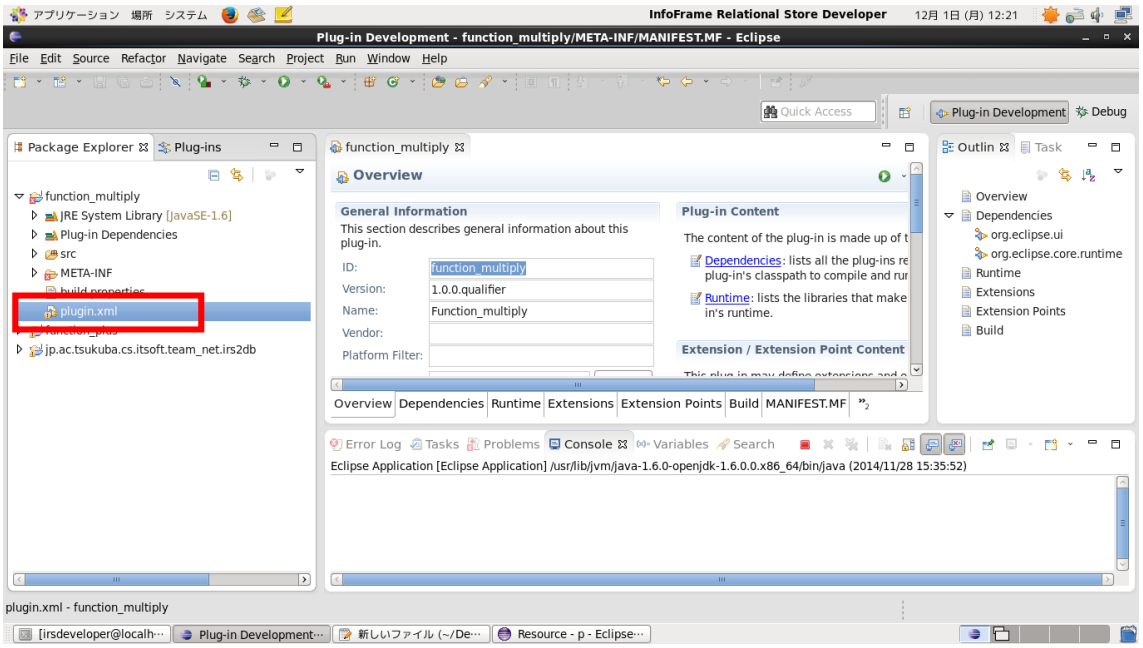
3. 指定がなければ Next



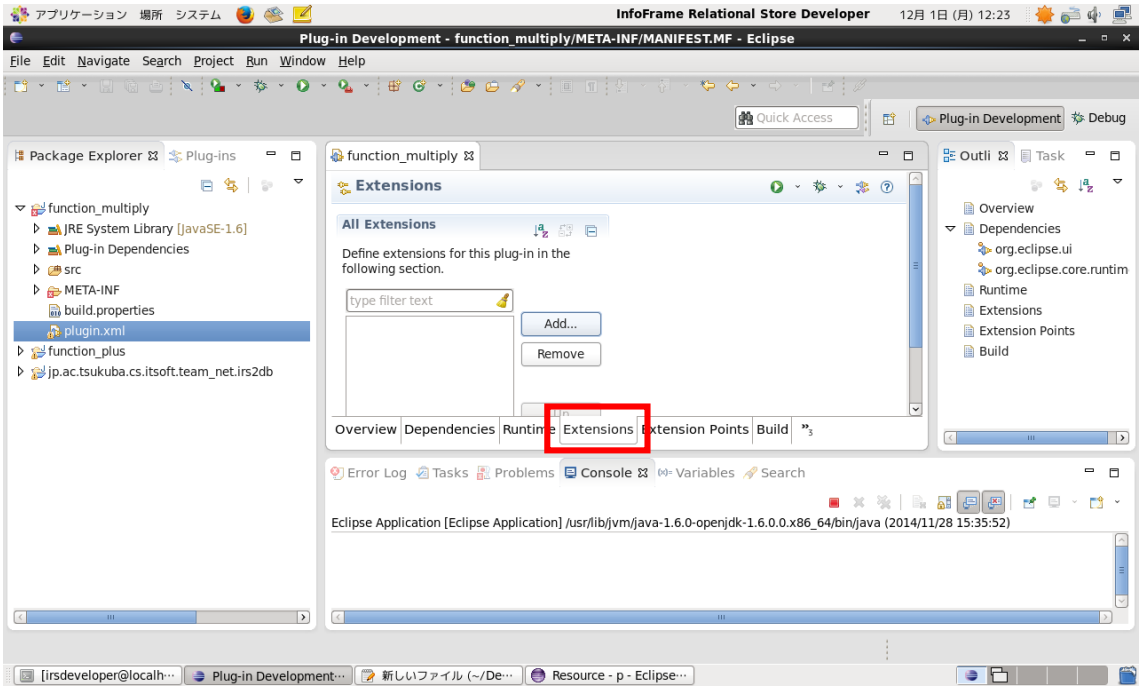
4. 「Create a plug-in using one of the templates」のチェックを外し、Finish ボタンを押すと、作成するプラグインのプロジェクトが作成される



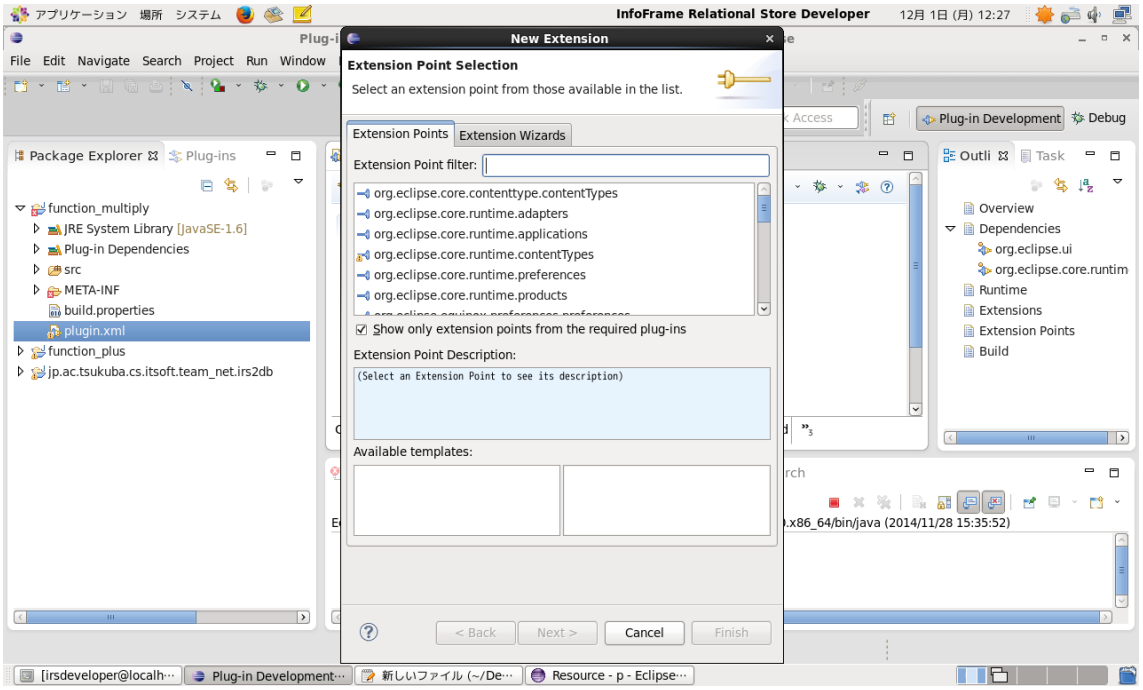
5. 作成するプラグインのプロジェクトを展開し、plugin.xml をダブルクリックして、プラグインの設定ファイルを開く



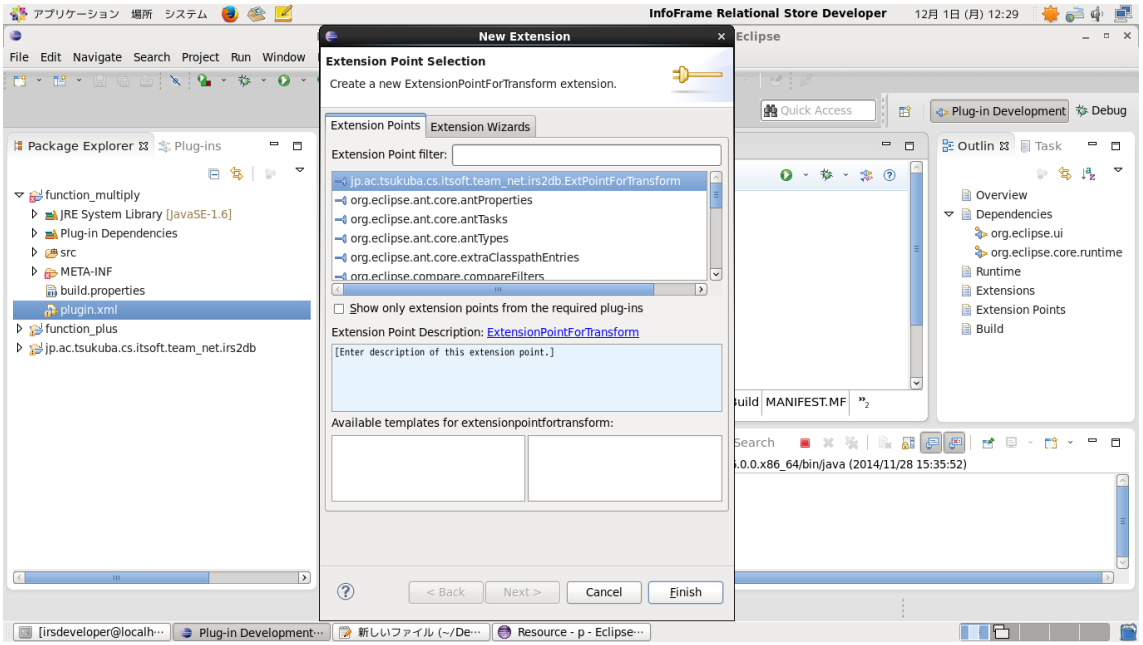
6. 「Extensions」タグを選択する



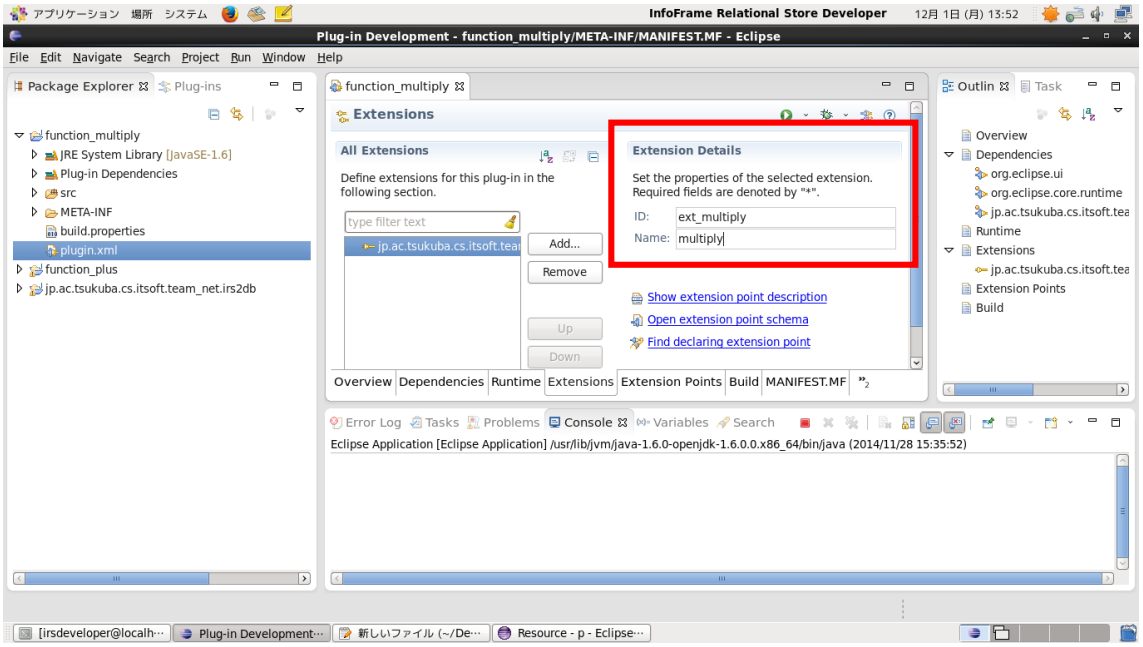
7. 「Add」ボタンをクリックして、Extension Points 選択画面を開く



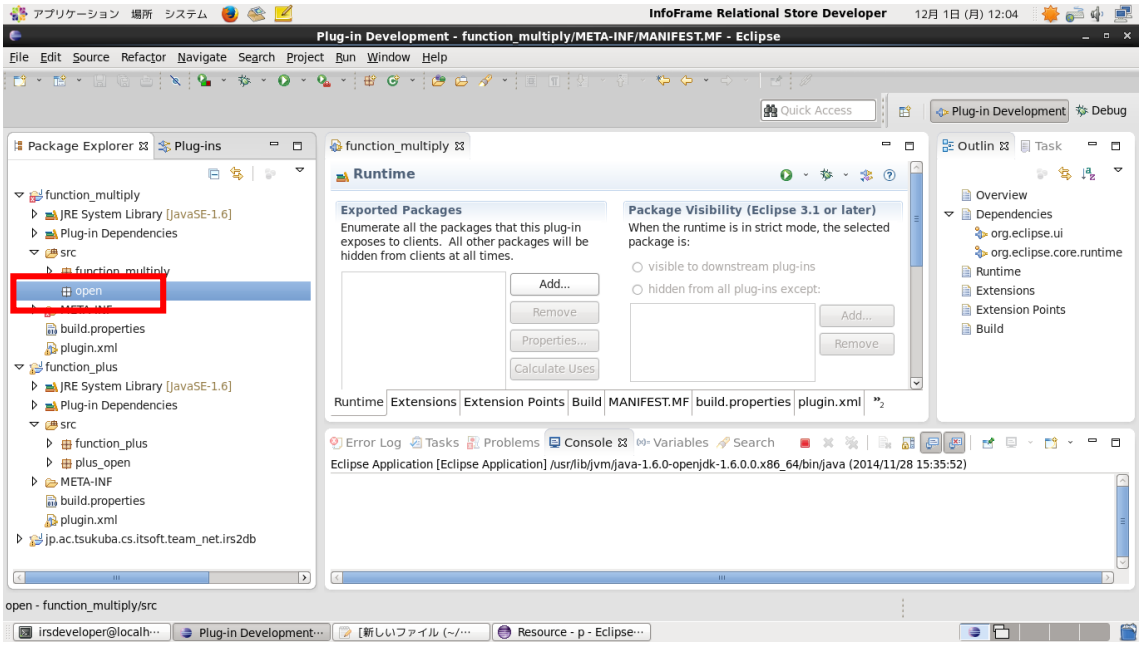
8. 「Show only extension points from the required plug-ins」のチェックを外し、「jp.ac.tsukuba.cs.itsoft.team_net.irs2db.ExtPointForTransform」を選んで、「Finish」ボタンを押す



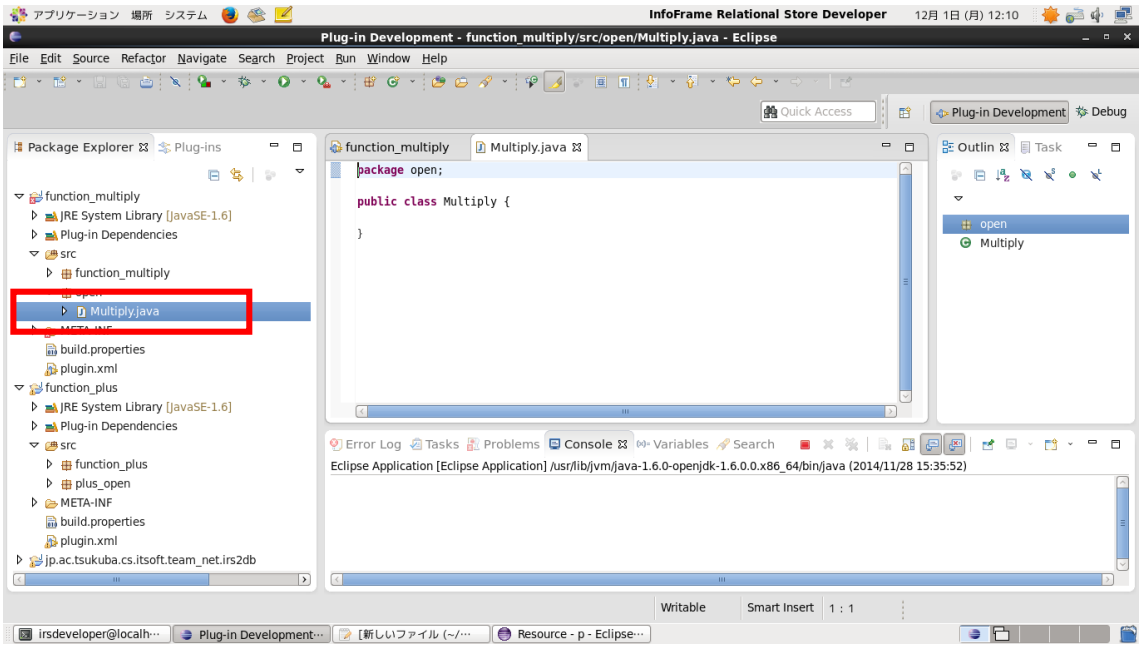
9. 拡張の ID と Name を入力して保存する



10. src の下に外部が見えるパッケージを新規作成する。パッケージ名は任意である。（本マニュアルでは「open」とした）



11. パッケージ「open」の下に NET ツールのインターフェースを実装するクラスを作成する。クラス名は任意である。（本マニュアルでは Multiply とした）



12. 作成したクラスの中身を書いて保存する

(1) クラスは NETObserver を継承する。

(2) コンストラクタで本 Transform ファンクションへの入力数と本 Transform ファンクションからの出力数の上限と下限を設定する。設定しない場合、デファルト値に

みなす。（デフォルト値として、上限は Integer.MAX_VALUE、下限は 0 である）

例：

```
public Multiply(){
  this.setIncoming_max(3);       //入力数の上限を3にする
  this.setIncoming_min(2);       //入力数の下限を2にする
  this.setOutgoing_max(5);       //出力数の上限を5にする
  this.setOutgoing_min(0);       //出力数の下限を0にする
}
```

（3） canExecute メソッドを実装する。このメソッドは当 Transform が実行できるかどうかチェックするメソッドである。メソッドのパラメータ List<Object> preResults は先行要素の計算結果リストであり、当ファンクションへの入力引数リストとする。

例：

乗算の条件として、すべての引数は数字であるかどうかチェックする。数字でない引数が存在する場合に false を返す。存在しない場合に true を返す。

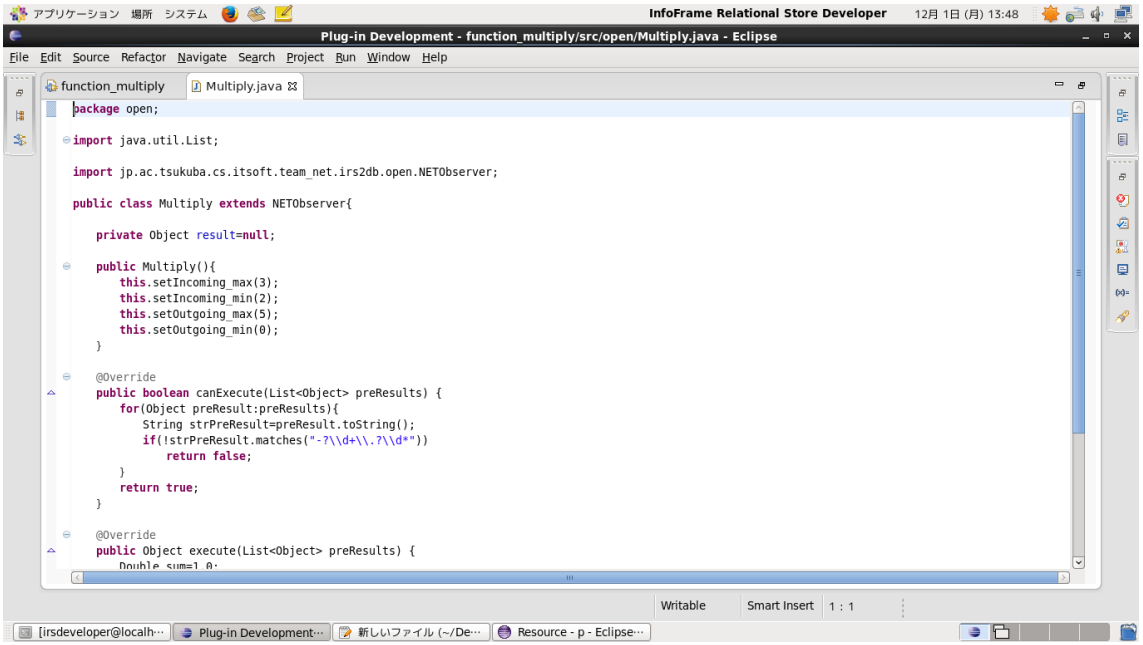
```
public boolean canExecute(List<Object> preResults) {
  for(Object preResult:preResults){
     String strPreResult=preResult.toString(); //引数をString型に変換
     if(!strPreResult.matches("-?¥¥d+¥¥.?¥¥d*")) //正規表現で数字であるかどうかチェック
        return false;
  }
  return true;
}
```

（4） execute メソッドを実装する。このメッソドは当 Transform が実行するメッソドである。メソッドのパラメータ List<Object> preResults は先行要素の計算結果リストであり、当ファンクションへの入力引数リストとする。

例：

引数の累乗を行い、計算結果を返す。

```
public Object execute(List<Object> preResults) {
  Double sum=1.0;
  for(Object obj:preResults){
     double x=Double.parseDouble(obj.toString()); //引数をDouble型に変換
     sum=sum*x;
  }
  result=sum.toString();
  return result;
}
```



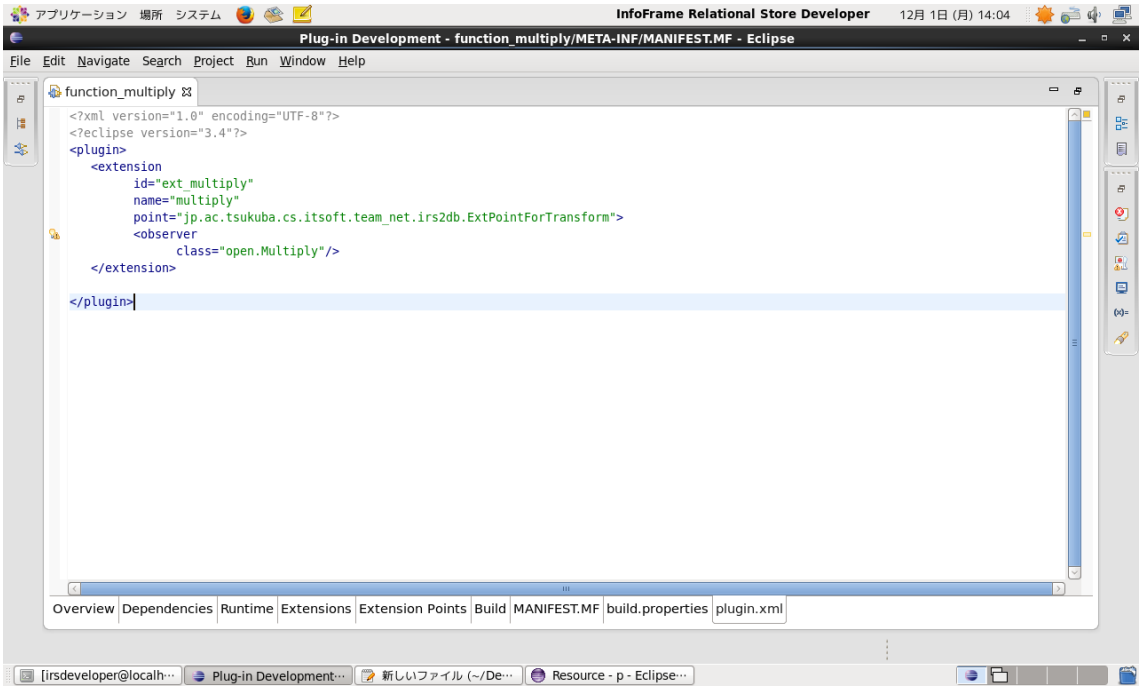
備考：

NET ツールが Transform プラグインを実行する時に、まず canExecute メソッドでプラグインが実行できるかどうか判断して、もし実行できると判断された場合、execute メソッドを実行する。

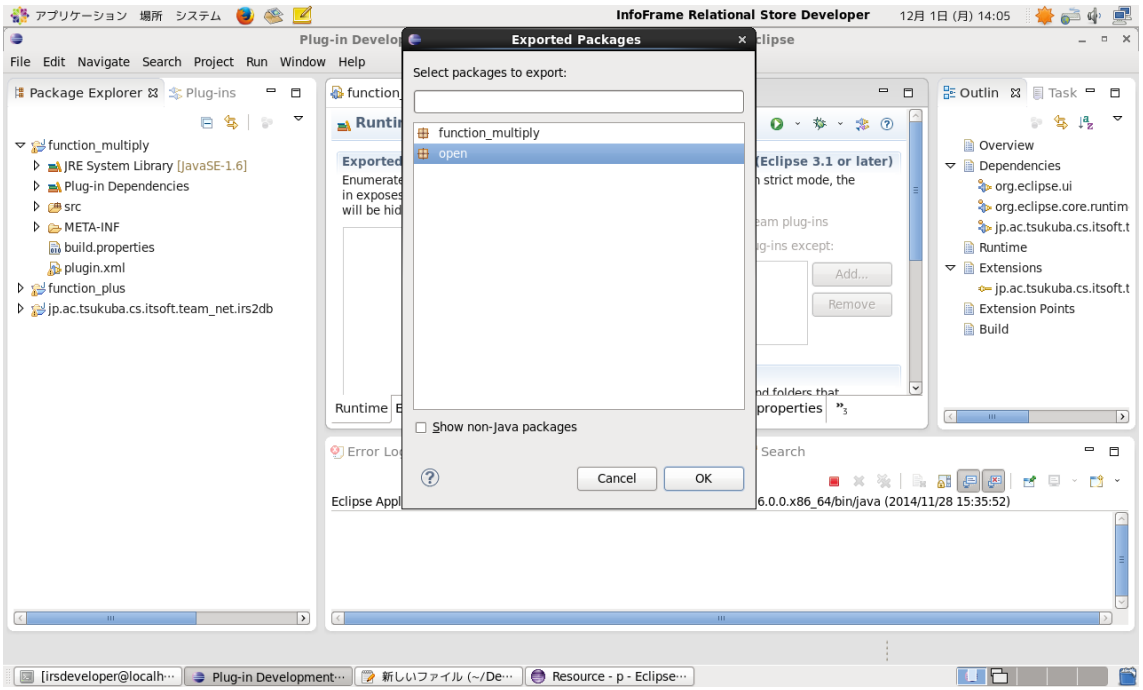
13. plugin.xml を開き、<extension>の中に<observer/>タグを作成して、class 属性の値として 11 に作成したクラスを指定する

    例：

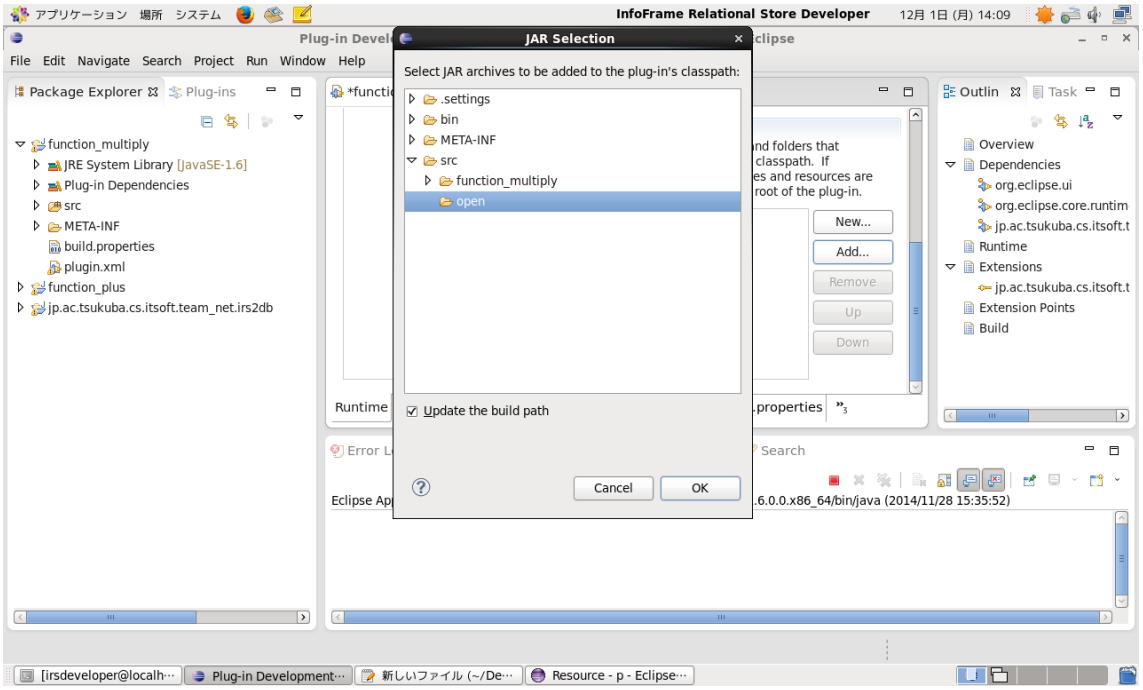
    <observer class="open.Multiply"/>



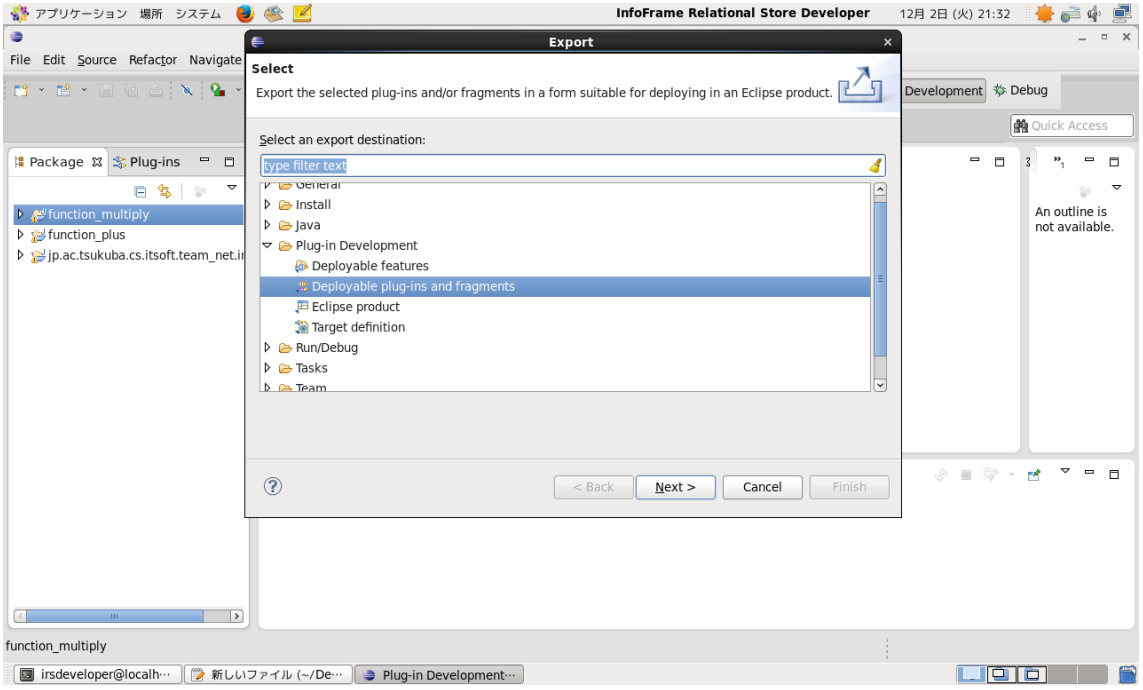
14. Runtime を選択し、Exported Packages エリアで「Add」ボタンを押して、10 で作成したパッケージを選び、「OK」ボタンを押す



15. Classpath エリアで「Add」ボタンを押し、10 で作成したパッケージを選び、「OK」ボタンを押して、以上の設定を保存する

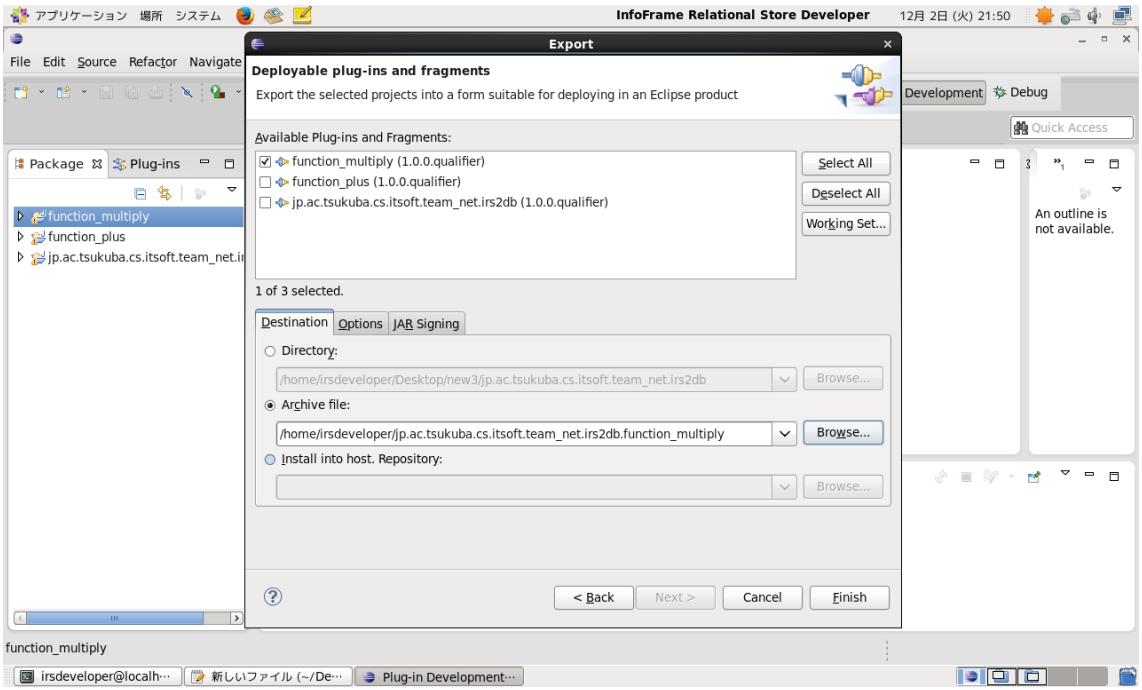


16. File → Export… → 「Plug-in Development」を展開 → 「Deployable plug-ins an fragments」を選択 → Next

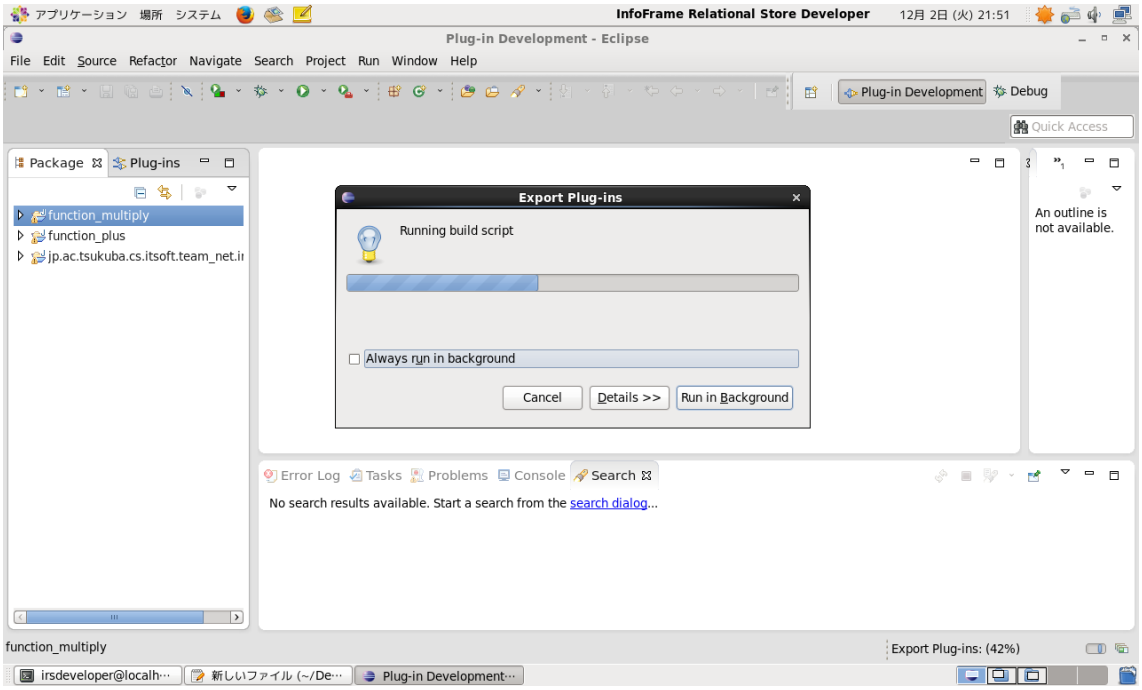


17. 上のエリアで作成するプラグインを選択し、下のエリアで「Destination」タグの「Archive File」をチェックし、エクスポートするアーカイブファイルの保存先を指定

する



18. 「Finish」ボタンを押すと、作成するプラグインをアーカイブする zip ファイルがエクスポートされる



19. エクスポートされた zip ファイルを解凍して、その中にある jar ファイルを Eclipse の

plugins フォルダに移動する

20. 以上で Transform プラグインの作成は終了である。Eclipse を再起動すると、Function ビューの Transform の下に作成した Transform プラグインが追加される